

Click and collect App

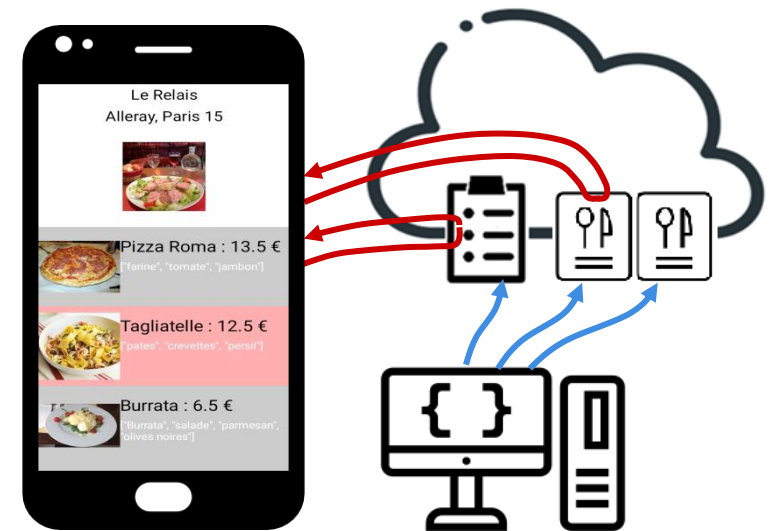
Design and build your own with App Inventor

Pierre Huguet

email : pierre.huguet50@gmail.com

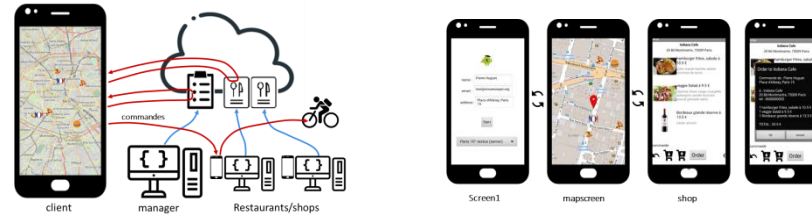
Site web : onvaessayer.org

Youtube : youtube.com/onvaessayer

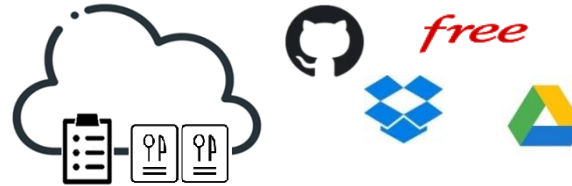


PLAN

- Introduction : décomposition de l'application



1. Création d'un site Web / serveur de données



2. Définition des données et préparation d'un jeu

- formats JSON et geoJSON
- modèles de données



3. Création de l'application mobile avec App Inventor

- algorithmes
- développement en étapes

```
initialize global URLgeoJSONCatalog to join "https://onvaessayer.github.io/" "gitshareData1/map_geojson"

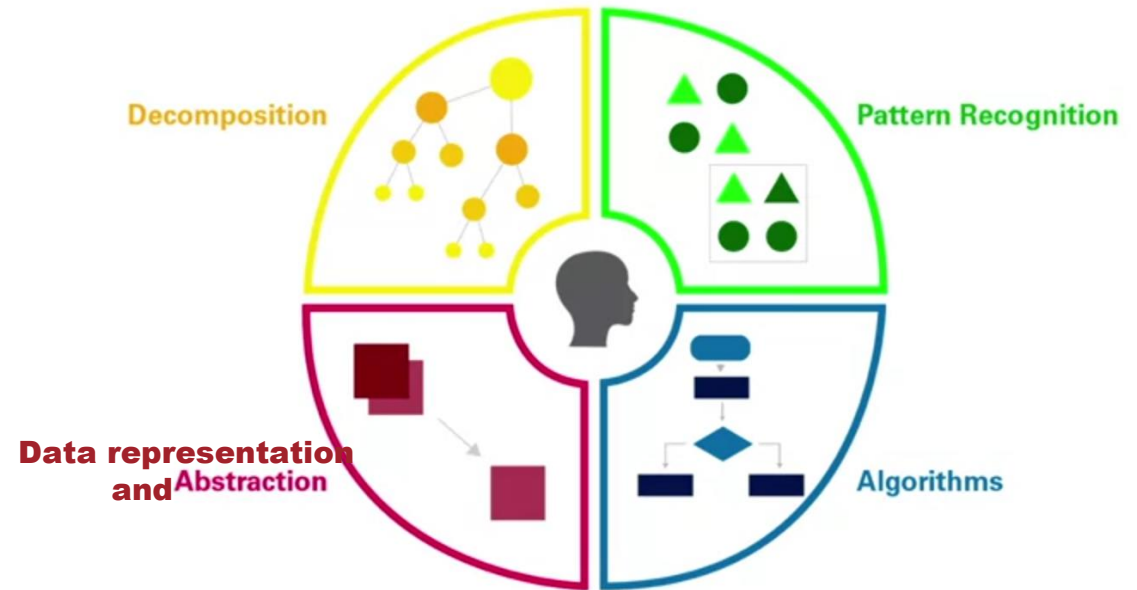
when Screen1.Initialize
do call Map1.LoadFromURL url get global URLgeoJSONCatalog
```

PLAN

- Introduction : décomposition de l'application

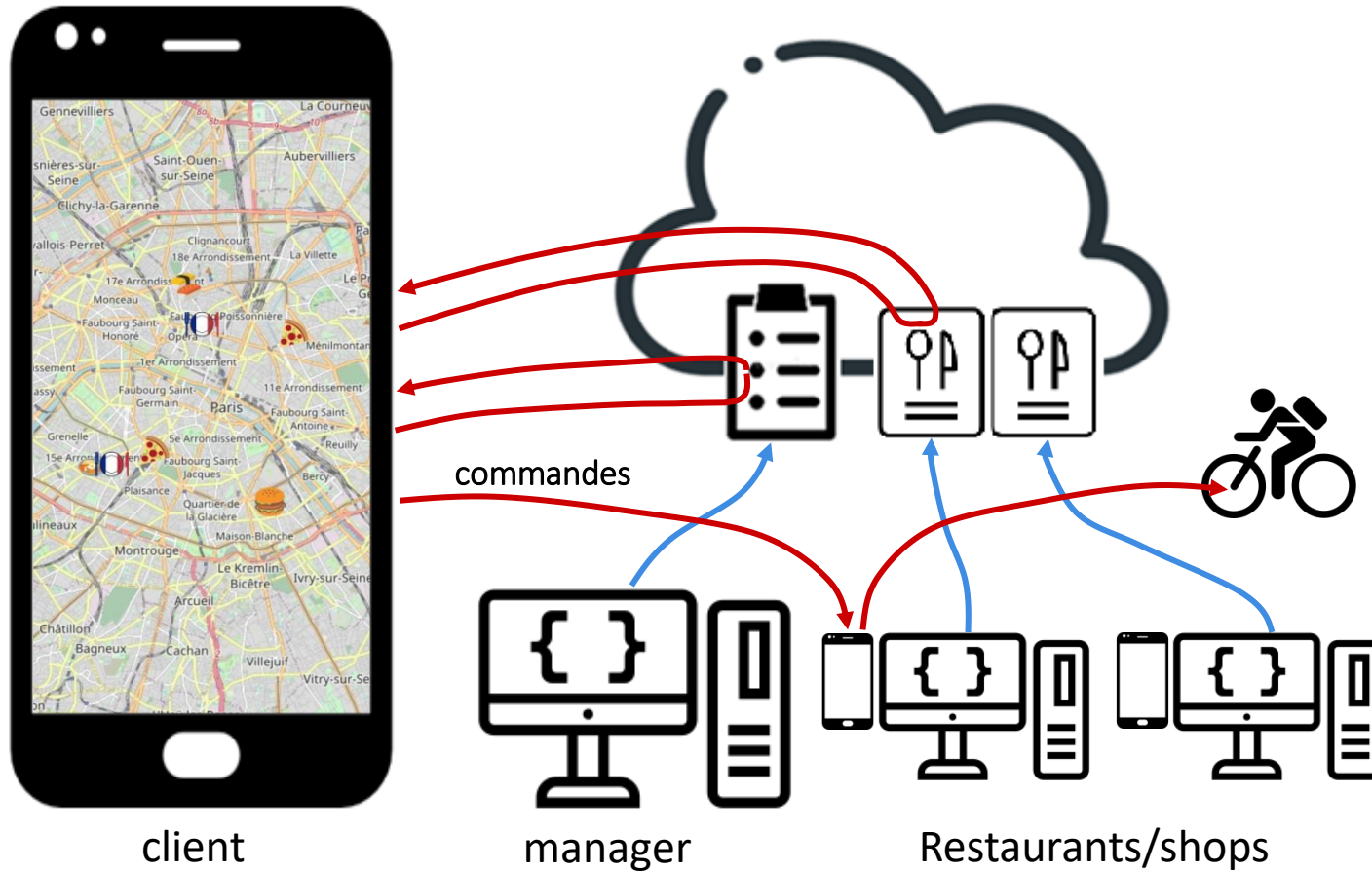
1. Création d'un site Web / serveur de données
 - formats JSON et geoJSON
 - modèles de données
2. Définition des données et préparation d'un jeu
 - algorithmes
 - développement en étapes
3. Création de l'application mobile avec App Inventor

Pillars of Computational Thinking

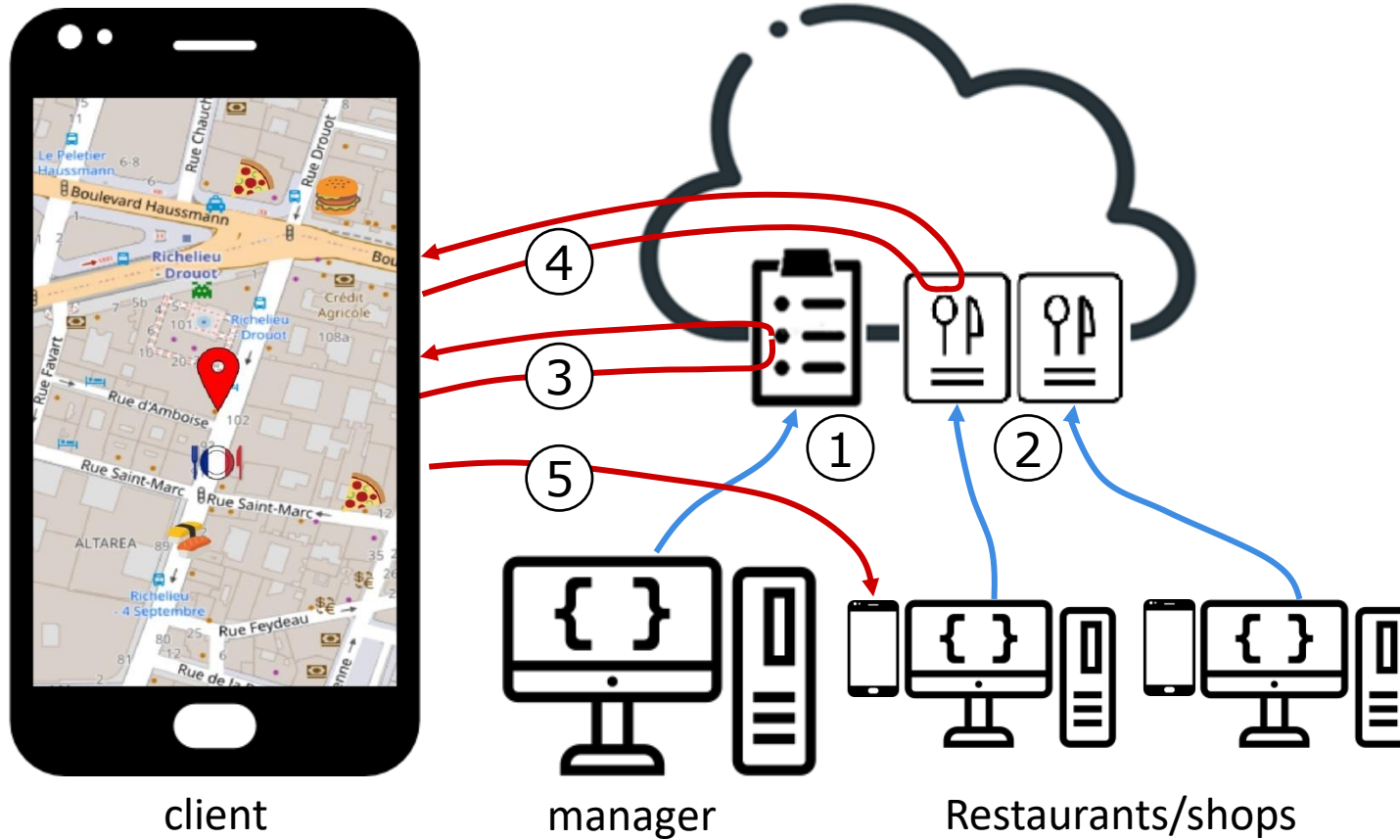


Jeannette M. Wing: Computational thinking. Commun. ACM 49(3) [1]: 33-35 (2006)
Property of Penn Engineering

DÉCOMPOSITION DE L'APPLICATION : LES FLUX

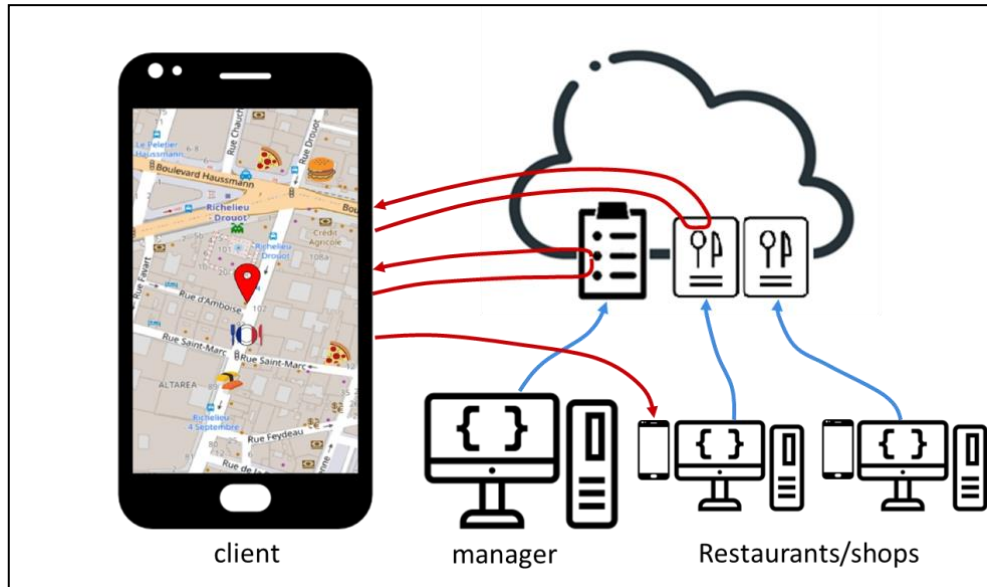


DÉCOMPOSITION DE L'APPLICATION : LES FLUX

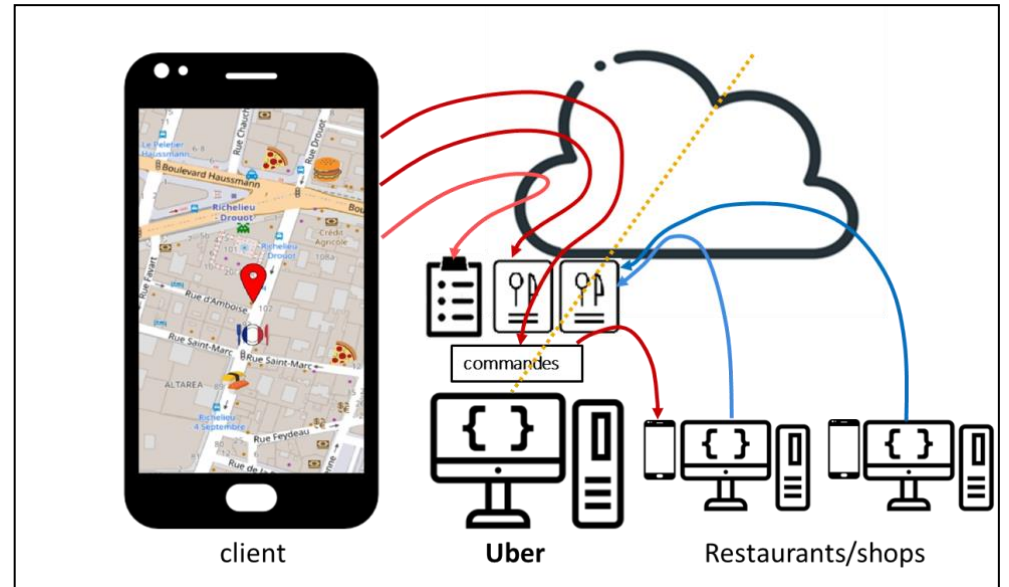


CHOIX D'ARCHITECTURE : MODE RÉPARTI / MODE CENTRALISÉ

Ce que l'on va faire

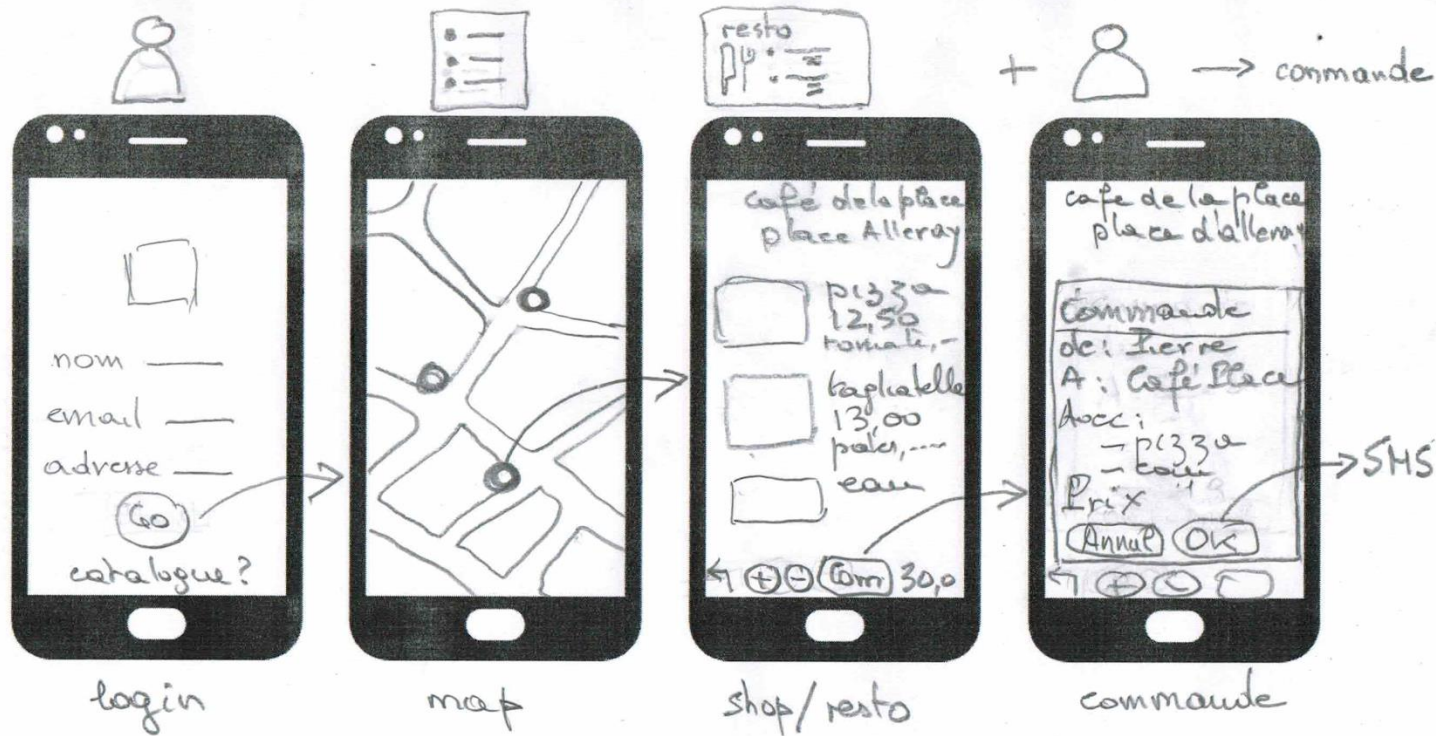


applis de type UBER, click and collect, ...



- analyse comparative
 - où est le pouvoir, comment sont répartis les tâches et les flux financiers ?
 - En fait, quel service paie t-on à Uber ?
- Les choix effectués en début de projet sont structurants

DÉCOMPOSITION : PROTOTYPAGE PAPIER



DÉCOMPOSITION DE L'APPLICATION



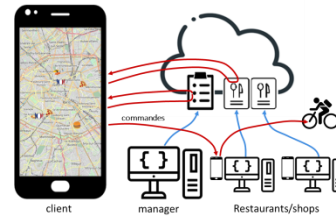
Screen1

mapscreen

shop

PLAN : INTRODUCTION

- Introduction : décomposition de l'application
 - Présentation de l'application
 - Décomposition des flux
 - Plan de travail
 - Pré-requis, moyens utilisés
 - Ressources fournies
- 1. Création d'un site Web / serveur de données
- 2. Définition des données et préparation d'un jeu
 - formats JSON et geoJSON
 - modèles de données
- 3. Création de l'application mobile avec App Inventor
 - algorithmes
 - développement en étapes



```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData1/map_geojson"]

when Screen1.Initialize
do call Map1.LoadFromURL
url get global URLgeoJSONCatalog
```

PLAN DES CHAPITRES 1 ET 2

- 00:00
 - Introduction : décomposition de l'application

- 09:38** 1. Création d'un site Web / serveur de données
 - 10:38
 - créer un compte github
 - 12:04
 - créer un repository "username".github.io
 - 13:10
 - créer et partager un fichier index.html

- 14:47** 2. Définition des données et préparation d'un jeu de test
 - 15:10
 - Définition et organisation des données
 - 15:41
 - formats JSON & geoJSON
 - 16:02
 - modèle de restaurant au format JSON
 - 17:54
 - Modèle de catalogue d'objets géolocalisés au format geoJSON
 - 21:15
 - télécharger les données sur le serveur Github
 - 23:02
 - choix des propriétés geoJSON pour app inventor

- 24:54** 3. Création de l'application mobile avec App Inventor

PLAN

- Introduction : décomposition de l'application

09:38 1. Création d'un site Web / serveur de données

14:47 2. Définition des données et préparation d'un jeu

24:54 3. Création de l'application mobile avec App Inventor

25:29 3.1 V1 : visualiser le catalogue des restaurants sur une carte

33:45 3.2 V2a : afficher un restaurant : nom, adresse, image et liste de plats

44:58 3.3 V2b : codage défensif, modèle de données,

51:59 adresses relatives, Dropbox & Google Drive

1:01:23 3.4 V3a : identifier et enregistrer l'utilisateur et la carte

1:11:44 recentrer la carte sur la position de l'utilisateur

1:18:32 3.5 V3b : préparer et passer une commande

1:33:11 3.6 V3c : bonus

PLAN : CHAPITRE 3

24:54

25:29 3.1 V1 : visualiser le catalogue des restaurants sur une carte

33:45 3.2 V2a : afficher un restaurant : nom, adresse, image et liste de plats

44:58 3.3 V2b : modèle, contrôle et gestion des données

3.3.1 codage défensif, modèle de données,

3.3.2 adresses relatives,

3.3.3 Dropbox & Google Drive

1:01:23 3.4 V3a : profil utilisateur

3.4.1 enregistrement du profil utilisateur et de

3.4.2 enregistrement de la carte

3.4.3 centrer la carte sur la position de l'utilisateur

1:18:32 3.5 V3b : passer commande

3.5.1 Design 3.5.2 restriction aux plats définis comme objets

3.5.3 Analyse fonctionnelle, pseudo-code

3.5.4 Programmation 3.5.5 essais de la version3b

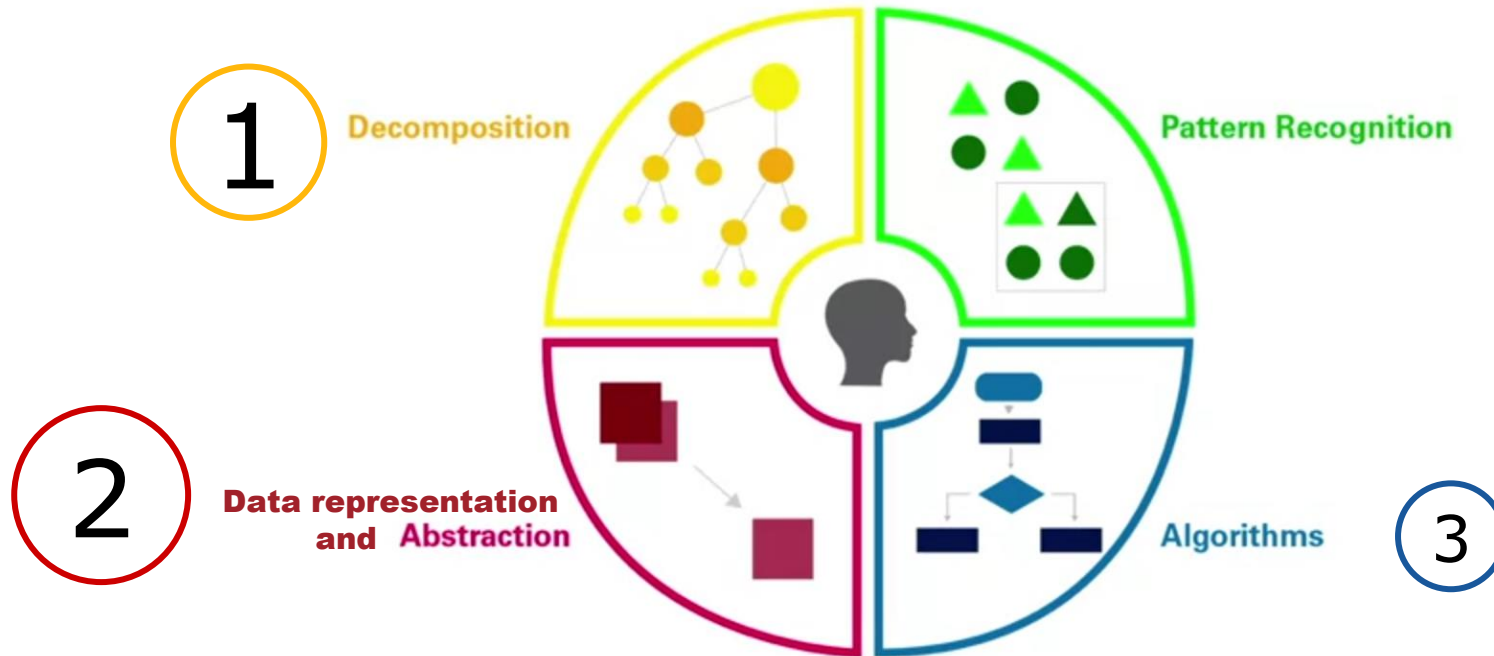
1:33:11 3.6 V3c : bonus

3.6.1 paiements, partage de l'appli, 3.6.2 vérification du n° de téléphone,

3.6.3 partage de catalogues, 3.6.4 affichage amélioré

PRINCIPALES ACTIVITÉS PROPOSÉES POUR LA RÉALISATION

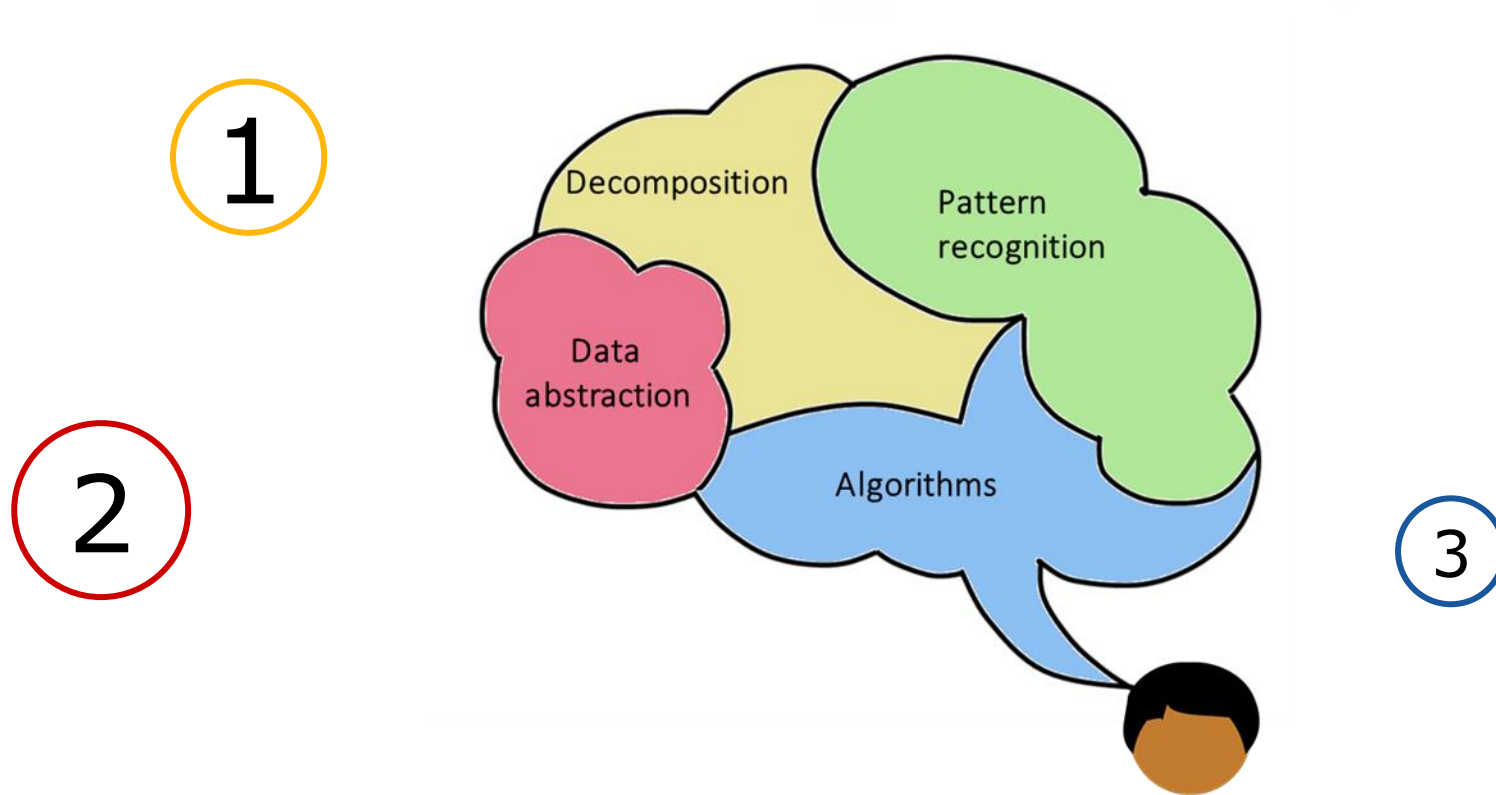
Pillars of Computational Thinking



Jeannette M. Wing: Computational thinking. Commun. ACM 49(3) [1]: 33-35 (2006)
Property of Penn Engineering

PRINCIPALES ACTIVITÉS PROPOSÉES POUR LA RÉALISATION

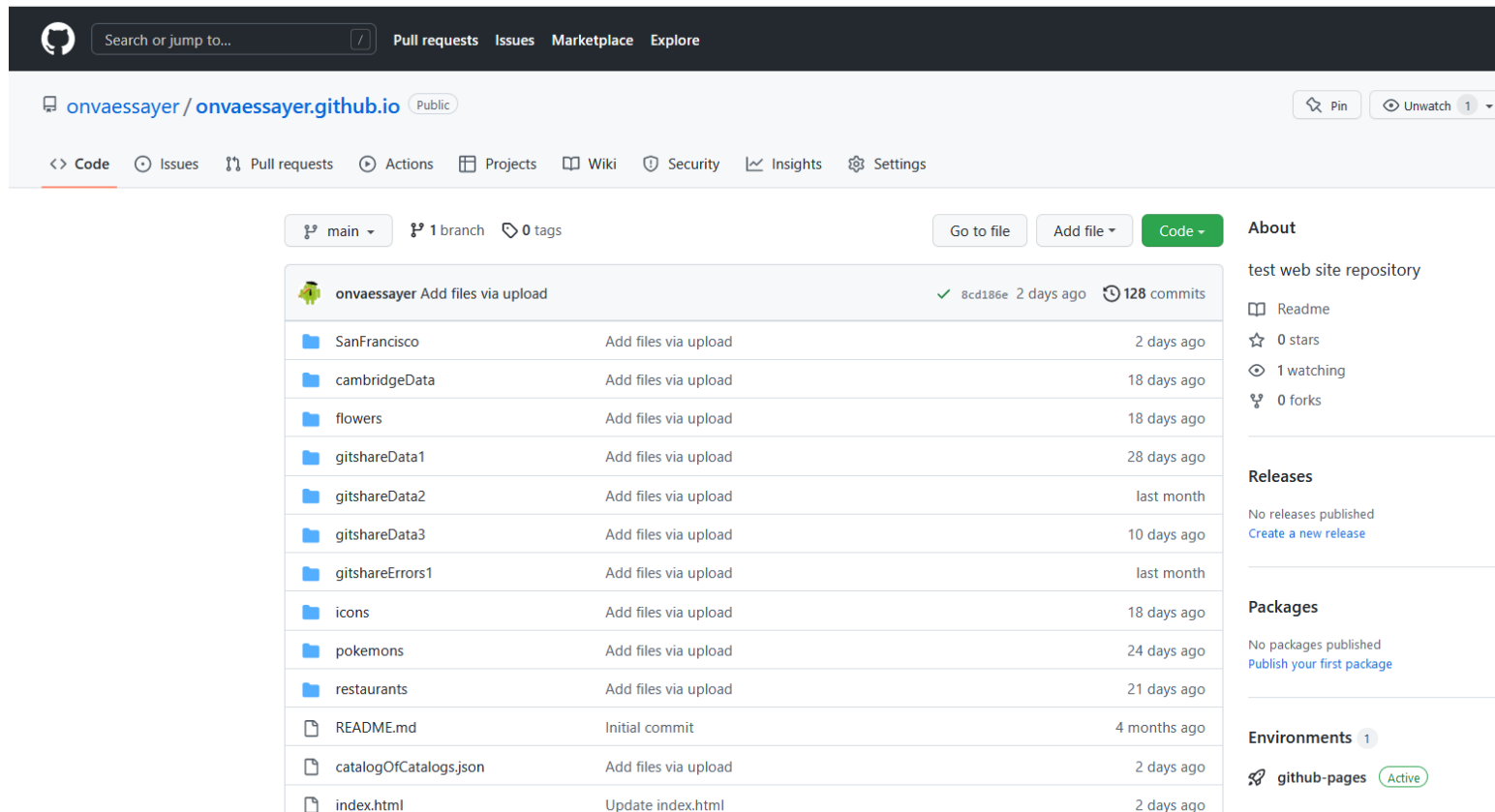
Pillars of Computational Thinking



RESSOURCES UTILISÉES : gratuites et/ou libres d'accès

Github : <https://github.com/>

(inscription gratuite nécessaire avec un email)



The screenshot shows the GitHub interface for a public repository named 'onvaessayer/onvaessayer.github.io'. The repository is described as a 'test web site repository'. It contains 128 commits, with the most recent one by user '8cd186e' made 2 days ago. The repository has 0 stars, 1 watcher, and 0 forks. The file list includes several folders for data and images, a README.md file, and a catalogOfCatalogs.json file. The right sidebar shows sections for 'About', 'Releases', 'Packages', and 'Environments', with 'github-pages' currently active.

File/Folder	Commit Message	Time Ago
SanFrancisco	Add files via upload	2 days ago
cambridgeData	Add files via upload	18 days ago
flowers	Add files via upload	18 days ago
gitshareData1	Add files via upload	28 days ago
gitshareData2	Add files via upload	last month
gitshareData3	Add files via upload	10 days ago
gitshareErrors1	Add files via upload	last month
icons	Add files via upload	18 days ago
pokemons	Add files via upload	24 days ago
restaurants	Add files via upload	21 days ago
README.md	Initial commit	4 months ago
catalogOfCatalogs.json	Add files via upload	2 days ago
index.html	Update index.html	2 days ago

RESSOURCES UTILISÉES : gratuites et/ou libres d'accès

App Inventor : <https://appinventor.mit.edu/> (inscription optionnelle avec un email)

MIT APP INVENTOR

Projects ▾ Connect ▾ Build ▾ Settings ▾ Help ▾

gitshare01 Screen1 ▾ Add Screen ... Remove Screen Publish to Gallery

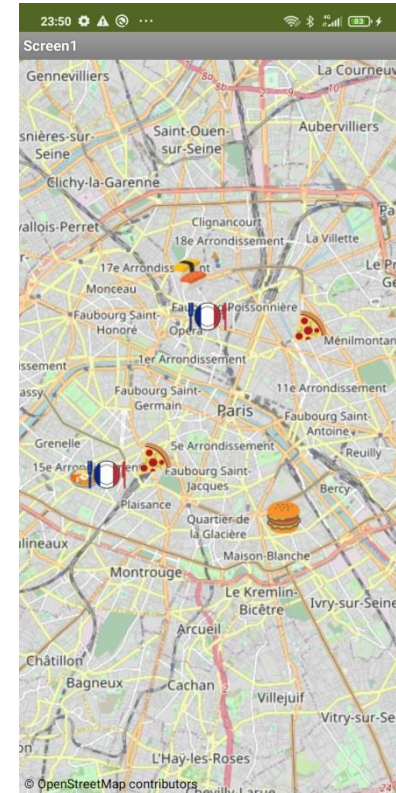
Blocks

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Dictionaries
 - Colors
 - Variables
 - Procedures
- Screen1
 - Map1
- Any component

Viewer

```
initialize global URLgeoJSONCatalog to join "https://onvaessayer.github.io/" "gitshareData1/map.geojson"
```

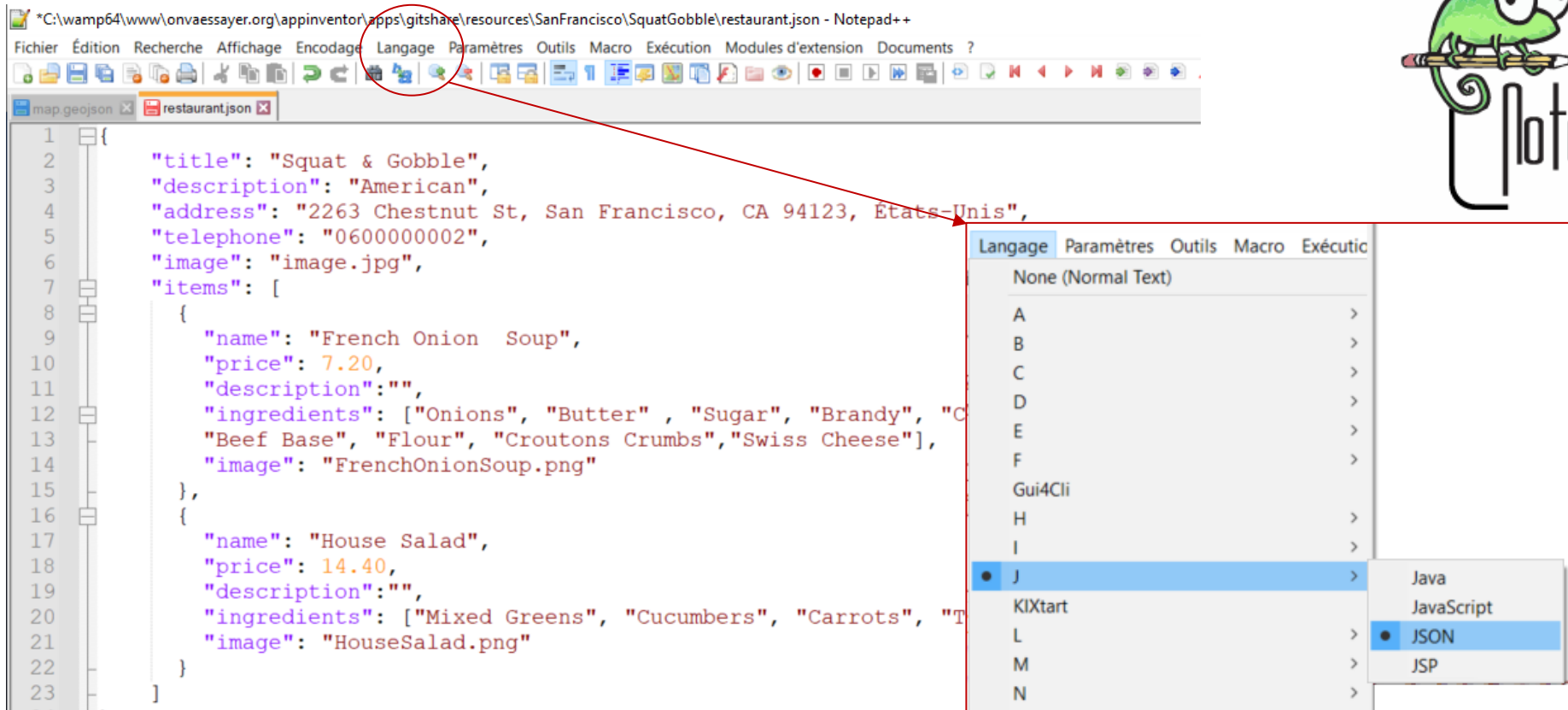
```
when Screen1.Initialize do call Map1.LoadFromURL url get global URLgeoJSONCatalog
```



RESSOURCES UTILISÉES : gratuites et/ou libres d'accès

Notepad++ : <https://notepad-plus-plus.org/>

(à télécharger)



The screenshot shows the Notepad++ interface with a JSON file open. The file content is as follows:

```
1 {
2   "title": "Squat & Gobble",
3   "description": "American",
4   "address": "2263 Chestnut St, San Francisco, CA 94123, États-Unis",
5   "telephone": "0600000002",
6   "image": "image.jpg",
7   "items": [
8     {
9       "name": "French Onion Soup",
10      "price": 7.20,
11      "description": "",
12      "ingredients": ["Onions", "Butter", "Sugar", "Brandy", "C",
13      "Beef Base", "Flour", "Croutons Crumbs", "Swiss Cheese"],
14      "image": "FrenchOnionSoup.png"
15    },
16    {
17      "name": "House Salad",
18      "price": 14.40,
19      "description": "",
20      "ingredients": ["Mixed Greens", "Cucumbers", "Carrots", "T",
21      "image": "HouseSalad.png"
22    }
23  ]
24 }
```

The 'Langage' menu is open, showing the following options:

- None (Normal Text)
- A
- B
- C
- D
- E
- F
- Gui4Cli
- H
- I
- J (selected)
- KIXtart
- L
- M
- N

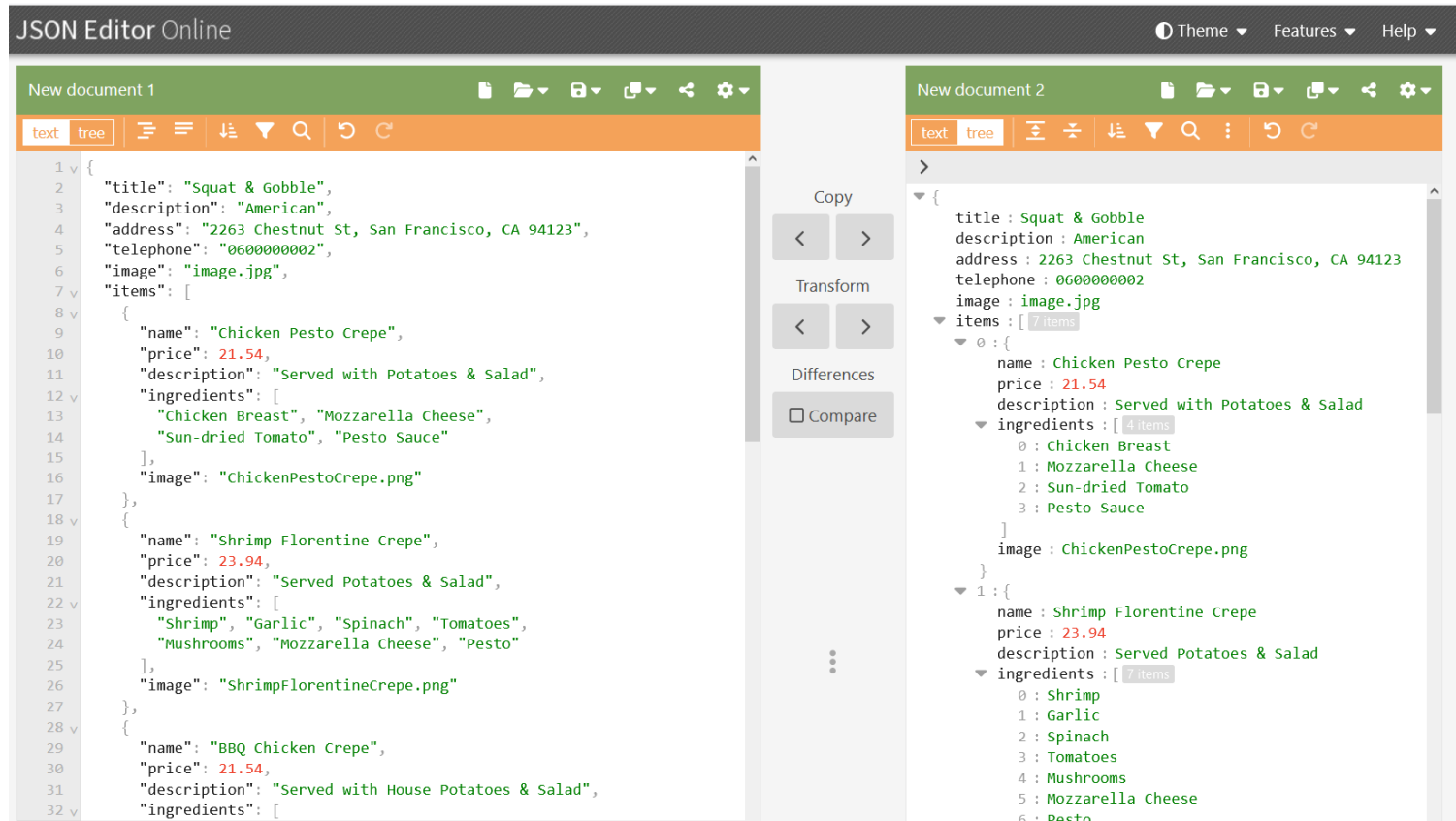
The 'J' option is expanded, showing the following sub-options:

- Java
- JavaScript
- JSON (selected)
- JSP

RESSOURCES UTILISÉES : gratuites et/ou libres d'accès

JSON editor online : <https://jsoneditoronline.org/>

(utilisation en ligne)



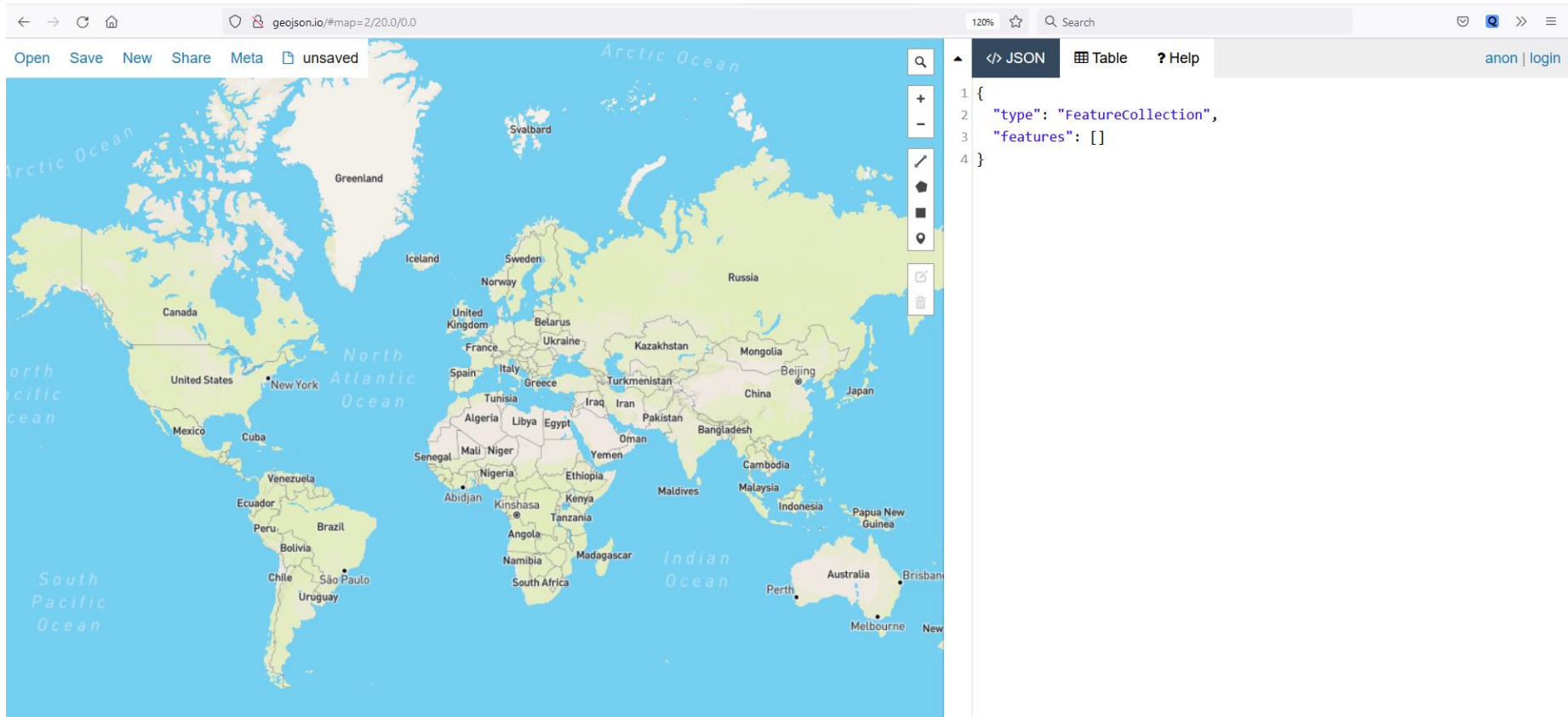
The screenshot displays the JSON Editor Online interface. It features two side-by-side panes: 'New document 1' on the left and 'New document 2' on the right. The left pane shows a JSON document in text view, with line numbers 1 through 32. The right pane shows the same JSON document in tree view, with expandable nodes for 'items' and 'ingredients'. A central toolbar contains buttons for 'Copy', 'Transform', 'Differences', and 'Compare'. The top navigation bar includes 'Theme', 'Features', and 'Help' dropdown menus.

```
1 {
2   "title": "Squat & Gobble",
3   "description": "American",
4   "address": "2263 Chestnut St, San Francisco, CA 94123",
5   "telephone": "0600000002",
6   "image": "image.jpg",
7   "items": [
8     {
9       "name": "Chicken Pesto Crepe",
10      "price": 21.54,
11      "description": "Served with Potatoes & Salad",
12      "ingredients": [
13        "Chicken Breast", "Mozzarella Cheese",
14        "Sun-dried Tomato", "Pesto Sauce"
15      ],
16      "image": "ChickenPestoCrepe.png"
17    },
18    {
19      "name": "Shrimp Florentine Crepe",
20      "price": 23.94,
21      "description": "Served Potatoes & Salad",
22      "ingredients": [
23        "Shrimp", "Garlic", "Spinach", "Tomatoes",
24        "Mushrooms", "Mozzarella Cheese", "Pesto"
25      ],
26      "image": "ShrimpFlorentineCrepe.png"
27    },
28    {
29      "name": "BBQ Chicken Crepe",
30      "price": 21.54,
31      "description": "Served with House Potatoes & Salad",
32      "ingredients": [
```

RESSOURCES UTILISÉES : gratuites et/ou libres d'accès

geoJSON.io : <http://geojson.io>

(utilisation en ligne)



The screenshot displays the geoJSON.io web application. The browser address bar shows the URL `geojson.io/#map=2/20.0/0.0`. The application interface includes a map on the left and a JSON editor on the right. The JSON editor shows the following code:

```
1 {  
2   "type": "FeatureCollection",  
3   "features": []  
4 }
```

SUPPORTS PÉDAGOGIQUES ET PRÉREQUIS

- Pour ce projet, il est préférable d'avoir une première expérience en programmation, plutôt avec App Inventor
- mais
 - toutes les notions essentielles sont expliquées,
 - le code source est fourni et réutilisable à chaque étape,
 - les jeux de données nécessaires sont fournis
<http://onvaessayer.org/appinventot?res=gitshare>
 - Un support de cours est proposé :
<http://onvaessayer.org/appinventor?app=gitshare>
- La version la plus récente de cette vidéo est à l'adresse :
<http://onvaessayer.org/appinventor?video=gitshare>







Créer un site web et un serveur
de données avec Github

PLAN DES CHAPITRES 1 ET 2

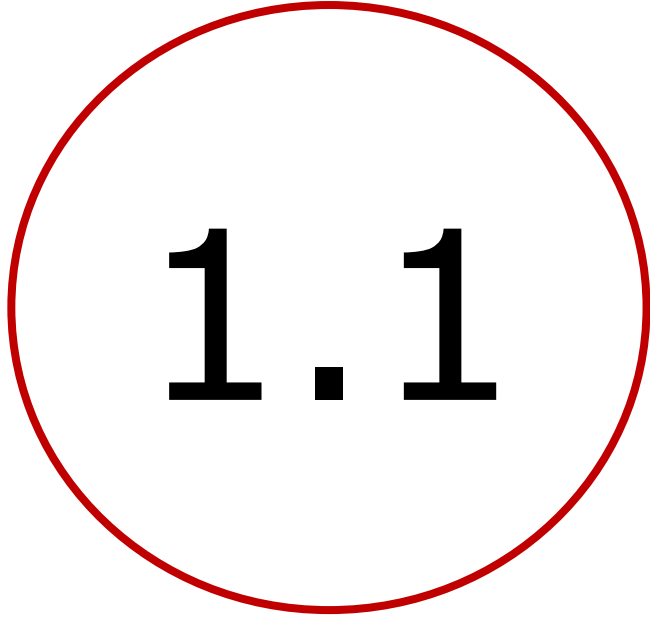
- Introduction : décomposition de l'application
1. Création d'un site Web / serveur de données
 - créer un compte github
 - créer un repository "username".github.io
 - créer et partager un fichier index.html
 2. Définition des données et préparation d'un jeu de test
 - Définition et organisation des données
 - formats JSON & geoJSON
 - modèle de restaurant au format JSON
 - Modèle de catalogue d'objets géolocalisés au format geoJSON
 - télécharger les données sur le serveur Github
 - choix des propriétés geoJSON pour app inventor
 3. Création de l'application mobile avec App Inventor

1. SERVEURS, ADRESSES OU URL

Adresse = URL = Uniform Resource Locator

-  github
 - <http://onvaessayer.github.io/gitshareData3/map.geojson>
- *free* pages perso de free
 - <http://pierre.huguet.free.fr/gitshare/restaurants/map.geojson>
-  serveur sur OVH
 - <http://onvaessayer.org/appinventor/apps/gitshare/resources/restaurants/map.geojson>
-  Dropbox
 - <https://www.dropbox.com/s/p8oizwazpme7xpg/?dl=0> (copy address to web browser)
-  Google Drive
 - <https://drive.google.com/file/d/1QbflQiM0goh-YAqE87X0bQmg1sq6lv7i/view?usp=sharing>





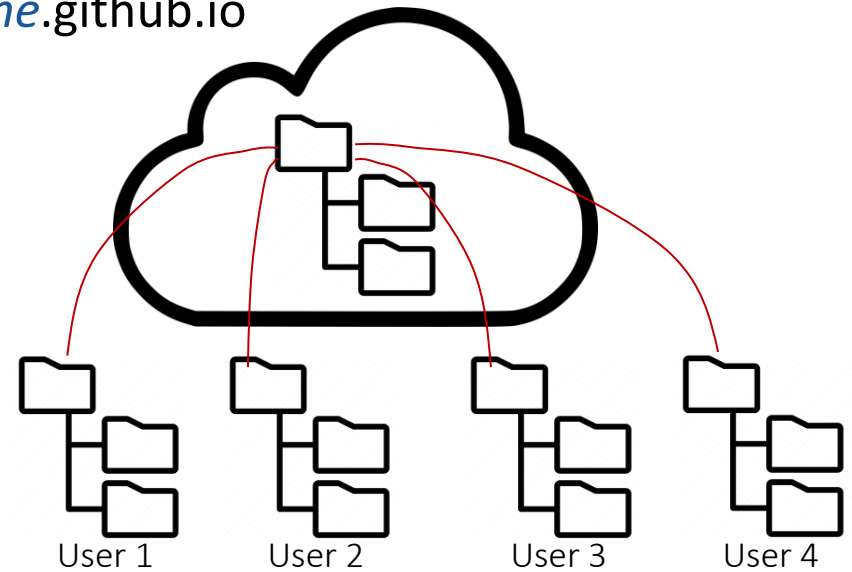
Créer un compte github

PLAN DES CHAPITRES 1 ET 2

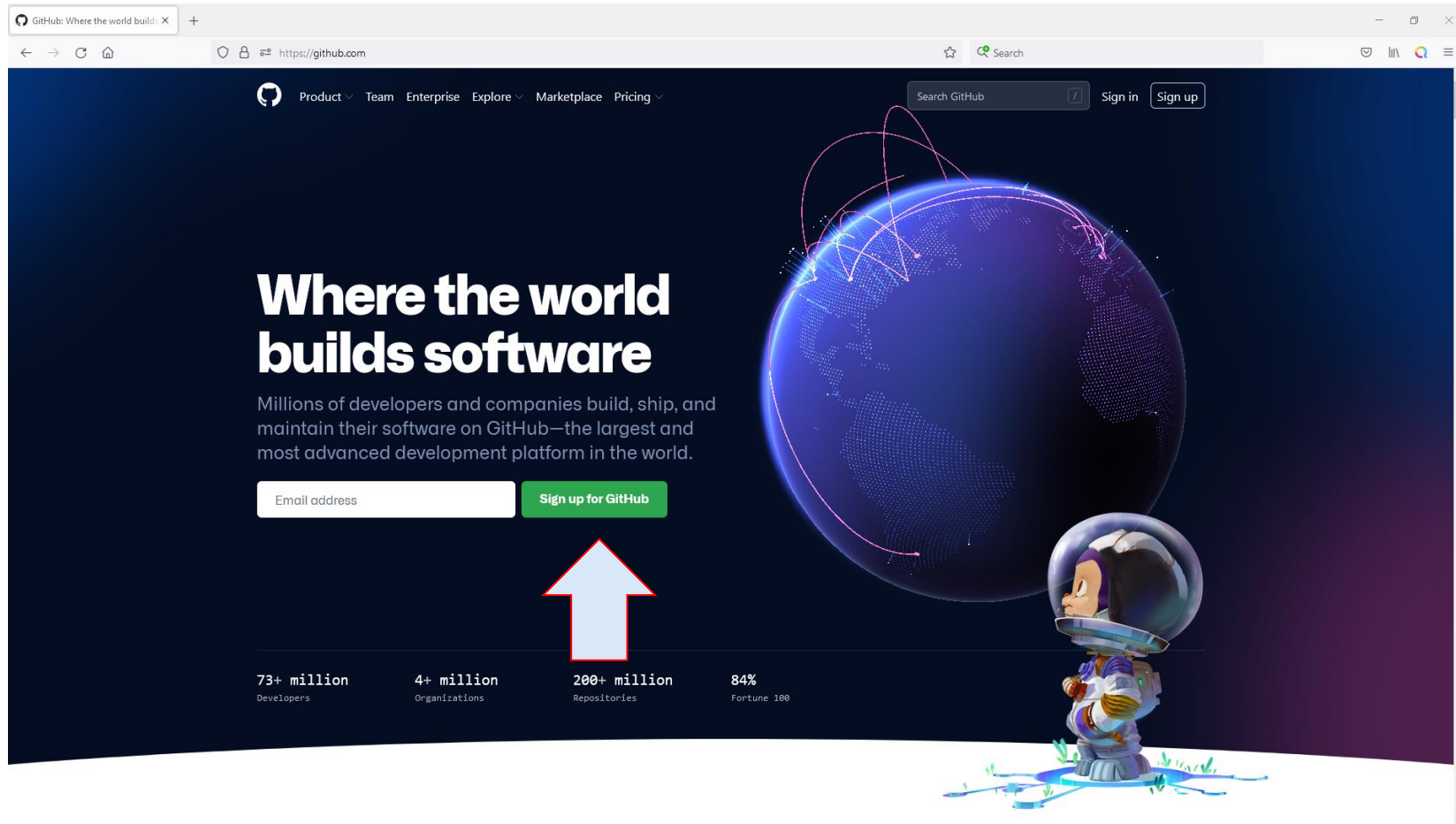
- Introduction : décomposition de l'application
- 1. Création d'un site Web / serveur de données
 - créer un compte github
 - créer un repository "username".github.io
 - créer et partager un fichier index.html
- 2. Définition des données et préparation d'un jeu de test
 - Définition et organisation des données
 - formats JSON & geoJSON
 - modèle de restaurant au format JSON
 - Modèle de catalogue d'objets géolocalisés au format geoJSON
 - télécharger les données sur le serveur Github
 - choix des propriétés geoJSON pour app inventor
- 3. Création de l'application mobile avec App Inventor

1.1 CRÉER UN SITE WEB : QU'EST-CE QUE GITHUB ?

- GitHub est un site web et un service du cloud qui aide les développeurs
 - à stocker et à gérer leur code,
 - à suivre et contrôler les modifications.
- Github pages
 - permet d'utiliser l'adresse `https://username.github.io` comme un site web.
username est le nom de l'utilisateur
(ici : `https://onvaessayer.github.io/`)
- Git, créé par Linus Torvalds, est un logiciel open-source
- GitHub est une société à but lucratif qui offre un service d'hébergement de référentiel Git basé sur le cloud.



1.1 Créer un compte Github : <https://github.com>



1.1 Créer un compte Github : CHOISIR UN 'username'

email

password

username

news

contrôles

création

Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ pierre@onvaessayer.org

Create a password
✓

Enter a username
✓ onvaessayerTest

Would you like to receive product updates and announcements via email?
Type "y" for yes or "n" for no
✓ n

Verify your account

✓

Create account

1.1 Créer un compte Github : ENTRER LE CODE REÇU PAR EMAIL

email

password

username

news

contrôles

création

Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ pierre@onvaessayer.org

Create a password
✓ ●●●●●●●●●●

Enter a username
✓ onvaessayerTest

Would you like to receive product updates and announcements via email?
Type "y" for yes or "n" for no
✓ n

Verify your account
✓

Create account

Validation email : code

You're almost done!
We sent a launch code to pierre@onvaessayer.org

→ Enter code

Didn't get your email? [Resend the code](#) or [update your email address](#).

1.1 Créer un compte Github : JUST ME

email

password

username

news

contrôles

création

Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ pierre@onvaessayer.org

Create a password
✓ ●●●●●●●●●●

Enter a username
✓ onvaessayerTest

Would you like to receive product updates and announcements via email?
Type "y" for yes or "n" for no
✓ n

Verify your account

✓

Create account



How many team members will be working with you?
This will help us guide you to the tools that are best suited for your projects.

Just me 2 - 5 5 - 10
10 - 20 20 - 50 50+

Are you a student or teacher?
Student Teacher

Continue

1.1 Créer un compte Github : PAS D'OPTION SPÉCIFIQUE

email

password

username

news

contrôles

création

Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ pierre@onvaessayer.org

Create a password
✓ ●●●●●●●●●●

Enter a username
✓ onvaessayerTest

Would you like to receive product updates and announcements via email?
Type "y" for yes or "n" for no
✓ n

Verify your account

✓

Create account

What specific features are you interested in using?

Select all that apply so we can point you to the right GitHub plan.

- Collaborative coding
Codespaces, Pull requests, Notifications, Code review, Code review assignments, Code owners, Draft pull requests, Protected branches, and more.
- Automation and CI/CD
Actions, Packages, APIs, GitHub Pages, GitHub Marketplace, Webhooks, Hosted runners, Self-hosted runners, Secrets management, and more.
- Security
Private repos, 2FA, Required reviews, Required status checks, Code scanning, Secret scanning, Dependency graph, Dependabot alerts, and more.
- Client Apps
GitHub Mobile, GitHub CLI, and GitHub Desktop.
- Project Management
Projects, Labels, Milestones, Issues, Unified Contribution Graph, Org activity graph, Org dependency insights, Repo insights, Wikis, and GitHub Insights.
- Team Administration
Organizations, Invitations, Team sync, Custom roles, Domain verification, Audit Log API, Repo creation restriction, and Notification restriction.
- Community
GitHub Marketplace, GitHub Sponsors, GitHub Learning Lab, Electron, and Atom.

Continue



1.1 Créer un compte Github : CONTINUE FOR FREE

email

password

username

news

contrôles

création

Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ pierre@onvaessayer.org

Create a password
✓ ●●●●●●●●●●

Enter a username
✓ onvaessayerTest

Would you like to receive product updates and announcements via email?
Type "y" for yes or "n" for no
✓ n

Verify your account

✓

Create account



Free

- Unlimited public/private repositories
- 2,000 CI/CD minutes/month
Free for public repositories
- 500MB of Packages storage
Free for public repositories
- Community support

Continue for free

1.1 Créer un compte Github : OK

The screenshot shows the GitHub homepage. At the top, there is a dark navigation bar with the GitHub logo, a search bar, and links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. On the left side, there is a sidebar with the heading 'Create your first project' and a sub-heading 'Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.' Below this are two buttons: 'Create repository' (green) and 'Import repository' (blue). Underneath is a section for 'Recent activity' with a sub-heading 'When you take actions across GitHub, we'll provide links to that activity here.' The main content area features a large light green banner with the text 'Learn Git and GitHub without any code!' and a sub-heading 'Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.' There are two buttons: 'Read the guide' (green) and 'Start a project' (white). Below the banner is a navigation bar with 'Following' and 'For you (Beta)'. The 'For you' section is active and contains a card titled 'Introduce yourself' with the text 'The easiest way to introduce yourself on GitHub is by creating a README in a repository about you! You can start here:'. Below this is a code editor showing a README template for 'onvaessayerTest / README.md' with five numbered lines of text: '1 - 👋 Hi, I'm @onvaessayerTest', '2 - 🐙 I'm interested in ...', '3 - 📖 I'm currently learning ...', '4 - 🍷 I'm looking to collaborate on ...', and '5 - 📧 How to reach me ...'. There are 'Dismiss this' and 'Continue' buttons at the bottom right of the card. Below the card is another section titled 'Discover interesting projects and people to populate your personal news feed.' with the text 'Your news feed helps you keep up with recent activity on repositories you watch or star and people you follow.' and an 'Explore GitHub' button.



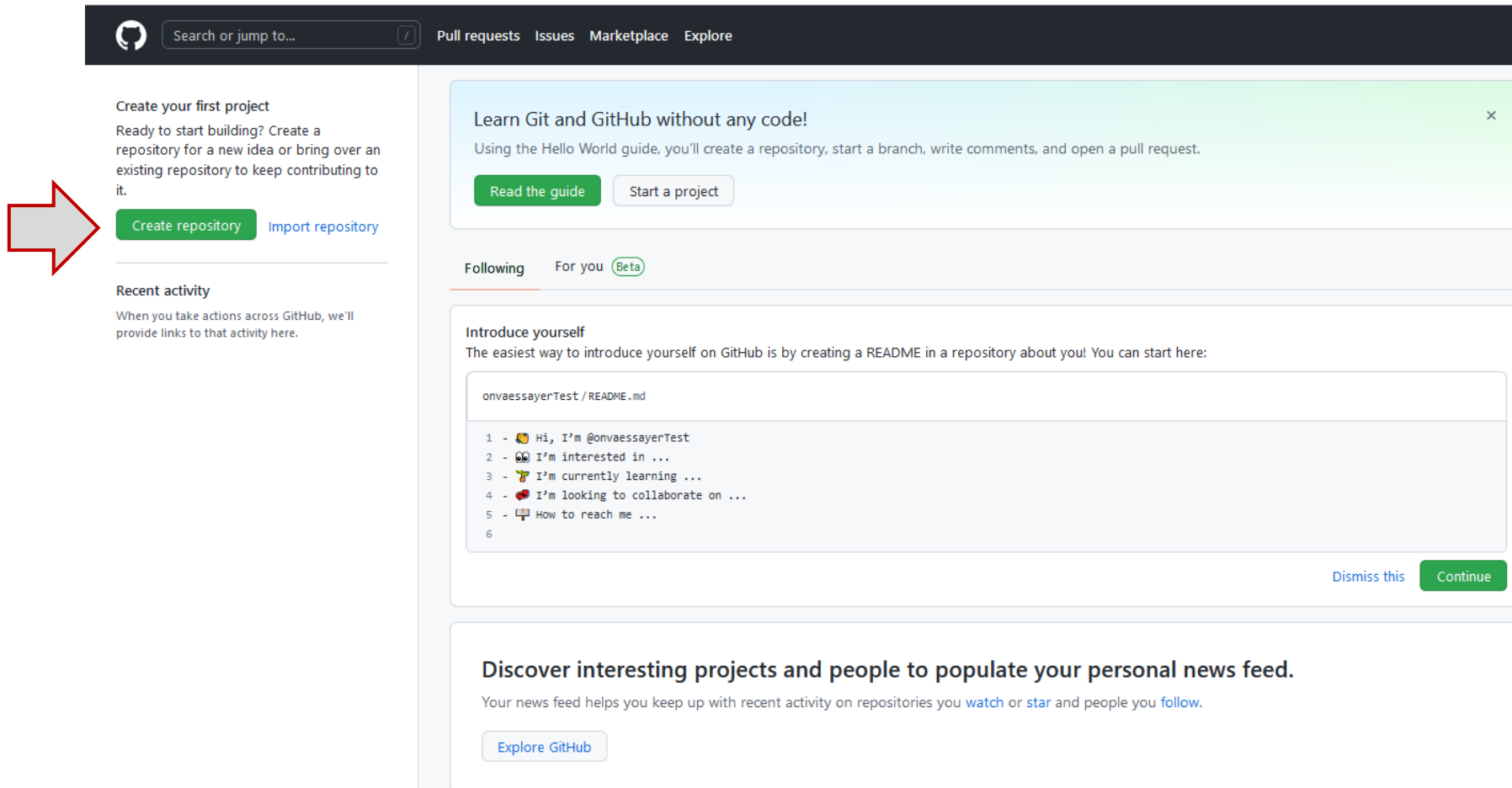
1.2

Créer un repository partagé
<https://username.github.io>

PLAN DES CHAPITRES 1 ET 2

- Introduction : décomposition de l'application
- 1. Création d'un site Web / serveur de données
 - créer un compte github
 - créer un repository "username".github.io
 - créer et partager un fichier index.html
- 2. Définition des données et préparation d'un jeu de test
 - Définition et organisation des données
 - formats JSON & geoJSON
 - modèle de restaurant au format JSON
 - Modèle de catalogue d'objets géolocalisés au format geoJSON
 - télécharger les données sur le serveur Github
 - choix des propriétés geoJSON pour app inventor
- 3. Création de l'application mobile avec App Inventor

1.2 Création d'un repository "*username*".github.io



The screenshot shows the GitHub homepage. At the top, there is a dark navigation bar with the GitHub logo, a search bar, and links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the navigation bar, the main content area is divided into several sections. On the left, there is a sidebar with the heading 'Create your first project' and a sub-heading 'Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.' Below this, there are two buttons: 'Create repository' (highlighted with a red arrow) and 'Import repository'. Further down, there is a section for 'Recent activity'. The main content area features a light blue banner for 'Learn Git and GitHub without any code!' with a 'Read the guide' button and a 'Start a project' button. Below this, there is a 'Following' section with a 'Beta' badge. The 'Introduce yourself' section provides instructions on creating a README file and shows a preview of a README template with a list of items to include. At the bottom, there is a section for 'Discover interesting projects and people to populate your personal news feed.' with an 'Explore GitHub' button.

Search or jump to... Pull requests Issues Marketplace Explore

Create your first project
Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.

Create repository Import repository

Recent activity
When you take actions across GitHub, we'll provide links to that activity here.

Learn Git and GitHub without any code!
Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

Read the guide Start a project

Following For you Beta

Introduce yourself
The easiest way to introduce yourself on GitHub is by creating a README in a repository about you! You can start here:

```
onvaessayerTest / README.md
```

```
1 - 👋 Hi, I'm @onvaessayerTest
2 - 🤖 I'm interested in ...
3 - 📖 I'm currently learning ...
4 - 🍷 I'm looking to collaborate on ...
5 - 📧 How to reach me ...
6
```

Dismiss this Continue

Discover interesting projects and people to populate your personal news feed.
Your news feed helps you keep up with recent activity on repositories you watch or star and people you follow.

Explore GitHub

1.2 Création d'un repository "*username*".github

Create repository

username.github.io

→ (commentaire)
→ public

→ add readme



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).



Owner * / Repository name *

Great repository names are short and memorable. Need inspiration? How about [crispy-goggles](#)?

Description (optional)

- Public**
Anyone on the internet can see this repository. You choose who can commit.
- Private**
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

- Add a README file**
This is where you can write a long description for your project. [Learn more](#).

Add .gitignore

Choose which files not to track from a list of templates. [Learn more](#).

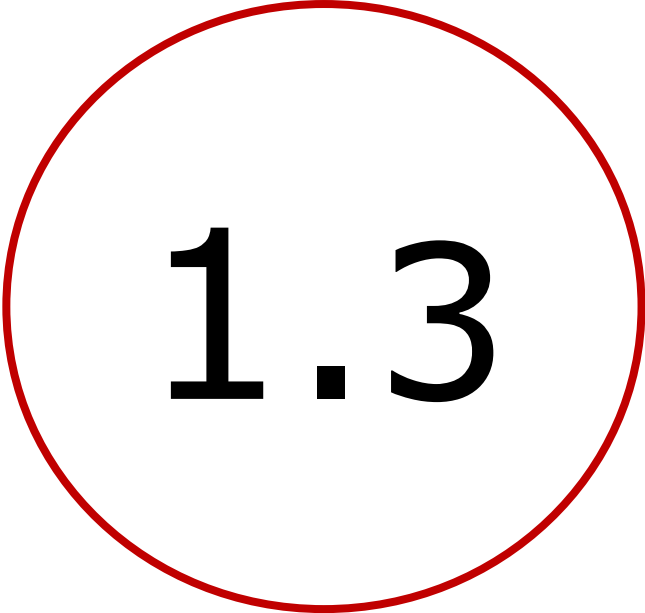
Choose a license

A license tells others what they can and can't do with your code. [Learn more](#).

This will set `main` as the default branch. Change the default name in your [settings](#).

ⓘ You are creating a public repository in your personal account.

Create repository



1.3

Créer un fichier
dans le répertoire partagé

PLAN DES CHAPITRES 1 ET 2

- Introduction : décomposition de l'application
- 1. Création d'un site Web / serveur de données
 - créer un compte github
 - créer un repository "username".github.io
 - créer et partager un fichier index.html
- 2. Définition des données et préparation d'un jeu de test
 - Définition et organisation des données
 - formats JSON & geoJSON
 - modèle de restaurant au format JSON
 - Modèle de catalogue d'objets géolocalisés au format geoJSON
 - télécharger les données sur le serveur Github
 - choix des propriétés geoJSON pour app inventor
- 3. Création de l'application mobile avec App Inventor

1.3 Créer un fichier index.html (si add readme sélectionné)

The screenshot shows the GitHub interface for a repository named 'onvaessayerTest'. At the top, there are navigation links for Projects, Wiki, Security, Insights, and Settings. Below these, there are buttons for 'main', '1 branch', and '0 tags'. On the right, there are buttons for 'Go to file', 'Add file', and 'Code'. The repository name 'onvaessayerTest' is followed by 'Initial commit', the commit hash '3830974', the time 'now', and '1 commit'. A table lists the files: 'README.md' with 'Initial commit' and 'now'. The content of the README.md file is displayed in a large box, showing the title 'onvaessayerTest.github.io' and the subtitle 'test static server on github'.

Projects Wiki Security Insights Settings

main 1 branch 0 tags

Go to file Add file Code

onvaessayerTest Initial commit 3830974 now 1 commit

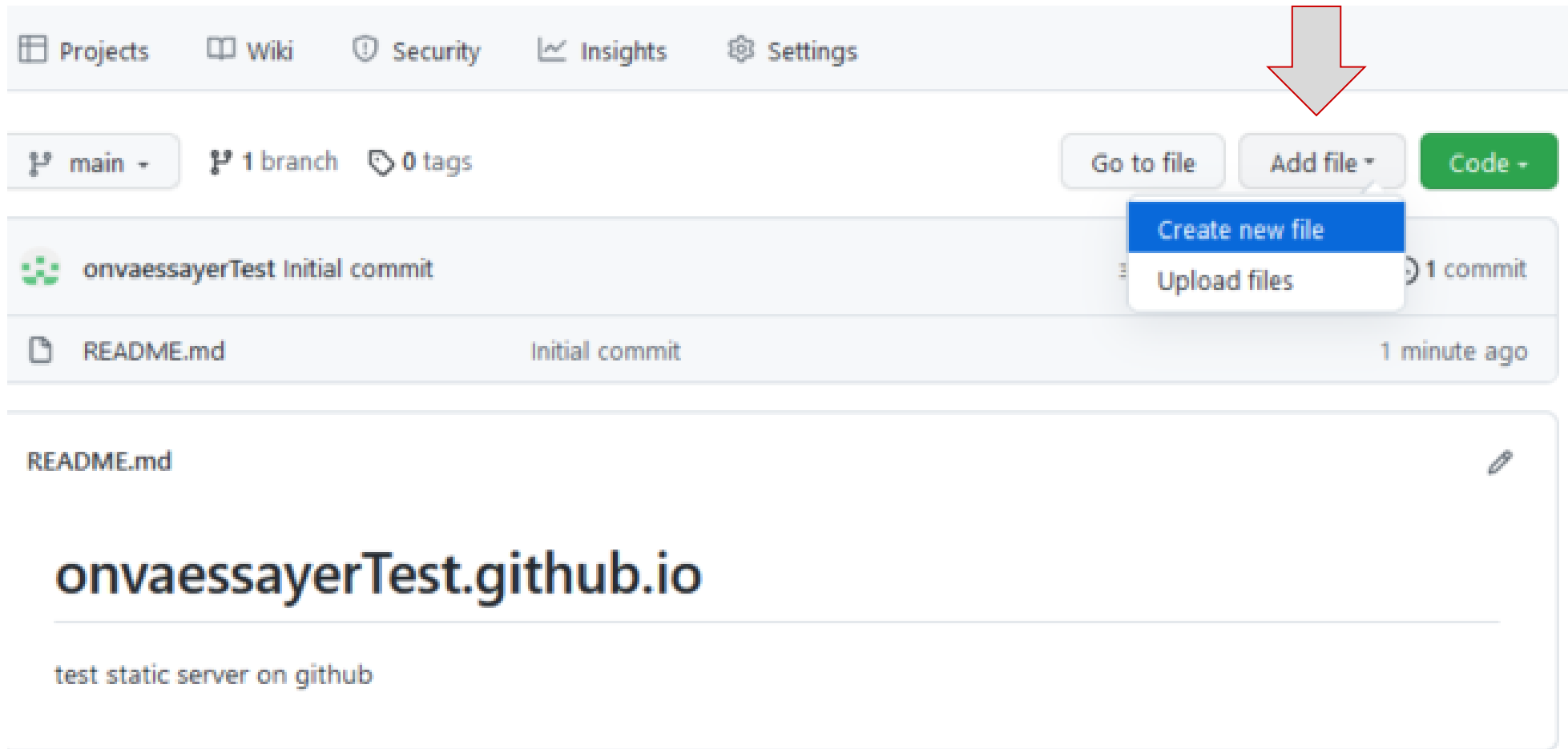
README.md	Initial commit	now
-----------	----------------	-----

README.md

onvaessayerTest.github.io

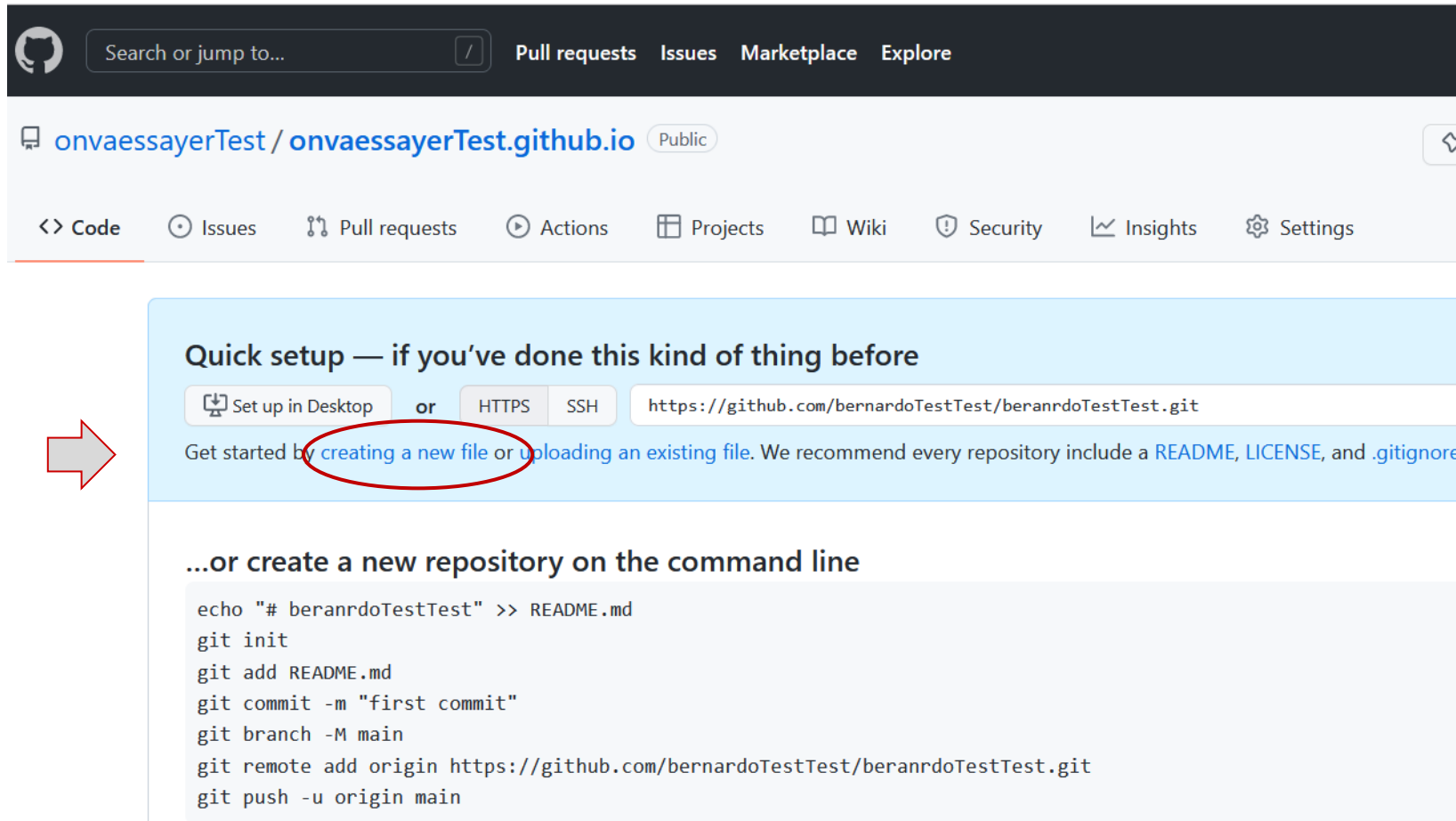
test static server on github

1.3 Créer un fichier index.html (si add readme sélectionné)



The screenshot shows the GitHub interface for a repository named 'onvaessayerTest'. At the top, there are navigation links for Projects, Wiki, Security, Insights, and Settings. Below these, there are buttons for 'main', '1 branch', and '0 tags'. On the right side, there are buttons for 'Go to file', 'Add file', and 'Code'. The 'Add file' dropdown menu is open, showing two options: 'Create new file' (highlighted in blue) and 'Upload files'. Below the dropdown, there is a commit history section showing a commit for 'README.md' with the message 'Initial commit' and a timestamp of '1 minute ago'. The main content area shows the 'README.md' file content, which includes the title 'onvaessayerTest.github.io' and the text 'test static server on github'.

1.3 Créer un fichier index.html (si add readme NON sélectionné)



Search or jump to... Pull requests Issues Marketplace Explore

onvaessayerTest / onvaessayerTest.github.io Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Quick setup — if you've done this kind of thing before

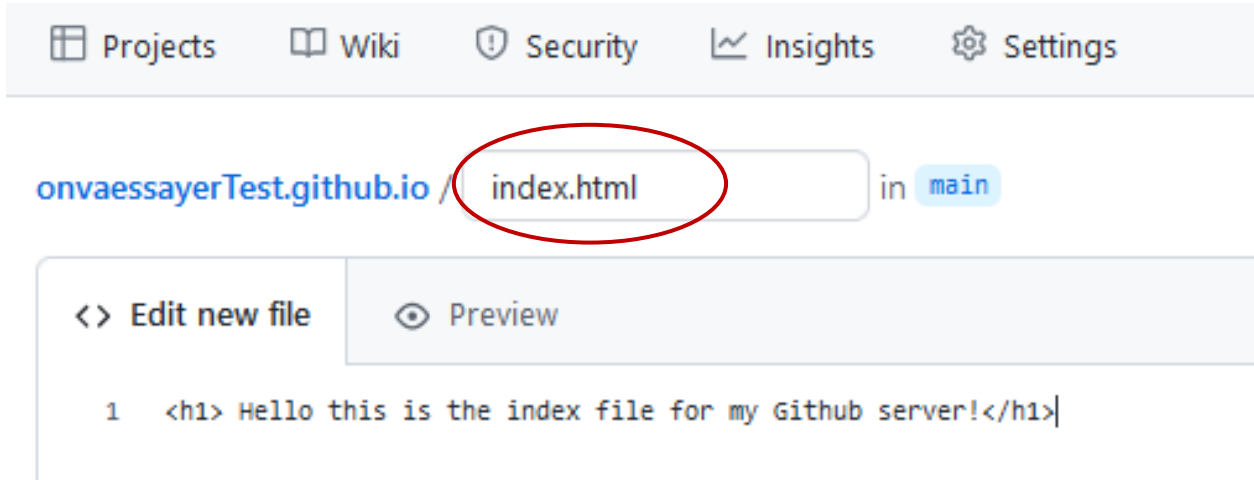
Set up in Desktop or HTTPS SSH <https://github.com/bernardoTestTest/beranrdoTestTest.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

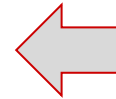
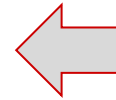
...or create a new repository on the command line

```
echo "# beranrdoTestTest" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/bernardoTestTest/beranrdoTestTest.git
git push -u origin main
```

1.3 Créer un fichier index.html



The screenshot shows the GitHub web interface for editing a new file. At the top, there are navigation links: Projects, Wiki, Security, Insights, and Settings. Below this, the repository path is shown as `onvaessayerTest.github.io / index.html` in the `main` branch. The `index.html` text is circled in red. Below the path, there are two tabs: `<> Edit new file` (which is active) and `Preview`. The code editor shows a single line of HTML: `1 <h1> Hello this is the index file for my Github server!</h1>`. A red arrow points from the `index.html` text to the right, and another red arrow points from the code editor to the right.



Texte html

1.3 Créer un fichier index.html

The screenshot shows the GitHub interface for creating a new file. At the top, there are navigation links: Projects, Wiki, Security, Insights, and Settings. Below this, the repository name 'onvaessayerTest.github.io' is shown, followed by a text input field containing 'index.html' (circled in red) and a dropdown menu set to 'main'. Below the repository path, there are two tabs: 'Edit new file' (active) and 'Preview'. The code editor shows a single line of HTML: `<h1> Hello this is the index file for my Github server!</h1>`. Below the code editor is the 'Commit new file' dialog, which includes a text input field with 'Create index.html', a larger text area for an optional description, and two buttons: 'Commit new file' (highlighted in green) and 'Cancel'. Three red arrows point from the right side of the image to the file name, the code editor, and the 'Commit new file' button.

onvaessayerTest.github.io / index.html in main

<> Edit new file Preview

```
1 <h1> Hello this is the index file for my Github server!</h1>
```

Commit new file

Create index.html

Add an optional extended description...

Commit new file Cancel

Texte html

Commit / valider

1.3 Créer un fichier index.html

Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

onvaessayerTest Create index.html 2e8d629 now 2 commits

README.md	Initial commit	7 minutes ago
index.html	Create index.html	now

README.md

onvaessayerTest.github.io

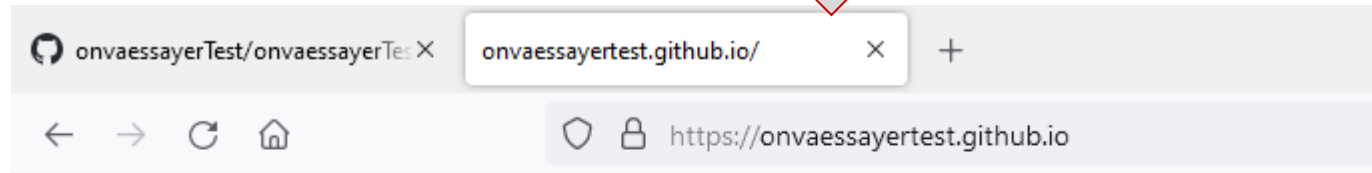
test static server on github

PLAN DES CHAPITRES 1 ET 2

- Introduction : décomposition de l'application
1. Création d'un site Web / serveur de données
 - créer un compte github
 - créer un repository "username".github.io
 - créer et partager un fichier index.html
 2. Définition des données et préparation d'un jeu de test
 - Définition et organisation des données
 - formats JSON & geoJSON
 - modèle de restaurant au format JSON
 - Modèle de catalogue d'objets géolocalisés au format geoJSON
 - télécharger les données sur le serveur Github
 - choix des propriétés geoJSON pour app inventor
 3. Création de l'application mobile avec App Inventor

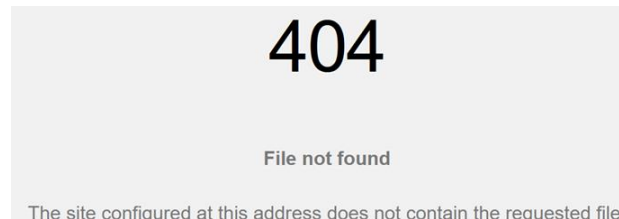
1.3 AFFICHER index.html DANS UN NAVIGATEUR

- ouvrir un navigateur et aller à l'adresse à l'adresse username.github.io (dans cet exemple onvaessayerTest.github.io)



Hello this is the index file for my Github server!

- Attention**, il faut parfois plusieurs minutes entre la création ou l'importation d'un fichier et sa disponibilité sur Internet






Définir et préparer les données
catalogue géolocalisé et restaurants

PLAN DES CHAPITRES 1 ET 2

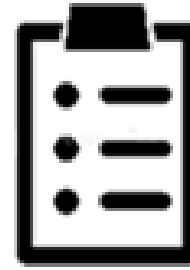
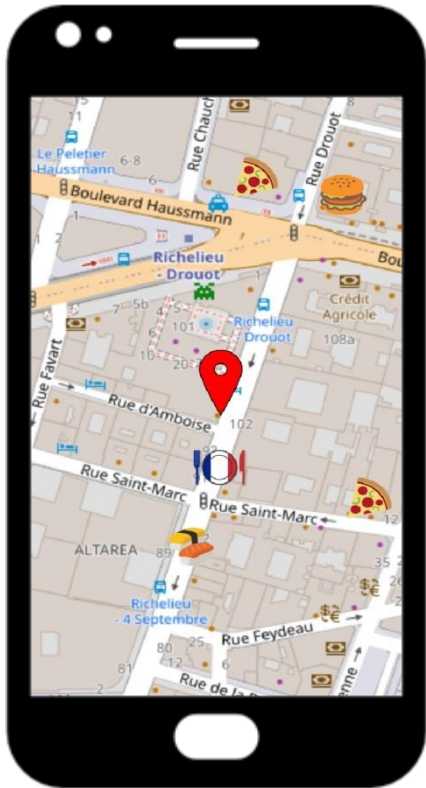
- Introduction : décomposition de l'application
- 1. Création d'un site Web / serveur de données
 - créer un compte github
 - créer un repository "username".github.io
 - créer et partager un fichier index.html
- 2. Définition des données et préparation d'un jeu de test
 - Définition et organisation des données
 - formats JSON & geoJSON
 - modèle de restaurant au format JSON
 - Modèle de catalogue d'objets géolocalisés au format geoJSON
 - télécharger les données sur le serveur Github
 - choix des propriétés geoJSON pour app inventor
- 3. Création de l'application mobile avec App Inventor



2.1

Définition et organisation
des données

2.1 DÉFINITION DES DONNÉES : CATALOGUES ET RESTAURANTS (SHOPS)



Catalogue des restaurants

- title (name)
- description (lien)
- geo-location
 - longitude, latitude



Données d'un restaurant

- title (name)
- description (category)
- address
- image
- liste des plats
 - name, price, ...
 - name, price, ...

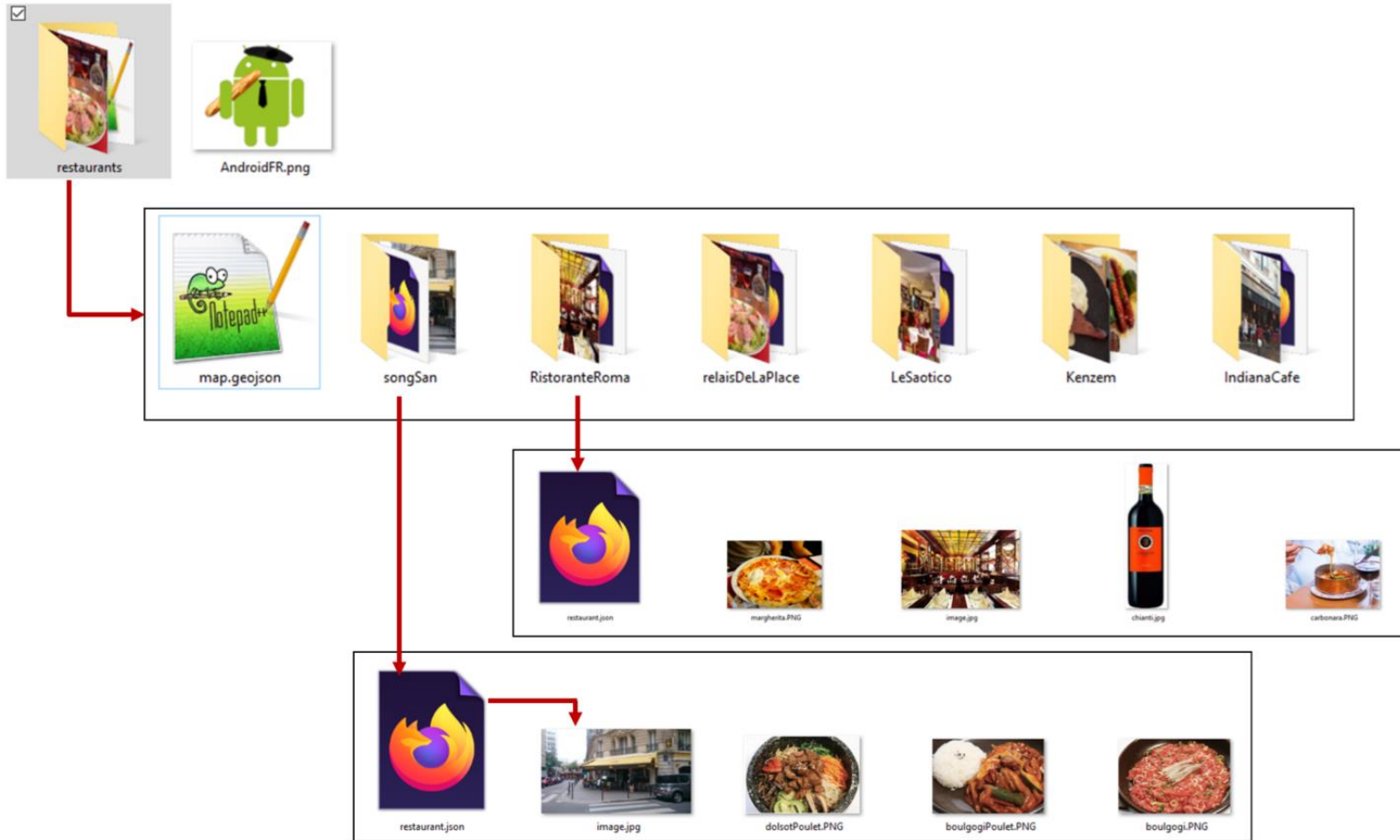
2.1 TYPES DE DONNÉES : IMAGES, FICHIERS JSON ET GEOJSON




2.1 TYPES DE DONNÉES : IMAGES, FICHIERS JSON ET GEOJSON (géolocalisées)



2.1 ORGANISATION : catalogue et un répertoire par restaurant






2.2

JSON & geoJSON

PLAN DES CHAPITRES 1 ET 2

- Introduction : décomposition de l'application
- 1. Création d'un site Web / serveur de données
 - créer un compte github
 - créer un repository "username".github.io
 - créer et partager un fichier index.html
- 2. Définition des données et préparation d'un jeu de test
 - Définition et organisation des données
 - formats JSON & geoJSON
 - modèle de restaurant au format JSON
 - Modèle de catalogue d'objets géolocalisés au format geoJSON
 - télécharger les données sur le serveur Github
 - choix des propriétés geoJSON pour app inventor
- 3. Création de l'application mobile avec App Inventor



2.3

modèle de restaurant
au format JSON

PLAN DES CHAPITRES 1 ET 2

- Introduction : décomposition de l'application
1. Création d'un site Web / serveur de données
 - créer un compte github
 - créer un repository "username".github.io
 - créer et partager un fichier index.html
 2. Définition des données et préparation d'un jeu de test
 - Définition et organisation des données
 - formats JSON & geoJSON
 - modèle de restaurant au format JSON
 - Modèle de catalogue d'objets géolocalisés au format geoJSON
 - télécharger les données sur le serveur Github
 - choix des propriétés geoJSON pour app inventor
 3. Création de l'application mobile avec App Inventor

JSON EDITOR ONLINE : <https://jsoneditoronline.org>

The image displays the JSON Editor Online interface with two document panes. The left pane, titled 'New document 1', shows the raw JSON code for a restaurant menu. The right pane, titled 'New document 2', shows the same data in a tree view. A central toolbar contains 'Copy', 'Transform', and 'Differences' buttons.

```
1 {
2   {
3     "title": "SongSan",
4     "description": "Restaurant Coréen",
5     "address": "rue Marmontel",
6     "telephone": "0600000005",
7     "image": "image.jpg",
8     "items": [
9       {
10        "name": "Dolsot poulet",
11        "price": 12,
12        "description": "",
13        "ingredients": ["poulet", "carottes", "soja"],
14        "image": "dolsotPoulet.PNG"
15      },
16      {
17        "name": "Bulgogi",
18        "price": 18,
19        "description": "émincé de boeuf sauce soja",
20        "ingredients": ["boeuf", "soja"],
21        "image": "bulgogi.PNG"
22      },
23      {
24        "name": "Bulgogi poulet",
25        "price": 18,
26        "description": "blanc de poulet, sauce soja",
27        "ingredients": ["poulet", "soja"],
28        "image": "bulgogiPoulet.PNG"
29      }
30    ]
31  }
32 }
33
34
35
```

The tree view on the right shows the following structure:

- Root object with 5 properties: title, description, address, telephone, image.
- items array with 3 items:
 - Item 0: Dolsot poulet (price: 12, description: "", ingredients: poulet, carottes, soja, image: dolsotPoulet.PNG)
 - Item 1: Bulgogi (price: 18, description: émincé de boeuf sauce soja, ingredients: boeuf, soja, image: bulgogi.PNG)
 - Item 2: Bulgogi poulet (price: 18, description: blanc de poulet, sauce soja, ingredients: poulet, soja, image: bulgogiPoulet.PNG)

JSON EDITOR ONLINE : <https://jsoneditoronline.org>

The image displays the JSON Editor Online interface, split into two panes. The left pane, titled 'New document 1', shows a JSON document with a syntax error highlighted in red at line 18, column 4. The error message at the bottom reads: 'expected \',\' or \'}\' after property value in object at line 18 column 4'. The right pane, titled 'New document 2', shows the tree view of the JSON document, with the 'ingredients' array expanded to show three items: 'Dolsot poulet', 'Boulgogi', and 'Boulgogi poulet'. A context menu is open over the 'Boulgogi' item, showing options like 'Edit key', 'Edit value', 'Cut', 'Copy', 'Paste', 'Remove', 'Duplicate', 'Extract', 'Sort', 'Transform', 'Insert before', and 'Insert after'. The interface also includes a toolbar with icons for file operations and a 'powered by CodeMirror' watermark.

```
1 {
2   {
3     "title": "SongSan",
4     "description": "Restaurant Coréen",
5     "address": "rue Marmontel",
6     "telephone": "0600000005",
7     "image": "image.jpg",
8     "items": [
9       {
10        "name": "Dolsot poulet",
11        "price": 12,
12        "description": "",
13        "ingredients": ["poulet", "carottes", "soja"],
14        "image": "dolsotPoulet.PNG"
15      },
16      {
17        "name": "Boulgogi"
18        "price": 18,
19        "description": "émincé de boeuf sauce soja",
20        "ingredients": ["boeuf", "soja"],
21        "image": "boulgogi.PNG"
22      },
23      {
24        "name": "Boulgogi poulet",
25        "price": 18,
26        "description": "blanc de poulet, sauce soja",
27        "ingredients": ["poulet", "soja"],
28        "image": "boulgogiPoulet.PNG"
29      }
30    ]
31  }
32 }
```

Copy
Transform
Differences
Enable

items > 0 > ingredients > 2

```
{
  title : SongSan
  description : Restaurant Coréen
  address : rue Marmontel
  telephone : 0600000005
  image : image.jpg
  items : [ 3 items
    0 : {
      name : Dolsot poulet
      price : 12
      description : value
      ingredients : [ 3 items
        0 : poulet
        1 : carottes
        2 : soja
      ]
      image : dolsotPoulet.PNG
    }
    1 : {
      name : Boulgogi
      price : 18
      description : émincé de boeuf s
      ingredients : [ 2 items ]
      image : boulgogi.PNG
    }
    2 : { 5 props }
  ]
}
```

Edit key Edit value
Cut Copy Paste
Remove Insert:
Duplicate + Structure
Extract {} Object
Sort [] Array
Transform " Value
Insert before Insert after

⚠ expected \',\' or \'}\' after property value in object at line 18 column 4 Auto repair

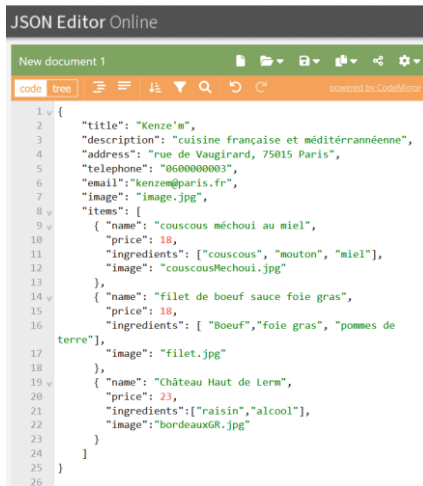
MODÈLE DE RESTAURANT AU FORMAT JSON

JSON

outil d'édition recommandé :

JSON editor online

<https://jsoneditoronline.org/>



```
{
  "title": "SongSan",
  "description": "Restaurant Coréen",
  "address": "rue Marmontel",
  "telephone": "0600000005",
  "image": "image.jpg",
  "items": [
    {
      "name": "Dolsot poulet",
      "price": 12,
      "description": "",
      "ingredients": ["poulet", "carottes", "soja"],
      "image": "dolsotPoulet.PNG"
    },
    {
      "name": "Boulgogi",
      "price": 18,
      "description": "émincé de boeuf sauce soja",
      "ingredients": ["boeuf", "soja"],
      "image": "boulgogi.PNG"
    }
  ]
}
```



2.4

geoJSON : données géolocalisées
création d'un catalogue avec geojson.io

PLAN DES CHAPITRES 1 ET 2

- Introduction : décomposition de l'application
- 1. Création d'un site Web / serveur de données
 - créer un compte github
 - créer un repository "username".github.io
 - créer et partager un fichier index.html
- 2. Définition des données et préparation d'un jeu de test
 - Définition et organisation des données
 - formats JSON & geoJSON
 - modèle de restaurant au format JSON
 - **Modèle de catalogue d'objets géolocalisés au format geoJSON**
 - télécharger les données sur le serveur Github
 - choix des propriétés geoJSON pour app inventor
- 3. Création de l'application mobile avec App Inventor

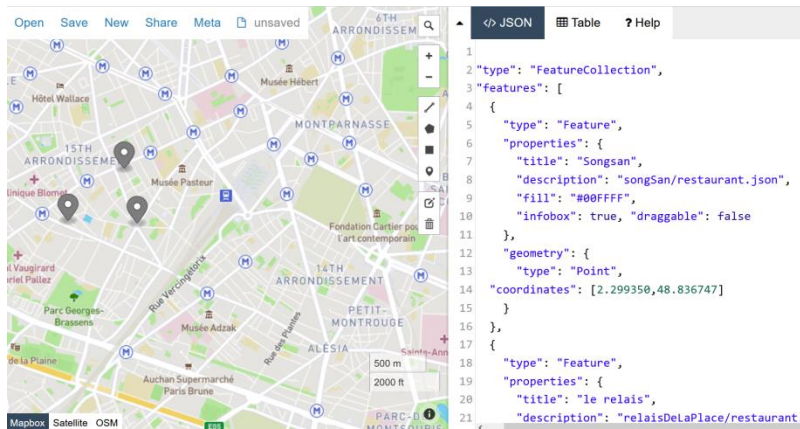
QUELQUES MOTS SUR JSON ET GEOJSON

geoJSON

outil d'édition recommandé :

[geoJSON.io](http://geojson.io)

<http://geojson.io>



```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "title": "le relais",
        "description": "https://onvaessayer.github.io/restaurants/relaisDeLaPlace/restaurant.json",
        "fill": "#0000FF", "infobox": true
      },
      "geometry": {
        "type": "Point", "coordinates": [2.3066380620002747, 48.836606046462514]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "title": "Kenze'm",
        "description": "https://onvaessayer.github.io/restaurants/Kenzem/restaurant.json",
        "fill": "#00FF00", "infobox": true
      },
      "geometry": {
        "type": "Point", "coordinates": [2.3052459955215454, 48.84039809611953]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "title": "Songsan",
        "description": "https://onvaessayer.github.io/restaurants/songSan/restaurant.json",
        "fill": "#00FFFF", "infobox": true
      },
      "geometry": {
        "type": "Point", "coordinates": [2.2993505001068115, 48.83674728250898]
      }
    }
  ]
}
```


GEOJSON ATTRIBUTES FOR APPINVENTOR

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "title": "Songsan",
        "description": "https://onvaessayer.github.io/gitshareGitDataset1/songSan/restaurant.json",
        "infobox": true,
        "fill": "#00FF00",
        "image": "https://onvaessayer.github.io/gitshareGitDataset1/icons/korean.png"
      },
      "geometry": {
        "type": "Point", "coordinates": [2.29935, 48.836747]
      }
    },
    { ... }
  ]
}
```

adresse (URL) du fichier JSON de ce restaurant

adresse (URL) de l'icône à afficher

2.5

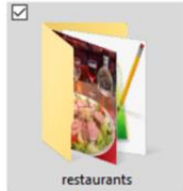
télécharger sur github

un catalogue geoJSON
et les données json de restaurants

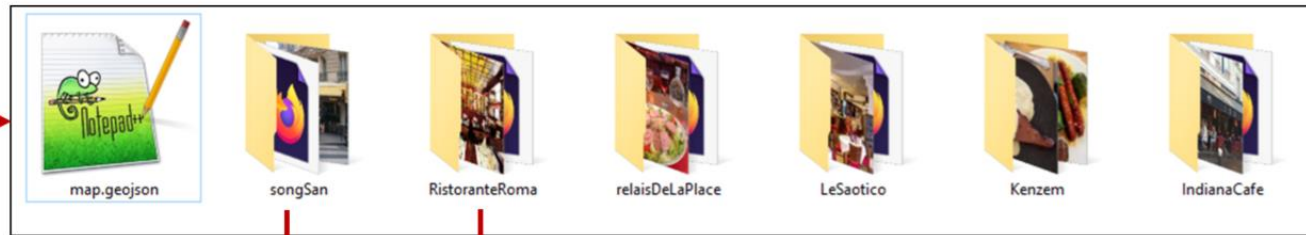
PLAN

- Introduction : décomposition de l'application
1. Création d'un site Web / serveur de données
 - créer un compte github
 - créer un repository "username".github.io
 - créer un fichier index.html
 - consulter le site (index.html) avec un navigateur
 2. Définition des données et préparation d'un jeu
 - Créer et décrire un restaurant (JSON)
 - le format JSON
 - créer un catalogue de restaurants géolocalisés (geoJSON)
 - télécharger les données sur le site Web (Github)
 - choix des propriétés du catalogue pour app inventor
 3. Création de l'application mobile avec App Inventor

2.5 CATALOGUE ET RÉPERTOIRES DE RESTAURANTS



répertoire des restaurants : <https://onvaessayer.Github.io/gitshareData1/map.geojson>



catalogue GeoJSON
et répertoires des
restaurants

Descriptif JSON
d'un restaurant,
image du restaurant
et images des plats

Descriptif JSON
d'un restaurant,
image du restaurant
et images des plats

2.5 UPLOAD/TÉLÉCHARGEMENT DE DONNÉES (RÉPERTOIRES ET FICHIERS)

The screenshot shows the GitHub interface for a repository named 'onvaessayerTest'. At the top, there are navigation links for Projects, Wiki, Security, Insights, and Settings. Below these, there are buttons for 'main', '1 branch', and '0 tags'. On the right side, there are buttons for 'Go to file', 'Add file', and 'Code'. A dropdown menu is open under 'Add file', showing options for 'Create new file' and 'Upload files'. The file list shows three files: 'README.md' (Initial commit, 10 minutes ago), 'index.html' (Create index.html, 3 minutes ago), and 'onvaessayerTest Create index.html' (2 commits, 2e8). Below the file list, the content of the 'README.md' file is displayed, showing the text 'onvaessayerTest.github.io' and 'test static server on github'.

Projects Wiki Security Insights Settings

main 1 branch 0 tags

Go to file Add file Code

Create new file
Upload files

onvaessayerTest Create index.html 2e8 2 commits

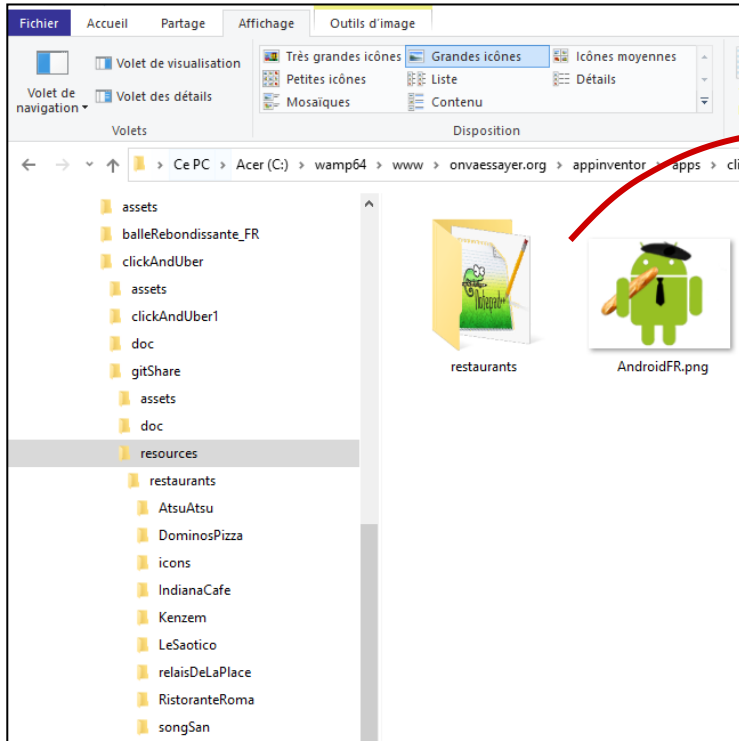
README.md	Initial commit	10 minutes ago
index.html	Create index.html	3 minutes ago

README.md

onvaessayerTest.github.io

test static server on github

2.5 UPLOAD/TÉLÉCHARGEMENT DE DONNÉES (RÉPERTOIRES ET FICHIERS)




Projects Wiki Security Insights Settings

onvaessayerTest.github.io /

Drag additional files here to add them to your repository
Or [choose your files](#)

- /restaurantsParis15b/Kenzem/bordeauxGR.jpg
- /restaurantsParis15b/Kenzem/hamburger.PNG
- /restaurantsParis15b/songSan/restaurant.json

 **Commit changes**


Add files via upload

Add an optional extended description...

Commit directly to the `main` branch.

Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes Cancel



2.5 CONSULTATION DES DONNÉES DEPUIS UN NAVIGATEUR

Index du repository

<https://onvaessayer.github.io>



Catalogue des restaurants (geoJSON)

<https://onvaessayer.github.io/restaurants/map.geojson>

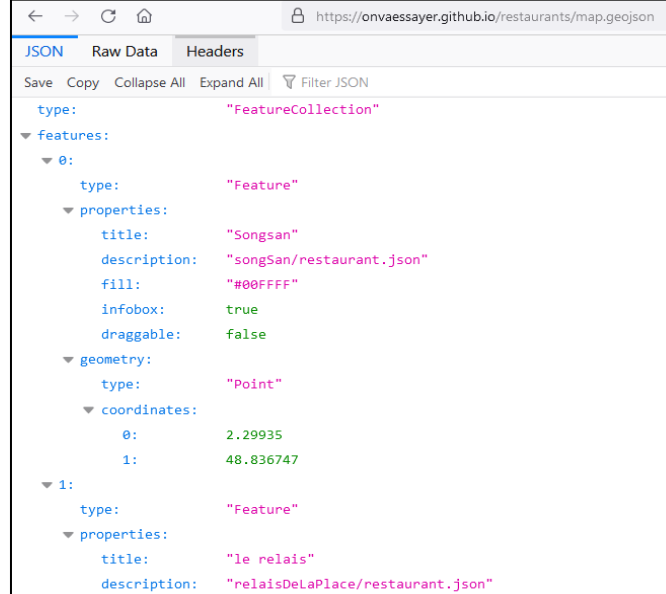
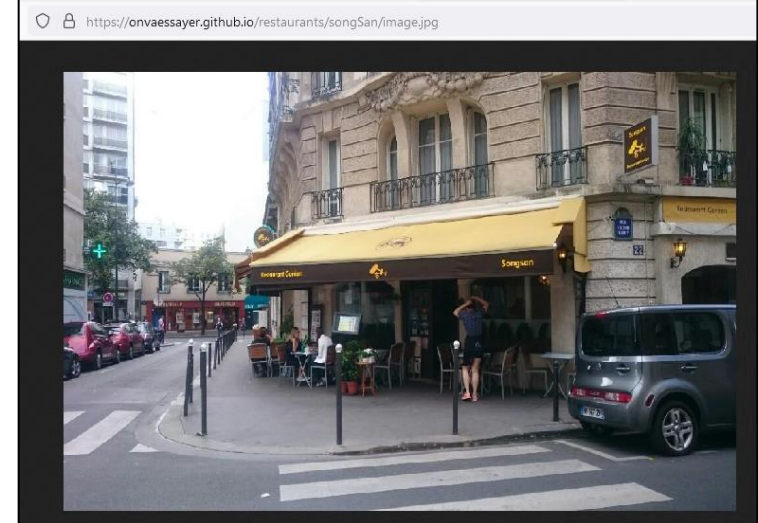


Image d'un restaurant

<https://onvaessayer.github.io/restaurants/songSan/image.jpg>



Un délai de plusieurs minutes est parfois nécessaire pour que le site Web soit mis à jour après un commit



2.6

Propriétés du catalogue geojson
lues automatiquement par App Inventor

PLAN

- Introduction : décomposition de l'application
- 1. Création d'un site Web / serveur de données
 - créer un compte github
 - créer un repository "username".github.io
 - créer un fichier index.html
 - consulter le site (index.html) avec un navigateur
- 2. Définition des données et préparation d'un jeu
 - Créer et décrire un restaurant (JSON)
 - le format JSON
 - créer un catalogue de restaurants géolocalisés (geoJSON)
 - télécharger les données sur le site Web (Github)
 - choix des propriétés du catalogue pour app inventor
- 3. Création de l'application mobile avec App Inventor

2.6 PROPRIÉTÉS GEOJSON DÉJÀ LUES PAR APPINVENTOR

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "title": "Songsan",
        "description": "http",
        "infobox": true,
        "fill": "#00FF00",
        "image": "korean.png"
      },
      "geometry": {
        "type": "Point", "coordinates": [2.29935, 48.83]
      }
    },
    { ... }
  ]
}
```

App Inventor :

Propriétés lues par la fonction de lecture geoJSON

- description becomes **Description**
- draggable becomes **Draggable**
- infobox becomes **EnableInfobox**
- fill becomes **FillColor**
- fill-opacity becomes **FillOpacity**
- image becomes **ImageAsset**
- stroke becomes **StrokeColor**
- stroke-opacity becomes **StrokeOpacity**
- stroke-width becomes **StrokeWidth**
- title becomes **Title**
- visible becomes **Visible**

2.6 GEOJSON : RÔLE OU UTILISATION DES ATTRIBUTS

```
{  
  "type": "FeatureCollection",  
  "features": [  
    {  
      "type": "Feature",  
      "properties": {  
        "title": "Songsan",  
        "description": "https://onvaessayer.github.io/gitshareGitDataset1/songSan/restaurant.json",  
        "infobox": true,  
        "fill": "#00FF00",  
        "image": "korean.png"  
      },  
      "geometry": {  
        "type": "Point", "coordinates": [2.29935, 48.836747]  
      }  
    },  
    { ... }  
  ]  
}
```

adresse (URL) du fichier JSON de ce restaurant

adresse (URL) de l'icone à afficher, ou media App Inventor



Création d'une application mobile
click and collect

PLAN

- Introduction : décomposition de l'application
- 1. Création d'un site Web / serveur de données
- 2. Définition des données et préparation d'un jeu
- 3. Création de l'application mobile avec App Inventor
 - 1. V1 : visualiser le catalogue des restaurants sur une carte
 - 2. V2a : sélectionner un restaurant et afficher ses nom, adresse, image et liste de plats
 - 3. V2b : codage défensif, modèle de données, adresses relatives, Dropbox & Google Drive
 - 4. V3a : identifier et enregistrer l'utilisateur et la carte
 - 5. V3b : préparer et passer une commande
 - 6. V3c : bonus

3.1

visualiser le catalogue
des restaurants sur une carte

PLAN

- Introduction : décomposition de l'application
- 1. Création d'un site Web / serveur de données
- 2. Définition des données et préparation d'un jeu
- 3. Création de l'application mobile avec App Inventor
 - 1. V1 : visualiser le catalogue des restaurants sur une carte
 - 2. V2a : sélectionner un restaurant et afficher ses nom, adresse, image et liste de plats
 - 3. V2b : codage défensif, modèle de données, adresses relatives, Dropbox & Google Drive
 - 4. V3a : identifier et enregistrer l'utilisateur et la carte
 - 5. V3b : préparer et passer une commande
 - 6. V3c : bonus

DESIGN : COMPOSANT MAP

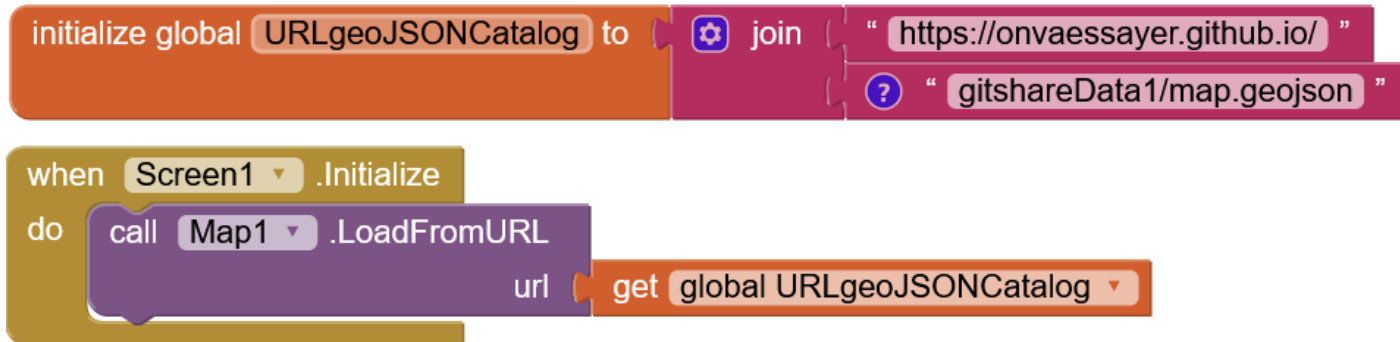
The screenshot displays the MIT App Inventor web interface. At the top, the MIT App Inventor logo is on the left, and navigation links for Projects, Connect, Build, Settings, and Help are in the center. On the right, there are links for My Projects, View Trash, Guide, Report an Issue, and language settings. Below the navigation bar, the project name 'webshareWithGit1' is shown, along with buttons for Screen1, Add Screen..., Remove Screen, and Publish to Gallery. The interface is divided into four main panels: Palette, Viewer, Components, and Properties.

- Palette:** A sidebar on the left containing various UI components. The 'Maps' category is selected, showing options like Circle, FeatureCollection, LineString, Map (highlighted), Marker, Navigation, Polygon, and Rectangle.
- Viewer:** A central area showing a mobile phone emulator. The screen displays a map of Paris with a search bar and a list of nearby locations. A checkbox 'Display hidden components in Viewer' is visible above the emulator.
- Components:** A panel on the right showing the hierarchy of components on the screen. It lists 'Screen1' and 'Map1' with 'Rename' and 'Delete' buttons.
- Properties:** A panel on the far right showing the configuration options for the selected 'Map1' component. Properties include CenterFromString (48.84,2.34), EnablePan (checked), EnableRotation (unchecked), EnableZoom (checked), Height (Fill parent...), Width (Fill parent...), LocationSensor (None...), MapType (Roads), Rotation (0.0), ScaleUnits (Metric), ShowScale (unchecked), ShowUser (unchecked), and ZoomLevel (11).

PROGRAMME : MAP LOAD FROM URL



PROGRAMME : MAP LOAD FROM URL

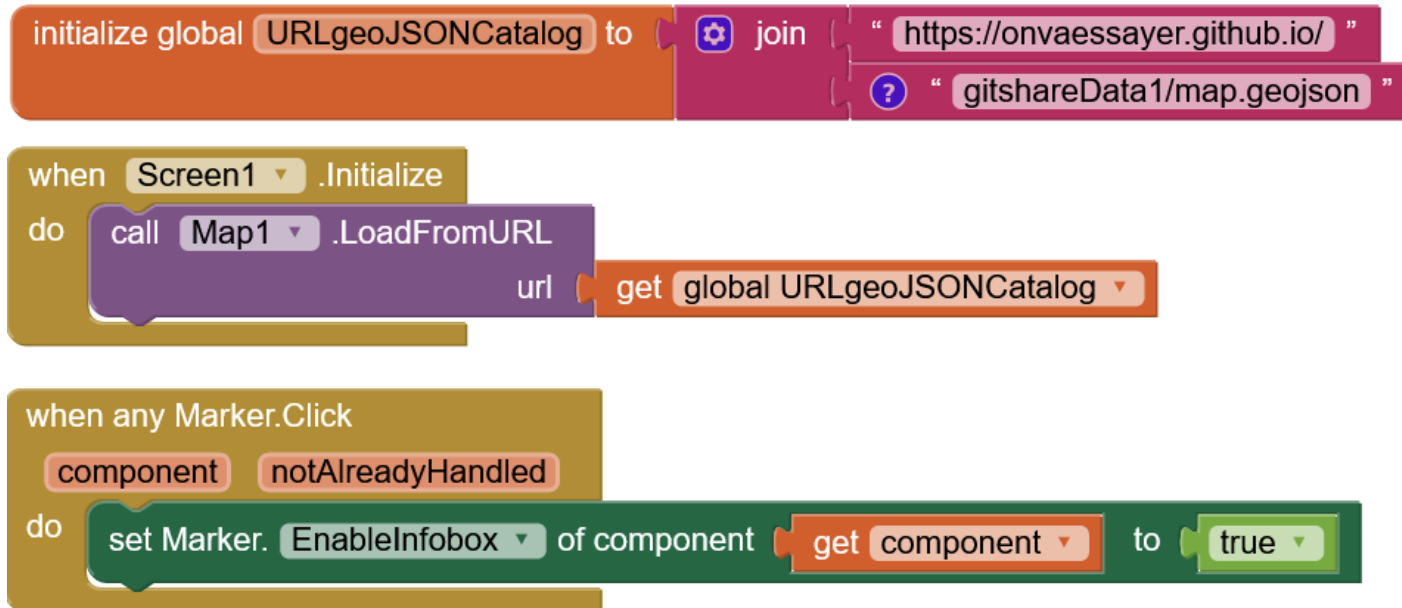


Propriétés de chaque objet du catalogue geoJSON qui sont lues par LoadFromURL :

- `description` becomes `Description`
- `draggable` becomes `Draggable`
- `infobox` becomes `EnableInfobox`
- `fill` becomes `FillColor`
- `fill-opacity` becomes `FillOpacity`
- `image` becomes `ImageAsset`
- `stroke` becomes `StrokeColor`
- `stroke-opacity` becomes `StrokeOpacity`
- `stroke-width` becomes `StrokeWidth`
- `title` becomes `Title`
- `visible` becomes `Visible`

- **texte libre** utilisé pour enregistrer l'adresse du fichier de chaque restaurant
- `true/false` : déplaçable ou non
- `true/false` : affiche title et description quand on clique sur cet objet
- couleur de remplissage
- **adresse web** de l'icone affiché sur la carte (ou nom du media appinventor)
- couleur de contour
- transparence/opacité
- largeur du contour
- **title** : nom du restaurant
- `true/false` : visible ou pas

PROGRAMME : MAP LOAD FROM URL



CODE SOURCE DE L'APPLICATION

le code source de l'application (gitshare1.aia) et la version exécutable (gitshare1.apk) sont à l'adresse

- <http://onvaessayer.org/appinventor?res=gitshare>

Voir également sur github

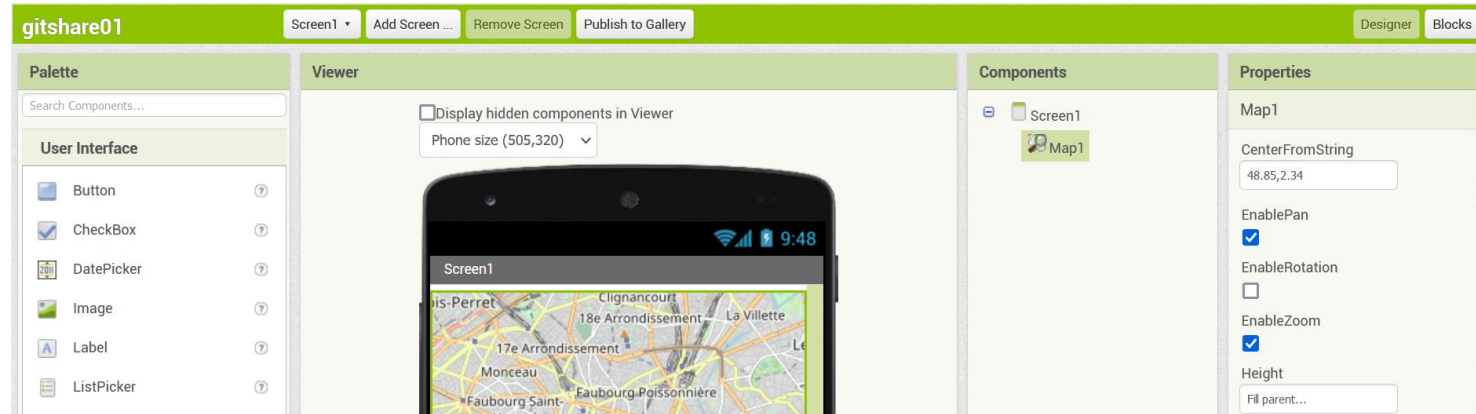
- <https://onvaessayer.github.io/gitshare1.aia> et <https://onvaessayer.github.io/gitshare1.apk>

Vous avez aussi un accès direct à l'application dans App Inventor depuis l'onglet projets :

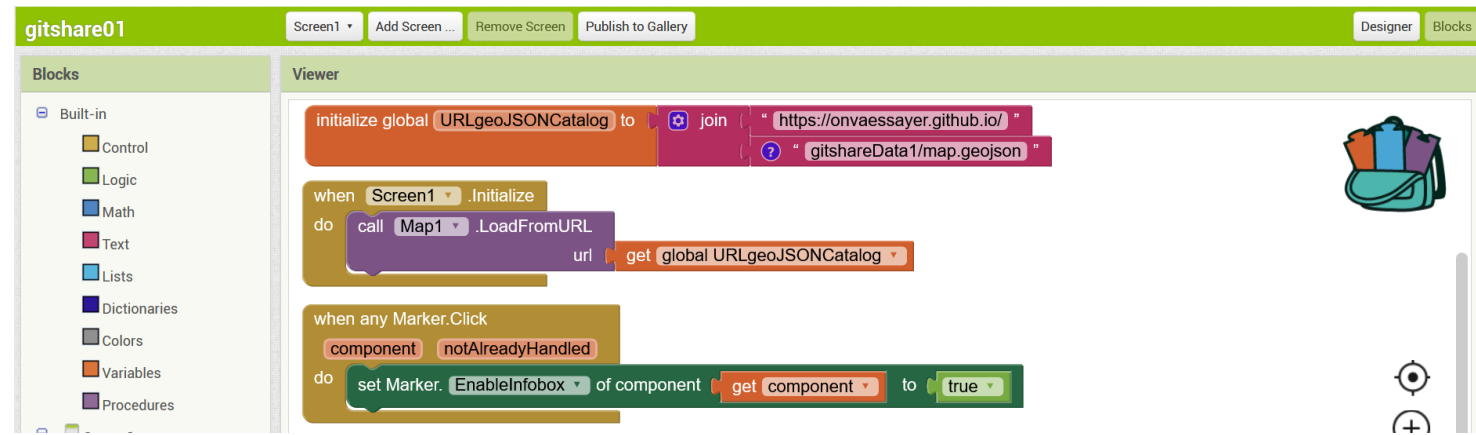
- import for repository
- ajouter repository <http://onvaessayer.org/apps/githare/>
- et charger gitshare1

AFFICHAGE D'UNE CARTE AVEC LES RESTAURANTS (GITSHARE 1)

Design :
(composant map)



Blocs :
(algorithmme)



TÉLÉCHARGER L'APPLICATION DEPUIS UN REPOSITORY (GITSHARE 1)

The screenshot shows the MIT App Inventor web interface. At the top, there are navigation tabs for 'Start new project', 'Move To Trash', 'View Trash', and 'Log Out'. Below these is a green bar with the text 'Create a Project from a Template'. On the left, a 'Projects' list contains various project names, including 'gitshare02', 'gitshare1', 'gitshare01', 'gitshare03', 'gitshare01_', 'gitshare3', 'gitshareSolutuuiion1', 'gitsharemini_copy', 'gitsharemini', 'ConvertisseurMonnaie_0007_v4', 'gitshare2', 'gitshare3_backup', 'testLoadFromURL', 'clickAndUber3', 'test7', 'gitshare1_a', 'pokemonNamesV2', 'test6', 'test7b', and 'test7a'. The main area is dominated by a dialog box titled 'Built-in Templates'. This dialog has a search bar and a list of templates. The 'gitshare' template is selected, and its URL is highlighted: <http://onvaessayer.org/appinventor/apps/gitshare/>. A preview of the 'gitshare' app is shown, displaying a 'Welcome to DIY Book Club!' screen with 'Write Review' and 'Read Reviews' buttons. The dialog has 'Cancel' and 'OK' buttons at the bottom. The background shows a list of timestamps on the right, including '2:07:05 PM', '0:03:42 PM', '0:00:10 PM', ':45:30 PM', ':00:56 PM', ':34:23 PM', ':11:40 PM', '1:26:55 AM', '1:26:34 AM', '4:16:00 PM', '36:49 PM', '10:33:29 AM', '7:48:40 PM', '11:26:07 AM', '12:48:24 AM', '12:24:38 AM', '4:22:22 PM', '0:28:28 PM', and '2:59:18 PM'. At the bottom, there are two dates: 'Apr 14, 2022, 6:46:52 PM' and 'Apr 22, 2022, 11:59:55 AM'.

TÉLÉCHARGER L'APPLICATION DEPUIS UN REPOSITORY (GITSHARE 1)

The screenshot shows the MIT App Inventor web interface. At the top, there are navigation tabs: "Start new project", "Move To Trash", "View Trash", and "Log". Below these is a green bar with "Create a Project from a Template".

A modal dialog is open, titled "Create a Project from a Template". It has a dropdown menu at the top showing "http://onvaessayer.org/appinventor/apps/gitshare/". To the right of the dropdown is a button "Remove this repository".

The dialog contains two repository options:

- gitshare1**: share data with git : step 1 geoJSON catalog for click and collect. This option is highlighted in blue.
- gitshare2**: share data with git : step 2 JSON data items for click and collect.

Below the text descriptions is a preview image of a mobile application interface showing a map with several location pins. The map labels include "Rue Le Peletier", "Rue de la Gra", "Rue Chauvart", "Richelieu Drouot", "Rue Favart", and "Vienn".

At the bottom of the dialog are "Cancel" and "OK" buttons.

In the background, on the left, there is a "Projects" list with checkboxes next to various project names like "gitshare02", "gitshare1", "gitshare01", etc. On the right, there is a vertical list of timestamps, including "2:07:05 PM", "0:03:42 PM", "0:00:10 PM", "4:45:30 PM", "1:00:56 PM", "1:34:23 PM", "1:11:40 PM", "1:26:55 AM", "1:26:34 AM", "4:16:00 PM", "36:49 PM", "10:33:29 AM", "7:48:40 PM", "11:26:07 AM", "12:48:24 AM", "12:24:38 AM", "4:22:22 PM", "10:28:28 PM", and "2:59:18 PM".

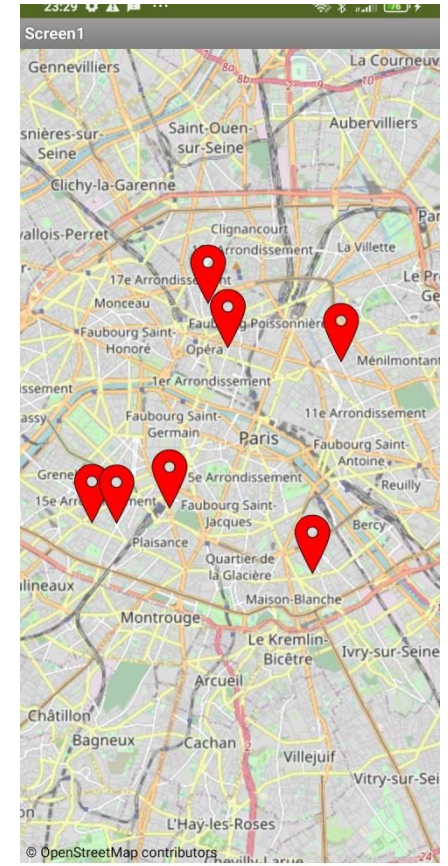
At the bottom of the interface, there are two date and time stamps: "Apr 14, 2022, 6:46:52 PM" and "Apr 22, 2022, 11:59:55 AM".

VERSIONS DU CATALOGUE, DU PLUS SIMPLE AU PLUS COMPLET

base (title + description) :

<https://onvaessayer.github.io/gitshareData1/map1.geojson>

```
1 {
2   "type": "FeatureCollection",
3   "features": [
4     {
5       "type": "Feature",
6       "properties": {
7         "title": "Songsan",
8         "description": "https://onvaessayer.github.io/gitshareData1/songSan/restaurant.json"
9       },
10      "geometry": {
11        "type": "Point", "coordinates": [2.29935, 48.836747]
12      }
13    },
14    {
15      "type": "Feature",
16      "properties": {
17        "title": "le relais",
18        "description": "https://onvaessayer.github.io/gitshareData1/relaisDeLaPlace/restaurant.json"
19      },
20      "geometry": {
21        "type": "Point", "coordinates": [2.306638, 48.836606]
22      }
23    }
24  ]
25 }
```

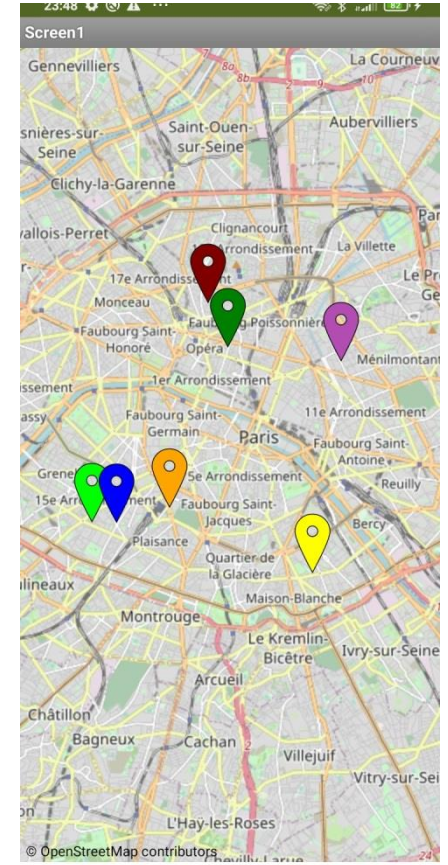


VERSIONS DU CATALOGUE, DU PLUS SIMPLE AU PLUS COMPLET

color (title + description + fill) :

<https://onvaessayer.github.io/gitshareData1/map1a.geojson>

```
1 {
2   "type": "FeatureCollection",
3   "features": [
4     {
5       "type": "Feature",
6       "properties": {
7         "title": "Songsan",
8         "description": "https://onvaessayer.github.io/gitshareData1/songSan/restaurant.json",
9
10        "fill": "#00FF00" ← Icon color (hexa #aabbcc)
11      },
12      "geometry": {
13        "type": "Point", "coordinates": [2.29935, 48.836747]
14      }
15    },
16    {
17      "type": "Feature",
18      "properties": {
19        "title": "le relais",
20        "description": "https://onvaessayer.github.io/gitshareData1/relaisDeLaPlace/restaurant.json",
21
22        "fill": "#0000FF"
23      },
24      "geometry": {
25        "type": "Point", "coordinates": [2.306638, 48.836606]
26      }
27    }
28  ]
29 }
```



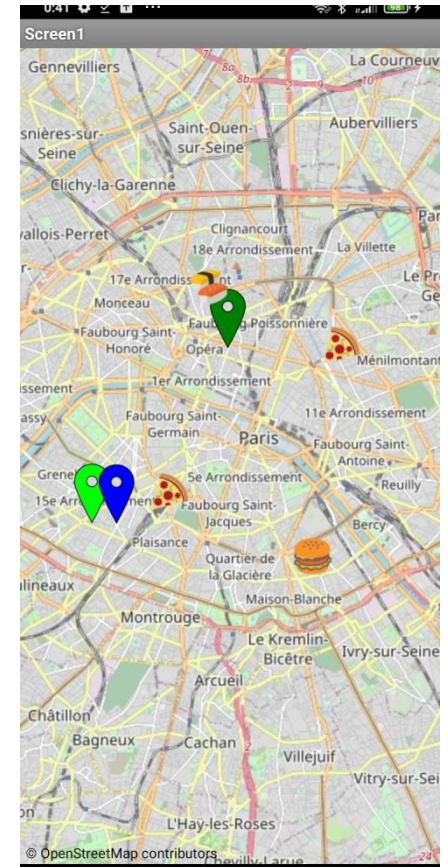
VERSIONS DU CATALOGUE, DU PLUS SIMPLE AU PLUS COMPLET

local icons (title + description + local image)

<https://onvaessayer.github.io/gitshareData1/map2.geojson>

```
1 {
2   "type": "FeatureCollection",
3   "features": [
4     {
5       "type": "Feature",
6       "properties": {
7         "title": "Songsan",
8         "description": "https://onvaessayer.github.io/gitshareData1/songSan/restaurant.json",
9
10        "fill": "#00FF00",
11        "image": "korean.png"
12      },
13      "geometry": {
14        "type": "Point", "coordinates": [2.29935, 48.836747]
15      }
16    },
17    {
18      "type": "Feature",
19      "properties": {
20        "title": "le relais",
21        "description": "https://onvaessayer.github.io/gitshareData1/relaisDeLaPlace/restaurant.json",
22
23        "fill": "#0000FF",
24        "image": "frenchFood.png"
25      },
26      "geometry": {
27        "type": "Point", "coordinates": [2.306638, 48.836606]
28      }
29    }
30  ]
31 }
```

← Image file name in app media



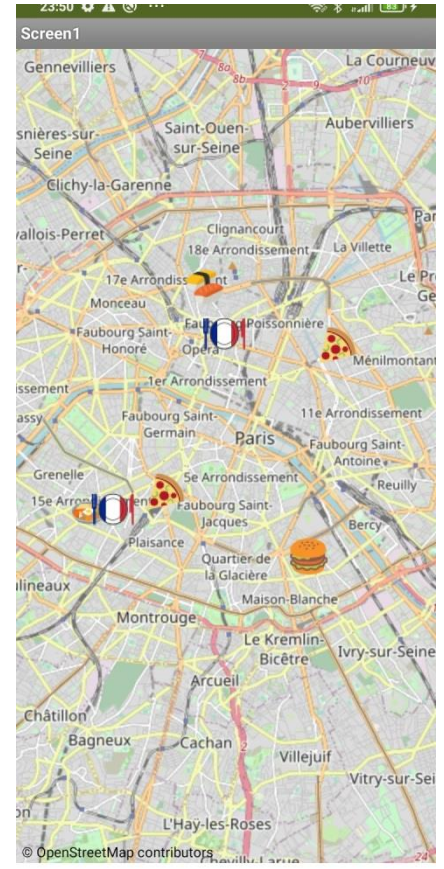
VERSIONS DU CATALOGUE, DU PLUS SIMPLE AU PLUS COMPLET

local icons (title + description + web image)

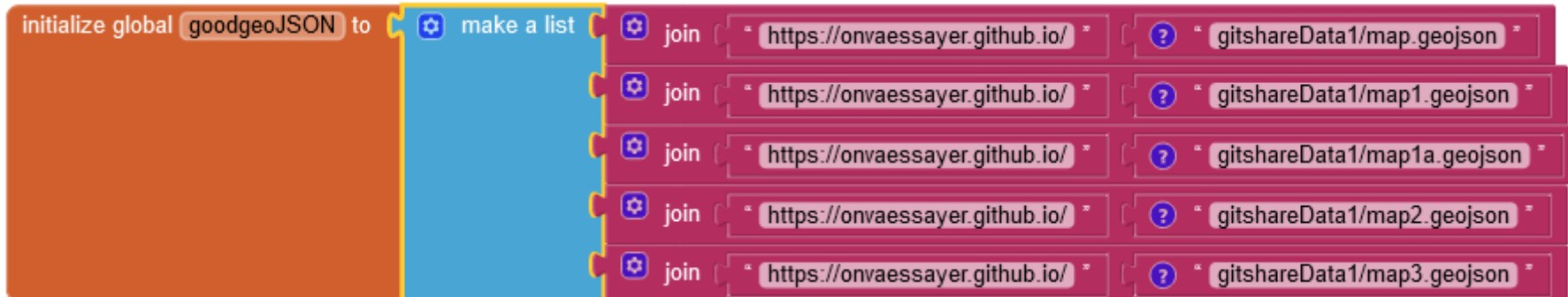
<https://onvaessayer.github.io/gitshareData1/map3.geojson>

```
1 {
2   "type": "FeatureCollection",
3   "features": [
4     {
5       "type": "Feature",
6       "properties": {
7         "title": "Songsan",
8         "description": "https://onvaessayer.github.io/gitshareData1/songSan/restaurant.json",
9
10        "fill": "#00FF00",
11        "image": "https://onvaessayer.github.io/gitshareData1/icons/korean.png"
12      },
13      "geometry": {
14        "type": "Point", "coordinates": [2.29935, 48.836747]
15      }
16    },
17    {
18      "type": "Feature",
19      "properties": {
20        "title": "le relais",
21        "description": "https://onvaessayer.github.io/gitshareData1/relaisDeLaPlace/restaurant.json",
22
23        "fill": "#0000FF",
24        "image": "https://onvaessayer.github.io/gitshareData1/icons/frenchFood.png"
25      },
26      "geometry": {
27        "type": "Point", "coordinates": [2.306638, 48.836606]
28      }
29    }
30  ]
31 }
```

↑
Image URL on the web



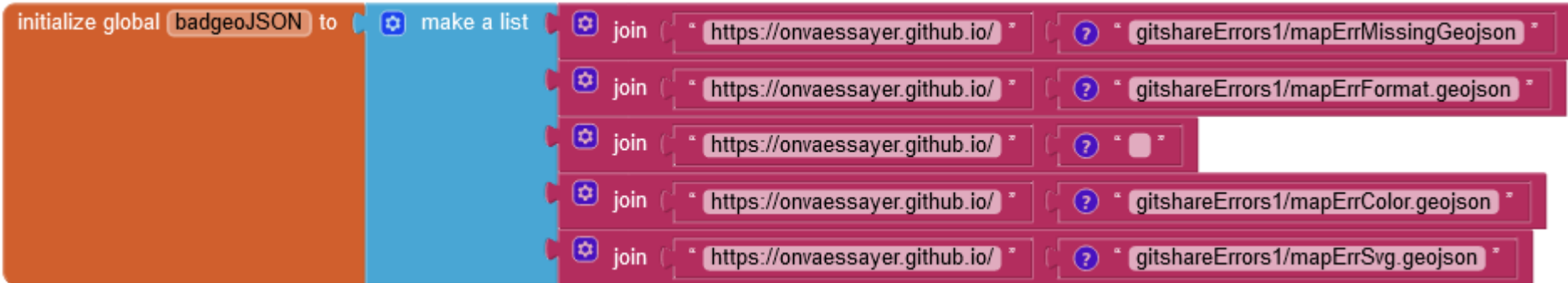
GEOJSON CATALOG URLs FOR TESTING gitshareDataset1



Scratch code for testing gitshareDataset1. It starts with an orange 'initialize global' block setting 'goodgeoJSON' to a blue 'make a list' block. Five 'join' blocks follow, each concatenating the URL 'https://onvaessayer.github.io/' with a specific dataset path: 'gitshareData1/map.geojson', 'gitshareData1/map1.geojson', 'gitshareData1/map1a.geojson', 'gitshareData1/map2.geojson', and 'gitshareData1/map3.geojson'.

```
initialize global goodgeoJSON to make a list  
join " https://onvaessayer.github.io/ " " gitshareData1/map.geojson "  
join " https://onvaessayer.github.io/ " " gitshareData1/map1.geojson "  
join " https://onvaessayer.github.io/ " " gitshareData1/map1a.geojson "  
join " https://onvaessayer.github.io/ " " gitshareData1/map2.geojson "  
join " https://onvaessayer.github.io/ " " gitshareData1/map3.geojson "
```

GEOJSON CATALOG URLs FOR TESTING gitshareErrors1



Scratch code for testing gitshareErrors1. It starts with an orange 'initialize global' block setting 'badgeoJSON' to a blue 'make a list' block. Five 'join' blocks follow, each concatenating the URL 'https://onvaessayer.github.io/' with a specific error path: 'gitshareErrors1/mapErrMissingGeojson', 'gitshareErrors1/mapErrFormat.geojson', an empty string, 'gitshareErrors1/mapErrColor.geojson', and 'gitshareErrors1/mapErrSvg.geojson'.

```
initialize global badgeoJSON to make a list  
join " https://onvaessayer.github.io/ " " gitshareErrors1/mapErrMissingGeojson "  
join " https://onvaessayer.github.io/ " " gitshareErrors1/mapErrFormat.geojson "  
join " https://onvaessayer.github.io/ " " "  
join " https://onvaessayer.github.io/ " " gitshareErrors1/mapErrColor.geojson "  
join " https://onvaessayer.github.io/ " " gitshareErrors1/mapErrSvg.geojson "
```

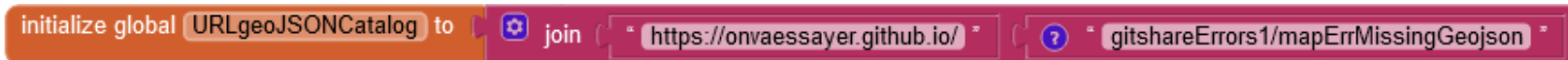
ERREURS POSSIBLES AVEC LE FICHER GEOJSON

- fichier manquant à cette URL → Error 1101 : unable to get a response
- pas d'accès à Internet → Error 1101 : unable to get a response
- contenu non conforme à geoJSON → hard crash
- format d'image non supporté (svg) → hard crash
- valeur de couleur invalide → hard crash

ERREURS POSSIBLES AVEC LE FICHER GEOJSON

- fichier manquant à cette URL → Error 1101 : unable to get a response

test geoJSON URL

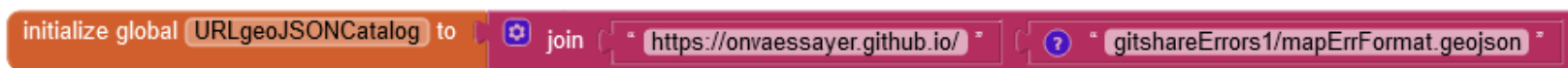


- pas d'accès à Internet → Error 1101 : unable to get a response

test geoJSON URL

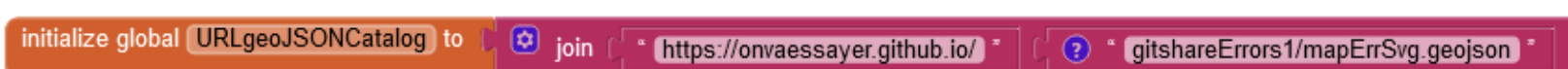
- contenu non conforme à geoJSON → hard crash

test geoJSON URL



- format d'image non supporté (svg) → hard crash

test geoJSON URL



- valeur de couleur invalide → hard crash

test geoJSON URL



ACTIONS DÉFENSIVES :

```
initialize global URLgeoJSONCatalog to  
  join ( " https://onvaessayer.github.io/ " )  
  ( " gitshareData1/map.geojson " )
```

```
when Screen1 .Initialize  
do call Map1 .LoadFromURL  
  url get global URLgeoJSONCatalog
```

```
when any Marker.Click  
  component notAlreadyHandled  
do set Marker. EnableInfobox of component  
  get component to true
```

ACTIONS DÉFENSIVES : évènement Error Occured

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/ ", "gitshareData1/map.geojson "]
```

```
when Screen1 .Initialize  
do call Map1 .LoadFromURL  
    url get global URLgeoJSONCatalog
```

```
when any Marker.Click  
    component notAlreadyHandled  
do set Marker. EnableInfobox of component get component to true
```

```
when Screen1 .ErrorOccurred  
    component functionName errorNumber message  
do call Notifier1 .ShowMessageDialog  
    message join [join [component : get component <br>]  
                  join [function : get functionName <br>]  
                  join [message : get message <br>]]  
    title join [Error : get errorNumber]  
    buttonText "OK"
```

Les erreurs gérées par App Inventor, l'évènement errorOccured de l'écran Screen1. On peut alors prévenir l'utilisateur,

ACTIONS DÉFENSIVES : évènement Error Occured

```
initialize global URLgeoJSONCatalog to join " https://onvaessayer.github.io/ " " gitshareData1/map.geojson "
```

```
when Screen1 .Initialize  
do call Map1 .LoadFromURL url get global URLgeoJSONCatalog
```

```
when any Marker.Click  
component notAlreadyHandled  
do set Marker. EnableInfobox of component get component to true
```

```
when Screen1 .ErrorOccurred  
component functionName errorNumber message  
do call Notifier1 .ShowMessag...
```

ACTIONS DÉFENSIVES : JEUX DE DONNÉES VALIDES POUR LES TESTS

```
initialize global URLgeoJSONCatalog to  
  join ( " https://onvaessayer.github.io/ " )  
  ( " gitshareData1/map.geojson " )
```

```
when Screen1 .Initialize  
do call Map1 .LoadFromURL  
  url ( get global URLgeoJSONCatalog )
```

```
when any Marker.Click  
  component notAlreadyHandled  
do set Marker. EnableInfobox of component  
  ( get component ) to true
```

```
when Screen1 .ErrorOccurred  
  component functionName errorNumber message  
do call Notifier1 .ShowMessag...
```

Valid Geojson catalogs

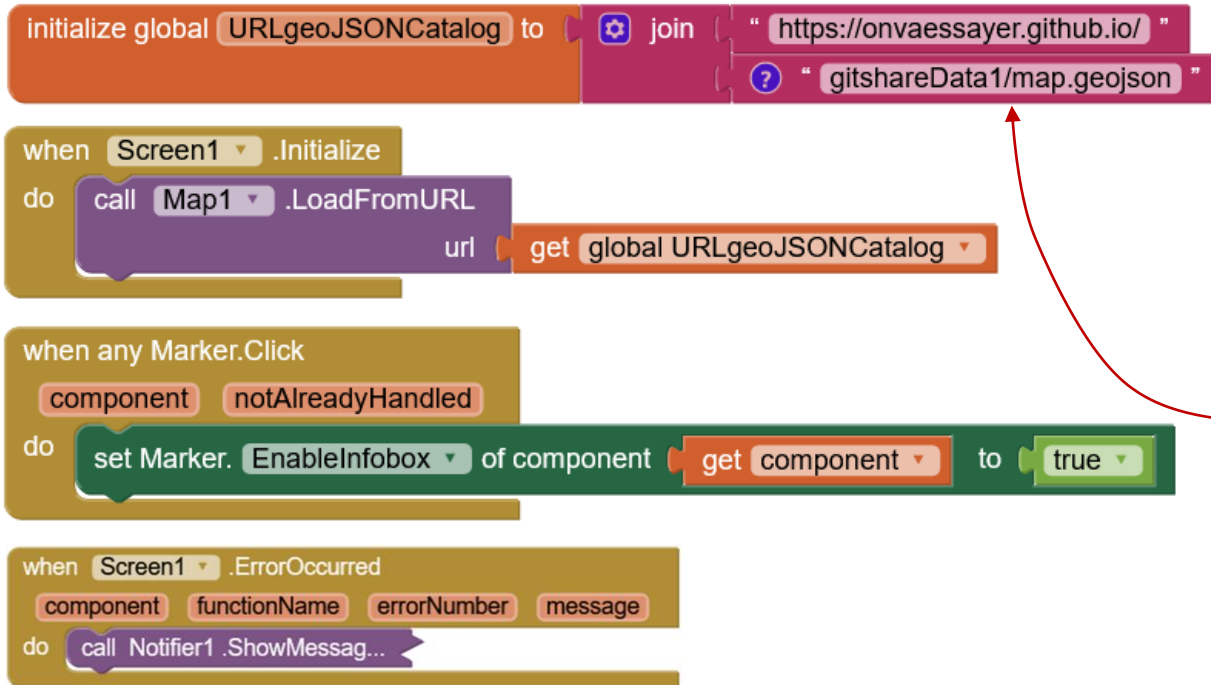
```
" gitshareData1/map1.geojson "
```

```
" gitshareData1/map1a.geojson "
```

```
" gitshareData1/map2.geojson "
```

```
" gitshareData1/map3.geojson "
```

ACTIONS DÉFENSIVES : JEUX DE DONNÉES ERRONÉES POUR LES TESTS




Valid Geojson catalogs

- “ gitshareData1/map1.geojson ”
- “ gitshareData1/map1a.geojson ”
- “ gitshareData1/map2.geojson ”
- “ gitshareData1/map3.geojson ”

Geojson catalogs with errors

- “ gitshareErrors1/map.geojson ”
- “ ”
- “ gitshareErrors1/mapErrMissingGeojson.geojson ”
- “ gitshareErrors1/mapErrFile.geojson ”
- “ gitshareErrors1/mapErrColor.geojson ”
- “ gitshareErrors1/mapErrSvg.geojson ”
- “ gitshareErrors1/mapErrWrongFile.geojson ”
- “ gitshareErrors1/mapErrMissingFile.geojson ”



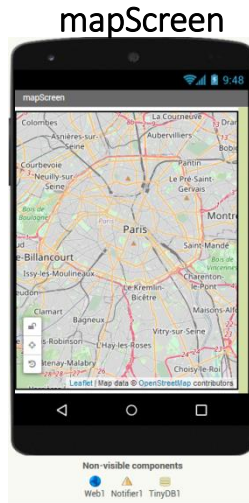
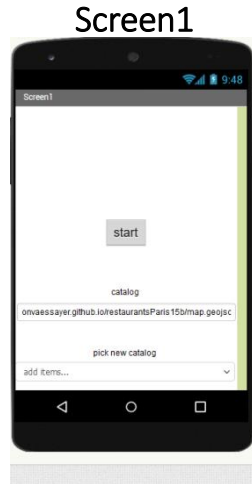
3.2

Afficher les données de restaurants
gitshare2a

PLAN

- Introduction : décomposition de l'application
- 1. Création d'un site Web / serveur de données
- 2. Définition des données et préparation d'un jeu
- 3. Création de l'application mobile avec App Inventor
 - 1. V1 : visualiser le catalogue des restaurants sur une carte
 - 2. V2a : sélectionner un restaurant et afficher ses nom, adresse, image et liste de plats
 - 3. V2b : codage défensif, modèle de données, adresses relatives, Dropbox & Google Drive
 - 4. V3a : identifier et enregistrer l'utilisateur et la carte
 - 5. V3b : préparer et passer une commande
 - 6. V3c : bonus

AFFICHER LES DONNÉES D'UN RESTAURANT : V1 → V2a



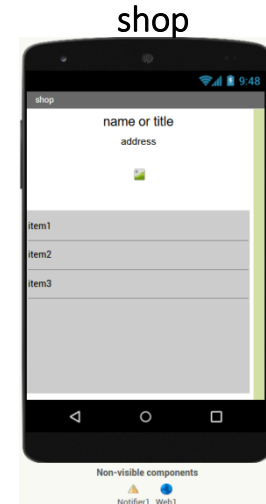
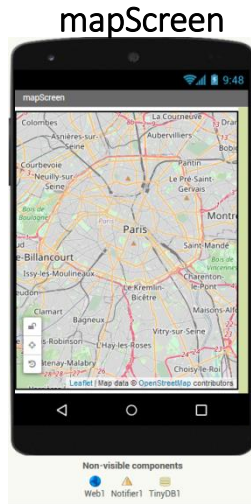
```
when Screen1.Initialize
do open another screen with start value screenName mapScreen
startValue ""
```

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData1/map.geojson"]
```

```
when mapScreen.Initialize
do call Map1.LoadFromURL
url get global URLgeoJSONCatalog
```

```
when any Marker.Click
component notAlreadyHandled
do set Marker.EnableInfoBox of component get component to true
```

AFFICHER LES DONNÉES D'UN RESTAURANT : V1 → V2a

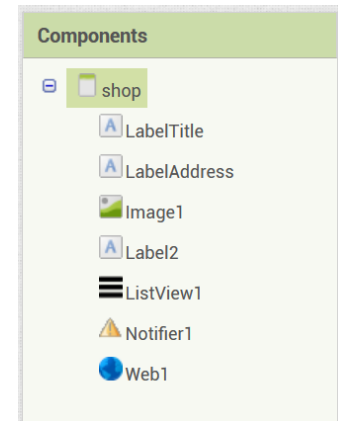


```
when Screen1.Initialize
do
  open another screen with start value screenName mapScreen
  startValue ""
```

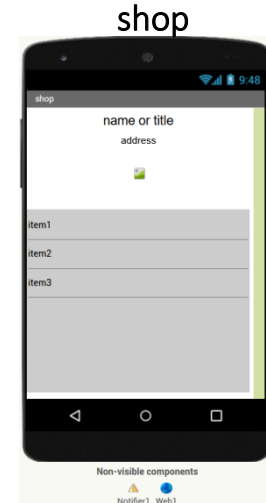
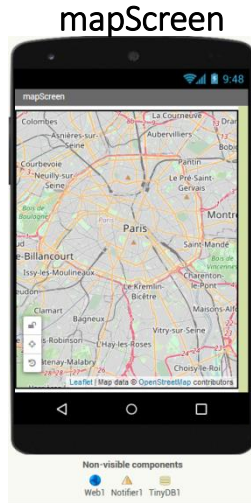
```
initialize global URLgeoJSONCatalog to
  join ["https://onvaessayer.github.io/", "gitshareData1/map.geojson"]

when mapScreen.Initialize
do
  call Map1.LoadFromURL
  url get global URLgeoJSONCatalog

when any Marker.Click
  component notAlreadyHandled
do
  set Marker.EnableInfobox of component get component to true
```



AFFICHER LES DONNÉES D'UN RESTAURANT : V1 → V2a

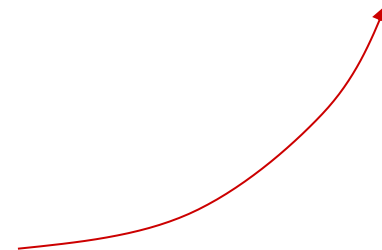


```
when Screen1.Initialize
do
  open another screen with start value screenName mapScreen
  startValue ""

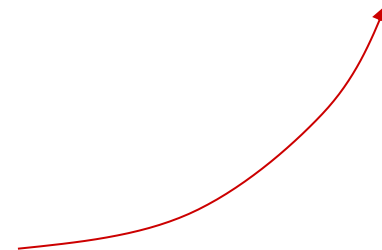
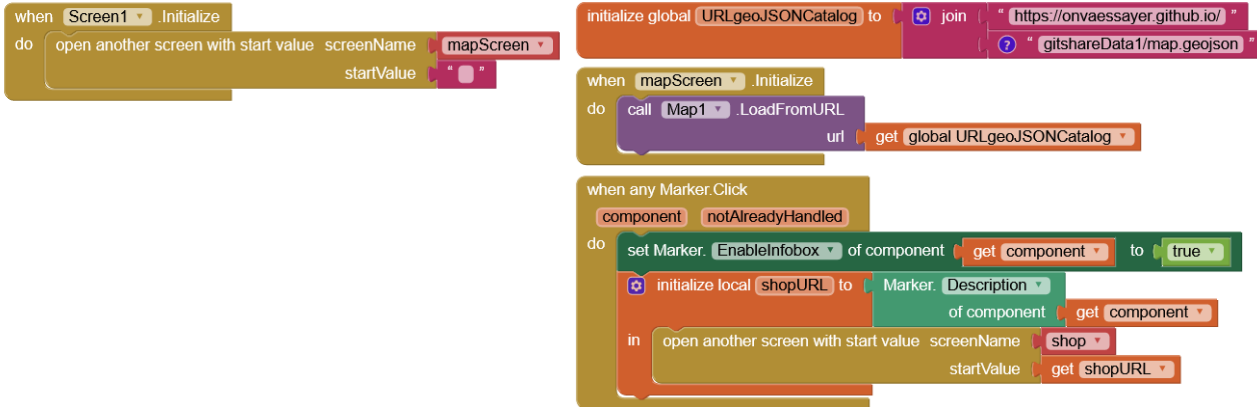
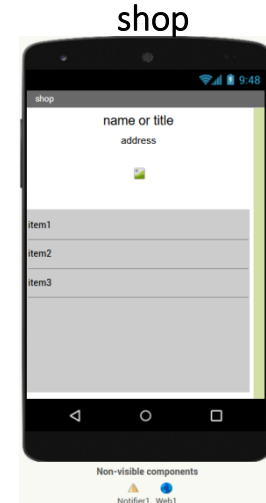
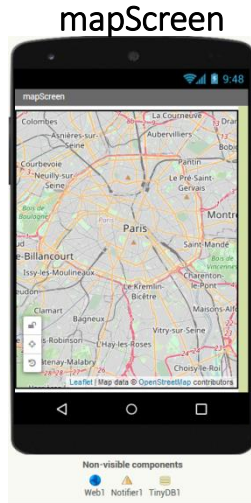
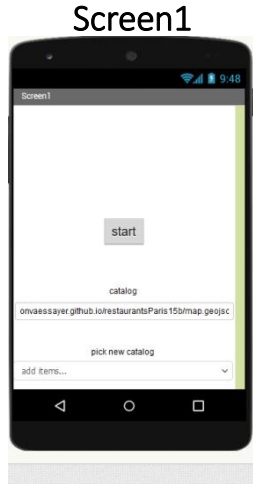
initialize global URLgeoJSONCatalog to join "https://onvaessayer.github.io/"
  "gitshareData1/map.geojson"

when mapScreen.Initialize
do
  call Map1.LoadFromURL
  uri get global URLgeoJSONCatalog

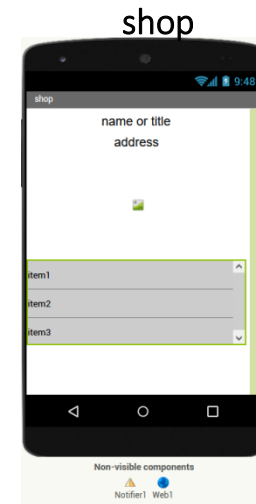
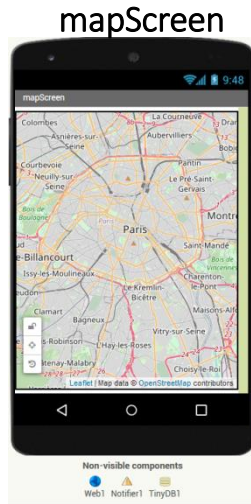
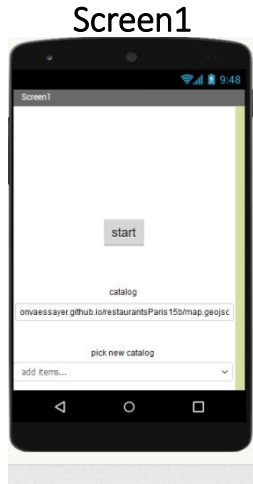
when any Marker.Click
component notAlreadyHandled
do
  set Marker.EnableInfobox of component get component to true
  initialize local shopURL to Marker.Description
  of component get component
  in
    open another screen with start value screenName shop
    startValue get shopURL
```



AFFICHER LES DONNÉES D'UN RESTAURANT : V1 → V2a



AFFICHER LES DONNÉES D'UN RESTAURANT : V1 → V2a



```
when Screen1.Initialize
do
  open another screen with start value screenName mapScreen
  startValue ""
```

```
initialize global URLgeoJSONCatalog to join "https://onvaessayer.github.io/"
  "gitshareData1/map.geojson"
```

```
when mapScreen.Initialize
do
  call Map1.LoadFromURL
  url get global URLgeoJSONCatalog
```

```
when any Marker.Click
  component notAlreadyHandled
do
  set Marker.EnableInfoBox of component get component to true
  initialize local shopURL to Marker.Description
  of component get component
  in
    open another screen with start value screenName shop
    startValue get shopURL
```

```
initialize global shop to create empty dictionary
initialize global shopURL to ""
initialize global items to create empty list
```

```
when Web1.GetText
  uri responseCode responseType responseContent
do
  set global shop to call Web1.JsonTextDecodeWithDictionaries jsonText get responseContent
  set LabelTitle.Text to get value for key "title" in dictionary get global shop or if not found "?"
  set LabelAddress.Text to get value for key "address" in dictionary get global shop or if not found "?"
  set Image1.Picture to get value for key "image"
  in dictionary get global shop
  or if not found "AndroidFR.png"
  set global items to get value for key "items" in dictionary get global shop or if not found create empty list
  set ListView1.Elements to get global items
```

```
when shop.Initialize
do
  set global shopURL to get start value
  set Web1.Uri to get global shopURL
  set Web1.SaveResponse to false
  call Web1.Get
```

{ JSON OBJECT } → App Inventor DICTIONARY

{ JSON text object }

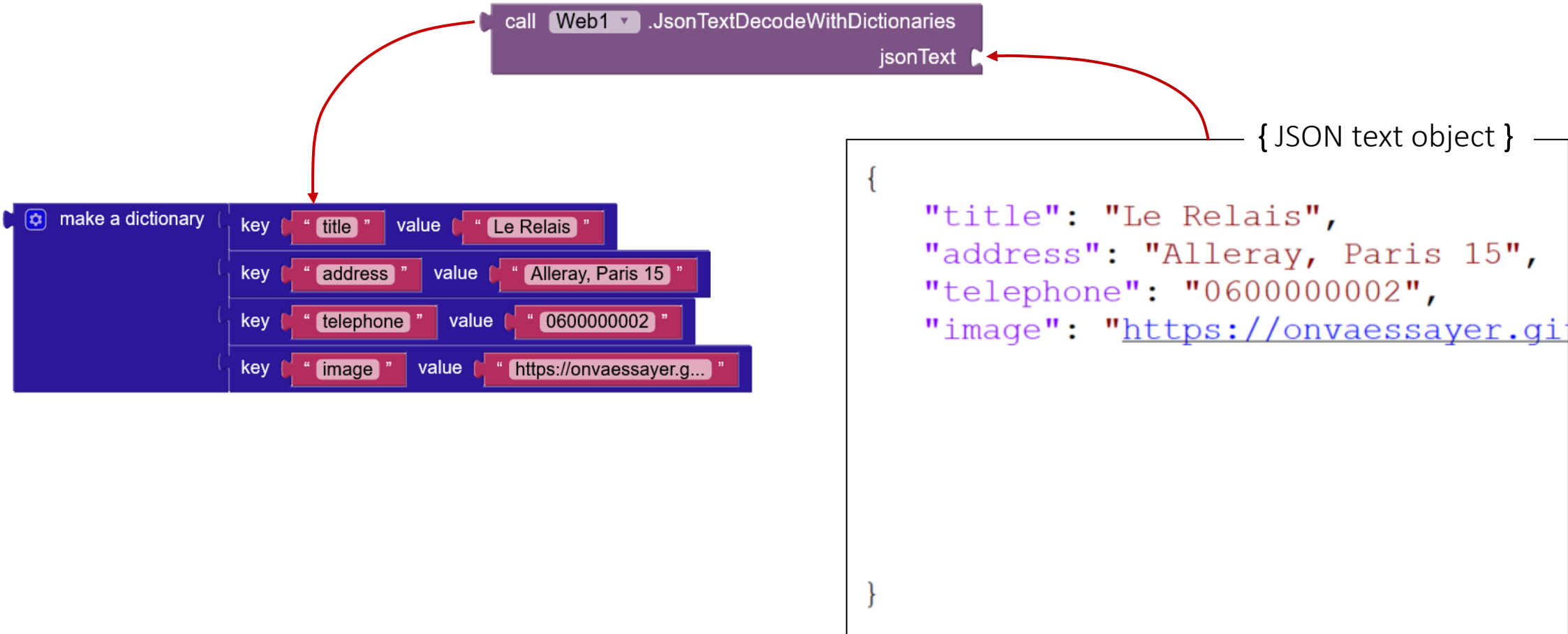
```
{  
  "title": "Le Relais",  
  "address": "Alleray, Paris 15",  
  "telephone": "0600000002",  
  "image": "https://onvaessayer.gi  
}
```

{ JSON OBJECT } → App Inventor DICTIONARY

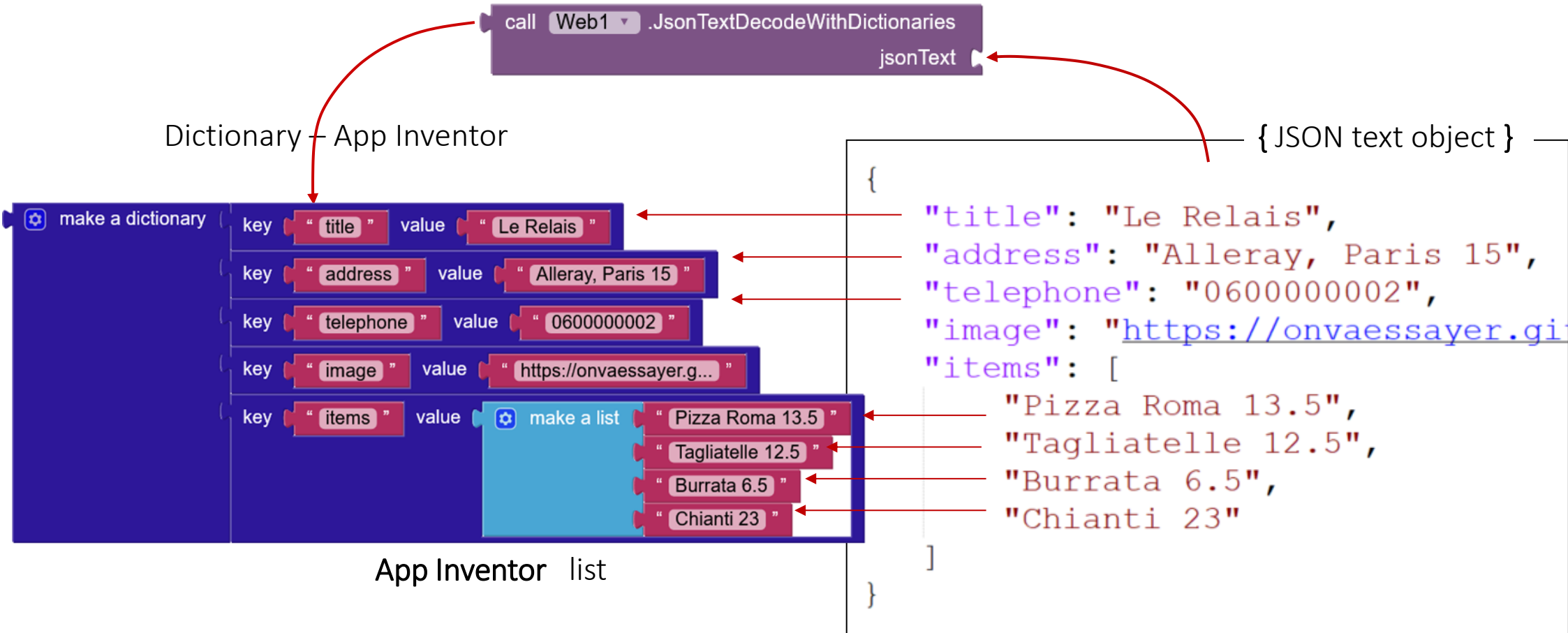
```
{  
  "title": "Le Relais",  
  "address": "Alleray, Paris 15",  
  "telephone": "0600000002",  
  "image": "https://onvaessayer.gi  
  "items": [  
    "Pizza Roma 13.5",  
    "Tagliatelle 12.5",  
    "Burrata 6.5",  
    "Chianti 23"  
  ]  
}
```

[JSON text list]

{ JSON OBJECT } → App Inventor DICTIONARY



{ JSON OBJECT , LIST } → App Inventor DICTIONARY, LIST



DICTIONARY & LIST BLOCKS

Blocks

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists**
 - Dictionaries**
 - Colors

initialize global **shop** to create empty dictionary

get value for key **"address"**
in dictionary **get global shop**
or if not found **"?"**

set value for key **"quantity"**
in dictionary **get global item**
to **1**

is a dictionary?

initialize global **items** to create empty list

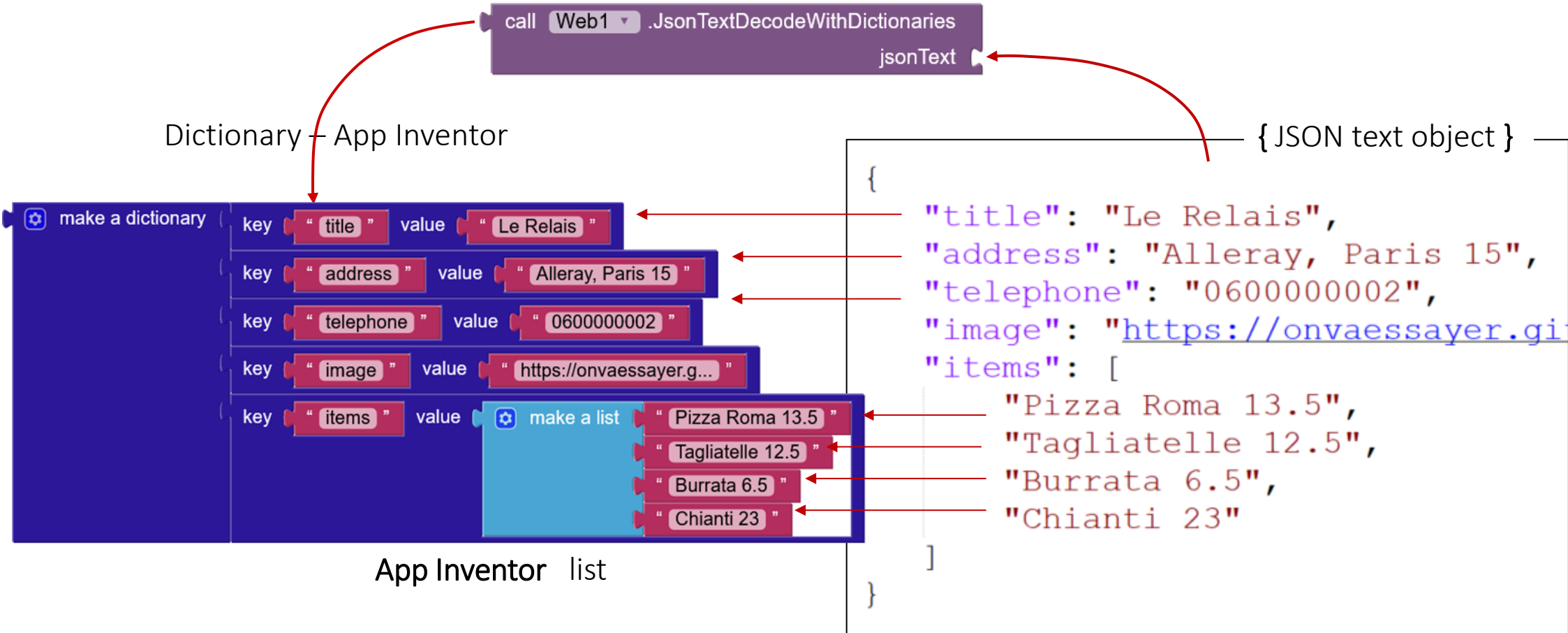
for each **item** in list **get global items**
do

select list item **list** **get global items**
index **1**

length of list **list** **get global items**

is a list? **thing** **get global items**

{ JSON OBJECT , LIST } → App Inventor DICTIONARY, LIST



GITSHARE2A : CONNECT (Screen1) & MAP (mapScreen)

Screen1 :

```
when Screen1 .Initialize
do
  open another screen with start value screenName mapScreen
  startValue " "

```

mapScreen :

```
initialize global URLgeoJSONCatalog to
  join " https://onvaessayer.github.io/ "
  " gitshareData 1/map.geojson "

when mapScreen .Initialize
do
  call Map1 .LoadFromURL
  url get global URLgeoJSONCatalog

when any Marker.Click
  component notAlreadyHandled
do
  set Marker. EnableInfobox of component get component to true
  initialize local shopURL to Marker. Description
  of component get component
  in
    open another screen with start value screenName shop
    startValue get shopURL

```

GITSHARE2A : DISPLAY RESTAURANT DATA (shop screen)

```
initialize global shop to create empty dictionary
initialize global items to create empty list
initialize global shopURL to ""

when Web1 .GotText
  url responseCode responseType responseContent
  do
    if responseCode = 200
      then
        set global shop to call Web1 .JsonTextDecodeWithDictionaries
          jsonText get responseContent
        if is a dictionary? get global shop
          then
            set LabelTitle .Text to get value for key "title" in dictionary get global shop or if not found "?"
            set LabelAddress .Text to get value for key "address" in dictionary get global shop or if not found "?"
            set Image1 .Picture to get value for key "image" in dictionary get global shop or if not found AndroidFR.png
            set global items to get value for key "items" in dictionary get global shop or if not found create empty list
            if is a list? thing get global items
              then
                set ListView1 .Elements to get global items
            else
              call Notifier1 .ShowMessag...
          else
            call Notifier1 .ShowMessag...
        else
          call Notifier1 .ShowMessag...

when shop .Initialize
  do
    set global shopURL to get start value
    set Web1 .Url to get global shopURL
    set Web1 .SaveResponse to false
    call Web1 .Get
```

3.3

Codage défensif , modèle objet des plats
adresses relatives, Dropbox& Google drive
gitshare2b

PLAN

- Introduction : décomposition de l'application
- 1. Création d'un site Web / serveur de données
- 2. Définition des données et préparation d'un jeu
- 3. Création de l'application mobile avec App Inventor
 - 1. V1 : visualiser le catalogue des restaurants sur une carte
 - 2. V2a : sélectionner un restaurant et afficher ses nom, adresse, image et liste de plats
 - 3. V2b : codage défensif, modèle de données, adresses relatives, Dropbox & Google Drive
 - 4. V3a : identifier et enregistrer l'utilisateur et la carte
 - 5. V3b : préparer et passer une commande
 - 6. V3c : bonus

CODAGE DÉFENSIF : SE PROTÉGER DES ERREURS

- L'application fonctionne avec les données fournies,
 - mais elle n'est pas protégée contre les erreurs possibles
 - elle n'est pas fiable
- Le programme doit se prémunir des erreurs
 - sur les données en entrée (créées et modifiées en permanence)
 - de fonctionnement (ex : pas d'accès à Internet)
 - des composants logiciels (ex : valeur ou format non pris en compte)
 - ...
- Nous examinerons ici des erreurs liées aux données de restaurants

3.3.1

Codage défensif

gitshare2b

PLAN

- Introduction : décomposition de l'application
- 1. Création d'un site Web / serveur de données
- 2. Définition des données et préparation d'un jeu
- 3. Création de l'application mobile avec App Inventor
 - 1. V1 : visualiser le catalogue des restaurants sur une carte
 - 2. V2a : sélectionner un restaurant et afficher ses nom, adresse, image et liste de plats
 - 3. V2b : codage défensif, modèle de données, adresses relatives, Dropbox & Google Drive
 - 4. V3a : identifier et enregistrer l'utilisateur et la carte
 - 5. V3c : préparer et passer une commande
 - 6. V3d : afficher les plats et partager les catalogues

TYPES D'ERREURS LIÉES AU FICHIER JSON DE CHAQUE RESTAURANT

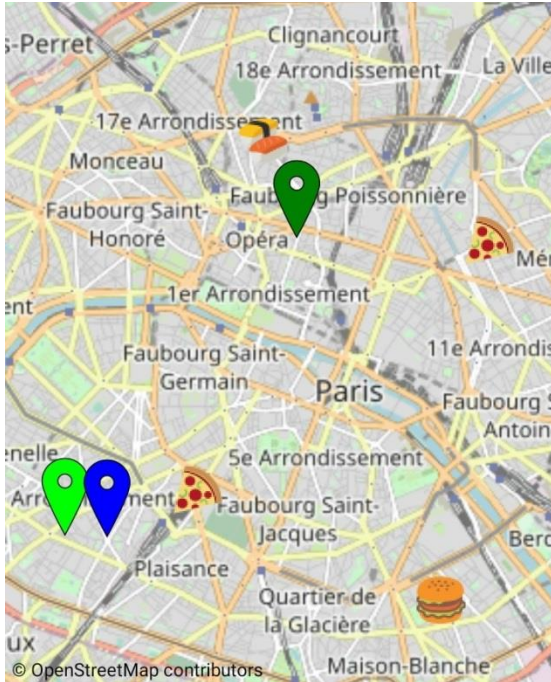
- Fichier non trouvé → response code = 404

Si lecture OK : response code =200

- contenu non conforme à JSON → Error 1105 : unable to decode text
→ traiter avec événement error.occured
- contenu ne correspond pas à un dictionnaire → tester dans le programme
- la clef items ne contient pas une liste → bad argument to Elements
→ à tester dans le programme
- un item n'est pas une chaine de caractères → hard crash
→ à tester dans le programme

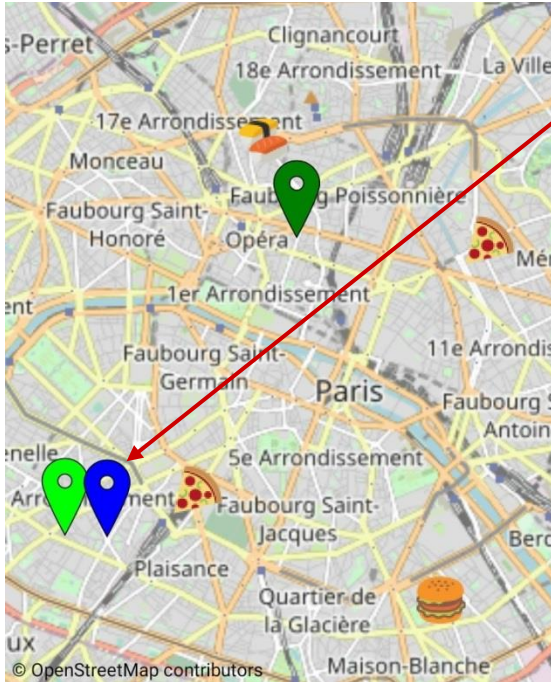
ERRORS / RESTAURANT JSON FILE

Catalogue geojson : <https://onvaessayer.github.io/gitshareErrors1/map.geojson>



ERR 1 : JSON FILE NOT FOUND / FICHIER NON TROUVÉ

Catalogue geojson : <https://onvaessayer.github.io/gitshareErrors1/map.geojson>

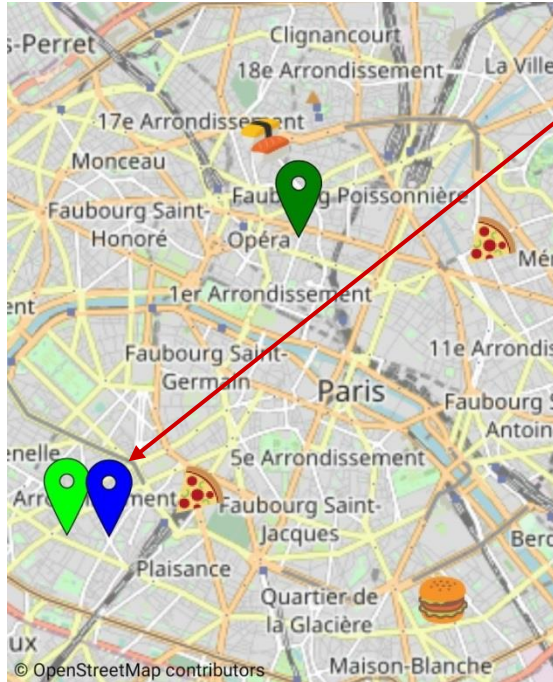


lerelais

problem : json file not found

ERR : JSON FILE NOT FOUND / FICHER NON TROUVÉ

Catalogue geojson : <https://onvaessayer.github.io/gitshareErrors1/map.geojson>



lerelais

problem : json file not found

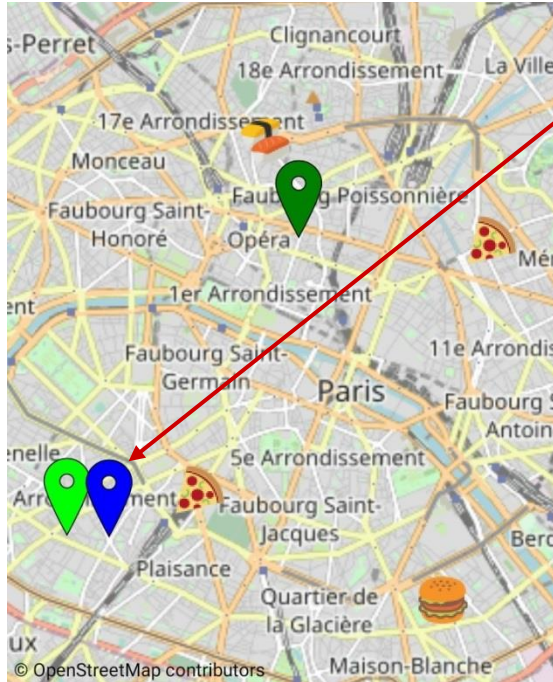
```
when Web1 .GotText
  url responseCode responseType responseContent
do
  if get responseCode = 200
  then
  else
```

Error message



ERR : JSON FILE NOT FOUND / FICHER NON TROUVÉ

Catalogue geojson : <https://onvaessayer.github.io/gitshareErrors1/map.geojson>



lerelais
problem : json file not found

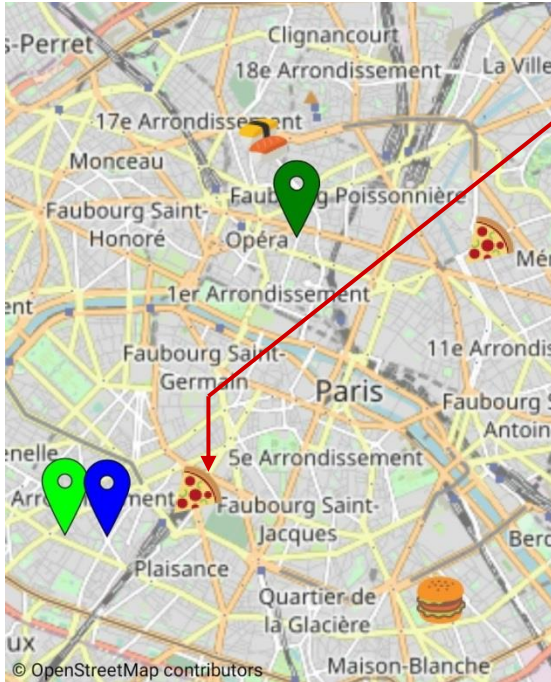
```
when Web1 .GotText
  url responseCode responseType responseContent
do
  if get responseCode = 200
  then
  else
    call Notifier1 .ShowMessageDialog
      message join join " Response code : " get responseCode
              join " <br> " get url
      title "shopURL"
      buttonText "OK"
```

Error message



ERR 2 : JSON FORMAT ERROR / ERREUR DE FORMAT

Catalogue geojson : <https://onvaessayer.github.io/gitshareErrors1/map.geojson>



Ristorante roma

problem : format error

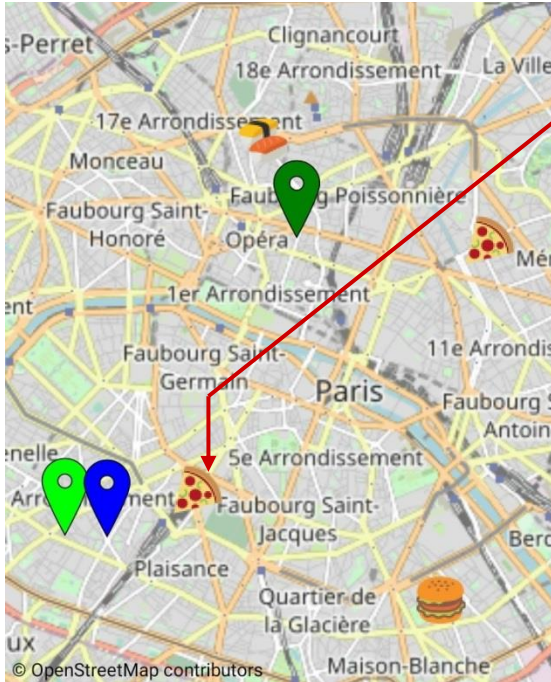
comma missing after email

virgule manquante après email

```
{  
  "title": "Le Roma",  
  "description": "Pizzeria",  
  "address": "avenue du Maine, Paris",  
  "image": "https://onvaessayer.github.io/gitshareGitDataset1/R",  
  "telephone": "0600000004",  
  "email": "ristoranteRoma@paris.fr"  
  "items": [  
    "pizza Margherita 8.40", "tagliatelles 7.90", "Chianti 13  
  ]  
}
```

ERR 2 : JSON FORMAT ERROR / ERREUR DE FORMAT

Catalogue geojson : <https://onvaessayer.github.io/gitshareErrors1/map.geojson>



Ristorante roma

problem : format error

comma missing after email

virgule manquante après email

```
{  
  "title": "Le Roma",  
  "description": "Pizzeria",  
  "address": "avenue du Maine, Paris",  
  "image": "https://onvaessayer.github.io",  
  "telephone": "0600000004",  
  "email": "ristoranteRoma@paris.fr"  
  "items": [  
    "pizza Margherita 8.40", "tagliatel  
  ]  
}
```

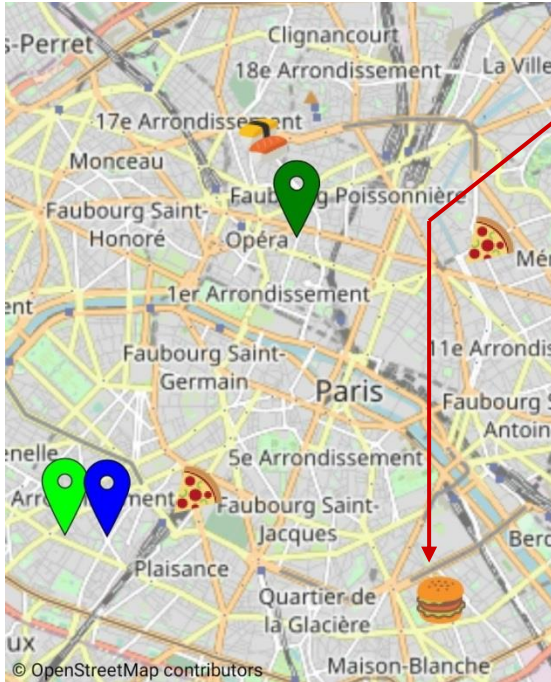
Error message

```
Error 1105: Unable to decode  
the JSON text: {  
  "title": "Indiana Cafe",  
  "description": "Restaurant  
américain",  
  "address": "rue de Tolbiac,  
75013 Paris",  
  "telephone": "0600000003",  
  "email": "indianaCafe@paris  
.fr",  
  "image":  
"https://onvaessayer.github.io  
/gitshareGitDataset1  
/IndianaCafe/image.jpg",  
  "items": [  
    "American burger : 12.5",  
    "veggie Salad 11.5",  
    "Bordeaux Haut de Lerm  
23"],  
}
```

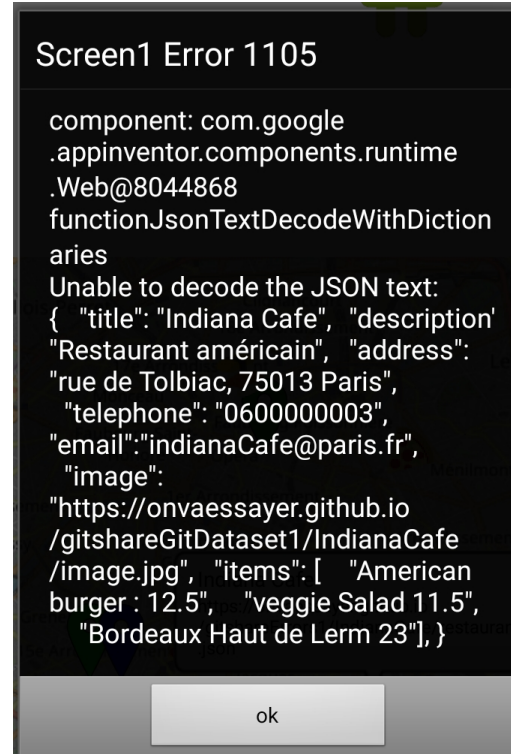
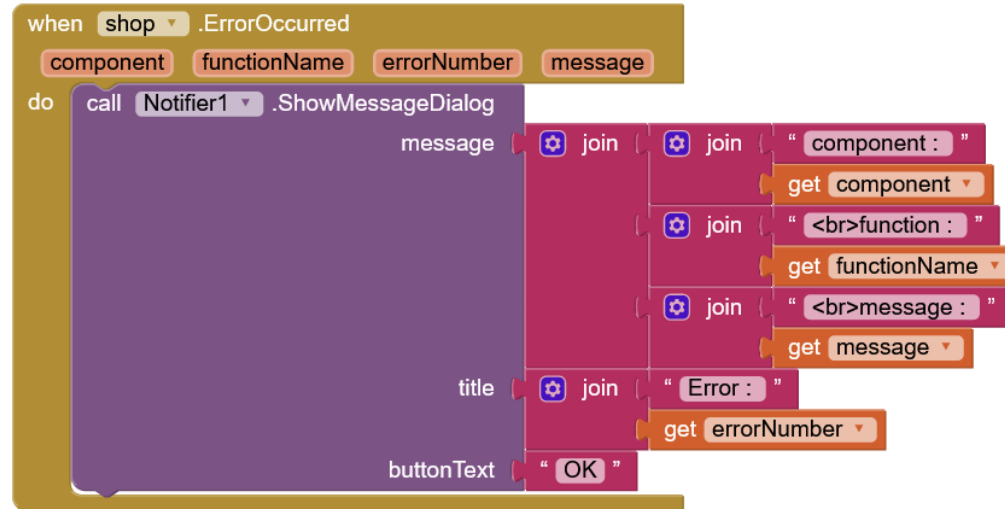
SCREEN1.ErrorOccured : CATCH ERRORS /ATTRAPER LES ERREURS

Catalogue geojson : <https://onvaessayer.github.io/gitshareErrors1/map.geojson>

Error message

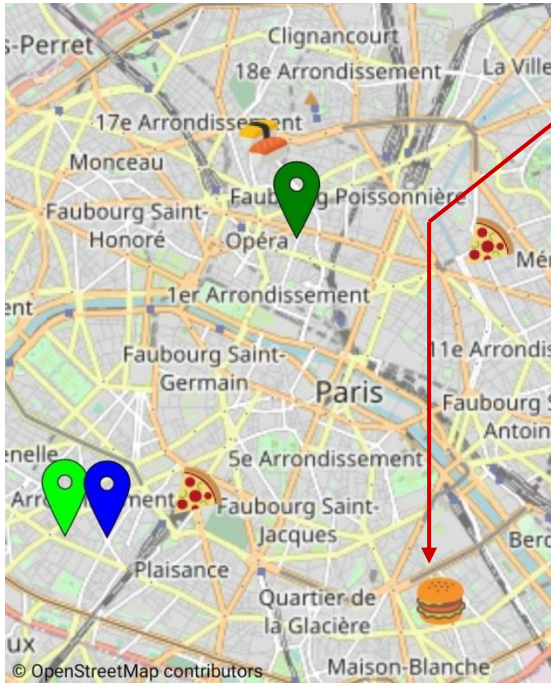


indianaCafe
problem : format error



ERR 3 : NOT A DICTIONARY / LE DÉCODAGE N'EST PAS UN DICTIONNAIRE

Catalogue geojson : <https://onvaessayer.github.io/gitshareErrors1/map.geojson>

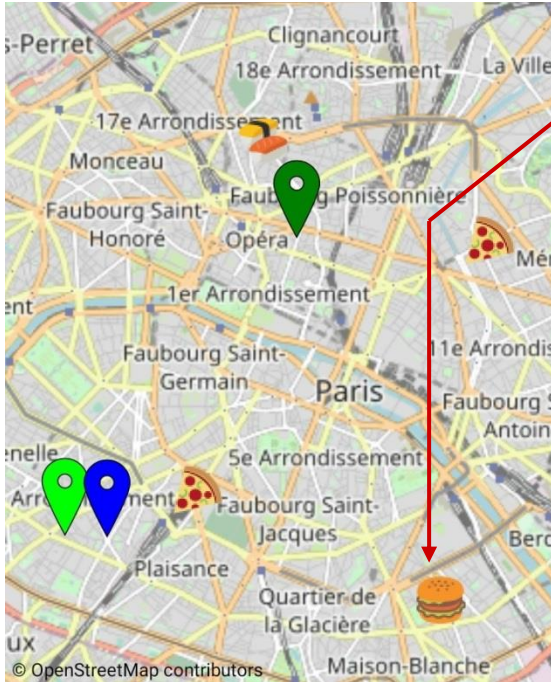


indianaCafe

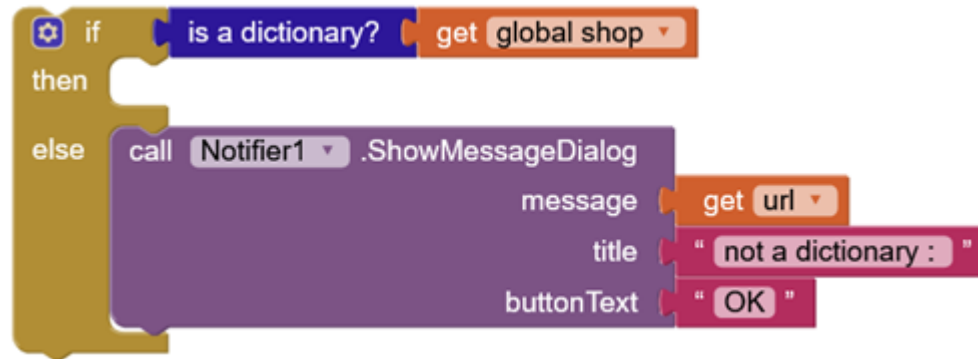
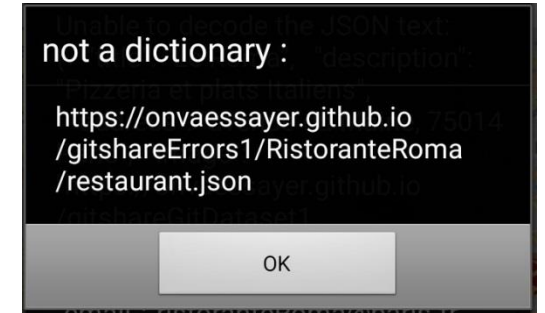
problem : format error

ERR 3 : NOT A DICTIONARY / LE DÉCODAGE N'EST PAS UN DICTIONNAIRE

Catalogue geojson : <https://onvaessayer.github.io/gitshareErrors1/map.geojson>



indianaCafe
problem : format error

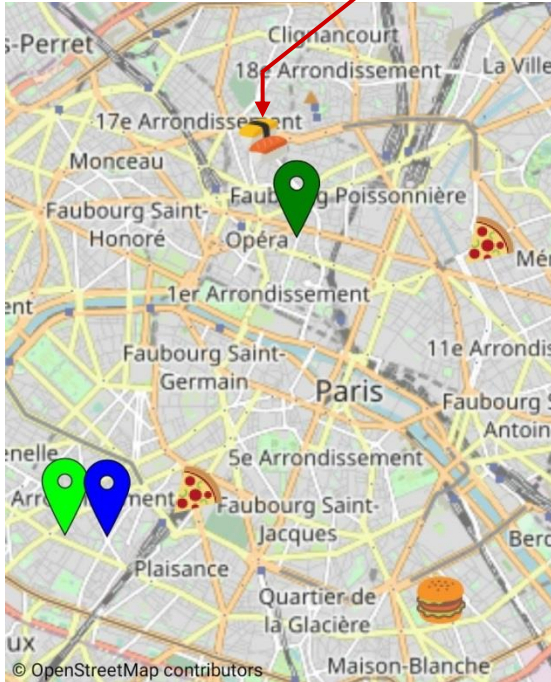


ERR 4 : ITEMS NOT A LIST / LES ITEMS NE SONT PAS UNE LISTE

Catalogue geojson : <https://onvaessayer.github.io/gitshareErrors1/map.geojson>

Momoka

problem : items n'est pas une liste



set ListView1 . Elements to get global items

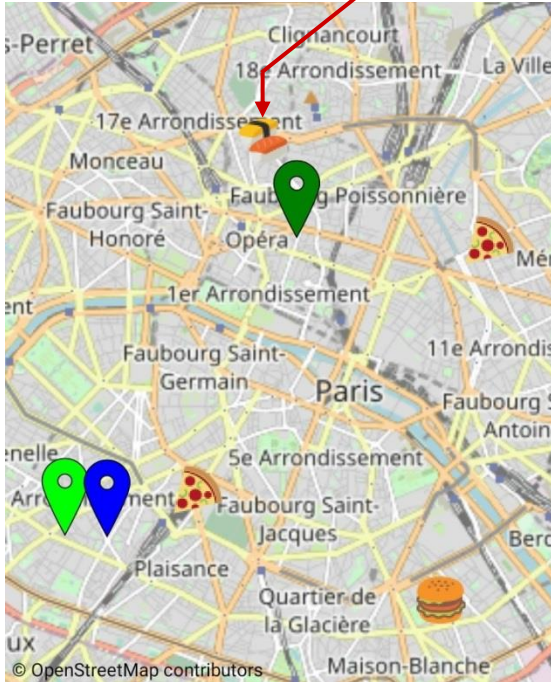
```
{  
  "title": "Momoka",  
  "description": "Restaurant Japonais",  
  "address": "rue Pigalle, 75009 Paris",  
  "telephone": "0600000005",  
  "image": "https://onvaessayer.github.io/gitshareGitDataset1/momoka",  
  "date": "2018 06 30",  
  "items": "item1, item2, ... as a string"  
}
```

ERR 4 : ITEMS NOT A LIST / LES ITEMS NE SONT PAS UNE LISTE

Catalogue geojson : <https://onvaessayer.github.io/gitshareErrors1/map.geojson>

Momoka

problem : items n'est pas une liste



Bad arguments to Elements

The operation Elements cannot accept the arguments: , ["item1, item2, ... as a string"]

End Application

```
{  
  "title": "Momoka",  
  "description": "Restaurant Japonais",  
  "address": "rue Pigalle, 75009 Paris",  
  "telephone": "0600000005",  
  "image": "https://onvaessayer.github.io/gitshareGitDataset1/momoka",  
  "date": "2018 06 30",  
  "items": "item1, item2, ... as a string"  
}
```

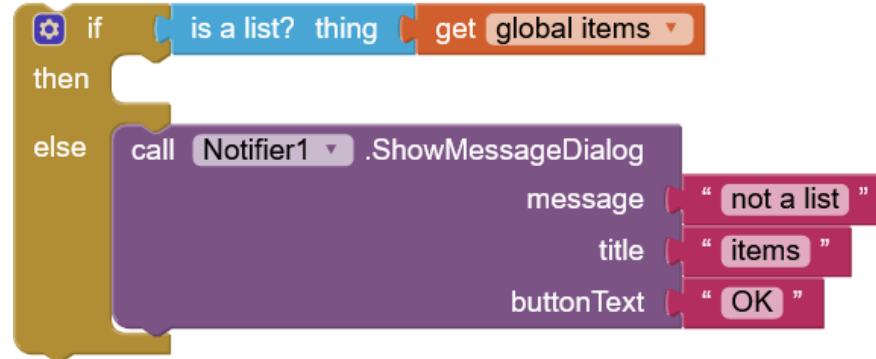
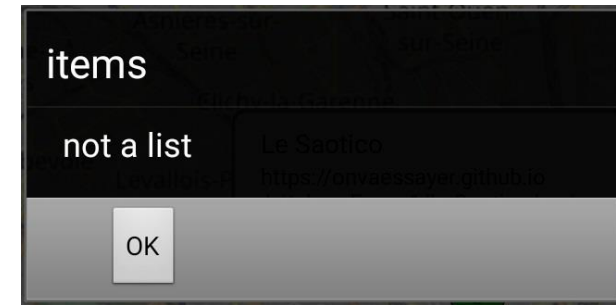
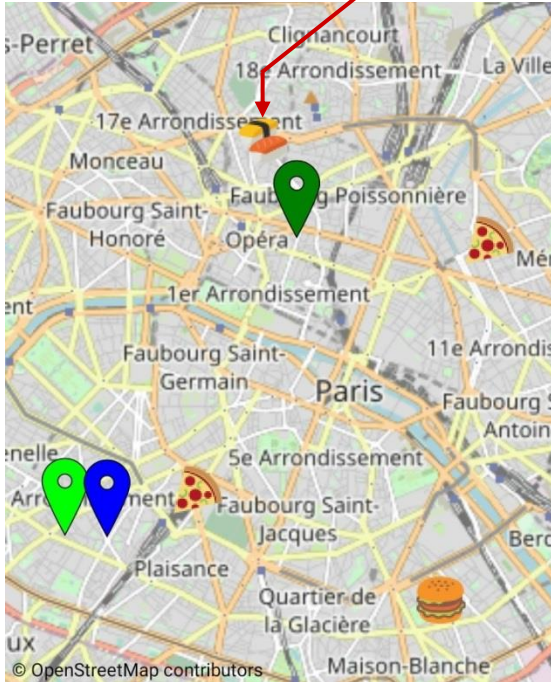
ERR 4 : ITEMS NOT A LIST / LES ITEMS NE SONT PAS UNE LISTE

Catalogue geojson : <https://onvaessayer.github.io/gitshareErrors1/map.geojson>

Error message

Momoka

problem : items n'est pas une liste

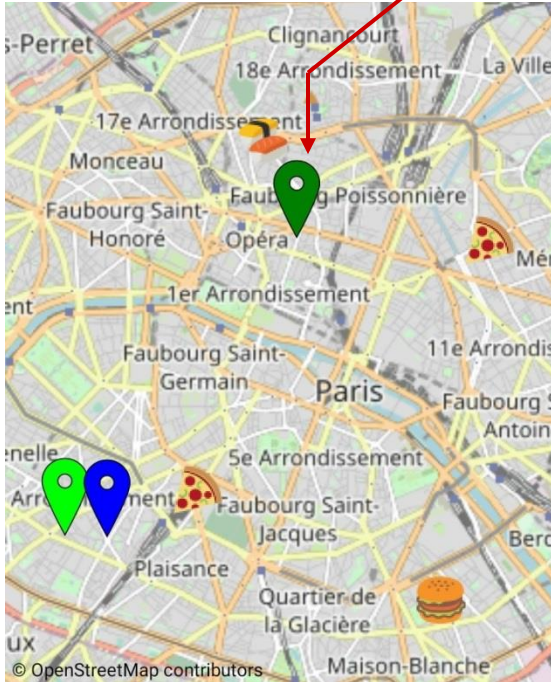


ERR 5 : ITEM NOT A STRING / N'EST PAS UNE CHAINE DE CARACTÈRES

Catalogue geojson : <https://onvaessayer.github.io/gitshareErrors1/map.geojson>

LeSaotico

problem : un item de la liste n'est pas une chaine de caractères



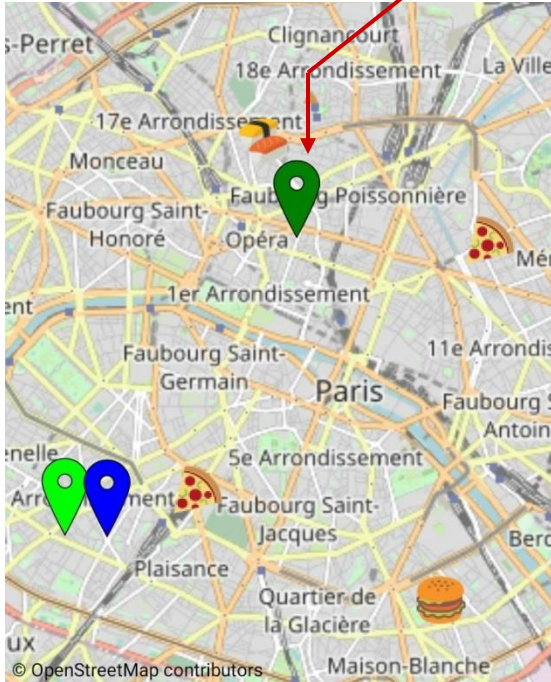
ERR 5 : ITEM NOT A STRING / N'EST PAS UNE CHAINE DE CARACTÈRES

Catalogue geojson : <https://onvaessayer.github.io/gitshareErrors1/map.geojson>

Error message

LeSaotico

problem : un item de la liste n'est pas une chaine de caractères



```
"items": [  
  {  
    "name": "bruchetta salade",  
    "price": 11.5,  
    "ingredients": ["salade",  
                   "tomate"],  
    "image": "bruchetta.PNG"  
  },  
  {  
    "name": "formule plat+dessert",  
    "price": 12.5,  
    "image": "formule.PNG"  
  }  
]
```

```
"items": [  
  "bruchetta et salade 11.5",  
  "formule plat+dessert 13.5"  
]
```

ERR 5 : ITEM NOT A STRING / N'EST PAS UNE CHAINE DE CARACTÈRES

Catalogue geojson : <https://onvaessayer.github.io/gitshareErrors1/map.geojson>

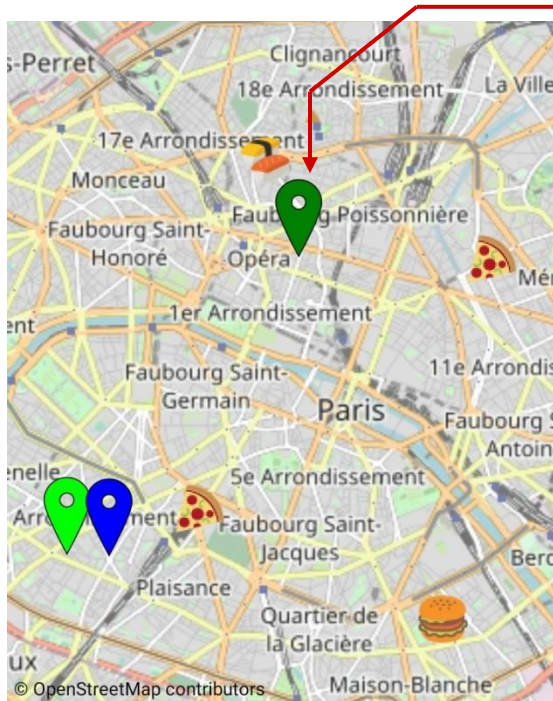
Error message

LeSaotico

problem : un item de la liste n'est pas une chaîne de caractères

```
set ListView1 . Elements to get global items
```

Application
crash,
no message



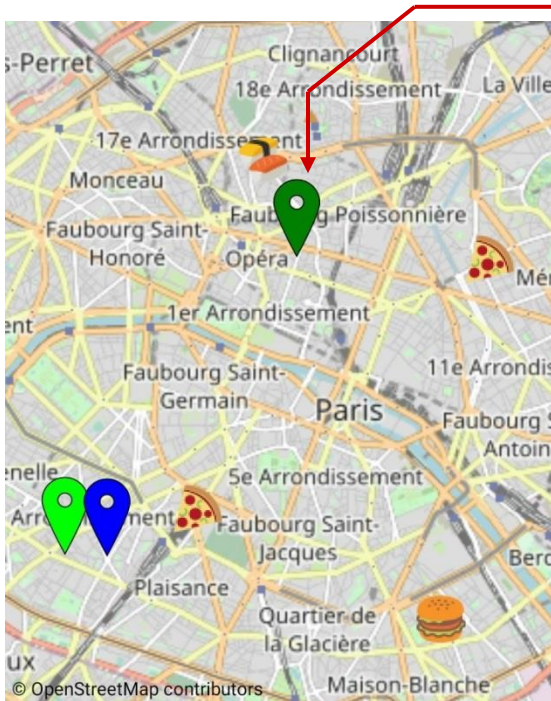
ERR 5 : ITEM NOT A STRING / N'EST PAS UNE CHAINE DE CARACTÈRES

Catalogue geojson : <https://onvaessayer.github.io/gitshareErrors1/map.geojson>

LeSaotico

problem : un item de la liste n'est pas
une chaine de caractères

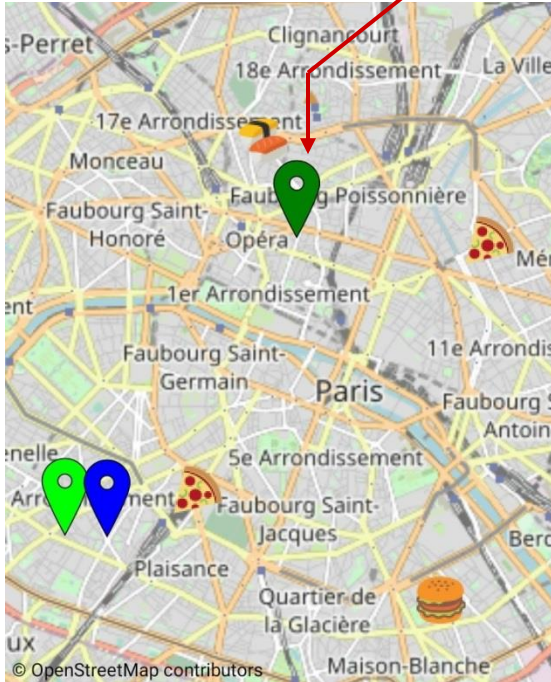
```
call setListviewElements myItems get global items
```



ERR 5 : ITEM NOT A STRING / N'EST PAS UNE CHAINE DE CARACTÈRES

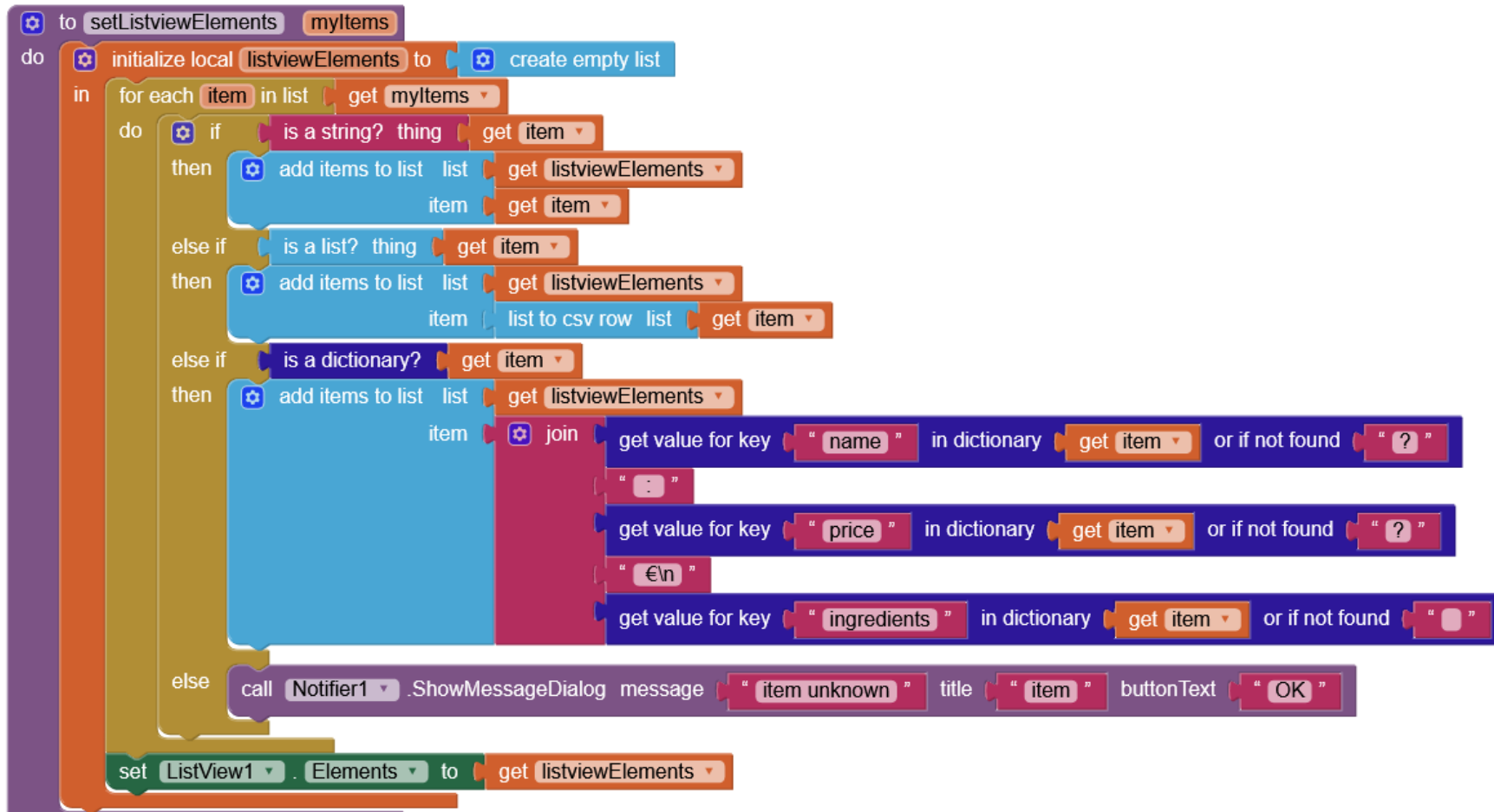
Catalogue geojson : <https://onvaessayer.github.io/gitshareErrors1/map.geojson>

LeSaotico



```
to setListViewElements myItems
do
  initialize local listViewElements to create empty list
  in for each item in list get myItems
  do
    if is a string? thing get item
    then
      add items to list list get listViewElements item get item
    else
      call Notifier1 .ShowMessageDialog
        message " an item is not a string "
        title " items "
        buttonText " OK "
  end
end
set ListView1 . Elements to get listViewElements
```

ERR 5 : ITEM NOT A STRING / N'EST PAS UNE CHAINE DE CARACTÈRES



```
to setListViewElements myItems
do
  initialize local listViewElements to create empty list
  in for each item in list get myItems
  do
    if is a string? thing get item
    then
      add items to list list get listViewElements
      item get item
    else if is a list? thing get item
    then
      add items to list list get listViewElements
      item list to csv row list get item
    else if is a dictionary? get item
    then
      add items to list list get listViewElements
      item
      join
      get value for key "name" in dictionary get item or if not found "?"
      " : "
      get value for key "price" in dictionary get item or if not found "?"
      " €\n "
      get value for key "ingredients" in dictionary get item or if not found " "
    else
      call Notifier1 . ShowMessageDialog message "item unknown" title "item" buttonText "OK"
  end
  set ListView1 . Elements to get listViewElements
end
```

ERR 5 : ITEM NOT A STRING / N'EST PAS UNE CHAINE DE CARACTÈRES

The image displays two Scratch code snippets. The left snippet, titled 'to setListviewElements myItems', is a function that iterates through a list of items. It uses conditional logic to handle different data types: strings, lists, and dictionaries. For strings and lists, it adds them directly to the listviewElements. For dictionaries, it joins the values for 'name', 'price', and 'ingredients' into a single string, separated by a dot, a newline, and a space. If an item is not a string, list, or dictionary, it shows a message dialog. The right snippet, titled 'when ListView1 AfterPicking', is an event handler that selects an item from the listviewElements based on the current selection index and sets the image of Image1 to the 'image' key of the selected dictionary item, or a blank image if not found.

```
to setListviewElements myItems
do
  initialize local listviewElements to create empty list
  in for each item in list get myItems
  do
    if is a string? thing get item
    then
      add items to list list get listviewElements
      item get item
    else if is a list? thing get item
    then
      add items to list list get listviewElements
      item list to csv row list get item
    else if is a dictionary? get item
    then
      add items to list list get listviewElements
      item
      join
      get value for key "name" in dictionary get item or if not found "?"
      "."
      get value for key "price" in dictionary get item or if not found "?"
      "€\n"
      get value for key "ingredients" in dictionary get item or if not found ""
    else
      call Notifier1.ShowDialog message "item unknown" title "item" buttonText "OK"
  set ListView1.Elements to get listviewElements
```

```
when ListView1 AfterPicking
do
  initialize local item to select list item list get global items
  index ListView1.SelectionIndex
  in
  if is a dictionary? get item
  then
    set Image1.Picture to get value for key "image"
    in dictionary get item
    or if not found ""
```

ERR 5 : ITEM NOT A STRING / N'EST PAS UNE CHAINE DE CARACTÈRES

The image displays a Scratch script for a Click&Collect app. The script is organized into several sections:

- Initialization:** Three global variables are initialized: 'shop' as an empty dictionary, 'items' as an empty list, and 'shopURL' as an empty string.
- Initialization Event:** A 'when shop Initialize' event triggers a 'do' block containing:
 - 'set global shopURL to get start value'
 - 'set Web1 . Url to get global shopURL'
 - 'set Web1 . SaveResponse to false'
 - 'call Web1 . Get'
- Web1 GotText Event:** A 'when Web1 GotText' event triggers a 'do' block:
 - 'if get responseCode = 200' then:
 - 'set global shop to call Web1 . JsonTextDecodeWithDictionaries' with 'jsonText' as 'get responseContent'.
 - 'if is a dictionary? get global shop' then:
 - 'set LabelTitle . Text to get value for key "title" in dictionary get global shop or if not found "?"'
 - 'set LabelAddress . Text to get value for key "address" in dictionary get global shop or if not found "?"'
 - 'set Image1 . Picture to get value for key "image" in dictionary get global shop or if not found AndroidFR.png'
 - 'set global items to get value for key "items" in dictionary get global shop or if not found create empty list'
 - 'if is a list? thing get global items' then:
 - 'call setListViewElements myItems get global items'
 - 'else call Notifier1 . ShowMessag...'
 - 'else call Notifier1 . ShowMessag...'
 - 'else call Notifier1 . ShowMessag...'

- Error Handling:** A 'when shop . ErrorOccurred' event triggers a 'do' block:
- 'call Notifier1 . ShowMessag...'
- ListView1 AfterPicking Event:** A 'when ListView1 . AfterPicking' event triggers a 'do' block:
- 'initialize local item to se...'
- ListView1 Elements:** A 'to setListViewElements myItems' block contains:
- 'do' block:
 - 'initialize local listViewElements to create empty list'
 - 'in for each item in list get m...'
 - 'set ListView1 . Elements to get listViewElements'

CODE gisthare 2a => 2b : traitement des erreurs

CODE gisthare 2a => 2b : traitement des erreurs

```
initialize global URLgeoJSONCatalog to join "https://onvaessayer.github.io/"
                                         "gistshareData1/map.geojson"

when Screen1.Initialize
do call Map1.LoadFromURL
   url get global URLgeoJSONCatalog

when any Marker.Click
component notAlreadyHandled
do set Marker.EnableInfoBox of component get component to true
   set Web1.Uri to Marker.Description of component get component
   set Web1.SaveResponse to false
   call Web1.Get

when Web1.GotText
url responseCode responseType responseContent
do if get responseCode = 200
   then set global shop to call Web1.JsonTextDecodeWithDictionaries jsonText get responseContent
        set LabelTitle.Text to get value for key "title" in dictionary get global shop or if not found "no title"
        set LabelAddress.Text to get value for key "address" in dictionary get global shop or if not found "no address"
        set Image1.Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
        set ListView1.Elements to get value for key "items" in dictionary get global shop or if not found create empty list
   else call Notifier1.ShowMessag...

initialize global shop to create empty dictionary

initialize global goodGeojson to make a list join "https://..."
initialize global badGeojson to make a list join "https://..."
```

CODE gisthare 2a => 2b : traitement des erreurs

The image displays several Scratch code blocks for an Android application, organized into four main sections:

- Initialization:** A block to initialize a global variable `URLgeoJSONCatalog` to a join of `https://onvaessayer.github.io/` and `gitshareData1/map.geojson`.
- Screen Initialization:** A `when Screen1.Initialize` block that calls `Map1.LoadFromURL` with the `URLgeoJSONCatalog` global.
- Marker Click:** A `when any Marker.Click` block that checks `notAlreadyHandled`, enables an info box, sets a web view's URL to the marker's description, sets `SaveResponse` to false, and calls `Web1.Get`.
- Error Handling:** A `when Screen1.ErrorOccurred` block that calls `Notifier1.ShowDialog` with a message containing component, function, and error details, and an "Error:" title.

Additional blocks include:

- Initializing a global `goodGeojson` to a list join of `https://...`.
- Initializing a global `badGeojson` to a list join of `https://...`.
- Initializing a global `shop` to an empty dictionary.
- A `when Web1.GotText` block that checks for a 200 response code. If successful, it decodes the JSON and sets UI elements (LabelTitle, LabelAddress, Image1, ListView1) based on dictionary keys. If not 200, it shows a message.

CODE gisthare 2a => 2b : traitement des erreurs

```
when Web1 . GotText
  initialize global shop to create empty dictionary
  url responseCode responseType responseContent
  do
    if get responseCode = 200
      then
        set global shop to call Web1 . JsonTextDecodeWithDictionaries jsonText get responseContent
        set LabelTitle . Text to get value for key "title" in dictionary get global shop or if not found "no title"
        set LabelAddress . Text to get value for key "address" in dictionary get global shop or if not found "no address"
        set Image1 . Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
        set ListView1 . Elements to get value for key "items" in dictionary get global shop or if not found create empty list
      else
        call Notifier1 . ShowMessag...
```

CODE gisthare 2a => 2b : traitement des erreurs

```
when Web1 GotText
  initialize global shop to create empty dictionary
  url responseCode responseType responseContent
  do
    if
      get responseCode = 200
    then
      set global shop to call Web1 .JsonTextDecodeWithDictionaries jsonText get responseContent
      set LabelTitle . Text to get value for key "title" in dictionary get global shop or if not found "no title"
      set LabelAddress . Text to get value for key "address" in dictionary get global shop or if not found "no address"
      set Image1 . Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
      set ListView1 . Elements to get value for key "items" in dictionary get global shop or if not found create empty list
    else
      call Notifier1 .ShowMessag...
```

CODE gisthare 2a => 2b : traitement des erreurs

```
when Web1 GotText
  initialize global shop to create empty dictionary
  url responseCode responseType responseContent
  do
    if
      get responseCode = 200
      then
        set global shop to call Web1 .JsonTextDecodeWithDictionaries jsonText get responseContent
      else
        if
          then
            set LabelTitle . Text to get value for key "title" in dictionary get global shop or if not found "no title"
            set LabelAddress . Text to get value for key "address" in dictionary get global shop or if not found "no address"
            set Image1 . Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
            set ListView1 . Elements to get value for key "items" in dictionary get global shop or if not found create empty list
          else
            call Notifier1 .ShowMessag...
```

CODE gisthare 2a => 2b : traitement des erreurs

```
when Web1 . GotText
  initialize global shop to create empty dictionary
  url responseCode responseType responseContent
do
  if
    get responseCode = 200
  then
    set global shop to call Web1 . JsonTextDecodeWithDictionaries jsonText get responseContent
    if not is a dictionary? get global shop
    then
    else
      set LabelTitle . Text to get value for key "title" in dictionary get global shop or if not found "no title"
      set LabelAddress . Text to get value for key "address" in dictionary get global shop or if not found "no address"
      set Image1 . Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
      set ListView1 . Elements to get value for key "items" in dictionary get global shop or if not found create empty list
    else
      call Notifier1 . ShowMessag...
```

CODE gisthare 2a => 2b : traitement des erreurs

```
when Web1 GotText
  initialize global shop to create empty dictionary
  url responseCode responseType responseContent
  do
    if
      get responseCode = 200
    then
      set global shop to call Web1.JsonTextDecodeWithDictionaries jsonText get responseContent
      if
        not is a dictionary? get global shop
      then
        call Notifier1.ShowAlert notice "shop is not a dictionary"
      else
        set LabelTitle.Text to get value for key "title" in dictionary get global shop or if not found "no title"
        set LabelAddress.Text to get value for key "address" in dictionary get global shop or if not found "no address"
        set Image1.Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
        set ListView1.Elements to get value for key "items" in dictionary get global shop or if not found create empty list
      else
        call Notifier1.ShowMessag...
```

CODE gisthare 2a => 2b : traitement des erreurs

```
when Web1 GotText
  uri
  responseCode
  responseType
  responseContent
do
  if
    get responseCode = 200
  then
    set global shop to call Web1.JsonTextDecodeWithDictionaries jsonText get responseContent
    if
      not is a dictionary? get global shop
    then
      call Notifier1.ShowAlert notice "shop is not a dictionary"
    else
      set LabelTitle.Text to get value for key "title" in dictionary get global shop or if not found "no title"
      set LabelAddress.Text to get value for key "address" in dictionary get global shop or if not found "no address"
      set Image1.Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
      set ListView1.Elements to get value for key "items" in dictionary get global shop or if not found create empty list
    end
  else
    call Notifier1.ShowMessag...
```

initialize global shop to create empty dictionary

initialize global items to create empty list

CODE gisthare 2a => 2b : traitement des erreurs

```
when Web1 GotText
  url
  statusCode
  responseType
  responseContent
do
  if statusCode = 200
  then
    set global shop to call Web1 .JsonTextDecodeWithDictionaries jsonText get responseContent
    if not is a dictionary? get global shop
    then
      call Notifier1 .ShowAlert notice "shop is not a dictionary"
    else
      set LabelTitle . Text to get value for key "title" in dictionary get global shop or if not found "no title"
      set LabelAddress . Text to get value for key "address" in dictionary get global shop or if not found "no address"
      set Image1 . Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
      set global items to
      set ListView1 . Elements to get value for key "items" in dictionary get global shop or if not found create empty list
  else
    call Notifier1 .ShowMessag...
```

CODE gisthare 2a => 2b : traitement des erreurs

The image shows a Scratch script for handling a web request. It starts with a 'when Web1 GotText' event. The script checks if the response code is 200. If not, it shows an alert. If yes, it decodes the JSON response into a dictionary. It then checks if the dictionary is a dictionary. If not, it shows an alert. If yes, it sets the title, address, and image of a label based on the dictionary values. It also sets a global list 'items' based on the dictionary value. Finally, it sets the elements of a list view.

```
when Web1 GotText
  uri
  responseCode
  responseType
  responseContent
do
  if
    get responseCode = 200
  then
    set global shop to call Web1 .JsonTextDecodeWithDictionaries jsonText get responseContent
    if
      not is a dictionary? get global shop
    then
      call Notifier1 .ShowAlert notice "shop is not a dictionary"
    else
      set LabelTitle . Text to get value for key "title" in dictionary get global shop or if not found "no title"
      set LabelAddress . Text to get value for key "address" in dictionary get global shop or if not found "no address"
      set Image1 . Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
      set global items to get value for key "items" in dictionary get global shop or if not found create empty list
      set ListView1 . Elements to
    else
      call Notifier1 .ShowMessag...
```

initialize global shop to create empty dictionary

initialize global items to create empty list

CODE gisthare 2a => 2b : traitement des erreurs

The image shows a Scratch script for handling a web request. It starts with a 'when Web1 GotText' event. The script checks if the response code is 200. If not, it shows an alert. If yes, it decodes the response content into a dictionary. It then checks if the dictionary is a dictionary. If not, it shows an alert. If yes, it sets the title, address, and image of a label, and sets the items of a list view. Finally, it sets the elements of a list view.

```
when Web1 GotText
  uri
  responseCode
  responseType
  responseContent
do
  if
    get responseCode = 200
  then
    set global shop to call Web1 .JsonTextDecodeWithDictionaries jsonText get responseContent
    if
      not is a dictionary? get global shop
    then
      call Notifier1 .ShowAlert notice "shop is not a dictionary"
    else
      set LabelTitle . Text to get value for key "title" in dictionary get global shop or if not found "no title"
      set LabelAddress . Text to get value for key "address" in dictionary get global shop or if not found "no address"
      set Image1 . Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
      set global items to get value for key "items" in dictionary get global shop or if not found create empty list
      set ListView1 . Elements to get global items
  else
    call Notifier1 .ShowMessag...
```

initialize global shop to create empty dictionary

initialize global items to create empty list

CODE gisthare 2a => 2b : traitement des erreurs

The image shows a Scratch script for handling a web request. It starts with a 'when Web1 GotText' event. The script checks the 'responseCode' and 'responseContent'. If the response code is 200, it decodes the JSON content into a dictionary for 'global shop'. It then checks if 'global shop' is a dictionary. If not, it shows an alert. If yes, it sets the title, address, and image of a label from the dictionary, and sets the items of a list view. If the response code is not 200, it shows a message.

```
when Web1 GotText
  url: responseCode, responseType, responseContent
  do
    if (get responseCode = 200)
      then
        set global shop to call Web1.JsonTextDecodeWithDictionaries jsonText: get responseContent
        if (not is a dictionary? get global shop)
          then
            call Notifier1.ShowAlert notice: "shop is not a dictionary"
          else
            set LabelTitle.Text to get value for key "title" in dictionary get global shop or if not found "no title"
            set LabelAddress.Text to get value for key "address" in dictionary get global shop or if not found "no address"
            set Image1.Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
            set global items to get value for key "items" in dictionary get global shop or if not found create empty list
            if
              then
                set ListView1.Elements to get global items
            else
            else
              call Notifier1.ShowMessag...
```

CODE gisthare 2a => 2b : traitement des erreurs

```
when Web1 . GotText
  uri responseCode responseType responseContent
do
  if
    get responseCode = 200
  then
    set global shop to call Web1 . JsonTextDecodeWithDictionaries jsonText get responseContent
    if not is a dictionary? get global shop
    then
      call Notifier1 . ShowAlert notice "shop is not a dictionary"
    else
      set LabelTitle . Text to get value for key "title" in dictionary get global shop or if not found "no title"
      set LabelAddress . Text to get value for key "address" in dictionary get global shop or if not found "no address"
      set Image1 . Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
      set global items to get value for key "items" in dictionary get global shop or if not found create empty list
      if is a list? thing get global items
      then
        set ListView1 . Elements to get global items
      else
    else
      call Notifier1 . ShowMessag...
```

CODE gisthare 2a => 2b : traitement des erreurs

```
when Web1 .GotText
  uri responseCode responseType responseContent
do
  if
    get responseCode = 200
  then
    set global shop to call Web1 .JsonTextDecodeWithDictionaries jsonText get responseContent
    if
      not is a dictionary? get global shop
    then
      call Notifier1 .ShowAlert notice "shop is not a dictionary"
    else
      set LabelTitle .Text to get value for key "title" in dictionary get global shop or if not found "no title"
      set LabelAddress .Text to get value for key "address" in dictionary get global shop or if not found "no address"
      set Image1 .Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
      set global items to get value for key "items" in dictionary get global shop or if not found create empty list
      if
        is a list? thing get global items
      then
        set ListView1 .Elements to get global items
      else
        call Notifier1 .ShowAlert notice "items is not a list"
    else
      call Notifier1 .ShowMessage...
  else
    call Notifier1 .ShowMessage...
```

initialize global shop to create empty dictionary

initialize global items to create empty list

CODE gisthare 2a => 2b : traitement des erreurs

```
when Web1 GotText
  uri responseCode responseType responseContent
do
  if (get responseCode = 200)
  then
    set global shop to call Web1 .JsonTextDecodeWithDictionaries jsonText get responseContent
    if (not is a dictionary? get global shop)
    then
      call Notifier1 .ShowAlert notice "shop is not a dictionary"
    else
      set LabelTitle . Text to get value for key "title" in dictionary get global shop or if not found "no title"
      set LabelAddress . Text to get value for key "address" in dictionary get global shop or if not found "no address"
      set Image1 . Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
      set global items to get value for key "items" in dictionary get global shop or if not found create empty list
      if (is a list? thing get global items)
      then
        set ListView1 . Elements to get global items
      else
        call Notifier1 .ShowAlert notice "items is not a list"
    else
      call Notifier1 .ShowMessage...

  initialize global shop to create empty dictionary
  initialize global items to create empty list
  to setListViewElements myItems
  do
```

CODE gisthare 2a => 2b : traitement des erreurs

The image shows a Scratch script for handling a web request. The script is organized into several nested blocks:

- when Web1 . GotText**: The main event trigger.
- url**: A block containing `responseCode`, `responseType`, and `responseContent`.
- do**: A large block containing the main logic:
 - if** `get responseCode = 200`:
 - then** `set global shop to call Web1 . JsonTextDecodeWithDictionaries jsonText get responseContent`
 - if** `not is a dictionary? get global shop`:
 - then** `call Notifier1 . showAlert notice "shop is not a dictionary"`
 - else**:
 - `set LabelTitle . Text to get value for key "title" in dictionary get global shop or if not found "no title"`
 - `set LabelAddress . Text to get value for key "address" in dictionary get global shop or if not found "no address"`
 - `set Image1 . Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"`
 - `set global items to get value for key "items" in dictionary get global shop or if not found create empty list`
 - if** `is a list? thing get global items`:
 - then** `call setListviewElements myItems get global items`
 - else** `call Notifier1 . showAlert notice "items is not a list"`
 - else** `call Notifier1 . ShowMessag...`

CODE gisthare 2a => 2b : traitement des erreurs

The image displays two Scratch code snippets. The left snippet (2a) shows a sequence of operations: a 'when Web1 GotText' event triggers a 'do' loop. Inside, it checks if 'responseCode' is 200. If not, it shows an alert 'shop is not a dictionary'. If yes, it sets 'global shop' to the decoded JSON. It then checks if 'shop' is a dictionary; if not, it shows an alert. If it is, it extracts 'title', 'address', and 'image' fields. It also checks if 'items' is a list; if not, it shows an alert. The right snippet (2b) shows a 'do' loop that initializes 'listviewElements' and iterates over 'myItems'. For each item, it checks if it's a string, a list, or a dictionary. If a string, it adds it to the list. If a list, it converts it to a CSV row. If a dictionary, it extracts the 'name' field. If none of these, it shows a message dialog 'item unknown' and breaks the loop. Finally, it sets 'ListView1 Elements' to 'listviewElements'.

```
when Web1 GotText
  url responseCode responseType responseContent
  do
    if get responseCode = 200
      then
        set global shop to call Web1 .JsonTextDecodeWithDictionaries jsonText get responseContent
        if not is a dictionary? get global shop
          then
            call Notifier1 .ShowAlert notice "shop is not a dictionary"
          else
            set LabelTitle . Text to get value for key "title" in dictionary get global shop or if not found "no title"
            set LabelAddress . Text to get value for key "address" in dictionary get global shop or if not found "no address"
            set Image1 . Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
            set global items to get value for key "items" in dictionary get global shop or if not found create empty list
            if is a list? thing get global items
              then
                call setListViewElements myItems get global items
              else
                call Notifier1 .ShowAlert notice "items is not a list"
            else
              call Notifier1 .ShowMessage...

initialize global shop to create empty dictionary
initialize global goodGeojson to make a list join "https://..."
initialize global items to create empty list
initialize global badGeojson to make a list join "https://..."

do
  to setListViewElements myItems
  do
    initialize local listviewElements to create empty list
    in for each item in list get myItems
    do
      if is a string? thing get item
        then
          add items to list list get listviewElements item get item
        else if is a list? thing get item
          then
            add items to list list get listviewElements item list to csv row list get item
        else if is a dictionary? get item
          then
            add items to list list get listviewElements
            item get value for key "name" in dictionary get item or if not found "?"
          else
            call Notifier1 .ShowMessageDialog
            message "item unknown"
            title "items"
            buttonText "OK"
            break
    set ListView1 . Elements to get listviewElements
```

CODE gisthare 2a => 2b : traitement des erreurs

The image displays several Scratch code blocks for an Android application, organized into four main sections:

- Initialization:** Two blocks at the top initialize global variables: `URLgeoJSONCatalogo` (pointing to a GitHub URL) and `goodGeojson` (pointing to a GeoJSON URL).
- Screen Initialization:** A `when Screen1.Initialize` block calls `Map1.LoadFromURL` with the `URLgeoJSONCatalogo` variable.
- Marker Click:** A `when any Marker.Click` block sets `Marker.EnableInfoBox` to true, updates `Web1.Url` with the marker's description, sets `Web1.SaveResponse` to false, and calls `Web1.Get`.
- Error Handling:** A `when Screen1.ErrorOccurred` block calls `Notifier1.ShowDialog` with a message containing the component name, function name, and error message, and a button labeled "OK".
- Web1.GetText:** A `when Web1.GetText` block checks the `responseCode`. If it's 200, it decodes the `responseContent` as JSON. It then checks if the result is a dictionary (for title and address) or a list (for items). If not a dictionary, it shows an alert "shop is not a dictionary". If not a list, it shows an alert "items is not a list".
- List Processing:** A `to setListViewElements myItems` block iterates through `myItems`. It checks if each item is a string, a list, or a dictionary. If a dictionary, it extracts the `name` and adds it to the list view. If not a dictionary, it shows an alert "item unknown".

3.3.2

Adresses relatives

gitshare2b

PLAN

- Introduction : décomposition de l'application
- 1. Création d'un site Web / serveur de données
- 2. Définition des données et préparation d'un jeu
- 3. Création de l'application mobile avec App Inventor
 - 1. V1 : visualiser le catalogue des restaurants sur une carte
 - 2. V2a : sélectionner un restaurant et afficher ses nom, adresse, image et liste de plats
 - 3. V2b : codage défensif, modèle de données, adresses relatives, Dropbox & Google Drive
 - 4. V3a : identifier et enregistrer l'utilisateur et la carte
 - 5. V3b : préparer et passer une commande
 - 6. V3c : bonus

ARBORESCENCE DES DONNÉES : CATALOGUE ET RESTAURANTS

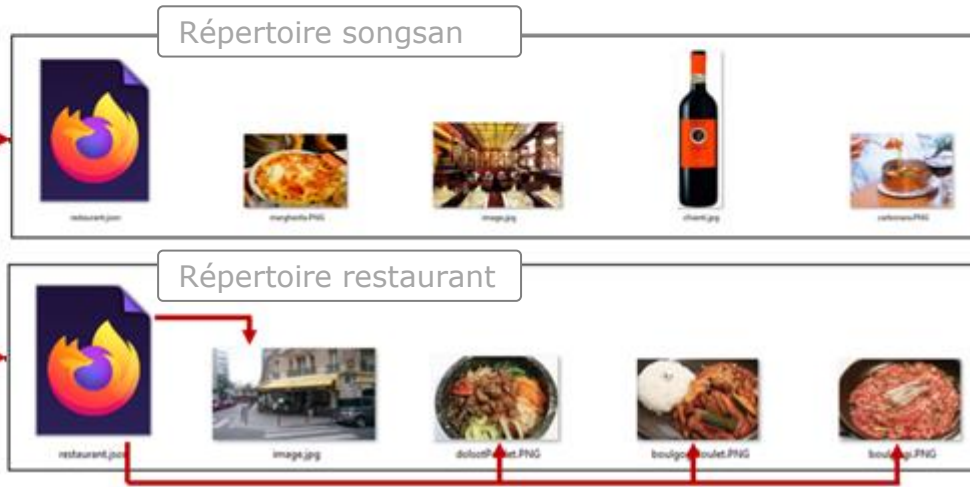


geoJSON catalog directory URL : <https://onvaessayer.Github.io/gitshareData1>



geoJSON catalog URL : <https://onvaessayer.Github.io/gitshareData1/map.geojson>

Restaurant JSON URL : <https://onvaessayer.Github.io/gitshareData1/songsan/restaurant.json>



Descriptif JSON d'un restaurant,
images du restaurant et des plats

Descriptif JSON d'un restaurant,
images du restaurant et des plats

ARBORESCENCE DES DONNÉES : CATALOGUE ET RESTAURANTS



geoJSON catalog directory URL : <https://onvaessayer.Github.io/gitshareData1>



geoJSON catalog URL : <https://onvaessayer.Github.io/gitshareData1/map.geojson>

Restaurant JSON URL : <https://onvaessayer.Github.io/gitshareData1/songsan/restaurant.json>

Restaurant File :

```
{
  "title": "SongSan",
  "description": "Restaurant Coréen",
  "address": "rue Marmontel",
  "telephone": "0600000005",
  "image": "https://onvaessayer.gitub.io/gitshareData1/image.jpg",
  "items": [
    {
      "name": "Dolsot poulet",
      "price": 12,
      "ingredients": ["poulet", "carottes", "soja"],
      "image": "https://onvaessayer.gitub.io/gitshareData1/dolsotPoulet.PNG"
    },
  ],
}
```

ARBORESCENCE DES DONNÉES : CATALOGUE ET RESTAURANTS



geoJSON catalog directory URL : <https://onvaessayer.Github.io/NEWDataDir>



geoJSON catalog URL : <https://onvaessayer.Github.io/NEWDataDir/map.geojson>

Restaurant JSON URL : <https://onvaessayer.Github.io/NEWDataDir/songsan/restaurant.json>

Restaurant File :

```
{
  "title": "SongSan",
  "description": "Restaurant Coréen",
  "address": "rue Marmontel",
  "telephone": "0600000005",
  "image": "https://onvaessayer.gitub.io/NEWDataDir/image.jpg",
  "items": [
    {
      "name": "Dolsot poulet",
      "price": 12,
      "ingredients": ["poulet", "carottes", "soja"],
      "image": "https://onvaessayer.gitub.io/NEWDataDir/dolsotPoulet.PNG"
    }
  ],
}
```

ARBORESCENCE DE DONNÉES : CATALOGUE ET RESTAURANTS



geoJSON catalog directory URL : <https://onvaessayer.Github.io/gitshareData1>



geoJSON catalog URL : <https://onvaessayer.Github.io/gitshareData1/map.geojson>

Relative address :

[songsan/restaurant.json](#)

Répertoire songsan



Descriptif JSON d'un restaurant,
images du restaurant et des plats

Répertoire restaurant



Descriptif JSON d'un restaurant,
images du restaurant et des plats

ARBORESCENCE DE DONNÉES : CATALOGUE ET RESTAURANTS



geoJSON catalog directory URL : <https://onvaessayer.Github.io/gitshareData1>



geoJSON catalog URL : <https://onvaessayer.Github.io/gitshareData1/map.geojson>

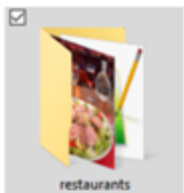
Relative address :

[songsan/restaurant.json](#)

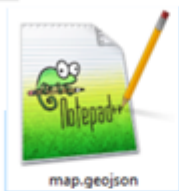
Restaurant File :

```
{
  "title": "SongSan",
  "description": "Restaurant Coréen",
  "address": "rue Marmontel",
  "telephone": "0600000005",
  "image": "image.jpg",
  "items": [
    {
      "name": "Dolsot poulet",
      "price": 12,
      "ingredients": ["poulet", "carottes", "soja"],
      "image": "dolsotPoulet.PNG"
    }
  ],
}
```

goodURL : RECONSTRUCTION DE L'ADRESSE ABSOLUE



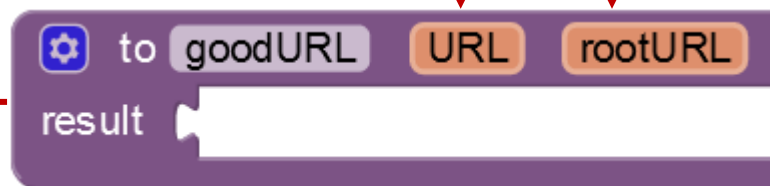
geoJSON catalog directory URL : <https://onvaessayer.Github.io/gitshareData1>



geoJSON catalog URL : <https://onvaessayer.Github.io/gitshareData1/map.geojson>

Relative address :

[songsan/restaurant.json](#)

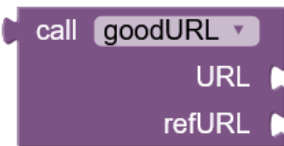


Restaurant JSON URL : <https://onvaessayer.Github.io/gitshareData1/songsan/restaurant.json>

RELATIVE → ABSOLUTE ADDRESS : GEOJSON CATALOG

refURL URLgeoJSONcatalog : <https://onvaessayer.github.io/gitshareData1/map.geojson>

absolute address



Relative address

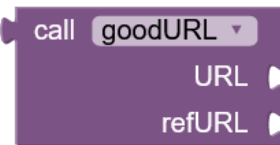
```
{  
  "type": "FeatureCollection",  
  "features": [  
    {  
      "type": "Feature",  
      "properties": {  
        "title": "Songsan",  
        "description": "songSan/restaurant.json",  
        "fill": "#00FF00", "image": "korean.png"  
      },  
      "geometry": {  
        "type": "Point",  
        "coordinates": [2.29935, 48.836747]  
      }  
    },  
  ],  
}
```

<https://onvaessayer.github.io/gitshareData1/songSan/restaurant.json>

RELATIVE → ABSOLUTE ADDRESS : RESTAURANT JSON FILE

refURL shopURL : <https://onvaessayer.github.io/gitshareData1/songSan/restaurant.json>

absolute address



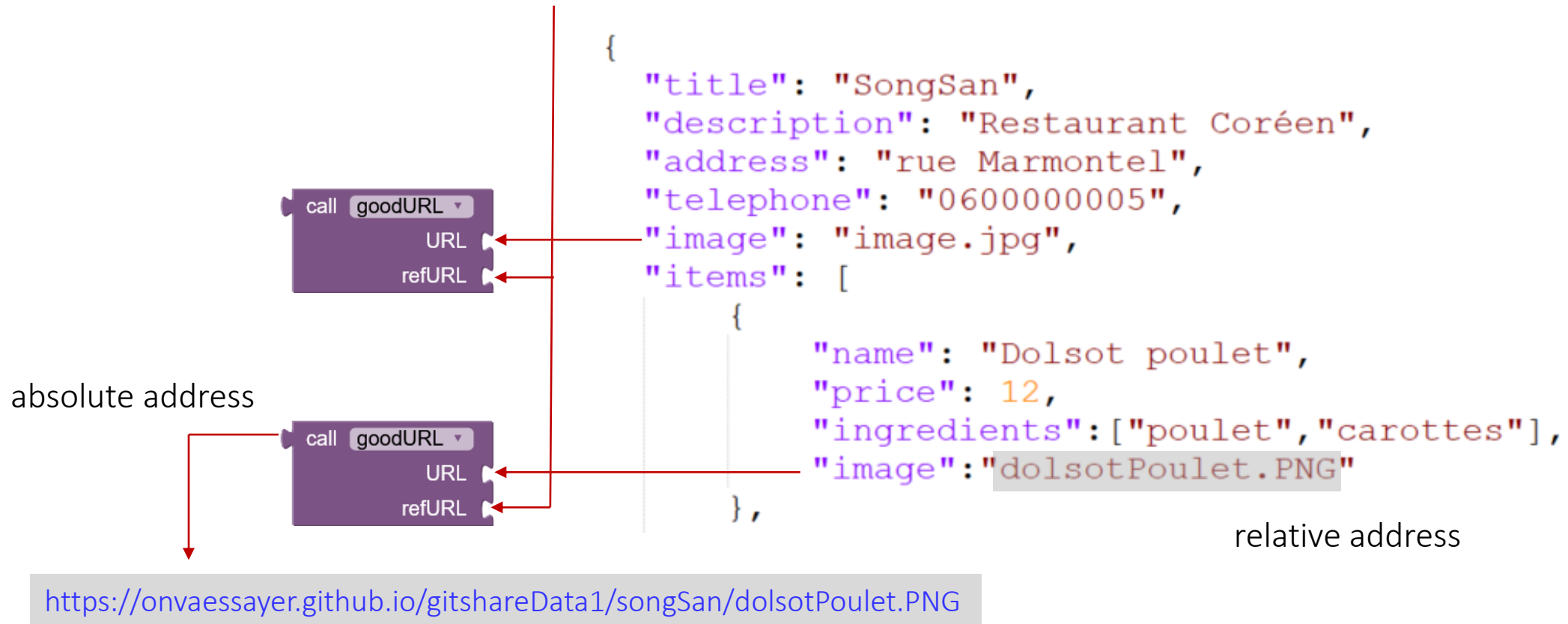
```
{
  "title": "SongSan",
  "description": "Restaurant Coréen",
  "address": "rue Marmontel",
  "telephone": "0600000005",
  "image": "image.jpg",
  "items": [
    {
      "name": "Dolsot poulet",
      "price": 12,
      "ingredients": ["poulet", "carottes"],
      "image": "dolsotPoulet.PNG"
    }
  ]
}
```

Relative address

<https://onvaessayer.github.io/gitshareData1/songSan/image.jpg>

RELATIVE → ABSOLUTE ADDRESS : RESTAURANT JSON FILE

refURL shopURL : <https://onvaessayer.github.io/gitshareData1/songSan/restaurant.json>



GEOJSON CATALOG : ABSOLUTE ADRESSES

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "title": "Songsan",
        "description": "https://onvaessayer.github.io/gitshareData1/songSan/restaurant.json",
        "infobox": true,
        "fill": "#00FF00",
        "image": "korean.png"
      },
      "geometry": {
        "type": "Point", "coordinates": [2.29935, 48.836747]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "title": "le relais",
        "description": "https://onvaessayer.github.io/gitshareData1/relaisDeLaPlace/restaurant.json",
        "infobox": true,
        "fill": "#0000FF",
        "image": "frenchFood.png"
      },
      "geometry": {
        "type": "Point", "coordinates": [2.306638, 48.836606]
      }
    }
  ],
}
```

GEOJSON CATALOG : RELATIVE ADRESSES

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "title": "Songsan",
        "description": "songSan/restaurant.json",
        "infobox": true,
        "fill": "#00FF00",
        "image": "korean.png"
      },
      "geometry": {
        "type": "Point", "coordinates": [2.29935, 48.836747]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "title": "le relais",
        "description": "relaisDeLaPlace/restaurant.json",
        "infobox": true,
        "fill": "#0000FF",
        "image": "frenchFood.png"
      },
      "geometry": {
        "type": "Point", "coordinates": [2.306638, 48.836606]
      }
    }
  ],
}
```

RESTAURANT JSON FILES : ABSOLUTE ADDRESSES

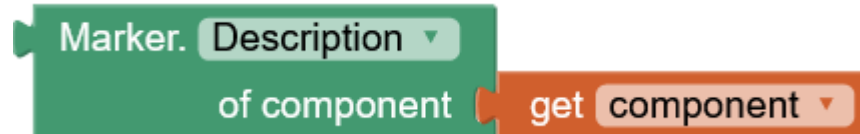
```
{
  "title": "SongSan",
  "description": "Restaurant Coréen",
  "address": "rue Marmontel",
  "telephone": "0600000005",
  "image": "https://onvaessayer.github.io/gitshareData2/songSan/image.jpg",
  "items": [
    {
      "name": "Dolsot poulet",
      "price": 12,
      "description": "",
      "ingredients": ["poulet", "carottes", "soja"],
      "image": "https://onvaessayer.github.io/gitshareData2/songSan/dolsotPoulet.PNG"
    },
    {
      "name": "Bulgogi",
      "price": 18,
      "description": "émincé de boeuf sauce soja",
      "ingredients": ["boeuf", "soja"],
      "image": "https://onvaessayer.github.io/gitshareData2/songSan/bulgogi.PNG"
    },
    {
      "name": "Bulgogi poulet",
      "price": 18,
      "description": "blanc de poulet, sauce soja",
      "ingredients": ["poulet", "soja"],
      "image": "https://onvaessayer.github.io/gitshareData2/songSan/bulgogiPoulet.PNG"
    }
  ]
}
```

RESTAURANT JSON FILES : RELATIVE ADRESSES

```
{
  "title": "SongSan",
  "description": "Restaurant Coréen",
  "address": "rue Marmontel",
  "telephone": "0600000005",
  "image": "image.jpg",
  "items": [
    {
      "name": "Dolsot poulet",
      "price": 12,
      "description": "",
      "ingredients": ["poulet", "carottes", "soja"],
      "image": "dolsotPoulet.PNG"
    },
    {
      "name": "Boulgogi",
      "price": 18,
      "description": "émincé de boeuf sauce soja",
      "ingredients": ["boeuf", "soja"],
      "image": "boulgogi.PNG"
    },
    {
      "name": "Boulgogi poulet",
      "price": 18,
      "description": "blanc de poulet, sauce soja",
      "ingredients": ["poulet", "soja"],
      "image": "boulgogiPoulet.PNG"
    }
  ]
}
```

CATALOGUE GEOJSON : ADRESSES RELATIVES

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "title": "Songsan",
        "description": "songSan/restaurant.json",
        "infobox": true,
        "fill": "#00FF00",
        "image": "korean.png"
      },
      "geometry": {
        "type": "Point", "coordinates": [2.29935, 48.836747]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "title": "le relais",
        "description": "relaisDeLaPlace/restaurant.json",
        "infobox": true,
```



CATALOGUE GEOJSON : ADRESSES RELATIVES

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "title": "Songsan",
        "description": "songSan/restaurant.json",
        "infobox": true,
        "fill": "#00FF00",
        "image": "korean.png"
      },
      "geometry": {
        "type": "Point", "coordinates": [
          12.566666666666667, 45.766666666666666
        ]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "title": "le relais",
        "description": "relaisDeLaPlace/restaurant.json",
        "infobox": true,

```

call goodURL

URL

Marker. Description of component

get component

refURL

get global URLgeoJSONCatalog

CATALOGUE GEOJSON : ADRESSES RELATIVES

Ref URL :

URLgeoJSONCatalog

<https://onvaessayer.github.io/gitshareData1/map.geojson>

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "title": "Songsan",
        "description": "songSan/restaurant.json",
        "infobox": true,
        "fill": "#00FF00",
        "image": "korean.png"
      },
      "geometry": {
        "type": "Point", "coordinates": [12.5625, 45.7625]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "title": "le relais",
        "description": "relaisDeLaPlace/restaurant.json",
        "infobox": true,

```

call goodURL

URL

Marker. Description

of component

get component

refURL

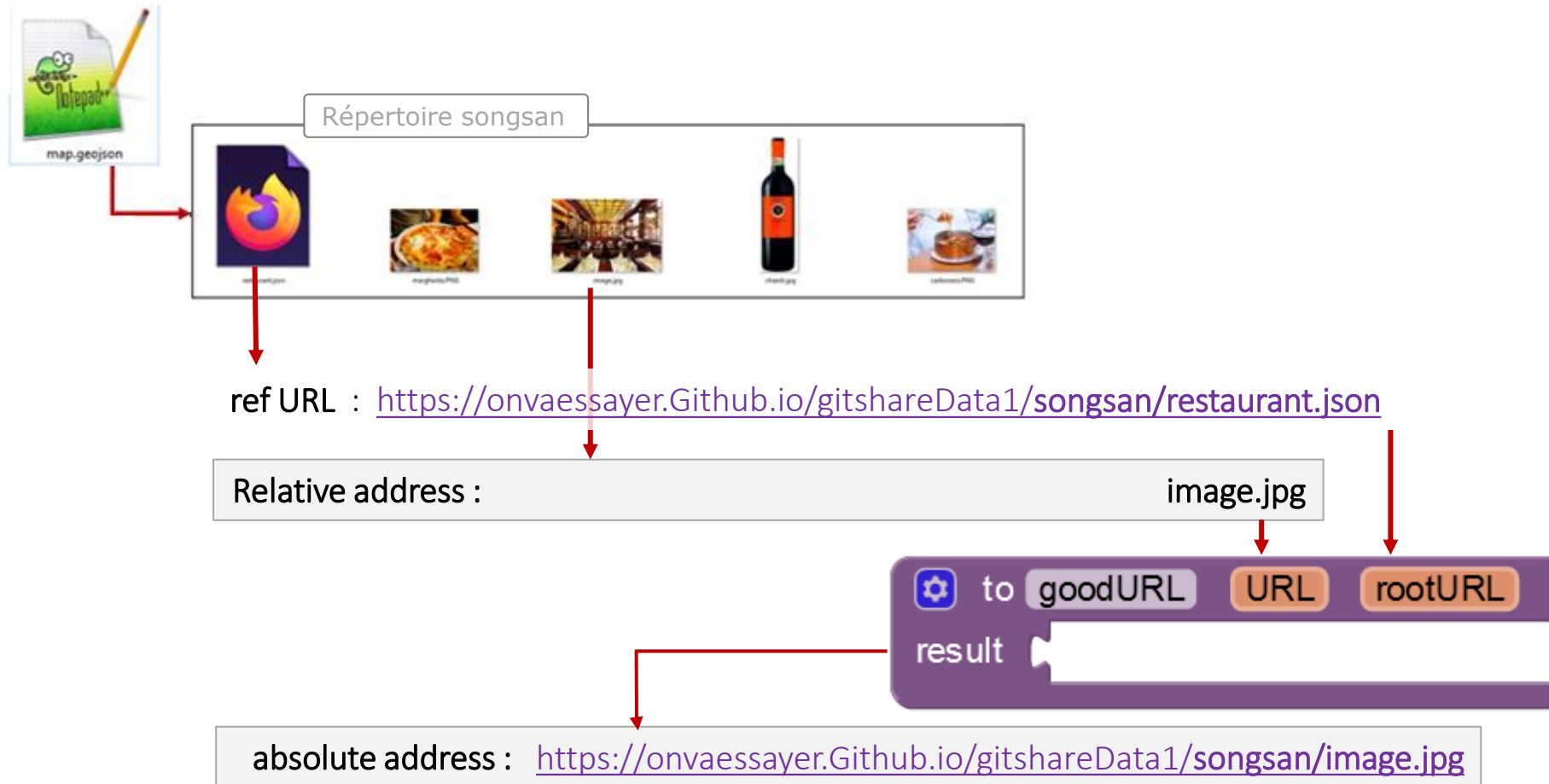
get global URLgeoJSONCatalog

CATALOGUE GEOJSON : ADRESSES ABSOLUES

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "title": "Songsan",
        "description": "https://onvaessayer.github.io/gitshareData1/songSan/restaurant.json",
        "infobox": true,
        "fill": "#00FF00",
        "image": "korean.png"
      },
      "geometry": {
        "type": "Point", "coordinates": [2.29935, 48.836747]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "title": "le relais",
        "description": "https://onvaessayer.github.io/gitshareData1/relaisDeLaPlace/restaurant.json",
        "infobox": true,

```

ARBORESCENCE DE DONNÉES : CATALOGUE ET RESTAURANTS



RESTAURANT AU FORMAT JSON : ADRESSE ABSOLUE

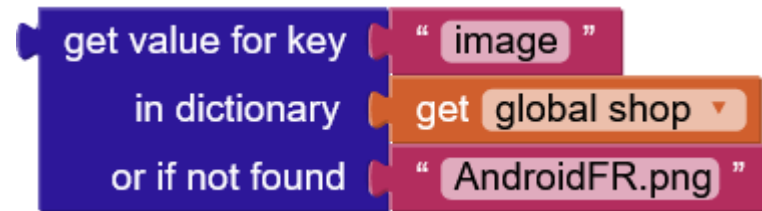
```
{  
  "title": "SongSan",  
  "description": "Restaurant Coréen",  
  "address": "rue Marmontel",  
  "telephone": "0600000005",  
  "image": "https://onvaessayer.github.io/gitshareData1/songSan/image.jpg",  
  "items": [  
    "Dolsot poulet 12",  
    "Bulgogi 18",  
    "Bulgogi poulet 18"  
  ]  
}
```

RESTAURANT AU FORMAT JSON : ADRESSE RELATIVE

```
{  
  "title": "SongSan",  
  "description": "Restaurant Coréen",  
  "address": "rue Marmontel",  
  "telephone": "0600000005",  
  "image": "image.jpg",  
  "items": [  
    "Dolsot poulet 12",  
    "Bulgogi 18",  
    "Bulgogi poulet 18"  
  ]  
}
```

RESTAURANT AU FORMAT JSON : ADRESSE RELATIVE

```
{  
  "title": "SongSan",  
  "description": "Restaurant Coréen",  
  "address": "rue Marmontel",  
  "telephone": "0600000005",  
  "image": "image.jpg",  
  "items": [  
    "Dolsot poulet 12",  
    "Bulgogi 18",  
    "Bulgogi poulet 18"  
  ]  
}
```

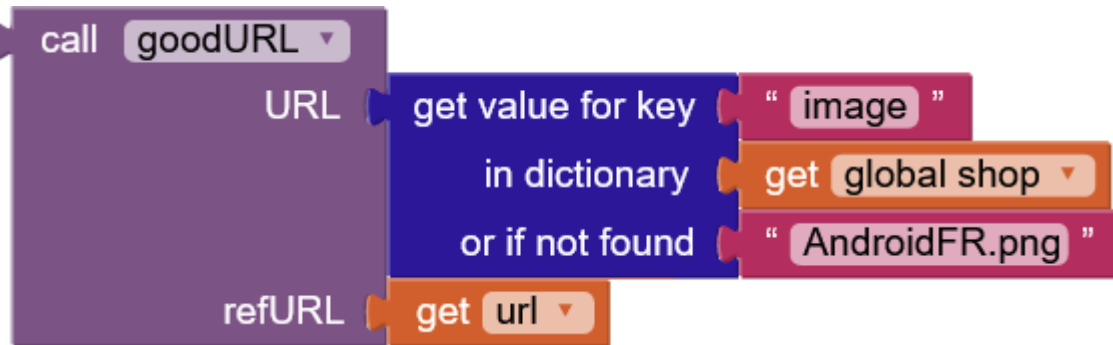


RESTAURANT AU FORMAT JSON : ADRESSE RELATIVE

Ref URL :

URL du restaurant <https://onvaessayer.github.io/gitshareData1/songSan/restaurant.json>

```
{  
  "title": "SongSan",  
  "description": "Restaurant Coréen",  
  "address": "rue Marmontel",  
  "telephone": "0600000005",  
  "image": "image.jpg",  
  "items": [  
    "Dolsot poulet 12",  
    "Bulgogi 18",  
    "Bulgogi poulet"  
  ]  
}
```

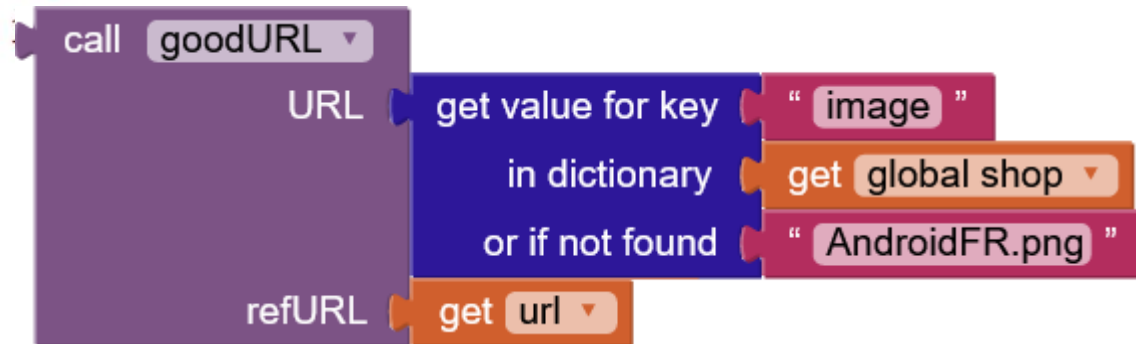


RESTAURANT AU FORMAT JSON : ADRESSE RELATIVE

Ref URL :

URL du restaurant <https://onvaessayer.github.io/gitshareData1/songSan/restaurant.json>

```
{  
  "title": "SongSan",  
  "description": "Restaurant Coréen",  
  "address": "rue Marmontel",  
  "telephone": "0600000005",  
  "image": "image.jpg",  
  "items": [  
    "Dolsot poulet 12",  
    "Bulgogi 18",  
    "Bulgogi poulet"  
  ]  
}
```



RESTAURANT AU FORMAT JSON : ADRESSE ABSOLUE

```
{  
  "title": "SongSan",  
  "description": "Restaurant Coréen",  
  "address": "rue Marmontel",  
  "telephone": "0600000005",  
  "image": "https://onvaessayer.github.io/gitshareData1/songSan/image.jpg",  
  "items": [  
    "Dolsot poulet 12",  
    "Bulgogi 18",  
    "Bulgogi poulet 18"  
  ]  
}
```

CODE gisthare 2a => 2b : adresses relatives /goodURL

CODE gisthare 2a => 2b : adresses relatives /goodURL

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/ ", "gitshareData1/map.geojson "]
```

```
when Screen1 .Initialize  
do call Map1 .LoadFromURL url get global URLgeoJSONCatalog
```

```
when any Marker.Click  
component notAlreadyHandled  
do set Marker. EnableInfobox of component get component to true  
set Web1 . Url to Marker. Description of component get component  
set Web1 . SaveResponse to false  
call Web1 .Get
```

```
when Web1 .GotText  
url responseCode responseType responseContent
```

```
initialize global shop to create empty dictionary
```

```
initialize global
```

```
initialize global
```

```
when Screen
```

```
component
```

```
do call Not
```

```
then set global shop to call Web1 .JsonTextDecodeWithDictionaries jsonText get responseContent
```

CODE gisthare 2a => 2b : adresses relatives /goodURL

```
initialize global URLgeoJSONCatalog to join [" https://onvaessayer.github.io/ ", " gitshareData1/map.geojson "
```

```
when Screen1 .Initialize  
do call Map1 .LoadFromURL url get global URLgeoJSONCatalog
```

```
when any Marker.Click  
component notAlreadyHandled  
do set Marker. EnableInfobox of component get component to true  
set Web1 . Url to Marker. Description of component get component  
set Web1 . SaveResponse to false  
call Web1 .Get
```

```
to goodURL URL refURL  
result
```

```
when Web1 .GotText  
url responseCode responseType responseContent
```

```
initialize global shop to create empty dictionary
```

```
initialize global
```

```
initialize global
```

```
when Screen  
component
```

```
do call Not
```

```
then set global shop to call Web1 .JsonTextDecodeWithDictionaries jsonText get responseContent
```

CODE gisthare 2a => 2b : adresses relatives /goodURL

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData1/map.geojson"]
```

```
when Screen1 .Initialize do call Map1 .LoadFromURL url get global URLgeoJSONCatalog
```

```
when any Marker.Click component notAlreadyHandled do set Marker. EnableInfoBox of component get component to true set Web1 . Url to Marker. Description of component set Web1 . SaveResponse to false call Web1 .Get
```

```
to goodURL URL refURL result
```

```
initialize global
```

```
initialize global
```

```
when Screen
```

```
component
```

```
do call Not
```

```
when Web1 .GotText url responseCode responseType responseContent
```

```
initialize global shop to create empty dictionary
```

```
then set global shop to call Web1 .JsonTextDecodeWithDictionaries jsonText get responseContent
```

CODE gisthare 2a => 2b : adresses relatives /goodURL

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/ ", "gitshareData1/map.geojson "]
```

```
when Screen1 .Initialize do call Map1 .LoadFromURL url get global URLgeoJSONCatalog
```

```
when any Marker.Click component notAlreadyHandled do set Marker. EnableInfobox of component get component to true set Web1 . Url to call goodURL URL Marker. Description of component get component refURL get global URLgeoJSONCatalog set Web1 . SaveResponse to false call Web1 .Get
```

```
to goodURL URL refURL result
```

```
when Web1 .GotText url responseCode responseType responseContent
```

```
initialize global shop to create empty dictionary
```

```
then set global shop to call Web1 .JsonTextDecodeWithDictionaries jsonText get responseContent
```

CODE gisthare 2a => 2b : adresses relatives /goodURL

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gistshareData1/map.geojson"]

when Screen1.Initialize
do
  call Map1.LoadFromURL url: get global URLgeoJSONCatalog

when any Marker.Click
  component notAlreadyHandled
do
  set Marker.EnableInfoBox of component get component to true
  set Web1.Url to call goodURL URL: Marker.Description of component refURL: get global URLgeoJSONCatalog
  set Web1.SaveResponse to false
  call Web1.Get
```

```
to goodURL URL refURL
result
```

```
when Web1.GetText
  url responseCode responseType responseContent
do
  if get responseCode = 200
  then
    set global shop to call Web1.JsonTextDecodeWithDictionaries jsonText: get responseContent
    if not is a dictionary? get global shop
    then
      call Notifier1.ShowAlert notice: "shop is not a dictionary"
    else
      set LabelTitle.Text to get value for key "title" in dictionary get global shop or if not found "no title"
      set LabelAddress.Text to get value for key "address" in dictionary get global shop or if not found "no address"
      set Image1.Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
      set global items to get value for key "items"
```

```
initialize global shop to create empty dictionary
```

```
initialize global items to create empty list
```

```
initialize global goodGeojson to make a list join "https://..."
initialize global badGeojson to make a list join "https://..."

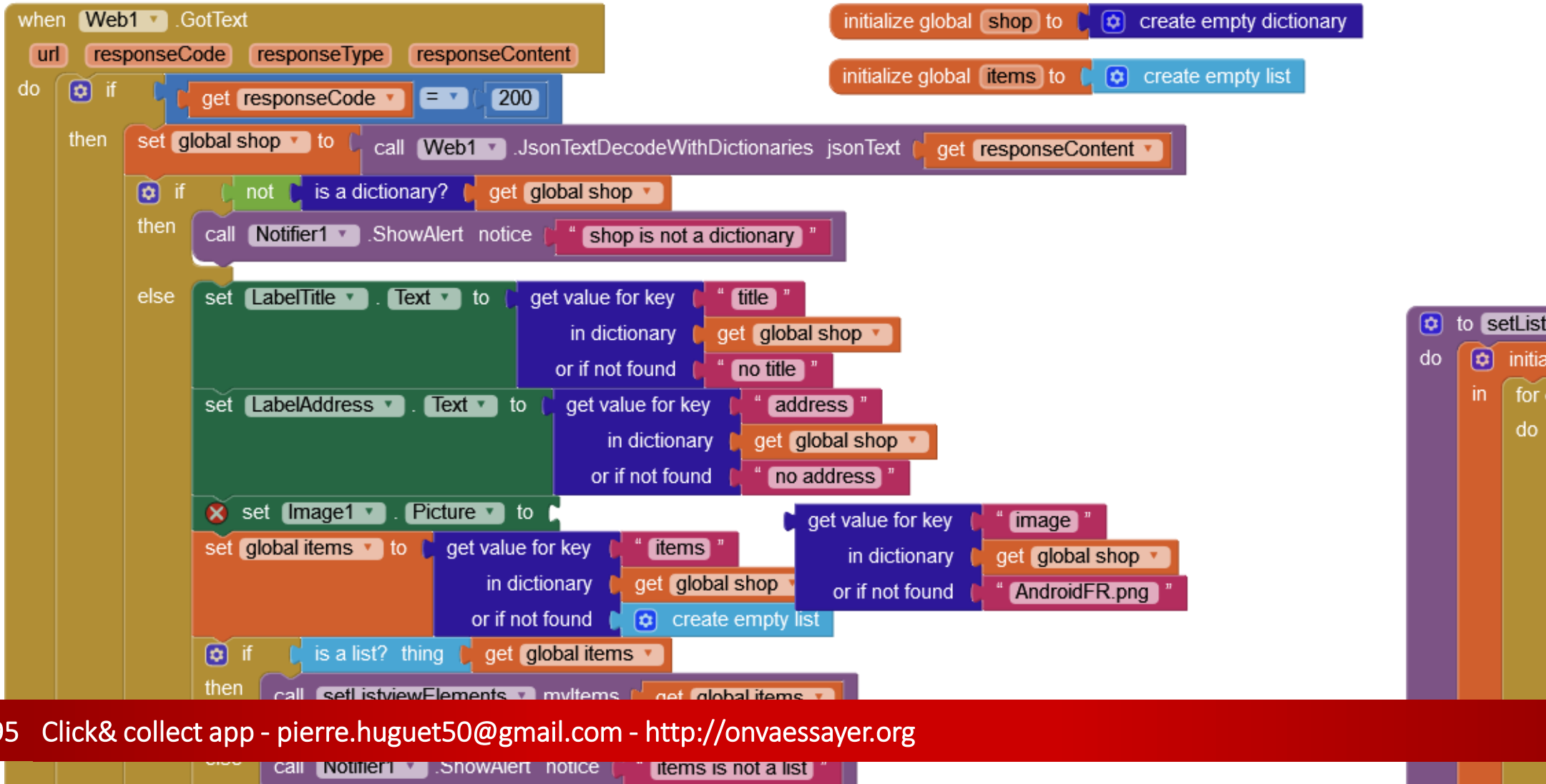
when Screen1.ErrorOccurred
  component functionName errorNumber message
do
  call Notifier1.ShowDialog message: join [component: get component "<br>", functionName: get functionName "<br>", message: "\n" get message "<br>"] title: "Error:" errorNumber: get errorNumber buttonText: "OK"
```

```
to setListViewElements myItems
do
  initialize local listViewElements to create empty list
  in for each item in list get myItems
  do
    if is a string? thing get item
    then
      add items to list list: get listViewElements item get item
    else if is a list? thing get item
    then
      add items to list list: get listViewElements item list to csv row list: get item
    else if is a dictionary? get item
```


CODE gisthare 2a => 2b : adresses relatives /goodURL

```
when Web1 GotText
  url
  responseCode
  responseType
  responseContent
do
  if
    get responseCode = 200
  then
    set global shop to call Web1.JsonTextDecodeWithDictionaries jsonText get responseContent
    if
      not is a dictionary? get global shop
    then
      call Notifier1.ShowAlert notice "shop is not a dictionary"
    else
      set LabelTitle.Text to get value for key "title" in dictionary get global shop or if not found "no title"
      set LabelAddress.Text to get value for key "address" in dictionary get global shop or if not found "no address"
      set Image1.Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
      set global items to get value for key "items" in dictionary get global shop or if not found create empty list
  initialize global shop to create empty dictionary
  initialize global items to create empty list
```

CODE gisthare 2a => 2b : adresses relatives /goodURL



The image shows a Scratch script for a web scraper. The main script is a 'when green flag clicked' block with a 'do' loop. Inside the loop, it checks if the 'responseCode' is 200. If yes, it sets a global variable 'shop' to the JSON response content. It then checks if 'shop' is a dictionary. If not, it shows an alert. If yes, it sets 'LabelTitle', 'LabelAddress', and 'Image1' to values from the dictionary. It also sets a global variable 'items' to the 'items' key in the dictionary. Finally, it checks if 'items' is a list and calls 'setListViewElements'.

Two separate initialization blocks are shown at the top right:

- initialize global **shop** to create empty dictionary
- initialize global **items** to create empty list

The main script blocks include:

- when **Web1** .GotText
- url, **responseCode**, **responseType**, **responseContent**
- do if (get **responseCode** = 200)
- then set **global shop** to call **Web1** .JsonTextDecodeWithDictionaries jsonText (get **responseContent**)
- if (not is a dictionary? get **global shop**)
- then call **Notifier1** .ShowAlert notice ("shop is not a dictionary")
- else set **LabelTitle** .Text to (get value for key "title" in dictionary get **global shop** or if not found "no title")
- set **LabelAddress** .Text to (get value for key "address" in dictionary get **global shop** or if not found "no address")
- set **Image1** .Picture to (get value for key "image" in dictionary get **global shop** or if not found "AndroidFR.png")
- set **global items** to (get value for key "items" in dictionary get **global shop** or if not found create empty list)
- if (is a list? thing get **global items**)
- then call **setListViewElements** myItems (get **global items**)

CODE gisthare 2a => 2b : adresses relatives /goodURL

```
when Web1 GotText
  url
  responseCode
  responseType
  responseContent
  do
    if
      get responseCode = 200
    then
      set global shop to call Web1.JsonTextDecodeWithDictionaries jsonText get responseContent
      if
        not is a dictionary? get global shop
      then
        call Notifier1.ShowAlert notice "shop is not a dictionary"
      else
        set LabelTitle.Text to get value for key "title" in dictionary get global shop or if not found "no title"
        set LabelAddress.Text to get value for key "address" in dictionary get global shop or if not found "no address"
        set Image1.Picture to call goodURL
          URL get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
          refURL get url
        set global items to get value for key "items"
```

initialize global shop to create empty dictionary

initialize global items to create empty list

CODE gisthare 2a => 2b : adresses relatives /goodURL

```
initialize global URLgeoJSONCatalog to join "https://onvaessayer.github.io/"  
"gitshareData1/map.geojson"
```

```
when Screen1.Initialize  
do call Map1.LoadFromURL  
url get global URLgeoJSONCatalog
```

```
when any Marker.Click  
component notAlreadyHandled  
do set Marker.EnableInfoBox of component get component to true  
set Web1.Uri to call goodURL URL Marker.Description  
of component get component  
refURL get global URLgeoJSONCatalog  
set Web1.SaveResponse to false  
call Web1.Get
```

```
to goodURL URL refURL  
result
```

```
initialize global goodGeojson to make a list join "https://..."
```

```
initialize global badGeojson to make a list join "https://..."
```

```
when Screen1.ErrorOccurred  
component functionName errorNumber message  
do call Notifier1.ShowDialog  
message join "component:" get component "<br>"  
join "function:" get functionName "<br>"  
join "message:" "\n" get message "<br>"  
title join "Error:" get errorNumber  
buttonText "OK"
```

```
when Web1.GotText  
initialize global shop to create empty dictionary
```

```
url responseCode responseType responseContent  
do if get responseCode = 200  
then set global shop to call Web1.JsonTextDecodeWithDictionaries jsonText get responseContent  
if not is a dictionary? get global shop  
then call Notifier1.ShowAlert notice "shop is not a dictionary"  
else set Label1.title .Text to get value for key "title"  
in dictionary get global shop  
or if not found "no title"  
set Label1.Address .Text to get value for key "address"  
in dictionary get global shop  
or if not found "no address"  
set Image1.Picture to call goodURL URL get value for key "image"  
in dictionary get global shop  
or if not found "AndroidIFR.png"  
refURL get uri  
set global items to get value for key "items"  
in dictionary get global shop  
or if not found create empty list  
if is a list? thing get global items  
then call setListViewElements myItems get global items  
else call Notifier1.ShowAlert notice "items is not a list"  
else call Notifier1.ShowMessage
```

```
initialize global items to create empty list
```

```
to getListViewElements myItems  
do initialize local listViewElements to create empty list  
in for each item in list get myItems  
do if is a string? thing get item  
then add items to list list get listViewElements item get item  
else if is a list? thing get item  
then add items to list list get listViewElements item list to csv row list get item  
else if is a dictionary? get item  
then add items to list list get listViewElements  
item get value for key "name"  
in dictionary get item  
or if not found "?"  
else call Notifier1.ShowDialog message "item unknown" title "items" buttonText "OK"  
break  
set ListView1.Elements to get listViewElements
```

CODE gisthare 2a => 2b : adresses relatives /goodURL

```
initialize global URLgeoJSONCatalog to join "https://onvaessayer.github.io/"
                                         "gitshareData1/map.geojson"

when Screen1.Initialize
do call Map1.LoadFromURL
   url get global URLgeoJSONCatalog

when any Marker.Click
component notAlreadyHandled
do set Marker.EnableInfoBox of component get component to true
   set Web1.Uri to call goodURL
   URL Marker.Description of component get component
   refURL get global URLgeoJSONCatalog
   set Web1.SaveResponse to false
   call Web1.Get

to goodURL URL refURL
result
```

```
initialize global goodGeojson to make a list join "https://..."
initialize global badGeojson to make a list join "https://..."

when Screen1.ErrorOccurred
component functionName errorNumber message
do call Notifier1.ShowMessage...
```

```
when Web1.GotText
url responseCode responseType responseContent
do if get responseCode = 200
   then set global shop to call Web1.JsonTextDecodeWithDictionaries jsonText get responseContent
        if not is a dictionary? get global shop
        then call Notifier1.ShowAlert notice "shop is not a dictionary"
        else set Label1.title .Text to get value for key "title"
              in dictionary get global shop
              or if not found "no title"
              set Label1.address .Text to get value for key "address"
              in dictionary get global shop
              or if not found "no address"
              set Image1.Picture to call goodURL
              URL get value for key "image"
              in dictionary get global shop
              or if not found "AndroidFR.png"
              refURL get uri
              set global items to get value for key "items"
              in dictionary get global shop
              or if not found create empty list
              if is a list? thing get global items
              then call setListViewElements myItems get global items
              else call Notifier1.ShowAlert notice "items is not a list"
        else call Notifier1.ShowMessage...
```

```
initialize global shop to create empty dictionary
initialize global items to create empty list

to setListViewElements myItems
do initialize local listViewElements to create empty list
   in for each item in list get m...
   set ListView1.Elements to get listViewElements
```

CODE gisthare 2a => 2b : adresses relatives /goodURL

initialize global URLgeoJSONCatalog to join <https://onvaessayer.github.io/>
gitshareData1/map.geojson

when Screen1.Initialize
do call Map1.LoadFromURL
url get global URLgeoJSONCatalog

when any Marker.Click
component notAlreadyHandled
do set Marker.EnableInfoBox of component get component to true
set Web1.Uri to Marker.Description of component get component
set Web1.SaveResponse to false
call Web1.Get

when Screen1.ErrorOccurred
component functionName errorNumber message
do call Notifier1.ShowMessage...
to goodURL URL refURL
result

initialize global goodGeojson to make a list join https://...
initialize global badGeojson to make a list join https://...

when Web1.GetText
url responseCode responseType responseContent
do if get responseCode = 200
then set global shop to call Web1.JsonTextDecodeWithDictionaries jsonText get responseContent
if not is a dictionary? get global shop
then call Notifier1.ShowAlert notice "shop is not a dictionary"
else set LabelTitle Text to get value for key title in dictionary get global shop or if not found no title
set LabelAddress Text to get value for key address in dictionary get global shop or if not found no address
set Image1.Picture to get value for key image in dictionary get global shop or if not found AndroidFR.png
set global items to get value for key items in dictionary get global shop or if not found create empty list
if is a list? thing get global items
then call setListViewElements myItems get global items
else call Notifier1.ShowAlert notice "items is not a list"
else call Notifier1.ShowMessage...
to setListViewElements myItems
do initialize local listViewElements to create empty list
in for each item in list get m...
set ListView1.Elements to get listViewElements

CODE gisthare 2a => 2b : adresses relatives /goodURL

initialize global URLgeoJSONCatalog to join "https://onvaessayer.github.io/"
"gitshareData1/map_geojson"

when Screen1.Initialize
do call Map1.LoadFromURL url get global URLgeoJSONCatalog

when any Marker.Click
component notAlreadyHandled
do set Marker.EnableInfoBox of component get component to true
set Web1.Url to Marker.Description of component get component
set Web1.SaveResponse to false
call Web1.Get

when Screen1.ErrorOccurred
component functionName errorNumber message
do call Notifier1.ShowMessage...
to goodURL URL refURL
result

initialize global goodGeojson to make a list join "https://..."

initialize global badGeojson to make a list join "https://..."

when Web1.GetText
url responseCode responseType responseContent
do if get responseCode = 200
then set global shop to call Web1.JsonTextDecodeWithDictionaries jsonText get responseContent
if not is a dictionary? get global shop
then call Notifier1.ShowAlert notice "shop is not a dictionary"
else set LabelTitle.Text to get value for key "title" in dictionary get global shop or if not found "no title"
set LabelAddress.Text to get value for key "address" in dictionary get global shop or if not found "no address"
set Image1.Picture to get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
set global items to get value for key "items" in dictionary get global shop or if not found create empty list
if is a list? thing get global items
then call setListViewElements myItems get global items
else call Notifier1.ShowAlert notice "items is not a list"
else call Notifier1.ShowMessage...

to setListViewElements myItems
do initialize local listViewElements to create empty list
in for each item in list get m...
set ListView1.Elements to get listViewElements

CODE gisthare 2a => 2b : adresses relatives /goodURL

initialize global URLgeoJSONCatalog to join "https://onvaessayer.github.io/" "gitshareData1/map.geojson"

when Screen1.Initialize
do call Map1.LoadFromURL
url get global URLgeoJSONCatalog

when any Marker.Click
component notAlreadyHandled
do set Marker.EnableInfoBox of component get component to true
set Web1.Uri to call goodURL.URL
Marker.Description of component get component
refURL get global URLgeoJSONCatalog
set Web1.SaveResponse to false
call Web1.Get

when Screen1.ErrorOccurred
component functionName errorNumber message
do call Notifier1.ShowMessage...

to goodURL URL refURL
result
do if starts at text get URL piece "http" ≠ 1
then initialize local path to ""
initialize local pathLength to 0
in if starts at text reverse get refURL piece "/" ≠ 0
then set pathLength to length get refURL - starts at text reverse get refURL piece "/"
set path to segment text get refURL start 1 length get pathLength
set URL to join get path "/" get URL

initialize global goodGeojson to make a list join "https://..."
initialize global badGeojson to make a list join "https://..."

when Web1.GetText
url responseCode responseType responseContent
do if get responseCode = 200
then set global shop to call Web1.JsonTextDecodeWithDictionaries jsonText get responseContent
if not is a dictionary? get global shop
then call Notifier1.ShowAlert notice "shop is not a dictionary"
else set LabelTitle.Text to get value for key "title" in dictionary get global shop or if not found "no title"
set LabelAddress.Text to get value for key "address" in dictionary get global shop or if not found "no address"
set Image1.Picture to call goodURL.URL
get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
refURL get url
set global items to get value for key "items" in dictionary get global shop or if not found create empty list
if is a list? thing get global items
then call setListViewElements myItems get global items
else call Notifier1.ShowAlert notice "items is not a list"
else call Notifier1.ShowMessage...

initialize global shop to create empty dictionary

initialize global items to create empty list

to setListViewElements myItems
do initialize local listViewElements to create empty list
in for each item in list get m...
set ListView1.Elements to get listViewElements

CODE gisthare 2a => 2b : adresses relatives /goodURL

```
initialize global URLgeoJSONCatalog to join "https://onvaessayer.github.io/" "gitshareData1/map.geojson"
```

```
when Screen1.Initialize  
do  
  call Map1.LoadFromURL url get global URLgeoJSONCatalog  
  
when any Marker.Click  
  component notAlreadyHandled  
do  
  set Marker.EnableInfoBox of component get component to true  
  set Web1.Url to call goodURL URL Marker.Description of component get component refURL get global URLgeoJSONCatalog  
  set Web1.SaveResponse to false  
  call Web1.Get
```

```
when Screen1.ErrorOccurred  
  component functionName errorNumber message  
do  
  call Notifier1.ShowMessage...
```

```
to goodURL URL refURL  
result do if starts at text get U...
```

```
initialize global goodGeojson to make a list join "https://..."  
initialize global badGeojson to make a list join "https://..."
```

```
when Web1.GetText  
  url responseType responseContent  
do  
  if get responseCode = 200  
  then  
    set global shop to call Web1.JsonTextDecodeWithDictionaries jsonText get responseContent  
    if not is a dictionary? get global shop  
    then  
      call Notifier1.ShowAlert notice "shop is not a dictionary"  
    else  
      set LabelTitle.Text to get value for key "title" in dictionary get global shop or if not found "no title"  
      set LabelAddress.Text to get value for key "address" in dictionary get global shop or if not found "no address"  
      set Image1.Picture to call goodURL URL get value for key "image" in dictionary get global shop or if not found "AndroidFR.png" refURL get uri  
      set global items to get value for key "items" in dictionary get global shop or if not found create empty list  
      if is a list? thing get global items  
      then  
        call setListViewElements myItems get global items  
      else  
        call Notifier1.ShowAlert notice "items is not a list"  
      else  
        call Notifier1.ShowMessage...
```

```
to setListViewElements myItems  
do  
  initialize local listViewElements to create empty list  
  in for each item in list get m...  
  set ListView1.Elements to get listViewElements
```

CODE gisthare 2a => 2b : adresses relatives /goodURL

```
initialize global URLgeoJSONCatalog to join ("https://onvaessayer.github.io/" "gitshareData1/map_geojson")
```

```
when Screen1 Initialize  
do call Map1 LoadFromURL url get global URLgeoJSONCatalog
```

```
when any Marker.Click  
component notAlreadyHandled  
do set Marker.EnableInfoBox of component get component to true  
set Web1.Url to call goodURL URL Marker.Description of component get component refURL get global URLgeoJSONCatalog  
set Web1.SaveResponse to false  
call Web1.Get
```

```
when Screen1 ErrorOccurred  
component functionName errorNumber message  
do call Notifier1.ShowMessage...
```

```
to goodURL URL refURL  
result do if starts at text get U...
```

```
initialize global goodGeojson to make a list join "https://..."
```

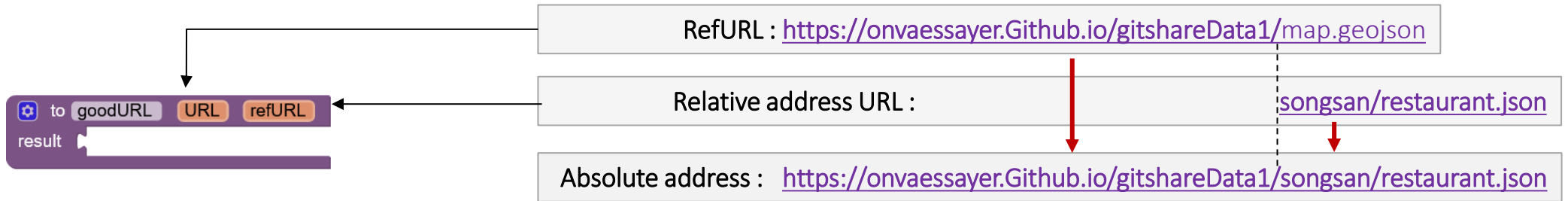
```
initialize global badGeojson to make a list join "https://..."
```

```
when Web1 .GetText  
url responseCode responseType responseContent  
do if get responseCode = 200  
then set global shop to call Web1.JsonTextDecodeWithDictionaries jsonText get responseContent  
if not is a dictionary? get global shop  
then call Notifier1.ShowAlert notice "shop is not a dictionary"  
else set LabelTitle.Text to get value for key "title" in dictionary get global shop or if not found "no title"  
set LabelAddress.Text to get value for key "address" in dictionary get global shop or if not found "no address"  
set Image1.Picture to call goodURL URL get value for key "image" in dictionary get global shop or if not found "AndroidFR.png" refURL get url  
set global items to get value for key "items" in dictionary get global shop or if not found create empty list  
if is a list? thing get global items  
then call setListviewElements myItems get global items  
else call Notifier1.ShowAlert notice "items is not a list"  
else call Notifier1.ShowMessage...
```

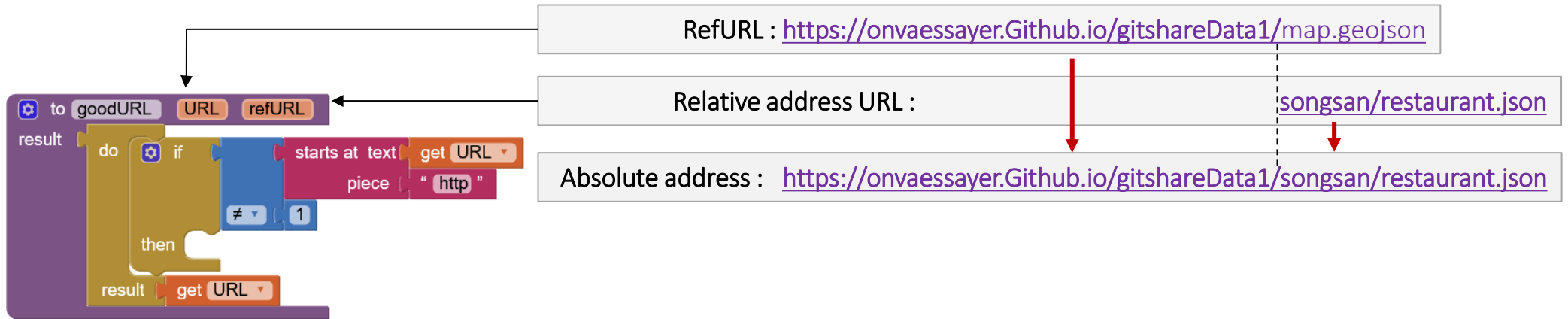
```
to setListviewElements myItems  
do initialize local listViewElements to create empty list  
in for each item in list get m...  
set ListView1.Elements to get listViewElements
```


FONCTION GOODURL : RECONSTRUIRE L'URL COMPLÈTE

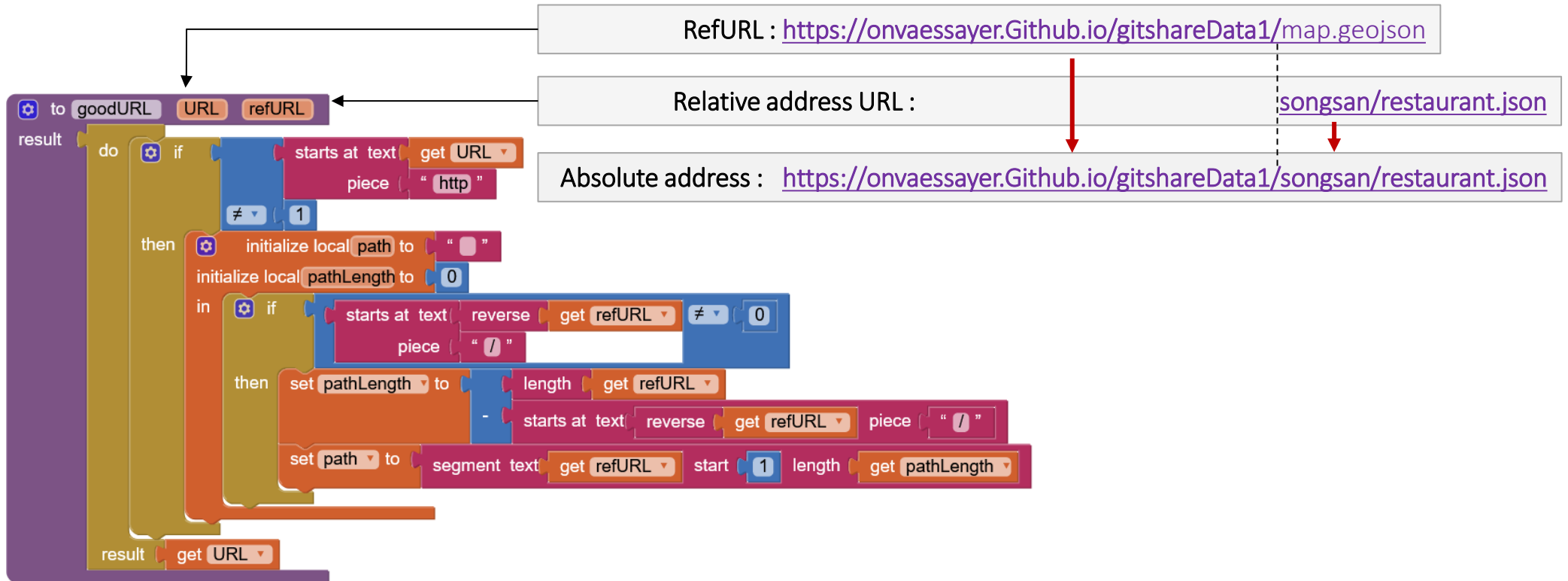
FUNCTION GOODURL : RECONSTRUIRE L'URL COMPLÈTE



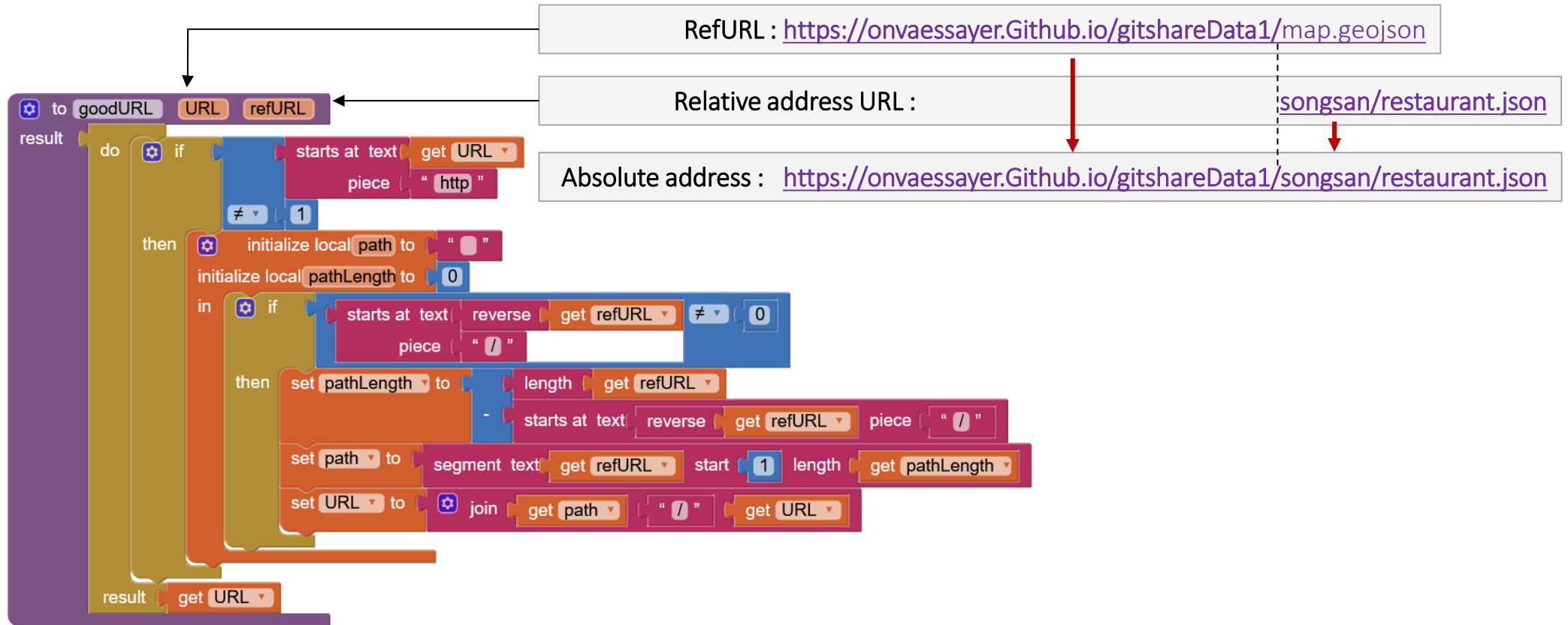
FONCTION GOODURL : RECONSTRUIRE L'URL COMPLÈTE



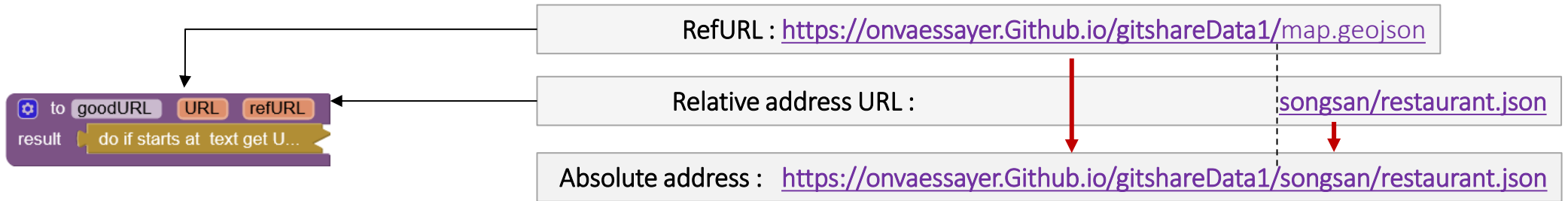
FONCTION GOODURL : RECONSTRUIRE L'URL COMPLÈTE



FONCTION GOODURL : RECONSTRUIRE L'URL COMPLÈTE



FUNCTION GOODURL : RECONSTRUIRE L'URL COMPLÈTE



FONCTION GOODURL : RECONSTRUIRE L'URL COMPLÈTE

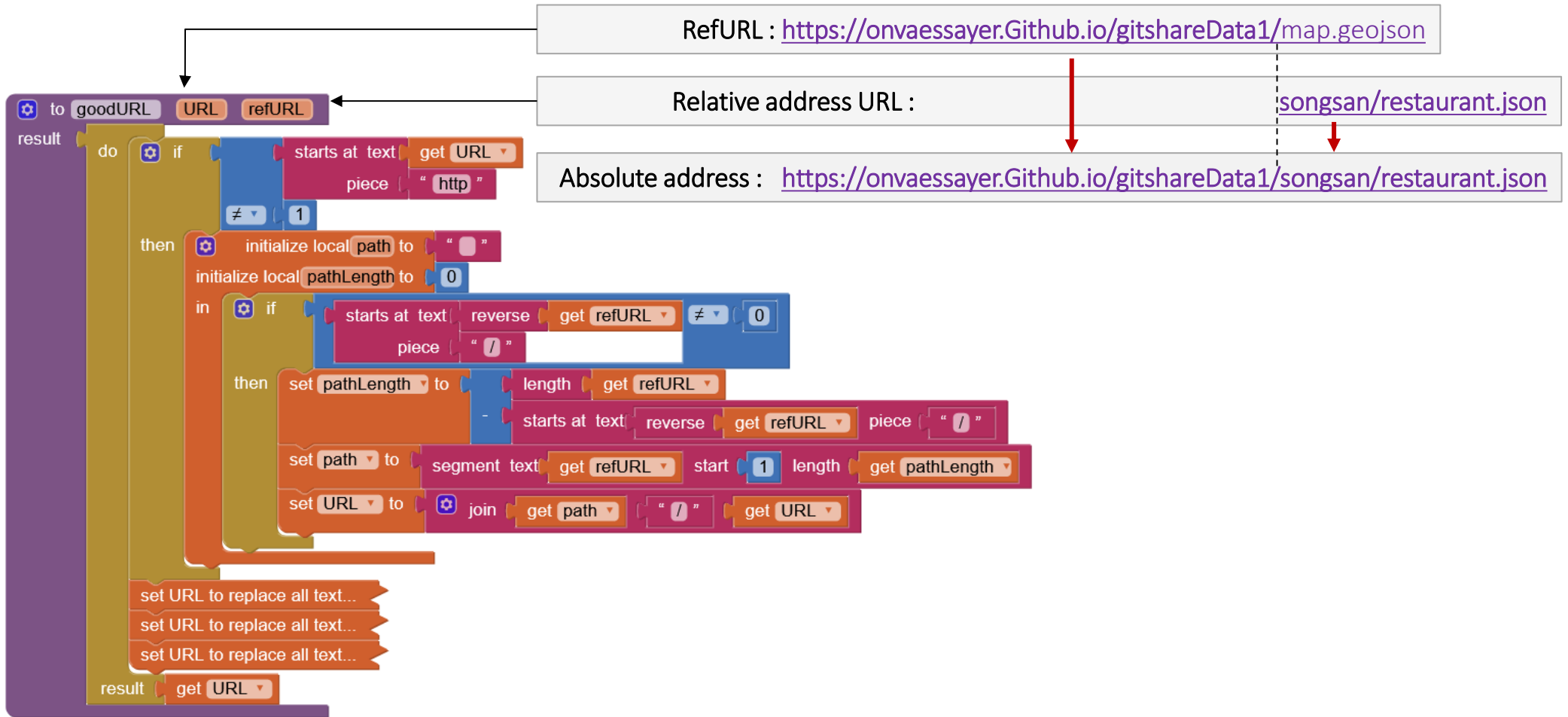
The diagram illustrates the reconstruction of a complete URL from a reference URL and a relative address. It shows three stages:

- RefURL :** <https://onvaessayer.Github.io/gitshareData1/map.geojson>
- Relative address URL :** [songsan/restaurant.json](#)
- Absolute address :** <https://onvaessayer.Github.io/gitshareData1/songsan/restaurant.json>

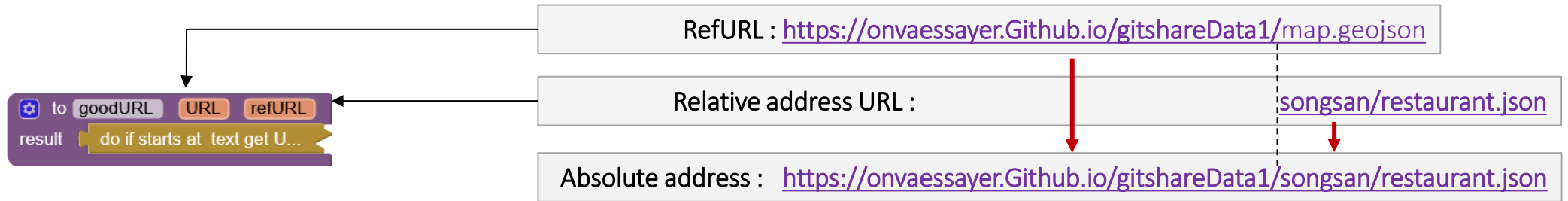
The Scratch script implements this logic:

- to goodURL URL refURL** (function definition)
- do** block:
 - if** `starts at text` `get URL` `piece` `"http"` `≠` `1`
 - then** block:
 - initialize local path** to `" "`
 - initialize local pathLength** to `0`
 - in** block:
 - if** `starts at text` `reverse` `get refURL` `≠` `0` `piece` `"/"`
 - then** block:
 - set pathLength** to `length` `get refURL` `-` `starts at text` `reverse` `get refURL` `piece` `"/"`
 - set path** to `segment text` `get refURL` `start` `1` `length` `get pathLength`
 - set URL** to `join` `get path` `"/"` `get URL`
 - set URL** to `replace all text` `get URL` `segment` `"?dl=0"` `replacement` `"?dl=1"`
 - set URL** to `replace all text` `get URL` `segment` `"https://drive.google.com/file/d/"` `replacement` `"https://drive.google.com/uc?export=download&id="`
 - set URL** to `replace all text` `get URL` `segment` `"/view?usp=sharing"` `replacement` `" "`
- result** `get URL`

FONCTION GOODURL : RECONSTRUIRE L'URL COMPLÈTE



FUNCTION GOODURL : RECONSTRUIRE L'URL COMPLÈTE



3.3.3

Adresses Dropbox & Google Drive

gitshare2b

PLAN

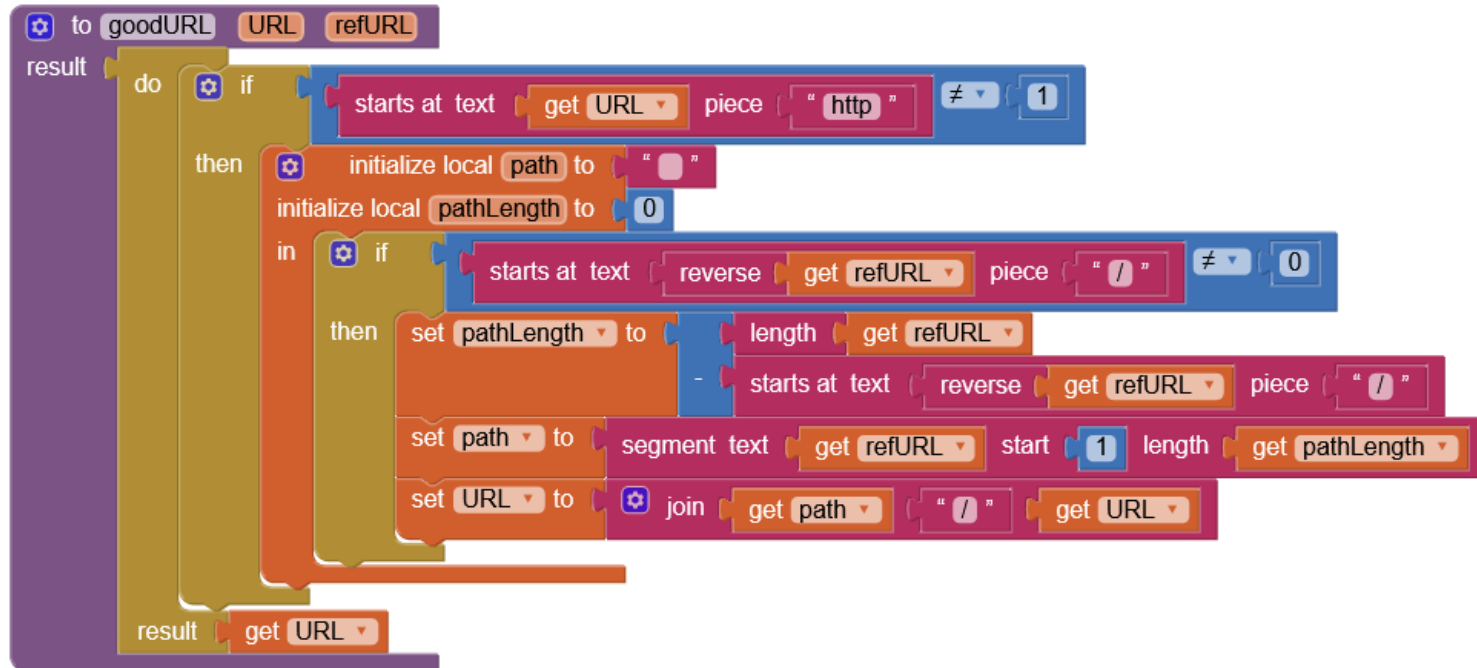
- Introduction : décomposition de l'application
- 1. Création d'un site Web / serveur de données
- 2. Définition des données et préparation d'un jeu
- 3. Création de l'application mobile avec App Inventor
 - 1. V1 : visualiser le catalogue des restaurants sur une carte
 - 2. V2a : sélectionner un restaurant et afficher ses nom, adresse, image et liste de plats
 - 3. V2b : codage défensif, modèle de données, adresses relatives, Dropbox & Google Drive
 - 4. V3a : identifier et enregistrer l'utilisateur et la carte
 - 5. V3b : préparer et passer une commande
 - 6. V3c : bonus

Dropbox / gitshareDropboxData

Importer Créer Organiser

Nom ↑	Modification	Qui y a accès ?
DominosPizza	☆ --	Vous unique...
IndianaCafe	☆ --	Vous unique...
LeSaotico	☆ --	Vous unique...
momoka	☆ --	Vous unique...
relaisDeLaPlace	☆ --	Vous unique...
RistoranteRoma	☆ --	Vous unique...
songSan	☆ --	Vous unique...
map.json	☆ 20/9/2022 17:22	Vous unique...

FONCTION GOODURL : CHANGE / DROPBOX



```
to goodURL URL refURL
  result
  do
    if starts at text (get URL) piece ("http") ≠ 1
    then
      initialize local path to ""
      initialize local pathLength to 0
      in if starts at text (reverse (get refURL)) piece ("/") ≠ 0
      then
        set pathLength to length (get refURL) - starts at text (reverse (get refURL)) piece ("/")
        set path to segment text (get refURL) start 1 length (get pathLength)
        set URL to join (get path) ("/") (get URL)
      end
    end
  end
  result (get URL)
end
```

The image shows a Scratch code block for a function named 'goodURL'. The function takes two arguments: 'URL' and 'refURL'. It starts with a 'do' block containing an 'if' block. The 'if' block checks if the 'URL' starts with 'http'. If true, it initializes local variables 'path' to an empty string and 'pathLength' to 0. It then enters an 'in' block with another 'if' block. This second 'if' block checks if the reverse of 'refURL' starts with '/'. If true, it calculates 'pathLength' as the length of 'refURL' minus the index of the first '/' in the reversed string. It then sets 'path' to the segment of 'refURL' starting at that index. Finally, it sets 'URL' to the concatenation of 'path', '/', and 'URL'. The function ends by returning the value of 'URL'.

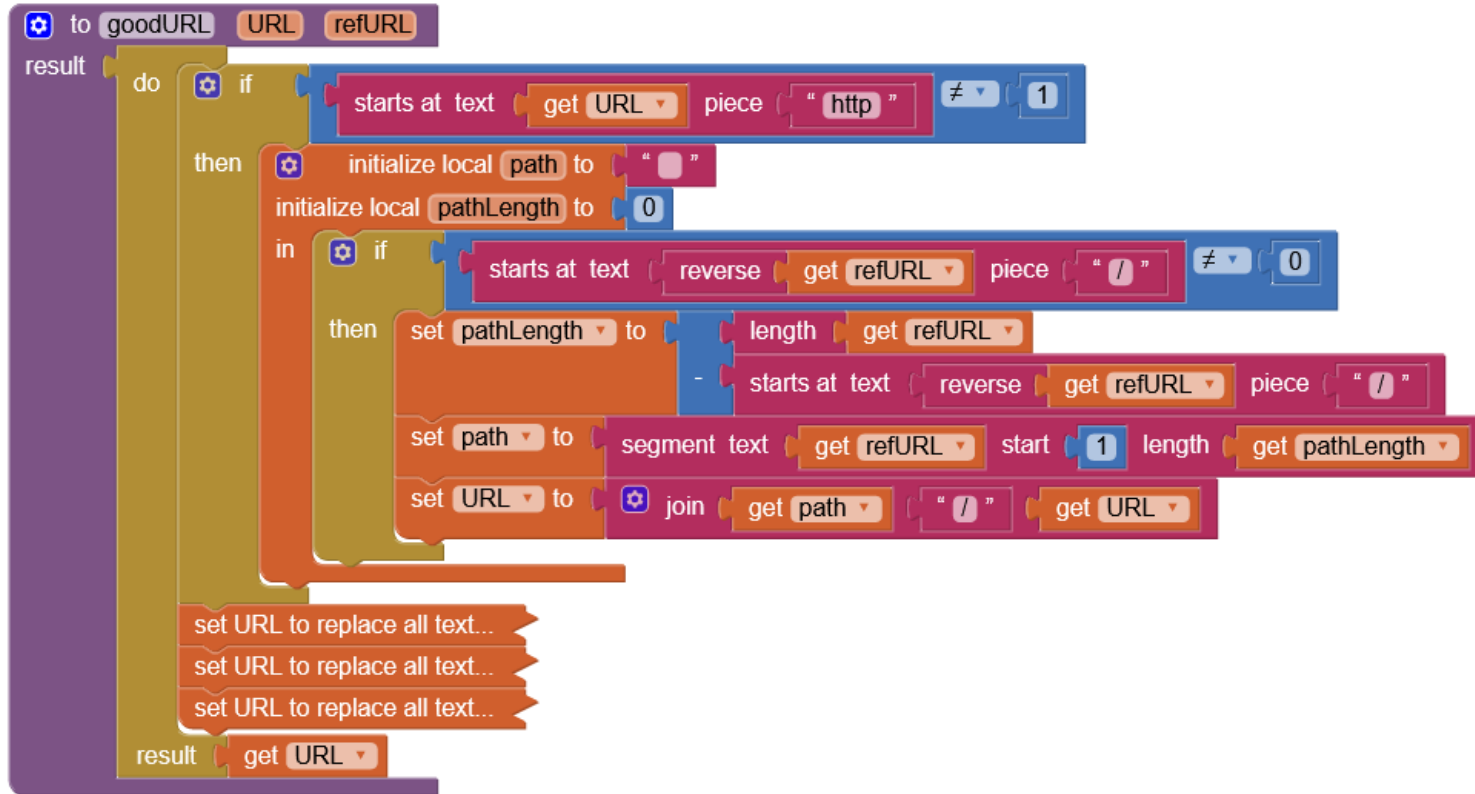
FUNCTION GOODURL : CHANGE / DROPBOX

```
to goodURL URL refURL
  result
  do
    if starts at text (get URL) piece ("http") ≠ 1
    then
      initialize local path to ""
      initialize local pathLength to 0
      in
        if starts at text (reverse (get refURL)) piece ("/") ≠ 0
        then
          set pathLength to (length (get refURL) - starts at text (reverse (get refURL)) piece ("/"))
          set path to (segment text (get refURL) start 1 length (get pathLength))
          set URL to (join (get path) ("/") (get URL))
        end
      end
    end
    set URL to (replace all text (get URL) segment ("?dl=0") replacement ("?dl=1"))
  end
  result (get URL)
end
```

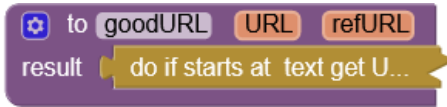
FONCTION GOODURL : CHANGE / GOOGLE DRIVE

```
to goodURL URL refURL
  result
  do
    if starts at text (get URL) piece ("http") ≠ 1
    then
      initialize local path to ""
      initialize local pathLength to 0
      in
        if starts at text (reverse (get refURL)) piece ("/") ≠ 0
        then
          set pathLength to (length (get refURL) - starts at text (reverse (get refURL)) piece ("/"))
          set path to (segment text (get refURL) start 1 length (get pathLength))
          set URL to (join (get path) ("/") (get URL))
      end
    end
    set URL to (replace all text (get URL) segment ("?dl=0") replacement ("?dl=1"))
    set URL to (replace all text (get URL) segment ("https://drive.google.com/file/d/ ") replacement ("https://drive.google.com/uc?export=download&id= "))
    set URL to (replace all text (get URL) segment ("/view?usp=sharing ") replacement (" "))
  end
  result (get URL)
end
```

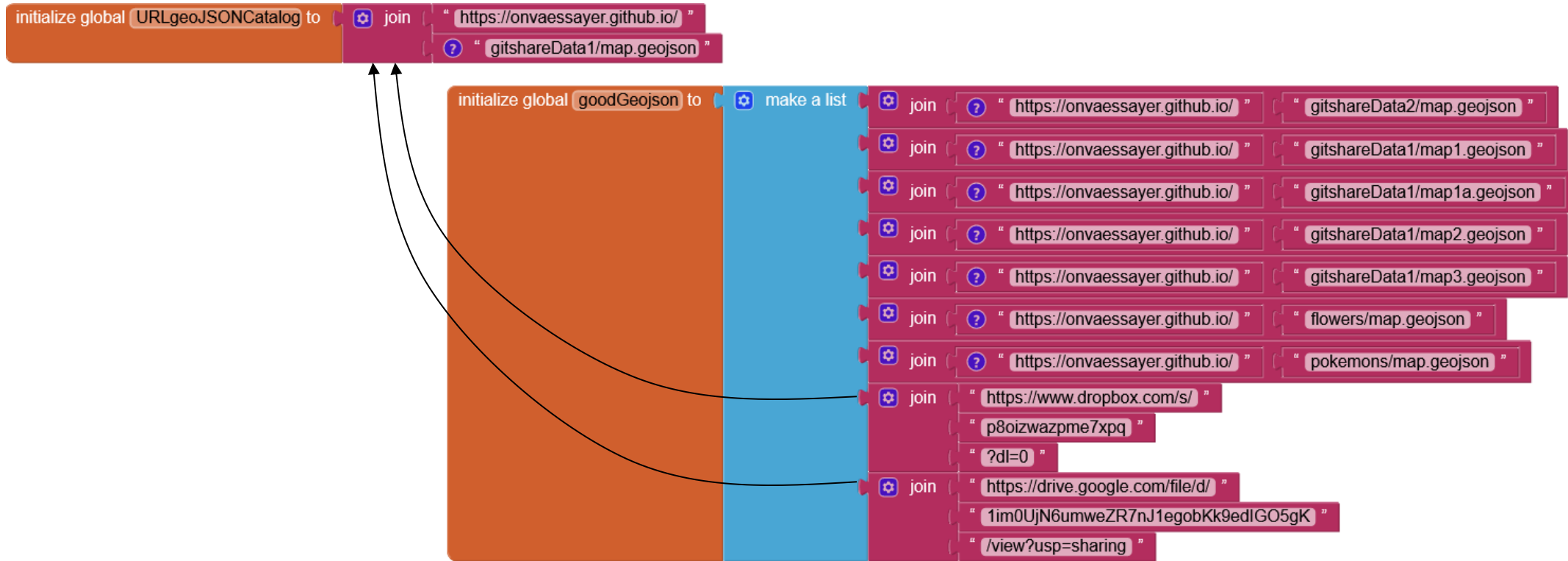
FONCTION GOODURL : RECONSTRUIRE L'URL COMPLÈTE



FONCTION GOODURL : RECONSTRUIRE L'URL COMPLÈTE



VERSION GITSHARE2B : TEST DATASETS, INCLUDING DROPBOX AND DRIVE



VERSION GITSHARE2B : FINAL

Viewer **Screen1**

```
when Screen1.Initialize
do
  open another screen with start value screenName
  startValue
```

Viewer **mapScreen**

```
initialize global URLgeoJSONCatalog to
join
  " https://drive.google.com/file/d/ "
  " 13nW1kQxoJ0jZc8RHxgpI9LlwEp0XWk2 "
  " /view?usp=sharing "

when mapScreen.Initialize
do
  call Map1.LoadFromURL
  url
  call goodURL
  URL
  refURL

when any Marker.Click
  component notAlreadyHandled
do
  set Marker.EnableInfoBox of component
  to true
  initialize local shopURL to
  Marker.Description
  of component
  in
  open another screen with start value screenName
  startValue
  call goodURL
  URL
  refURL

to goodURL URL refURL
result
  do if starts at text get U...
```

VERSION GITSHARE2B : FINAL

Viewer Shop screen

```
initialize global shop to create empty dictionary
initialize global items to create empty list
initialize global shopURL to ""


when Web1 GotText
  url responseCode responseType responseContent
  do
    if responseCode = 200
      then
        set global shop to call Web1 .JsonTextDecodeWithDictionaries
          jsonText get responseContent
        if is a dictionary? get global shop
          then
            set LabelTitle .Text to get value for key "title" in dictionary get global shop or if not found "?"
            set LabelAddress .Text to get value for key "address" in dictionary get global shop or if not found "?"
            set Image1 .Picture to call goodURL
              URL get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
              refURL get global shopURL
            set global items to get value for key "items" in dictionary get global shop or if not found create empty list
            if is a list? thing get global items
              then
                call setListViewElements myItems get global items
              else
                call Notifier1 .ShowMessag...
            else
              call Notifier1 .ShowMessag...
          else
            call Notifier1 .ShowMessag...

when shop Initialize
  do
    set global shopURL to get start value
    set Web1 .Uri to get global shopURL
    set Web1 .SaveResponse to false
    call Web1 .Get

when shop ErrorOccurred
  component functionName errorNumber message
  do
    call Notifier1 .ShowMessag...

when ListView1 AfterPicking
  do
    initialize local item to select list item list get global items
      index ListView1 .SelectionIndex
    in
      if is a dictionary? get item
        then
          set Image1 .Picture to call goodURL
            URL get value for key "image" in dictionary get item or if not found "AndroidFR.png"
            refURL get global shopURL

to setListViewElements myItems
  do
    initialize local listViewElements to create empty list
    in
      for each item in list get myItems
        do
          if is a string? thing get i...
            set ListView1 .Elements to get listViewElements
```



3.4

Profil utilisateur, catalogues
enregistrement de la carte

PLAN

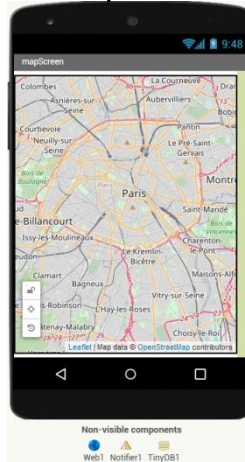
- Introduction : décomposition de l'application
- 1. Création d'un site Web / serveur de données
- 2. Définition des données et préparation d'un jeu
- 3. Création de l'application mobile avec App Inventor
 - 1. V1 : visualiser le catalogue des restaurants sur une carte
 - 2. V2a : sélectionner un restaurant et afficher ses nom, adresse, image et liste de plats
 - 3. V2b : codage défensif, modèle de données, adresses relatives, Dropbox & Google Drive
 - 4. V3a : identifier et enregistrer l'utilisateur et la carte
 - 5. V3b : préparer et passer une commande
 - 6. V3c : bonus

GITSHARE 2b

Screen1



mapScreen



shop



```
when Screen1 Initialize
do open another screen with start value screenName mapScreen
startValue ""
```

```
initialize global URLgeoJSONCatalog to join
["https://drive.google.com/file/d/
" "13nW1kQxoJ0jZc8RHxgpi9LlWp0XWk2"
"/view?usp=sharing"]
```

```
when mapScreen Initialize
do call Map1 LoadFromURL
url call goodURL
URL get global URLgeoJSONCatalog
refURL ""
```

```
when any Marker.Click
component notAlreadyHandled
do set Marker EnableInfoBox of component get component to true
initialize local shopURL to Marker Description of component get component
in open another screen with start value screenName shop
startValue call goodURL
URL get shopURL
refURL get global URLgeoJSONCatalog
```

```
initialize global shop to create empty dictionary
initialize global items to create empty list
when shop Initialize
do set Web1 Uri to get start value
set Web1 SaveResponse to false
call Web1 Get
```

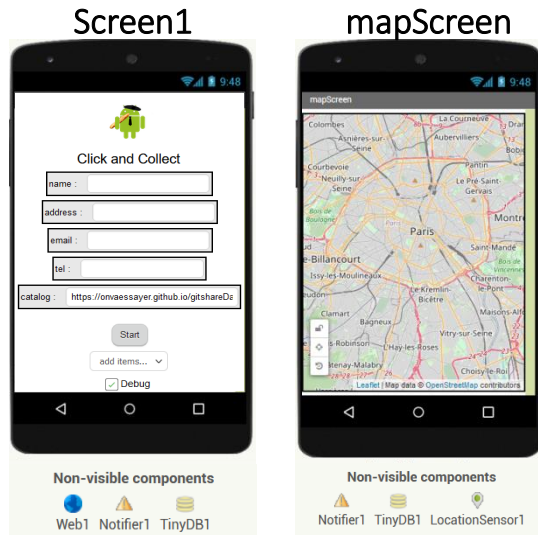
```
when Web1 GoText
url responseCode responseType responseContent
do if get responseCode == 200
then set global shop to call Web1 JsonTextDecodeWithDictionaries jsonText get responseContent
if is a dictionary? get global shop
then set LabelTitle Text to get value for key title in dictionary get global shop or if not found ?
set LabelAddress Text to get value for key address in dictionary get global shop or if not found
set Image1 Picture to call goodURL
URL get value for key image in dictionary get global shop or if not found AndroidFR.png
rootURL get url
set global items to
```

A large red circle is centered on the page, containing the text '3.4.1'.

3.4.1

Définir et enregistrer
le profil utilisateur

GITSHARE 3a : nouvelles fonctions / mapScreen



- lire l'URL du catalogue en paramètre d'appel

get start value

- enregistrer la position de la carte (à chaque déplacement)

& relire sa position (au démarrage de mapScreen)

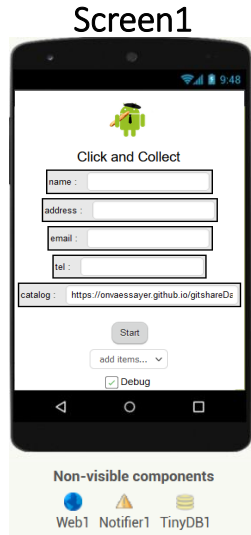
```
call TinyDB1 .StoreValue
tag "latitude"
valueToStore Map1 . Latitude
```

```
call TinyDB1 .GetValue
tag "latitude"
valueIfTagNotThere 48.85
```

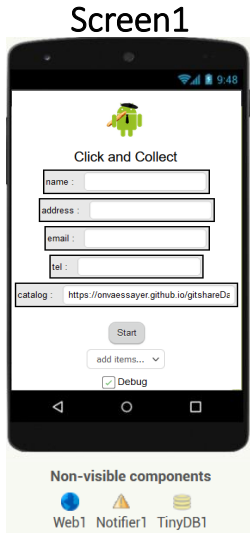
- centrer la carte sur la position de l'utilisateur

```
call Map1 .PanTo
latitude
longitude
zoom
```

GITSHARE 3a : nouvelles fonctions / Screen1



GITSHARE 3a : nouvelles fonctions / Screen1



- identifier l'utilisateur (comme un dictionnaire)

- nom,
- adresse,
- email,
- téléphone,
- catalogue geoJSON

TextBoxName . Text

TextBoxName . Text

TextBoxEmail . Text

TextBoxTelephone . Text

TextBoxCatalog . Text

- vérifier que le profil utilisateur est bien renseigné
- enregistrer le profil utilisateur

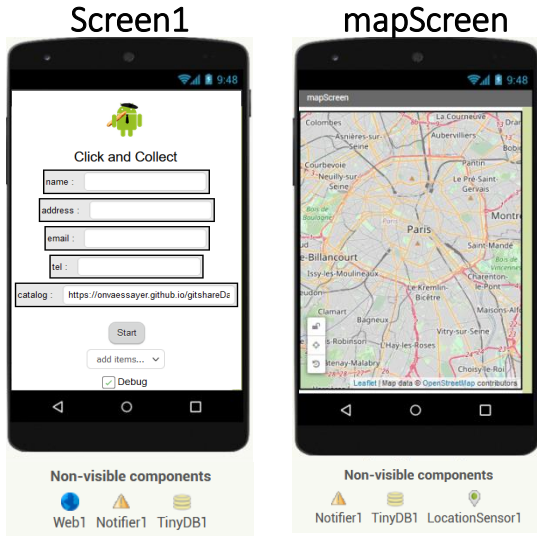
```
call TinyDB1 .StoreValue
tag "user"
valueToStore get user
```

relire le profil utilisateur au démarrage

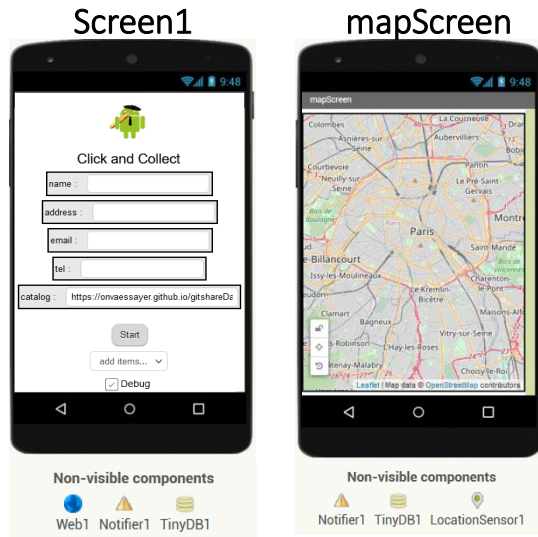
```
call TinyDB1 .GetValue
tag "user"
valueIfTagNotThere create empty dictionary
```

- choisir le catalogue dans une liste (un dictionnaire)
- gérer un mode "debug"

GITSHARE 3a : nouvelles fonctions / mapScreen



GITSHARE 3a : nouvelles fonctions / mapScreen



- lire l'URL du catalogue en paramètre d'appel
- enregistrer la position de la carte (à chaque déplacement)

get start value

& relire sa position (au démarrage de mapScreen)

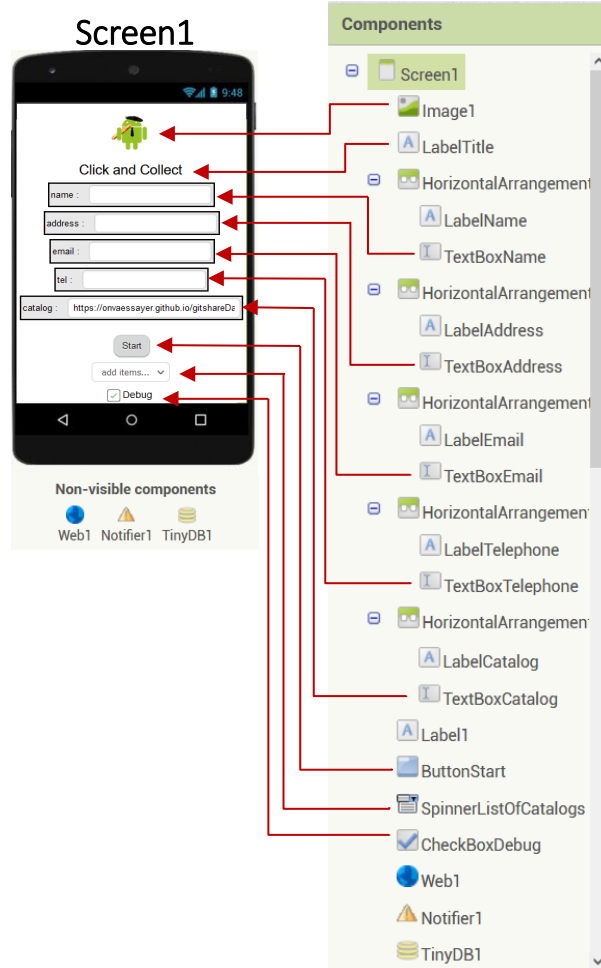
```
call TinyDB1 .StoreValue
tag "latitude"
valueToStore Map1 . Latitude
```

```
call TinyDB1 .GetValue
tag "latitude"
valueIfTagNotThere 48.85
```

- centrer la carte sur la position de l'utilisateur

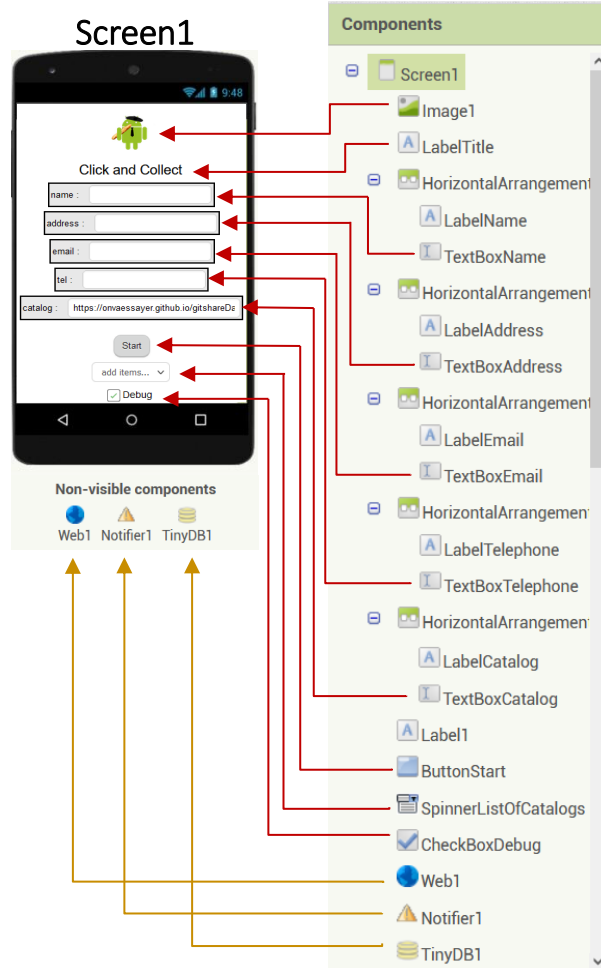
```
call Map1 .PanTo
latitude
longitude
zoom
```


GITSHARE 3a : Screen1 – interface utilisateur



- image,
- label pour le titre de l'application,
- zone de texte pour le nom (dans un arrangement) ,
- zone de texte pour l'adresse (dans un arrangement),
- zone de texte pour l'email (dans un arrangement),
- zone de texte pour le téléphone (dans un arrangement)
- zone de texte pour le catalogue geoJSON,
- Bouton start,
- Spinner pour le choix du catalogue,
- Checkbox pour activer/désactiver le debug,

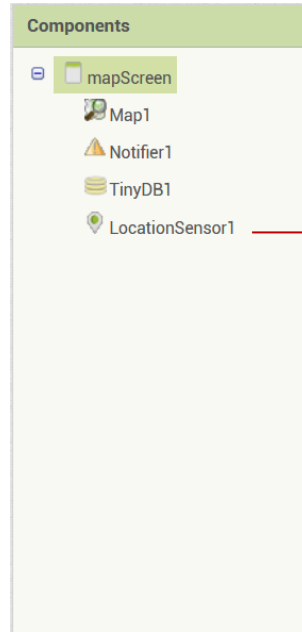
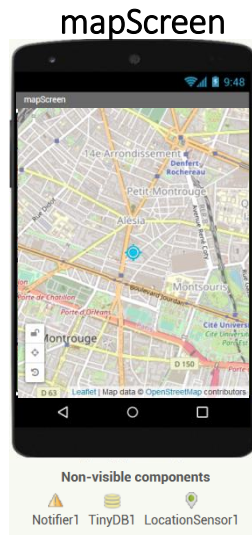
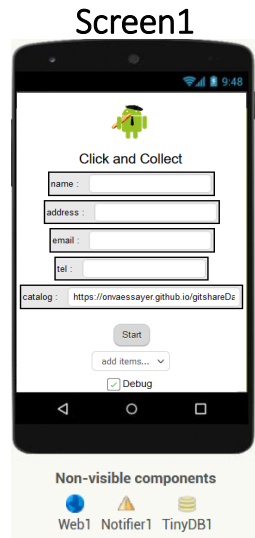
GITSHARE 3a : Screen1 – interface utilisateur



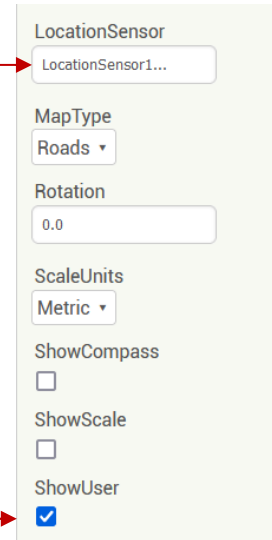
- image,
- label pour le titre de l'application,
- zone de texte pour le nom (dans un arrangement) ,
- zone de texte pour l'adresse (dans un arrangement),
- zone de texte pour l'email (dans un arrangement),
- zone de texte pour le téléphone (dans un arrangement)
- zone de texte pour le catalogue geoJSON,
- Bouton start,
- Spinner pour le choix du catalogue,
- Checkbox pour activer/désactiver le debug,

- Composant web pour la lecture de fichiers sur Internet,
- Notifier pour les messages,
- TinyDB pour enregistrer et relire le profil utilisateur

GITSHARE 3a : nouvelles fonctions / mapScreen



• Map :



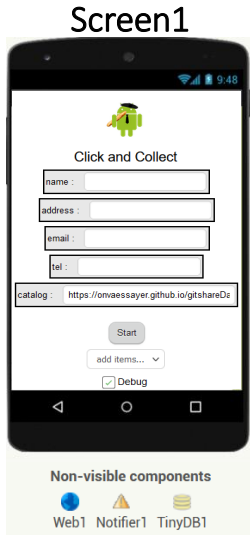
- notifier pour les messages,
- tinyDB pour enregistrer et relire la position de la carte
- location sensor pour la position de l'utilisateur

GITSHARE 3 : Screen1 - design

The screenshot shows the GitShare 3 Designer interface for designing a mobile screen named "Screen1".

- Top Bar:** "gitshare03a_" (left), "Screen1" (dropdown), "Add Screen ...", "Remove Screen", "Publish to Gallery" (right), "Designer" (tab), "Blocks" (tab).
- Palette (Left):** Search Components...; User Interface (Button, CheckBox, DatePicker, Image, Label, ListPicker, ListView, Notifier, PasswordTextBox, Slider, Spinner, Switch, TextBox, TimePicker, WebViewer); Layout; Media; Drawing and Animation.
- Viewer (Center):** "Display hidden components in Viewer" checkbox; A mobile phone mockup showing a form titled "Click and Collect" with fields for name, address, email, tel, and catalog (value: https://onvaessayer.github.io/gitshareDa...), a "Start" button, an "add items..." dropdown, and a checked "Debug" checkbox.
- Components (Right):** Tree view showing the hierarchy of UI elements: Screen1, Image1, LabelTitle, HorizontalArrangement (containing LabelName, TextBoxName), HorizontalArrangement (containing LabelAddress, TextBoxAddress), HorizontalArrangement (containing LabelEmail, TextBoxEmail), HorizontalArrangement (containing LabelTelephone, TextBoxTelephone), HorizontalArrangement (containing LabelCatalog). Buttons: Rename, Delete.
- Properties (Right):** Properties for "Screen1": AboutScreen (share with Git and...), AccentColor (Default), AlignHorizontal (Center : 3), AlignVertical (Center : 2), AppName (gitShare2b), BackgroundColor (Default), BackgroundImage (None...), BigDefaultText, BlocksToolkit (All), CloseScreenAnimation (Default), DefaultFileScope (App), HighContrast, Icon (AndroidFR.png...).
- Non-visible components (Bottom):** Web1, Notifier1, TinyDB1.

GITSHARE 3a : nouvelles fonctions / Screen1



- identifier l'utilisateur (comme un dictionnaire)

- nom,
- adresse,
- email,
- téléphone,
- catalogue geoJSON

TextBoxName . Text

TextBoxName . Text

TextBoxEmail . Text

TextBoxTelephone . Text

TextBoxCatalog . Text

- choisir le catalogue dans une liste (un dictionnaire)
- vérifier que le profil utilisateur est bien renseigné
- enregistrer le profil utilisateur

```
call TinyDB1 .StoreValue
tag "user"
valueToStore get user
```

relire le profil utilisateur au démarrage

```
call TinyDB1 .GetValue
tag "user"
valueIfTagNotThere create empty dictionary
```

- gérer un mode "debug"

GITSHARE 3a : SCREEN1

```
when Screen1.Initialize
do
  open another screen with start value screenName mapScreen
  startValue " "
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join "https://onvaessayer.github.io/" "gitshareData3/map.geojson"

when Screen1.Initialize
do open another screen with start value screenName mapScreen
startValue ""
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when Screen1.Initialize
do
  open another screen with start value screenName mapScreen
  startValue ""
```

```
to goodURL URL rootURL
result do if starts at text get U...
```


GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when Screen1.Initialize
do
  open another screen with start value screenName mapScreen
  startValue call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
```

```
to goodURL URL rootURL
result do if starts at text get U...
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when Screen1.Initialize
do
  open another screen with start value screenName mapScreen
  startValue call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""

when ButtonStart.Click
do
```

```
to goodURL URL rootURL
result do if starts at text get U...
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]
```

```
when Screen1.Initialize  
do
```

```
to goodURL URL rootURL  
result do if starts at text get U...
```

```
when ButtonStart.Click  
do open another screen with start value screenName  
startValue mapScreen  
call goodURL  
URL get global URLgeoJSONCatalog  
rootURL
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]
```

```
when Screen1.Initialize  
do set TextBoxCatalog.Text to call goodURL  
   URL get global URLgeoJSONCatalog  
   rootURL ""
```

```
to goodURL URL rootURL  
result do if starts at text get U...
```

```
when ButtonStart.Click  
do open another screen with start value screenName mapScreen  
   startValue call goodURL  
   URL get global URLgeoJSONCatalog  
   rootURL ""
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]  
  
when Screen1.Initialize  
do set TextBoxCatalog.Text to call goodURL  
   URL get global URLgeoJSONCatalog  
   rootURL ""
```

```
to goodURL URL rootURL  
result do if starts at text get U...
```

```
when ButtonStart.Click  
do open another screen with start value screenName mapScreen  
   startValue call goodURL  
   URL  
   rootURL ""
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
```

```
when ButtonStart.Click
do
  open another screen with start value screenName mapScreen
  startValue call goodURL
  URL TextBoxCatalog.Text
  rootURL ""
```

```
to goodURL URL rootURL
result do if starts at text get U...
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]
```

```
when Screen1.Initialize  
do  
  set TextBoxCatalog.Text to call goodURL  
  URL get global URLgeoJSONCatalog  
  rootURL ""
```

```
to goodURL URL rootURL  
result do if starts at text get U...
```

```
when ButtonStart.Click  
do  
  if  
  then open another screen with start value screenName mapScreen  
  startValue call goodURL  
  URL TextBoxCatalog.Text  
  rootURL ""
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]
```

```
when Screen1.Initialize  
do  
  set TextBoxCatalog.Text to call goodURL  
  URL get global URLgeoJSONCatalog  
  rootURL ""
```

```
to goodURL URL rootURL  
result do if starts at text get U...
```

```
when ButtonStart.Click  
do  
  if and  
  then open another screen with start value screenName mapScreen  
  startValue call goodURL  
  URL TextBoxCatalog.Text  
  rootURL ""
```


GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]  
  
when Screen1.Initialize  
do set TextBoxCatalog.Text to call goodURL  
    URL get global URLgeoJSONCatalog  
    rootURL ""
```

```
to goodURL URL rootURL  
result do if starts at text get U...
```

```
when ButtonStart.Click  
do if and  
    TextBoxName.Text  
then open another screen with start value screenName mapScreen  
    startValue call goodURL  
        URL TextBoxCatalog.Text  
        rootURL ""
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]  
  
when Screen1.Initialize  
do set TextBoxCatalog.Text to call goodURL  
    URL get global URLgeoJSONCatalog  
    rootURL ""
```

```
to goodURL URL rootURL  
result do if starts at text get U...
```

```
when ButtonStart.Click  
do if and  
    is empty TextBoxName.Text  
then open another screen with start value screenName mapScreen  
    startValue call goodURL  
        URL TextBoxCatalog.Text  
        rootURL ""
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]
```

```
when Screen1.Initialize  
do  
  set TextBoxCatalog.Text to call goodURL  
  URL get global URLgeoJSONCatalog  
  rootURL ""
```

```
to goodURL URL rootURL  
result do if starts at text get U...
```

```
when ButtonStart.Click  
do  
  if and not is empty TextBoxName.Text  
  then  
    open another screen with start value  
    screenName mapScreen  
    startValue call goodURL  
    URL TextBoxCatalog.Text  
    rootURL ""
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]  
  
when Screen1.Initialize  
do  
  set TextBoxCatalog.Text to call goodURL  
  URL get global URLgeoJSONCatalog  
  rootURL ""
```

```
to goodURL URL rootURL  
result do if starts at text get U...
```

```
when ButtonStart.Click  
do  
  if and [not isEmpty TextBoxName.Text, not isEmpty TextBoxAddress.Text]  
  then  
    open another screen with start value screenName mapScreen  
    startValue call goodURL  
    URL TextBoxCatalog.Text  
    rootURL ""
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
```

```
to goodURL URL rootURL
result do if starts at text get U...
```

```
when ButtonStart.Click
do
  if and [not is empty TextBoxName.Text, not is empty TextBoxAddress.Text, not is empty TextBoxEmail.Text]
  then
    open another screen with start value screenName mapScreen
    startValue call goodURL
    URL TextBoxCatalog.Text
    rootURL ""
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]
```

```
when Screen1.Initialize  
do  
  set TextBoxCatalog . Text to call goodURL  
  URL get global URLgeoJSONCatalog  
  rootURL ""
```

```
to goodURL URL rootURL  
result do if starts at text get U...
```

```
when ButtonStart.Click  
do  
  if and [not is empty TextBoxName . Text  
          not is empty TextBoxAddress . Text  
          not is empty TextBoxEmail . Text  
          not is empty TextBoxTelephone . Text]  
  then  
    open another screen with start value screenName mapScreen  
    startValue call goodURL  
    URL TextBoxCatalog . Text  
    rootURL ""
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
```

```
to goodURL URL rootURL
result do if starts at text get U...
```

```
when ButtonStart.Click
do
  if and [not is empty TextBoxName.Text, not is empty TextBoxAddress.Text, not is empty TextBoxEmail.Text, not is empty TextBoxTelephone.Text]
  then
    open another screen with start value screenName mapScreen
    startValue call goodURL
    URL TextBoxCatalog.Text
    rootURL ""
  else
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]
```

```
when Screen1.Initialize  
do  
  set TextBoxCatalog.Text to call goodURL  
  URL get global URLgeoJSONCatalog  
  rootURL ""
```

```
to goodURL URL rootURL  
result do if starts at text get U...
```

```
when ButtonStart.Click  
do  
  if and [not is empty TextBoxName.Text, not is empty TextBoxAddress.Text, not is empty TextBoxEmail.Text, not is empty TextBoxTelephone.Text]  
  then  
    open another screen with start value screenName mapScreen  
    startValue call goodURL  
    URL TextBoxCatalog.Text  
    rootURL ""  
  else  
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```


GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
```

```
to goodURL URL rootURL
result do if starts at text get U...
```

```
when ButtonStart.Click
do
  if and [not is empty TextBoxName.Text, not is empty TextBoxAddress.Text, not is empty TextBoxEmail.Text, not is empty TextBoxTelephone.Text]
  then
    initialize local user to
    in
    open another screen with start value screenName mapScreen
    startValue call goodURL
    URL TextBoxCatalog.Text
    rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
```

```
to goodURL URL rootURL
result do if starts at text get U...
```

```
when ButtonStart.Click
do
  if and [not is empty TextBoxName.Text, not is empty TextBoxAddress.Text, not is empty TextBoxEmail.Text, not is empty TextBoxTelephone.Text]
  then
    initialize local user to make a dictionary
    in
    open another screen with start value mapScreen
    startValue call goodURL
    URL TextBoxCatalog.Text
    rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
```

```
to goodURL URL rootURL
result do if starts at text get U...
```

```
when ButtonStart.Click
do
  if and [not is empty TextBoxName.Text, not is empty TextBoxAddress.Text, not is empty TextBoxEmail.Text, not is empty TextBoxTelephone.Text]
  then
    initialize local user to make a dictionary
    in
    open another screen with start value screenName mapScreen
    startValue call goodURL
    URL TextBoxCatalog.Text
    rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]  
  
when Screen1.Initialize  
do  
  set TextBoxCatalog.Text to call goodURL  
  URL get global URLgeoJSONCatalog  
  rootURL ""
```

```
to goodURL URL rootURL  
result do if starts at text get U...
```

```
when ButtonStart.Click  
do  
  if and [not is empty TextBoxName.Text, not is empty TextBoxAddress.Text, not is empty TextBoxEmail.Text, not is empty TextBoxTelephone.Text]  
  then  
    initialize local user to make a dictionary key "name" value  
    in  
    open another screen with start value screenName mapScreen startValue call goodURL URL TextBoxCatalog.Text rootURL ""  
  else  
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
```

```
to goodURL URL rootURL
result do if starts at text get U...
```

```
when ButtonStart.Click
do
  if and [not is empty TextBoxName.Text, not is empty TextBoxAddress.Text, not is empty TextBoxEmail.Text, not is empty TextBoxTelephone.Text]
  then
    initialize local user to make a dictionary
    key "name"
    value TextBoxName.Text

    in
    open another screen with start value screenName mapScreen
    startValue call goodURL
    URL TextBoxCatalog.Text
    rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
```

```
to goodURL URL rootURL
result do if starts at text get U...
```

```
when ButtonStart.Click
do
  if and [not is empty TextBoxName.Text, not is empty TextBoxAddress.Text, not is empty TextBoxEmail.Text, not is empty TextBoxTelephone.Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName.Text
    key "address" value TextBoxAddress.Text
    in
    open another screen with start value screenName mapScreen
    startValue call goodURL
    URL TextBoxCatalog.Text
    rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
```

```
to goodURL URL rootURL
result do if starts at text get U...
```

```
when ButtonStart.Click
do
  if and [not is empty TextBoxName.Text, not is empty TextBoxAddress.Text, not is empty TextBoxEmail.Text, not is empty TextBoxTelephone.Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName.Text
    key "address" value TextBoxAddress.Text
    key "email" value TextBoxEmail.Text
    in
    open another screen with start value screenName mapScreen
    startValue call goodURL
    URL TextBoxCatalog.Text
    rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.gejson"]

when Screen1.Initialize
do
  set TextBoxCatalog .Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
```

```
to goodURL URL rootURL
result do if starts at text get U...
```

```
when ButtonStart.Click
do
  if and [not is empty TextBoxName .Text, not is empty TextBoxAddress .Text, not is empty TextBoxEmail .Text, not is empty TextBoxTelephone .Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName .Text
    key "address" value TextBoxAddress .Text
    key "email" value TextBoxEmail .Text
    key "telephone" value TextBoxTelephone .Text
    in
    open another screen with start value screenName mapScreen
    startValue call goodURL
    URL TextBoxCatalog .Text
    rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```


GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
```

```
to goodURL URL rootURL
result do if starts at text get U...
```

```
when ButtonStart.Click
do
  if and [not is empty TextBoxName.Text, not is empty TextBoxAddress.Text, not is empty TextBoxEmail.Text, not is empty TextBoxTelephone.Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName.Text
    key "address" value TextBoxAddress.Text
    key "email" value TextBoxEmail.Text
    key "telephone" value TextBoxTelephone.Text
    key "catalog" value TextBoxCatalog.Text
    in
    open another screen with start value screenName mapScreen
    startValue call goodURL
    URL TextBoxCatalog.Text
    rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog .Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
```

```
to goodURL URL rootURL
result do if starts at text get U...
```

```
when ButtonStart.Click
do
  if and [not is empty TextBoxName .Text, not is empty TextBoxAddress .Text, not is empty TextBoxEmail .Text, not is empty TextBoxTelephone .Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName .Text
    key "address" value TextBoxAddress .Text
    key "email" value TextBoxEmail .Text
    key "telephone" value TextBoxTelephone .Text
    key "catalog" value TextBoxCatalog .Text
    in call TinyDB1 .StoreValue tag valueToStore
    open another screen with start value screenName mapScreen startValue call goodURL
    URL TextBoxCatalog .Text
    rootURL ""
  else
    call Notifier1 .ShowAlert notice "missing name, email, address or telephone"
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.gejson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
```

```
to goodURL URL rootURL
result do if starts at text get U...
```

```
when ButtonStart.Click
do
  if and [not isEmpty TextBoxName.Text, not isEmpty TextBoxAddress.Text, not isEmpty TextBoxEmail.Text, not isEmpty TextBoxTelephone.Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName.Text
    key "address" value TextBoxAddress.Text
    key "email" value TextBoxEmail.Text
    key "telephone" value TextBoxTelephone.Text
    key "catalog" value TextBoxCatalog.Text
    in call TinyDB1.StoreValue tag "user" valueToStore
    open another screen with start value screenName mapScreen startValue call goodURL
    URL TextBoxCatalog.Text
    rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.gejson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
```

```
to goodURL URL rootURL
result do if starts at text get U...
```

```
when ButtonStart.Click
do
  if and [not is empty TextBoxName.Text, not is empty TextBoxAddress.Text, not is empty TextBoxEmail.Text, not is empty TextBoxTelephone.Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName.Text
    key "address" value TextBoxAddress.Text
    key "email" value TextBoxEmail.Text
    key "telephone" value TextBoxTelephone.Text
    key "catalog" value TextBoxCatalog.Text
    in call TinyDB1.StoreValue tag "user" valueToStore get user
    open another screen with start value screenName mapScreen startValue call goodURL
    URL TextBoxCatalog.Text
    rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
```

```
to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
```

```
when ButtonStart.Click
do
  if and [not is empty TextBoxName.Text, not is empty TextBoxAddress.Text, not is empty TextBoxEmail.Text, not is empty TextBoxTelephone.Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName.Text
    key "address" value TextBoxAddress.Text
    key "email" value TextBoxEmail.Text
    key "telephone" value TextBoxTelephone.Text
    key "catalog" value TextBoxCatalog.Text
    in call TinyDB1.StoreValue tag "user" valueToStore get user
    open another screen with start value screenName mapScreen startValue call goodURL
    URL TextBoxCatalog.Text
    rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]
```

```
when Screen1.Initialize do set TextBoxCatalog .Text to call goodURL .URL get global URLgeoJSONCatalog .rootURL " " call readUserData
```

```
to goodURL URL rootURL result do if starts at text get U...
```

```
to readUserData do
```

```
when ButtonStart.Click do if and [not [TextBoxName .Text], not [TextBoxAddress .Text], not [TextBoxEmail .Text], not [TextBoxTelephone .Text]] then initialize local user to make a dictionary [key "name" value TextBoxName .Text, key "address" value TextBoxAddress .Text, key "email" value TextBoxEmail .Text, key "telephone" value TextBoxTelephone .Text, key "catalog" value TextBoxCatalog .Text] in call TinyDB1 .StoreValue tag "user" valueToStore get user open another screen with start value screenName mapScreen startValue call goodURL .URL TextBoxCatalog .Text .rootURL " " else call Notifier1 .ShowAlert notice "missing name, email, address or telephone"
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]
```

```
when Screen1.Initialize do set TextBoxCatalog .Text to call goodURL .URL get global URLgeoJSONCatalog .rootURL " " call readUserData
```

```
when ButtonStart.Click do if and [not [TextBoxName .Text], not [TextBoxAddress .Text], not [TextBoxEmail .Text], not [TextBoxTelephone .Text]] then initialize local user to make a dictionary [key "name" value TextBoxName .Text, key "address" value TextBoxAddress .Text, key "email" value TextBoxEmail .Text, key "telephone" value TextBoxTelephone .Text, key "catalog" value TextBoxCatalog .Text] in call TinyDB1 .StoreValue tag "user" valueToStore get user open another screen with start value screenName mapScreen startValue call goodURL .URL TextBoxCatalog .Text .rootURL " " else call Notifier1 .ShowAlert notice "missing name, email, address or telephone"
```

```
to goodURL URL rootURL result do if starts at text get U...
```

```
to readUserData do initialize local user to in
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.gejson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData

when ButtonStart.Click
do
  if and [not is empty TextBoxName.Text, not is empty TextBoxAddress.Text, not is empty TextBoxEmail.Text, not is empty TextBoxTelephone.Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName.Text
    key "address" value TextBoxAddress.Text
    key "email" value TextBoxEmail.Text
    key "telephone" value TextBoxTelephone.Text
    key "catalog" value TextBoxCatalog.Text
    in call TinyDB1.StoreValue tag "user" valueToStore get user
    open another screen with start value screenName mapScreen startValue call goodURL
    URL TextBoxCatalog.Text
    rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

```
to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1.GetValue tag "user"
  valueIfTagNotThere create empty dictionary
in
```


GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.gejson"]

when Screen1.Initialize
do
  set TextBoxCatalog .Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData

when ButtonStart.Click
do
  if and [not is empty TextBoxName .Text, not is empty TextBoxAddress .Text, not is empty TextBoxEmail .Text, not is empty TextBoxTelephone .Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName .Text
    key "address" value TextBoxAddress .Text
    key "email" value TextBoxEmail .Text
    key "telephone" value TextBoxTelephone .Text
    key "catalog" value TextBoxCatalog .Text
    in call TinyDB1.StoreValue tag "user" valueToStore get user
    open another screen with start value screenName mapScreen startValue call goodURL URL TextBoxCatalog .Text rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

```
to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1.GetValue tag "user" valueIfTagNotThere create empty dictionary
  in set TextBoxName .Text to get value for key "name" in dictionary get user or if not found ""
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog .Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData

when ButtonStart.Click
do
  if and [not is empty TextBoxName .Text, not is empty TextBoxAddress .Text, not is empty TextBoxEmail .Text, not is empty TextBoxTelephone .Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName .Text
    key "address" value TextBoxAddress .Text
    key "email" value TextBoxEmail .Text
    key "telephone" value TextBoxTelephone .Text
    key "catalog" value TextBoxCatalog .Text
    in call TinyDB1.StoreValue tag "user" valueToStore get user
    open another screen with start value screenName mapScreen startValue call goodURL URL TextBoxCatalog .Text rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

```
to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1.GetValue tag "user" valueIfTagNotThere create empty dictionary
  in
    set TextBoxName .Text to get value for key "name" in dictionary get user or if not found ""
    set TextBoxAddress .Text to get value for key "address" in dictionary get user or if not found ""
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog .Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData

when ButtonStart.Click
do
  if and [not is empty TextBoxName .Text, not is empty TextBoxAddress .Text, not is empty TextBoxEmail .Text, not is empty TextBoxTelephone .Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName .Text
    key "address" value TextBoxAddress .Text
    key "email" value TextBoxEmail .Text
    key "telephone" value TextBoxTelephone .Text
    key "catalog" value TextBoxCatalog .Text
    in call TinyDB1.StoreValue tag "user" valueToStore get user
    open another screen with start value screenName mapScreen startValue call goodURL URL TextBoxCatalog .Text rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

```
to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1.GetValue tag "user" valueIfTagNotThere create empty dictionary
  in
    set TextBoxName .Text to get value for key "name" in dictionary get user or if not found ""
    set TextBoxAddress .Text to get value for key "address" in dictionary get user or if not found ""
    set TextBoxEmail .Text to get value for key "email" in dictionary get user or if not found ""
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

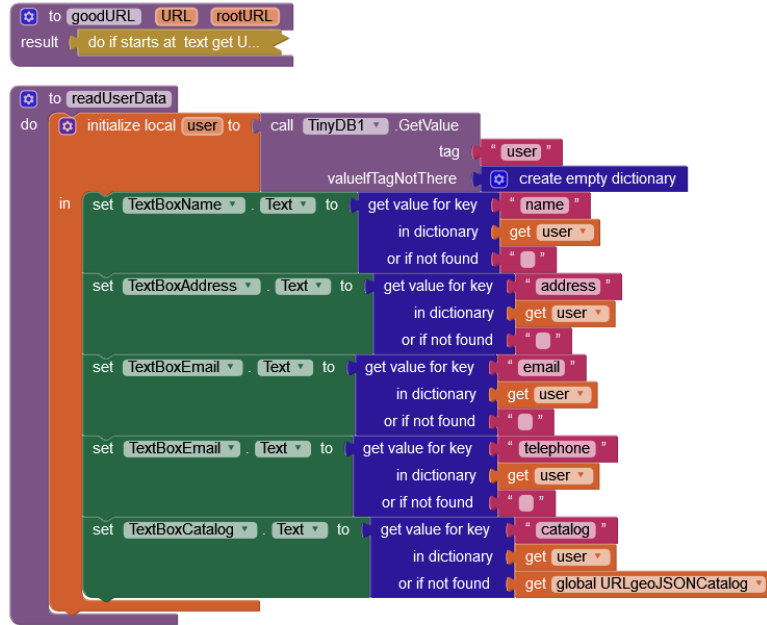
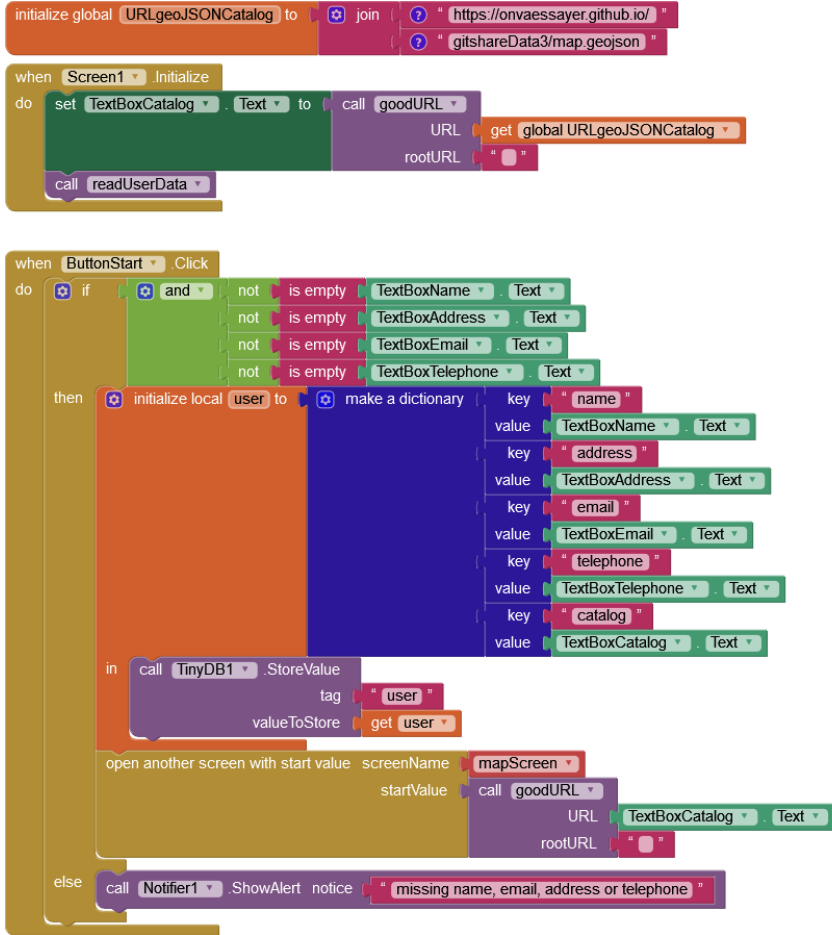
when Screen1.Initialize
do
  set TextBoxCatalog .Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData

when ButtonStart.Click
do
  if and [not is empty TextBoxName .Text, not is empty TextBoxAddress .Text, not is empty TextBoxEmail .Text, not is empty TextBoxTelephone .Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName .Text
    key "address" value TextBoxAddress .Text
    key "email" value TextBoxEmail .Text
    key "telephone" value TextBoxTelephone .Text
    key "catalog" value TextBoxCatalog .Text
    in call TinyDB1.StoreValue tag "user" valueToStore get user
    open another screen with start value screenName mapScreen startValue call goodURL URL TextBoxCatalog .Text rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

```
to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1.GetValue tag "user" valueIfTagNotThere create empty dictionary
  in
    set TextBoxName .Text to get value for key "name" in dictionary get user or if not found ""
    set TextBoxAddress .Text to get value for key "address" in dictionary get user or if not found ""
    set TextBoxEmail .Text to get value for key "email" in dictionary get user or if not found ""
    set TextBoxTelephone .Text to get value for key "telephone" in dictionary get user or if not found ""
```

GITSHARE 3a : SCREEN1



GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson "]

when Screen1.Initialize
do
  set TextBoxCatalog . Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData

when ButtonStart . Click
do
  if and [not is empty TextBoxName . Text, not is empty TextBoxAddress . Text, not is empty TextBoxEmail . Text, not is empty TextBoxTelephone . Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName . Text
    key "address" value TextBoxAddress . Text
    key "email" value TextBoxEmail . Text
    key "telephone" value TextBoxTelephone . Text
    key "catalog" value TextBoxCatalog . Text
    in call TinyDB1 . StoreValue
    tag "user"
    valueToStore get user
    open another screen with start value screenName mapScreen
    startValue call goodURL
    URL TextBoxCatalog . Text
    rootURL ""
  else
    call Notifier1 . ShowAlert notice "missing name, email, address or telephone "
```

```
to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1 . GetValue
  tag "user"
  valueIfTagNotThere create empty dictionary
  in
    set TextBoxName . Text to get value for key "name" in dictionary get user or if not found ""
    set TextBoxAddress . Text to get value for key "address" in dictionary get user or if not found ""
    set TextBoxEmail . Text to get value for key "email" in dictionary get user or if not found ""
    set TextBoxTelephone . Text to get value for key "telephone" in dictionary get user or if not found ""
    set TextBoxCatalog . Text to get value for key "catalog" in dictionary get user or if not found ""
    or if not found get global URLgeoJSONCatalog
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog .Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData

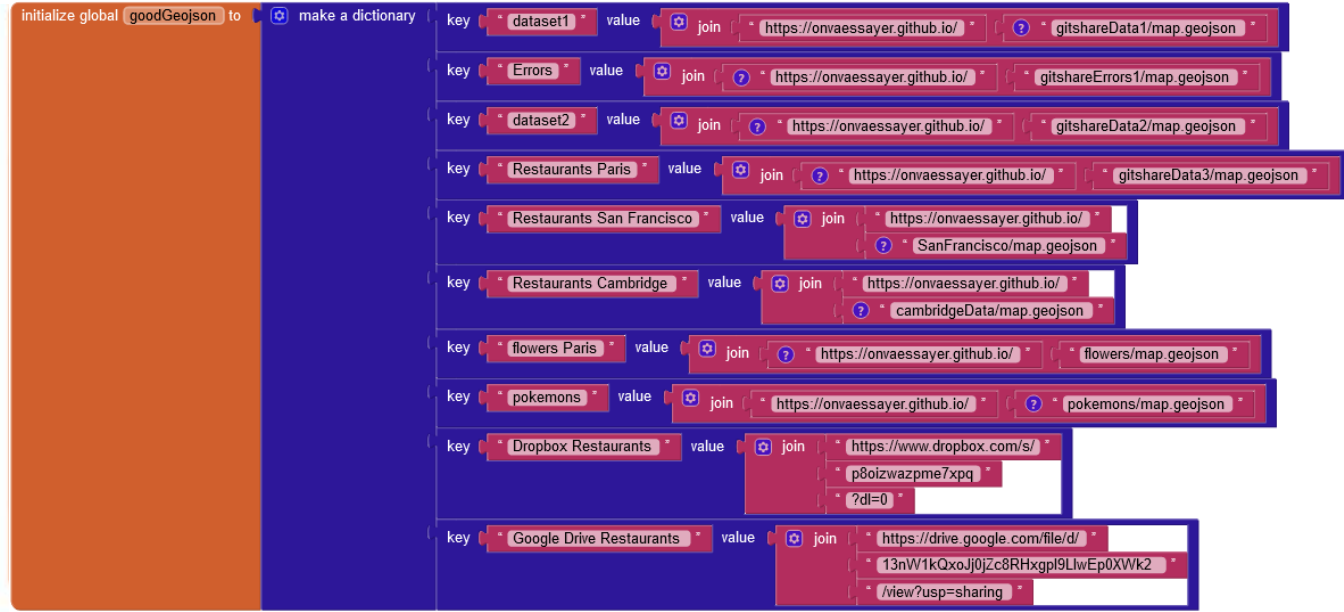
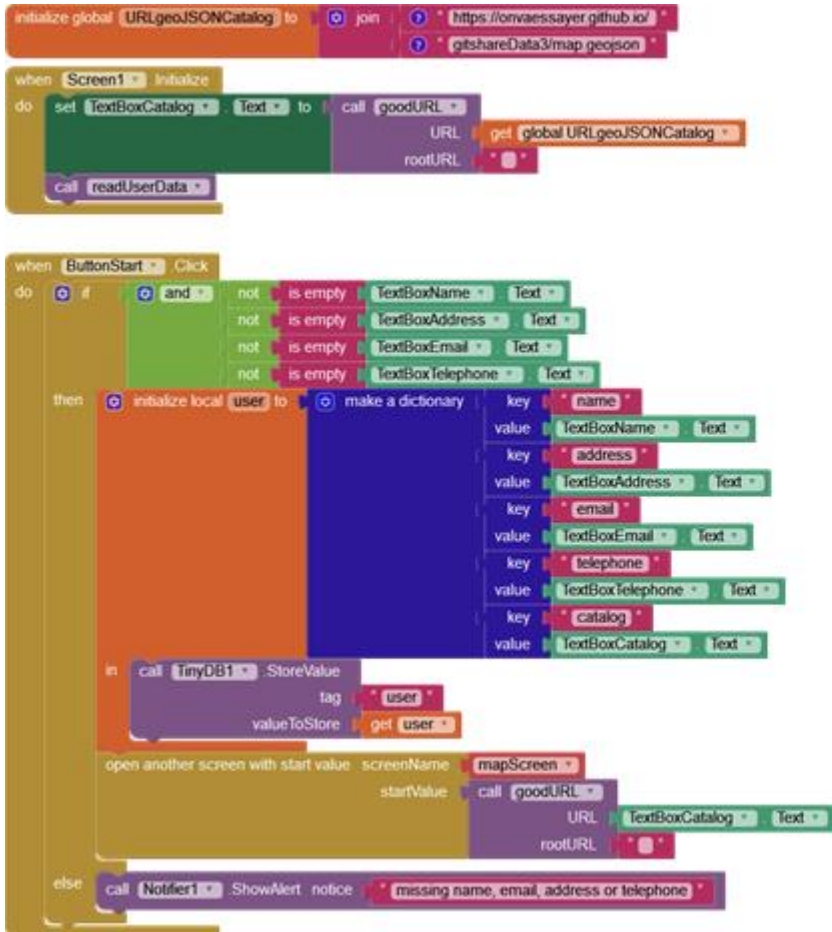
when ButtonStart.Click
do
  if and [not is empty TextBoxName .Text, not is empty TextBoxAddress .Text, not is empty TextBoxEmail .Text, not is empty TextBoxTelephone .Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName .Text
    key "address" value TextBoxAddress .Text
    key "email" value TextBoxEmail .Text
    key "telephone" value TextBoxTelephone .Text
    key "catalog" value TextBoxCatalog .Text
    in call TinyDB1.StoreValue tag "user" valueToStore get user
    open another screen with start value screenName mapScreen startValue call goodURL URL TextBoxCatalog .Text rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

```
initialize global goodGeojson to [{"dataset1":join "https://..."}]

to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1.GetValue tag "user" valueIfTagNotThere create empty dictionary
  in
    set TextBoxName .Text to get value for key "name" in dictionary get user or if not found ""
    set TextBoxAddress .Text to get value for key "address" in dictionary get user or if not found ""
    set TextBoxEmail .Text to get value for key "email" in dictionary get user or if not found ""
    set TextBoxTelephone .Text to get value for key "telephone" in dictionary get user or if not found ""
    set TextBoxCatalog .Text to get value for key "catalog" in dictionary get user or if not found get global URLgeoJSONCatalog
```

GITSHARE 3a : SCREEN1



GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1 Initialize
do
  set TextBoxCatalog Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL
  call readUserData
  set SpinnerListOfCatalogs Elements to get keys get global goodGeojson

when ButtonStart Click
do
  if and [not is empty TextBoxName Text, not is empty TextBoxAddress Text, not is empty TextBoxEmail Text, not is empty TextBoxTelephone Text]
  then
    initialize local user to make a dictionary
    key name value TextBoxName Text
    key address value TextBoxAddress Text
    key email value TextBoxEmail Text
    key telephone value TextBoxTelephone Text
    key catalog value TextBoxCatalog Text

    in call TinyDB1 StoreValue
    tag user
    valueToStore get user

    open another screen with start value screenName mapScreen
    startValue call goodURL
    URL TextBoxCatalog Text
    rootURL

  else
    call Notifier1 ShowAlert notice missing name, email, address or telephone
```

```
initialize global goodGeojson to make a dictionary
key "dataset1" value join ["https://onvaessayer.github.io/", "gitshareData1/map_geojson"]
key "Errors" value join ["https://onvaessayer.github.io/", "gitshareErrors1/map_geojson"]
key "dataset2" value join ["https://onvaessayer.github.io/", "gitshareData2/map_geojson"]
key "Restaurants Paris" value join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]
key "Restaurants San Francisco" value join ["https://onvaessayer.github.io/", "SanFrancisco/map_geojson"]
key "Restaurants Cambridge" value join ["https://onvaessayer.github.io/", "cambridgeData/map_geojson"]
key "flowers Paris" value join ["https://onvaessayer.github.io/", "flowers/map_geojson"]
key "pokemons" value join ["https://onvaessayer.github.io/", "pokemons/map_geojson"]
key "Dropbox Restaurants" value join ["https://www.dropbox.com/s/", "p8oizwazpme7xpq", "?dl=0"]
key "Google Drive Restaurants" value join ["https://drive.google.com/file/d/", "13nW1kQxoJ0jZc8RHxgpl9LwEp0XWk2", "/view?usp=sharing"]
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData
  set SpinnerListOfCatalogs.Elements to get keys get global goodGeojson

when ButtonStart.Click
do
  if and [not is empty TextBoxName.Text, not is empty TextBoxAddress.Text, not is empty TextBoxEmail.Text, not is empty TextBoxTelephone.Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName.Text
    key "address" value TextBoxAddress.Text
    key "email" value TextBoxEmail.Text
    key "telephone" value TextBoxTelephone.Text
    key "catalog" value TextBoxCatalog.Text
    in call TinyDB1.StoreValue tag "user" valueToStore get user
    open another screen with start value screenName mapScreen startValue call goodURL URL TextBoxCatalog.Text rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

```
initialize global goodGeojson to [{"dataset1":join "https://..."}]

to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1.GetValue tag "user" valueIfTagNotThere create empty dictionary
  in
    set TextBoxName.Text to get value for key "name" in dictionary get user or if not found ""
    set TextBoxAddress.Text to get value for key "address" in dictionary get user or if not found ""
    set TextBoxEmail.Text to get value for key "email" in dictionary get user or if not found ""
    set TextBoxTelephone.Text to get value for key "telephone" in dictionary get user or if not found ""
    set TextBoxCatalog.Text to get value for key "catalog" in dictionary get user or if not found global URLgeoJSONCatalog
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog .Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData
  set SpinnerListOfCatalogs .Elements to get keys get global goodGeojson

when ButtonStart .Click
do
  if and [not is empty TextBoxName .Text, not is empty TextBoxAddress .Text, not is empty TextBoxEmail .Text, not is empty TextBoxTelephone .Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName .Text
    key "address" value TextBoxAddress .Text
    key "email" value TextBoxEmail .Text
    key "telephone" value TextBoxTelephone .Text
    key "catalog" value TextBoxCatalog .Text
    in call TinyDB1 .StoreValue tag "user" valueToStore get user
    open another screen with start value screenName mapScreen startValue call goodURL URL TextBoxCatalog .Text rootURL ""
  else
    call Notifier1 .ShowAlert notice "missing name, email, address or telephone"
```

```
initialize global goodGeojson to [{"dataset1":join "https://..."}]

to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1 .GetValue tag "user" valueIfTagNotThere create empty dictionary
  in
    set TextBoxName .Text to get value for key "name" in dictionary get user or if not found ""
    set TextBoxAddress .Text to get value for key "address" in dictionary get user or if not found ""
    set TextBoxEmail .Text to get value for key "email" in dictionary get user or if not found ""
    set TextBoxTelephone .Text to get value for key "telephone" in dictionary get user or if not found ""
    set TextBoxCatalog .Text to get value for key "catalog" in dictionary get user or if not found get global URLgeoJSONCatalog

when SpinnerListOfCatalogs .AfterSelecting
do
  selection
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog .Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData
  set SpinnerListOfCatalogs .Elements to get keys get global goodGeojson

when ButtonStart .Click
do
  if and
    not is empty TextBoxName .Text
    not is empty TextBoxAddress .Text
    not is empty TextBoxEmail .Text
    not is empty TextBoxTelephone .Text
  then
    initialize local user to make a dictionary
    key "name"
    value TextBoxName .Text
    key "address"
    value TextBoxAddress .Text
    key "email"
    value TextBoxEmail .Text
    key "telephone"
    value TextBoxTelephone .Text
    key "catalog"
    value TextBoxCatalog .Text
  in
    call TinyDB1 .StoreValue
    tag "user"
    valueToStore get user
  open another screen with start value
  screenName mapScreen
  startValue call goodURL
  URL TextBoxCatalog .Text
  rootURL ""
  else
    call Notifier1 .ShowAlert notice "missing name, email, address or telephone"
```

```
initialize global goodGeojson to ["dataset1", "join", "https://..."]

to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1 .GetValue
  tag "user"
  valueIfTagNotThere create empty dictionary
  in
    set TextBoxName .Text to get value for key "name" in dictionary get user or if not found ""
    set TextBoxAddress .Text to get value for key "address" in dictionary get user or if not found ""
    set TextBoxEmail .Text to get value for key "email" in dictionary get user or if not found ""
    set TextBoxTelephone .Text to get value for key "telephone" in dictionary get user or if not found ""
    set TextBoxCatalog .Text to get value for key "catalog" in dictionary get user or if not found ""
    or if not found get global URLgeoJSONCatalog

when SpinnerListOfCatalogs .AfterSelecting
selection
do
  set TextBoxCatalog .Text to
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog .Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData
  set SpinnerListOfCatalogs .Elements to get keys get global goodGeojson

when ButtonStart .Click
do
  if and
    not is empty TextBoxName .Text
    not is empty TextBoxAddress .Text
    not is empty TextBoxEmail .Text
    not is empty TextBoxTelephone .Text
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName .Text
    key "address" value TextBoxAddress .Text
    key "email" value TextBoxEmail .Text
    key "telephone" value TextBoxTelephone .Text
    key "catalog" value TextBoxCatalog .Text
  in
    call TinyDB1 .StoreValue
    tag "user"
    valueToStore get user
  open another screen with start value
  screenName mapScreen
  startValue call goodURL
  URL TextBoxCatalog .Text
  rootURL ""
  else
    call Notifier1 .ShowAlert notice "missing name, email, address or telephone"
```

```
initialize global goodGeojson to ["dataset1", join "https://..."]

to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1 .GetValue
  tag "user"
  valueIfTagNotThere create empty dictionary
  in
    set TextBoxName .Text to get value for key "name" in dictionary get user or if not found ""
    set TextBoxAddress .Text to get value for key "address" in dictionary get user or if not found ""
    set TextBoxEmail .Text to get value for key "email" in dictionary get user or if not found ""
    set TextBoxTelephone .Text to get value for key "telephone" in dictionary get user or if not found ""
    set TextBoxCatalog .Text to get value for key "catalog" in dictionary get user or if not found get global URLgeoJSONCatalog

when SpinnerListOfCatalogs .AfterSelecting
selection
do
  set TextBoxCatalog .Text to get value for key in dictionary or if not found
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog .Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData
  set SpinnerListOfCatalogs .Elements to get keys get global goodGeojson

when ButtonStart .Click
do
  if and
    not is empty TextBoxName .Text
    not is empty TextBoxAddress .Text
    not is empty TextBoxEmail .Text
    not is empty TextBoxTelephone .Text
  then
    initialize local user to make a dictionary
    key "name"
    value TextBoxName .Text
    key "address"
    value TextBoxAddress .Text
    key "email"
    value TextBoxEmail .Text
    key "telephone"
    value TextBoxTelephone .Text
    key "catalog"
    value TextBoxCatalog .Text
  in
    call TinyDB1 .StoreValue
    tag "user"
    valueToStore get user
  open another screen with start value
  screenName mapScreen
  startValue call goodURL
  URL TextBoxCatalog .Text
  rootURL ""
  else
    call Notifier1 .ShowAlert notice "missing name, email, address or telephone"
```

```
initialize global goodGeojson to ["dataset1", join "https://..."]

to goodURL URL rootURL
result
  do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1 .GetValue
  tag "user"
  valueIfTagNotThere create empty dictionary
  in
    set TextBoxName .Text to get value for key "name" in dictionary get user or if not found ""
    set TextBoxAddress .Text to get value for key "address" in dictionary get user or if not found ""
    set TextBoxEmail .Text to get value for key "email" in dictionary get user or if not found ""
    set TextBoxTelephone .Text to get value for key "telephone" in dictionary get user or if not found ""
    set TextBoxCatalog .Text to get value for key "catalog" in dictionary get user or if not found get global URLgeoJSONCatalog

when SpinnerListOfCatalogs .AfterSelecting
selection
do
  set TextBoxCatalog .Text to get value for key in dictionary get global goodGeojson or if not found
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog .Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData
  set SpinnerListOfCatalogs .Elements to get keys get global goodGeojson

when ButtonStart .Click
do
  if and
    not is empty TextBoxName .Text
    not is empty TextBoxAddress .Text
    not is empty TextBoxEmail .Text
    not is empty TextBoxTelephone .Text
  then
    initialize local user to make a dictionary
    key "name"
    value TextBoxName .Text
    key "address"
    value TextBoxAddress .Text
    key "email"
    value TextBoxEmail .Text
    key "telephone"
    value TextBoxTelephone .Text
    key "catalog"
    value TextBoxCatalog .Text
  in
    call TinyDB1 .StoreValue
    tag "user"
    valueToStore get user
  open another screen with start value
  screenName mapScreen
  startValue call goodURL
  URL TextBoxCatalog .Text
  rootURL ""
  else
    call Notifier1 .ShowAlert notice "missing name, email, address or telephone"
```

```
initialize global goodGeojson to ["dataset1", join "https://..."]

to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1 .GetValue
  tag "user"
  valueIfTagNotThere create empty dictionary
  in
    set TextBoxName .Text to get value for key "name" in dictionary get user or if not found ""
    set TextBoxAddress .Text to get value for key "address" in dictionary get user or if not found ""
    set TextBoxEmail .Text to get value for key "email" in dictionary get user or if not found ""
    set TextBoxTelephone .Text to get value for key "telephone" in dictionary get user or if not found ""
    set TextBoxCatalog .Text to get value for key "catalog" in dictionary get user or if not found get global URLgeoJSONCatalog

when SpinnerListOfCatalogs .AfterSelecting
selection
do
  set TextBoxCatalog .Text to get value for key get selection in dictionary get global goodGeojson or if not found
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog . Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData
  set SpinnerListOfCatalogs . Elements to get keys get global goodGeojson

when ButtonStart . Click
do
  if and
    not is empty TextBoxName . Text
    not is empty TextBoxAddress . Text
    not is empty TextBoxEmail . Text
    not is empty TextBoxTelephone . Text
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName . Text
    key "address" value TextBoxAddress . Text
    key "email" value TextBoxEmail . Text
    key "telephone" value TextBoxTelephone . Text
    key "catalog" value TextBoxCatalog . Text
  in
    call TinyDB1 . StoreValue
    tag "user"
    valueToStore get user
  open another screen with start value
  screenName mapScreen
  startValue call goodURL
  URL TextBoxCatalog . Text
  rootURL ""
  else
    call Notifier1 . ShowAlert notice "missing name, email, address or telephone"
```

```
initialize global goodGeojson to {"dataset1":join "https://..."}

to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1 . GetValue
  tag "user"
  valueIfTagNotThere create empty dictionary
  in
    set TextBoxName . Text to get value for key "name" in dictionary get user or if not found ""
    set TextBoxAddress . Text to get value for key "address" in dictionary get user or if not found ""
    set TextBoxEmail . Text to get value for key "email" in dictionary get user or if not found ""
    set TextBoxTelephone . Text to get value for key "telephone" in dictionary get user or if not found ""
    set TextBoxCatalog . Text to get value for key "catalog" in dictionary get user or if not found get global URLgeoJSONCatalog
    set CheckBoxDebug . Checked to call TinyDB1 . GetValue
    tag "debug"
    valueIfTagNotThere true

when SpinnerListOfCatalogs . AfterSelecting
selection
do
  set TextBoxCatalog . Text to get value for key get selection
  in dictionary get global goodGeojson
  or if not found get global URLgeoJSONCatalog
```


GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog .Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData
  set SpinnerListOfCatalogs .Elements to get keys get global goodGeojson

when ButtonStart .Click
do
  if and [not is empty TextBoxName .Text, not is empty TextBoxAddress .Text, not is empty TextBoxEmail .Text, not is empty TextBoxTelephone .Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName .Text
    key "address" value TextBoxAddress .Text
    key "email" value TextBoxEmail .Text
    key "telephone" value TextBoxTelephone .Text
    key "catalog" value TextBoxCatalog .Text
  in
    call TinyDB1 .StoreValue
    tag "user"
    valueToStore get user
  open another screen with start value
  screenName mapScreen
  startValue call goodURL
  URL TextBoxCatalog .Text
  rootURL ""
  else
    call Notifier1 .ShowAlert notice "missing name, email, address or telephone"
```

```
initialize global goodGeojson to ["dataset1", join "https://..."]

to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1 .GetValue
  tag "user"
  valueIfTagNotThere create empty dictionary
  in
    set TextBoxName .Text to get value for key "name" in dictionary get user or if not found ""
    set TextBoxAddress .Text to get value for key "address" in dictionary get user or if not found ""
    set TextBoxEmail .Text to get value for key "email" in dictionary get user or if not found ""
    set TextBoxTelephone .Text to get value for key "telephone" in dictionary get user or if not found ""
    set TextBoxCatalog .Text to get value for key "catalog" in dictionary get user or if not found get global URLgeoJSONCatalog

when SpinnerListOfCatalogs .AfterSelecting
selection
do
  set TextBoxCatalog .Text to get value for key get selection in dictionary get global goodGeojson or if not found get global URLgeoJSONCatalog
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog . Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData
  set SpinnerListOfCatalogs . Elements to get keys get global goodGeojson

when ButtonStart . Click
do
  if and
    not is empty TextBoxName . Text
    not is empty TextBoxAddress . Text
    not is empty TextBoxEmail . Text
    not is empty TextBoxTelephone . Text
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName . Text
    key "address" value TextBoxAddress . Text
    key "email" value TextBoxEmail . Text
    key "telephone" value TextBoxTelephone . Text
    key "catalog" value TextBoxCatalog . Text
  in
    call TinyDB1 . StoreValue
    tag "user"
    valueToStore get user
  open another screen with start value
  screenName mapScreen
  startValue call goodURL
  URL TextBoxCatalog . Text
  rootURL ""
  else
    call Notifier1 . ShowAlert notice "missing name, email, address or telephone"
```

```
initialize global goodGeojson to [{"dataset1":join "https://..."}]

to goodURL URL rootURL
result do if starts at text get URL

to readUserData
do
  initialize local user to call TinyDB1 . GetValue
  tag "user"
  valueIfTagNotThere create empty dictionary
  in
    set TextBoxName . Text to get value for key "name" in dictionary get user or if not found ""
    set TextBoxAddress . Text to get value for key "address" in dictionary get user or if not found ""
    set TextBoxEmail . Text to get value for key "email" in dictionary get user or if not found ""
    set TextBoxTelephone . Text to get value for key "telephone" in dictionary get user or if not found ""
    set TextBoxCatalog . Text to get value for key "catalog" in dictionary get user or if not found get global URLgeoJSONCatalog

when SpinnerListOfCatalogs . AfterSelecting
selection
do
  set TextBoxCatalog . Text to get value for key get selection
  in dictionary get global goodGeojson
  or if not found get global URLgeoJSONCatalog

when CheckBoxDebug . Changed
do
  call TinyDB1 . StoreValue
  tag "debug"
  valueToStore CheckBoxDebug . Checked
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog . Text to call goodURL .
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData
  set SpinnerListOfCatalogs . Elements to get keys get global goodGeojson

when ButtonStart . Click
do
  if and [not is empty TextBoxName . Text, not is empty TextBoxAddress . Text, not is empty TextBoxEmail . Text, not is empty TextBoxTelephone . Text]
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName . Text
    key "address" value TextBoxAddress . Text
    key "email" value TextBoxEmail . Text
    key "telephone" value TextBoxTelephone . Text
    key "catalog" value TextBoxCatalog . Text
  in
    call TinyDB1 . StoreValue
    tag "user"
    valueToStore get user
  open another screen with start value screenName mapScreen
  startValue call goodURL
  URL TextBoxCatalog . Text
  rootURL ""
  else
    call Notifier1 . ShowAlert notice "missing name, email, address or telephone"
```

```
initialize global goodGeojson to [{"dataset1":join "https://..."}]

to goodURL URL rootURL
result do if starts at text get URL

to readUserData
do
  initialize local user to call TinyDB1 . GetValue
  tag "user"
  valueIfTagNotThere create empty dictionary
  in
    set TextBoxName . Text to get value for key "name" in dictionary get user or if not found ""
    set TextBoxAddress . Text to get value for key "address" in dictionary get user or if not found ""
    set TextBoxEmail . Text to get value for key "email" in dictionary get user or if not found ""
    set TextBoxTelephone . Text to get value for key "telephone" in dictionary get user or if not found ""
    set TextBoxCatalog . Text to get value for key "catalog" in dictionary get user or if not found get global URLgeoJSONCatalog

when SpinnerListOfCatalogs . AfterSelecting
selection
do
  set TextBoxCatalog . Text to get value for key get selection
  in dictionary get global goodGeojson
  or if not found get global URLgeoJSONCatalog

when CheckBoxDebug . Changed
do
  call TinyDB1 . StoreValue
  tag "debug"
  valueToStore CheckBoxDebug . Checked
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData
  set SpinnerListOfCatalogs.Elements to get keys get global goodGeojson

when ButtonStart.Click
do
  if and not is empty TextBoxName.Text
    not is empty TextBoxAddress.Text
    not is empty TextBoxEmail.Text
    not is empty TextBoxTelephone.Text
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName.Text
    key "address" value TextBoxAddress.Text
    key "email" value TextBoxEmail.Text
    key "telephone" value TextBoxTelephone.Text
    key "catalog" value TextBoxCatalog.Text
  in
    call TinyDB1.StoreValue
    tag "user"
    valueToStore get user
  open another screen with start value screenName mapScreen
  startValue call goodURL
  URL TextBoxCatalog.Text
  rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

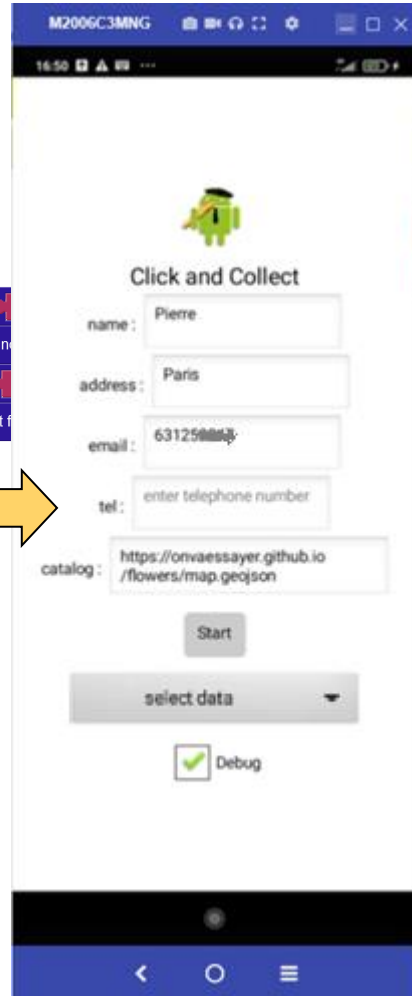
```
initialize global goodGeojson to {"select data":"","data..."}

to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1.GetValue
  tag "user"
  valueIfTagNotThere create empty dictionary
  in
    set TextBoxName.Text to get value for key "name" in dictionary get user or if not found
    set TextBoxAddress.Text to get value for key "address" in dictionary get user or if not found
    set TextBoxEmail.Text to get value for key "email" in dictionary get user or if not found
    set TextBoxTelephone.Text to get value for key "telephone" in dictionary get user or if not found
    set TextBoxCatalog.Text to get value for key "catalog" in dictionary get user or if not found
  or if not found get global URLgeoJSONCatalog

when SpinnerListOfCatalogs.AfterSelecting
selection
do
  set TextBoxCatalog.Text to get value for key get selection
  in dictionary get global goodGeojson
  or if not found get global URLgeoJSONCatalog

when CheckBoxDebug.Changed
do
  call TinyDB1.StoreValue
  tag "debug"
  valueToStore CheckBoxDebug.Checked
```



GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData
  set SpinnerListOfCatalogs.Elements to get keys get global goodGeojson

when ButtonStart.Click
do
  if and not isEmpty TextBoxName.Text
    not isEmpty TextBoxAddress.Text
    not isEmpty TextBoxEmail.Text
    not isEmpty TextBoxTelephone.Text
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName.Text
    key "address" value TextBoxAddress.Text
    key "email" value TextBoxEmail.Text
    key "telephone" value TextBoxTelephone.Text
    key "catalog" value TextBoxCatalog.Text
    in call TinyDB1.StoreValue tag "user" valueToStore get user
    open another screen with start value screenName mapScreen startValue call goodURL
    URL TextBoxCatalog.Text
    rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

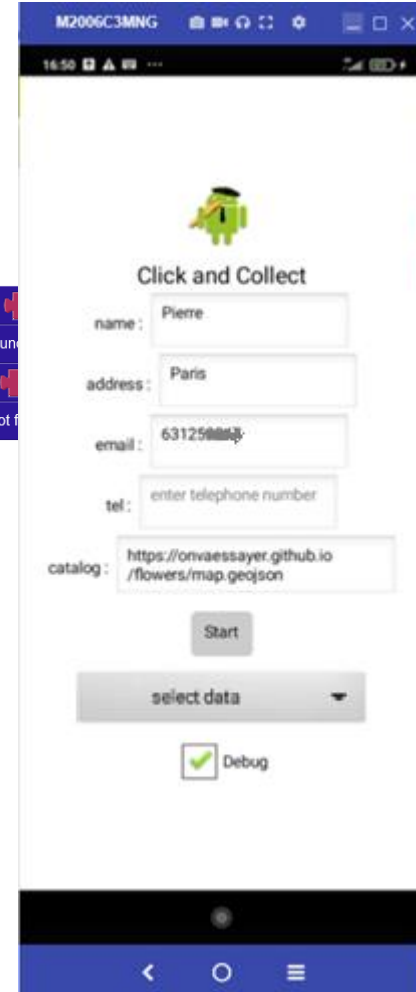
```
initialize global goodGeojson to {"select data":"","data..."}

to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1.GetValue tag "user" valueIfTagNotThere create empty dictionary
  in
    set TextBoxName.Text to get value for key "name" in dictionary get user or if not found
    set TextBoxAddress.Text to get value for key "address" in dictionary get user or if not found
    set TextBoxEmail.Text to get value for key "email" in dictionary get user or if not found
    set TextBoxTelephone.Text to get value for key "telephone" in dictionary get user or if not found
    set TextBoxCatalog.Text to get value for key "catalog" in dictionary get user or if not found
    get global URLgeoJSONCatalog

when SpinnerListOfCatalogs.AfterSelecting
selection
do
  set TextBoxCatalog.Text to get value for key get selection in dictionary get global goodGeojson or if not found
  get global URLgeoJSONCatalog

when CheckBoxDebug.Changed
do
  call TinyDB1.StoreValue tag "debug" valueToStore CheckBoxDebug.Checked
```



GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData
  set SpinnerListOfCatalogs.Elements to get keys get global goodGeojson

when ButtonStart.Click
do
  if and not is empty TextBoxName.Text
    not is empty TextBoxAddress.Text
    not is empty TextBoxEmail.Text
    not is empty TextBoxTelephone.Text
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName.Text
    key "address" value TextBoxAddress.Text
    key "email" value TextBoxEmail.Text
    key "telephone" value TextBoxTelephone.Text
    key "catalog" value TextBoxCatalog.Text
    in call TinyDB1.StoreValue tag "user" valueToStore get user
    open another screen with start value screenName mapScreen startValue call goodURL
    URL TextBoxCatalog.Text
    rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```



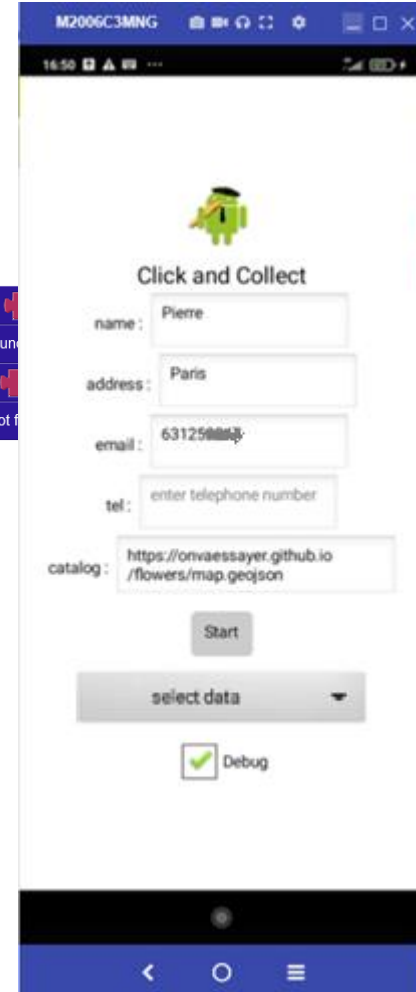
```
initialize global goodGeojson to {"select data":"","data..."}

to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1.GetValue tag "user"
  valueIfTagNotThere create empty dictionary
  in
    set TextBoxName.Text to get value for key "name" in dictionary get user or if not found
    set TextBoxAddress.Text to get value for key "address" in dictionary get user or if not found
    set TextBoxEmail.Text to get value for key "email" in dictionary get user or if not found
    set TextBoxTelephone.Text to get value for key "telephone" in dictionary get user or if not found
    set TextBoxCatalog.Text to get value for key "catalog" in dictionary get user or if not found
    get global URLgeoJSONCatalog

when SpinnerListOfCatalogs.AfterSelecting
selection
do
  set TextBoxCatalog.Text to get value for key get selection
  in dictionary get global goodGeojson
  or if not found get global URLgeoJSONCatalog

when CheckBoxDebug.Changed
do
  call TinyDB1.StoreValue tag "debug" valueToStore CheckBoxDebug.Checked
```



GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData
  set SpinnerListOfCatalogs.Elements to get keys get global goodGeojson

when ButtonStart.Click
do
  if and not is empty TextBoxName.Text
    not is empty TextBoxAddress.Text
    not is empty TextBoxEmail.Text
    not is empty TextBoxTelephone.Text
    not is empty TextBoxCatalog.Text
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName.Text
    key "address" value TextBoxAddress.Text
    key "email" value TextBoxEmail.Text
    key "telephone" value TextBoxTelephone.Text
    key "catalog" value TextBoxCatalog.Text
  in
    call TinyDB1.StoreValue tag "user" valueToStore get user
    open another screen with start value screenName mapScreen startValue call goodURL
    URL TextBoxCatalog.Text
    rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
```

```
initialize global goodGeojson to {"select data":"","data..."}

to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1.GetValue tag "user" valueIfTagNotThere create empty dictionary
  in
    set TextBoxName.Text to get value for key "name" in dictionary get user or if not found ""
    set TextBoxAddress.Text to get value for key "address" in dictionary get user or if not found ""
    set TextBoxEmail.Text to get value for key "email" in dictionary get user or if not found ""
    set TextBoxTelephone.Text to get value for key "telephone" in dictionary get user or if not found ""
    set TextBoxCatalog.Text to get value for key "catalog" in dictionary get user or if not found global URLgeoJSONCatalog

when SpinnerListOfCatalogs.AfterSelecting
selection
do
  set TextBoxCatalog.Text to get value for key get selection in dictionary get global goodGeojson or if not found global URLgeoJSONCatalog

when CheckBoxDebug.Changed
do
  call TinyDB1.StoreValue tag "debug" valueToStore CheckBoxDebug.Checked
```

GITSHARE 3a : SCREEN1

```
initialize global URLgeoJSONCatalog to join [" https://onvaessayer.github.io/ ", " gitshareData3/map_geojson "]

when Screen1 Initialize
do
  set TextBoxCatalog Text to call goodURL
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData
  set SpinnerListOfCatalogs Elements to get keys get global goodGeojson

when ButtonStart Click
do
  if and
    not is empty TextBoxName Text
    not is empty TextBoxAddress Text
    not is empty TextBoxEmail Text
    not is empty TextBoxTelephone Text
  then
    initialize local user to make a dictionary
    key " name " value TextBoxName Text
    key " address " value TextBoxAddress Text
    key " email " value TextBoxEmail Text
    key " telephone " value TextBoxTelephone Text
    key " catalog " value TextBoxCatalog Text
  in
    call TinyDB1 StoreValue
    tag " user "
    valueToStore get user
  open another screen with start value screenName mapScreen
  startValue call goodURL
  URL TextBoxCatalog Text
  rootURL ""
  else
    call Notifier1 ShowAlert notice " missing name, email, address or telephone "
```

```
initialize global goodGeojson to [" dataset1 ".join " https://... "]

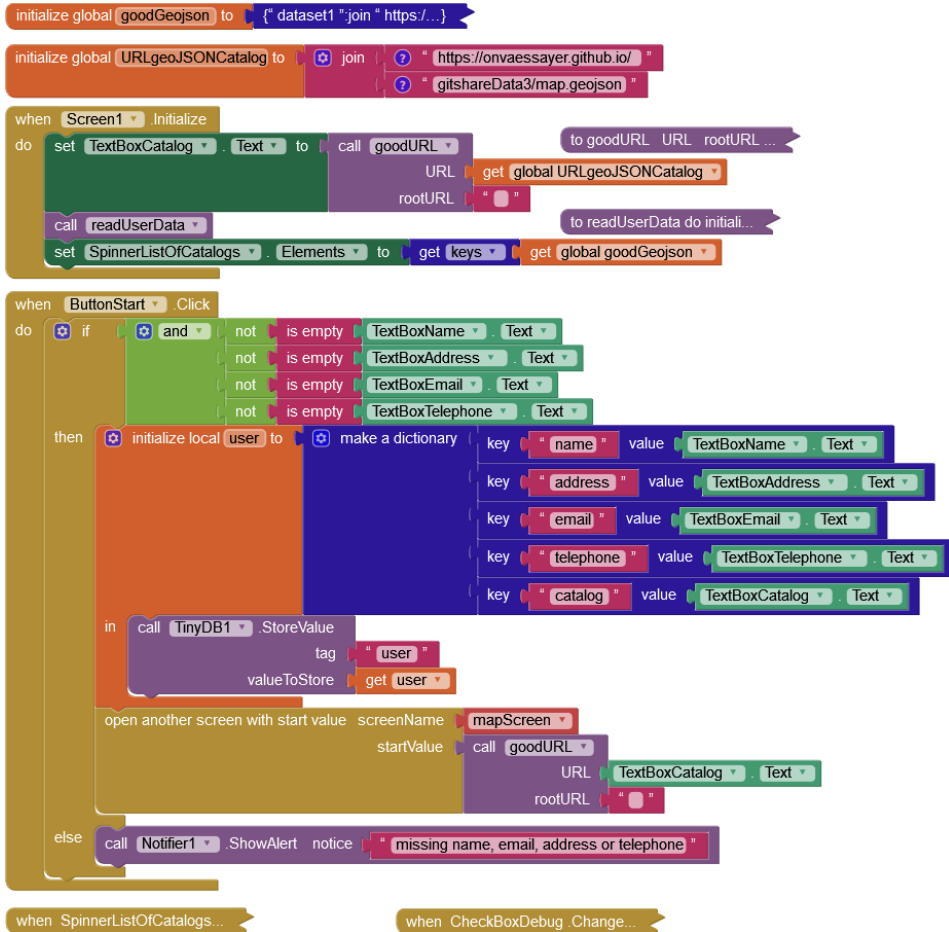
to goodURL URL rootURL
result do if starts at text get U...

to readUserData do initiali...

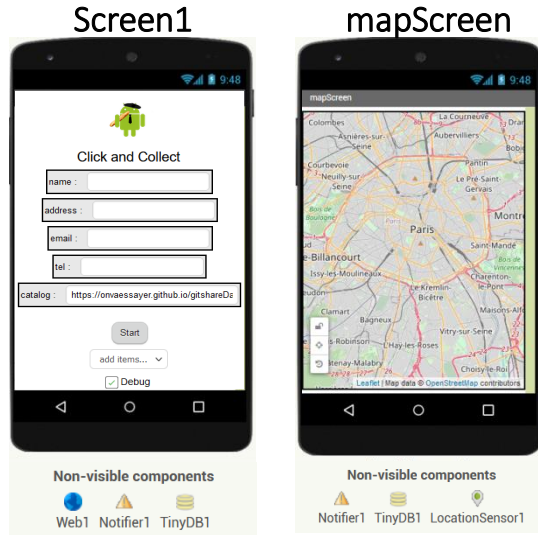
when SpinnerListOfCatalogs...

when CheckBoxDebug.Change...
```


GITSHARE 3a : SCREEN1



GITSHARE 3a : nouvelles fonctions / mapScreen



- lire l'URL du catalogue en paramètre d'appel
- enregistrer la position de la carte (à chaque déplacement)

```
get start value
```

& relire sa position (au démarrage de mapScreen)

```
call TinyDB1 .StoreValue
tag "latitude"
valueToStore Map1 . Latitude
```

```
call TinyDB1 .GetValue
tag "latitude"
valueIfTagNotThere 48.85
```

- centrer la carte sur la position de l'utilisateur

```
call Map1 .PanTo
latitude
longitude
zoom
```

GITSHARE 3a : mapScreen - design

gitshare03a mapScreen ▾ Add Screen ... Remove Screen Publish to Gallery Designer Blocks

Palette

Search Components...

User Interface

- Button
- CheckBox
- DatePicker
- Image
- Label
- ListPicker
- ListView
- Notifier
- PasswordTextBox
- Slider
- Spinner
- Switch
- TextBox
- TimePicker
- WebView

Layout

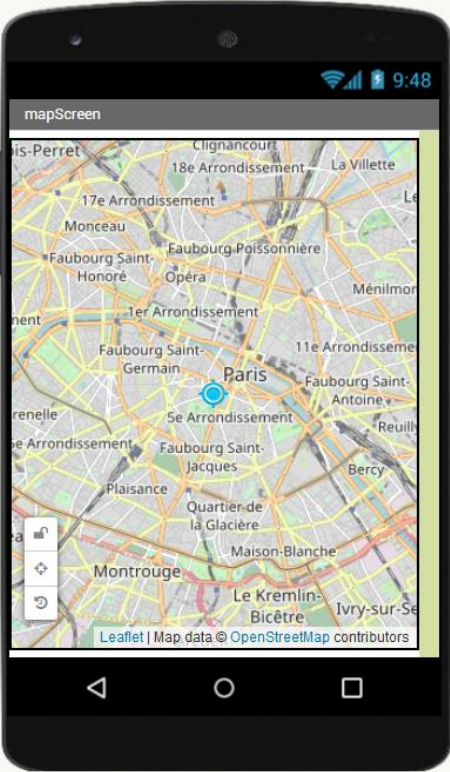
Media

Drawing and Animation

Sensors

Viewer

Display hidden components in Viewer



Components

- mapScreen
- Map1
- Notifier1
- TinyDB1
- LocationSensor1

Properties

mapScreen

AboutScreen

AlignHorizontal: Left : 1

AlignVertical: Top : 1

BackgroundColor: Default

BackgroundImage: None...

BigDefaultText:

CloseScreenAnimation: Default

HighContrast:

OpenScreenAnimation: Default

ScreenOrientation: Portrait

Scrollable:

ShowStatusBar:

Title: mapScreen

TitleVisible:

Media

- AndroidFR.png
- P_Food32.png
- crepe.png
- flower1.png
- flower5.png

Non-visible components

- Notifier1
- TinyDB1
- LocationSensor1

Rename Delete



3.4.1_b

Récupérer dans mapScreen le
catalogue électionné dans Screen1

GITSHARE 3a : MAPSCREEN

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when mapScreen.Initialize
do call Map1.LoadFromURL
    url call goodURL
        URL get global URLgeoJSONCatalog
        refURL ""
```

```
initialize global badgeoJSON to join ["https://onvaessayer.github.io/", "gitshareErrors1/map.geojson"]

initialize global goodGeojson to make a list join "https://..."
```

```
when any Marker.Click
component notAlreadyHandled
do set Marker.EnableInfoBox of component get component to true
    initialize local shopURL to Marker.Description of component get component
    in open another screen with start value screenName shop
        startValue call goodURL
            URL get shopURL
            refURL get global URLgeoJSONCatalog
```

```
to goodURL URL refURL
result do if starts at text get U...
```

GITSHARE 3a : MAPSCREEN

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]
```

```
initialize global badgeoJSON to join ["https://onvaessayer.github.io/", "gitshareErrors1/map.geojson"]
```

```
when mapScreen.Initialize  
do call Map1.LoadFromURL  
  url call goodURL  
      URL get global URLgeoJSONCatalog  
      refURL ""
```

```
when any Marker.Click  
  component notAlreadyHandled  
do set Marker.EnableInfobox of component get component to true  
  initialize local shopURL to Marker.Description  
  of component get component  
  in open another screen with start value screenName shop  
  startValue call goodURL  
            URL get shopURL  
            refURL get global URLgeoJSONCatalog
```

```
to goodURL URL refURL  
result do if starts at text get U...
```

GITSHARE 3a : MAPSCREEN

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]
```

```
when mapScreen.Initialize  
do call Map1.LoadFromURL  
  url call goodURL.URL  
  URL get global URLgeoJSONCatalog  
  refURL ""
```

```
when any Marker.Click  
  component notAlreadyHandled  
do  
  set Marker.EnableInfoBox of component get component to true  
  initialize local shopURL to Marker.Description of component  
  in open another screen with start value screenName shop  
    startValue call goodURL.URL  
    URL get shopURL  
    refURL get global URLgeoJSONCatalog
```

```
to goodURL URL refURL  
result do if starts at text get U...
```

GITSHARE 3a : MAPSCREEN

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]
```

```
when mapScreen.Initialize  
do  
  set global URLgeoJSONCatalog to get start value  
  call Map1.LoadFromURL  
    url call goodURL  
        URL get global URLgeoJSONCatalog  
        refURL ""
```

```
when any Marker.Click  
  component notAlreadyHandled  
do  
  set Marker.EnableInfoBox of component get component to true  
  initialize local shopURL to Marker.Description of component  
  in open another screen with start value  
    screenName shop  
    startValue call goodURL  
        URL get shopURL  
        refURL get global URLgeoJSONCatalog
```

```
to goodURL URL refURL  
result do if starts at text get U...
```


GITSHARE 3a : MAPSCREEN

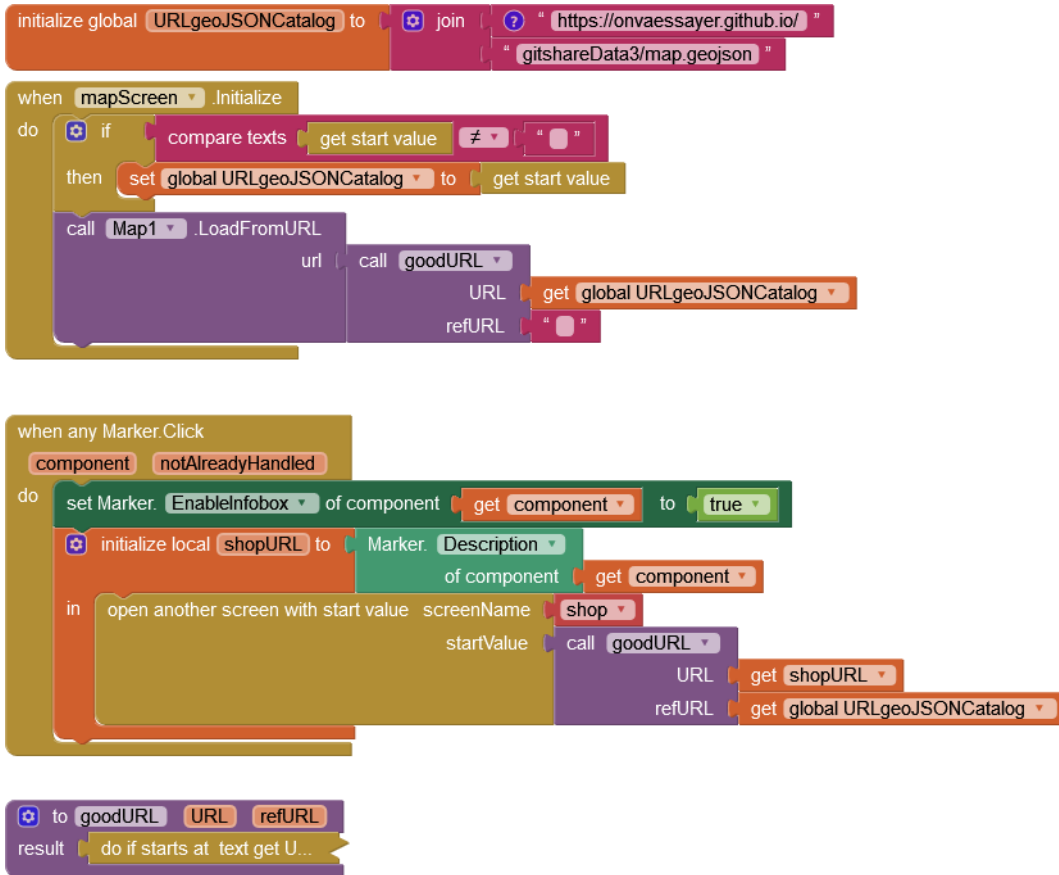
```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]
```

```
when mapScreen.Initialize  
do  
  if  
  then set global URLgeoJSONCatalog to get start value  
  call Map1.LoadFromURL  
    url call goodURL  
        URL get global URLgeoJSONCatalog  
        refURL ""
```

```
when any Marker.Click  
component notAlreadyHandled  
do  
  set Marker.EnableInfoBox of component get component to true  
  initialize local shopURL to Marker.Description of component get component  
  in open another screen with start value screenName shop  
    startValue call goodURL  
        URL get shopURL  
        refURL get global URLgeoJSONCatalog
```

```
to goodURL URL refURL  
result do if starts at text get U...
```

GITSHARE 3a : MAPSCREEN

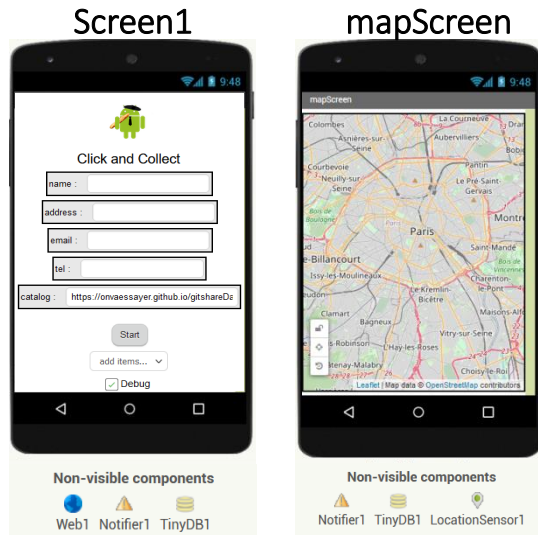




3.4.2

Enregistrer et relire
la position de la carte

GITSHARE 3a : nouvelles fonctions / mapScreen



- lire l'URL du catalogue en paramètre d'appel

get start value

- enregistrer la position de la carte (à chaque déplacement)

& relire sa position (au démarrage de mapScreen)

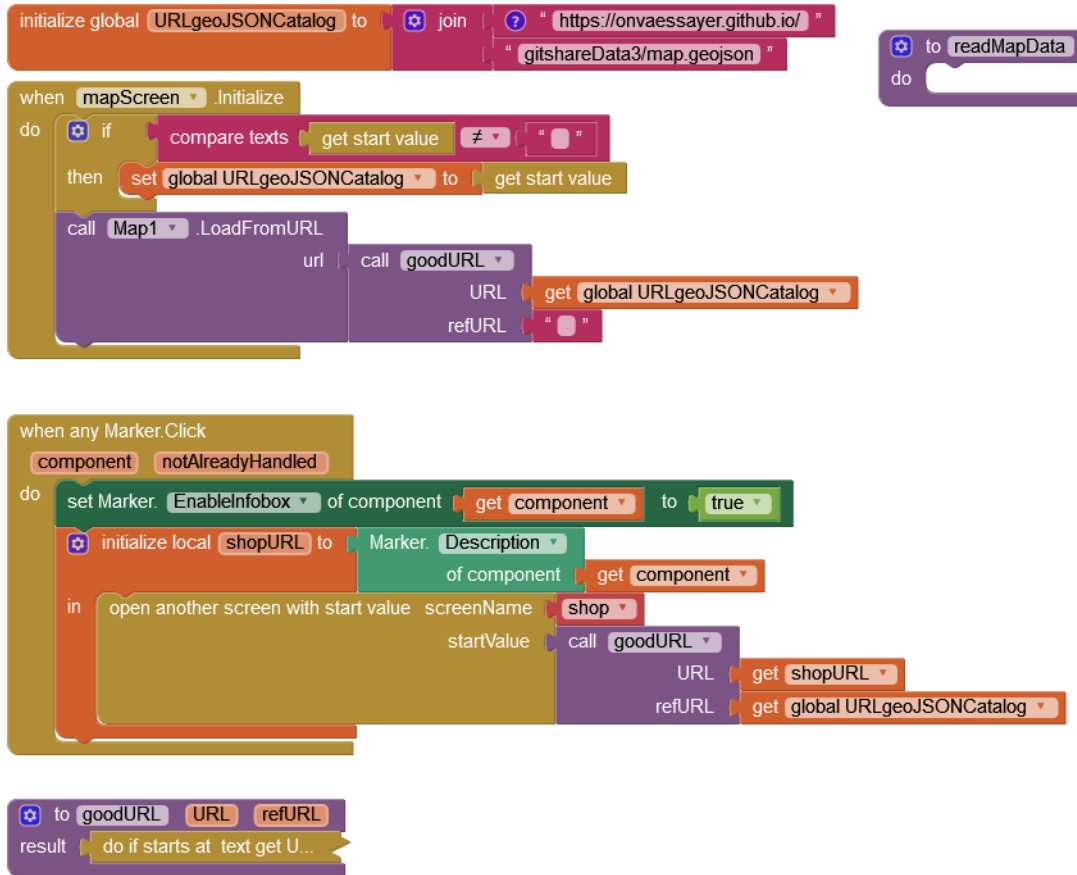
```
call TinyDB1 .StoreValue
tag "latitude"
valueToStore Map1 . Latitude
```

```
call TinyDB1 .GetValue
tag "latitude"
valueIfTagNotThere 48.85
```

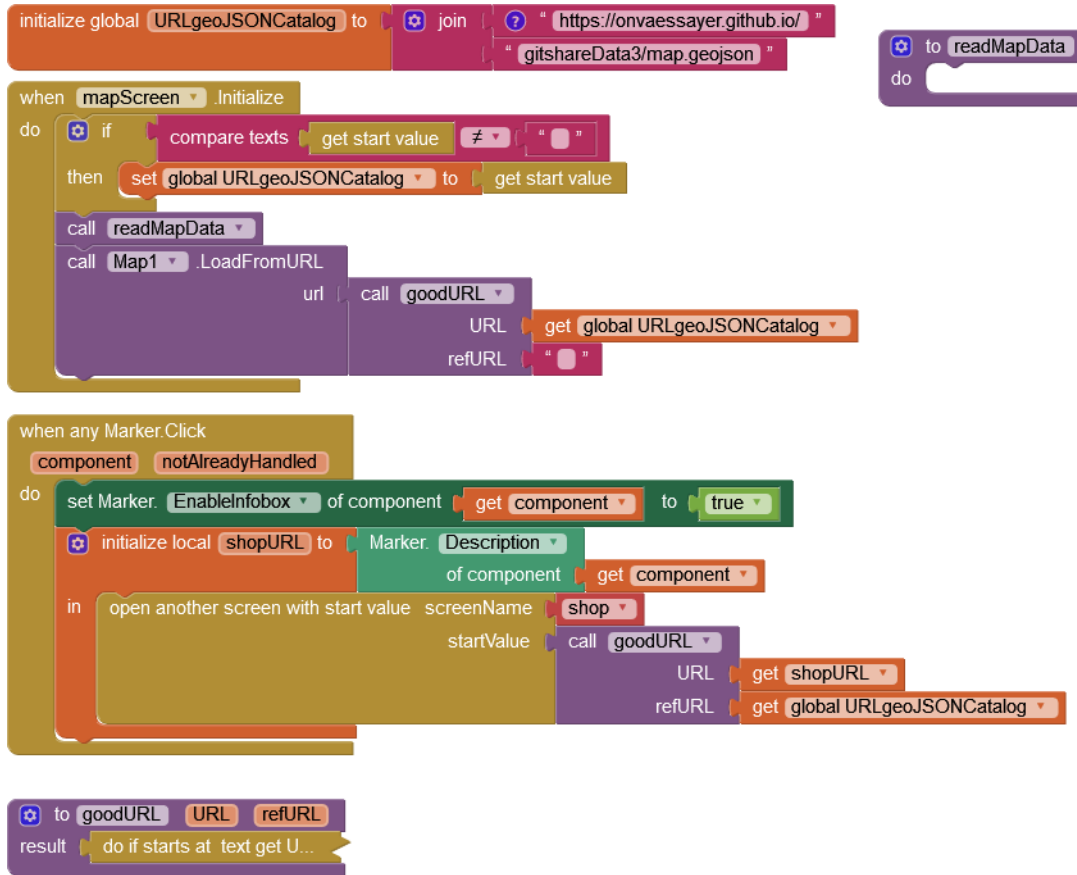
- centrer la carte sur la position de l'utilisateur

```
call Map1 .PanTo
latitude
longitude
zoom
```

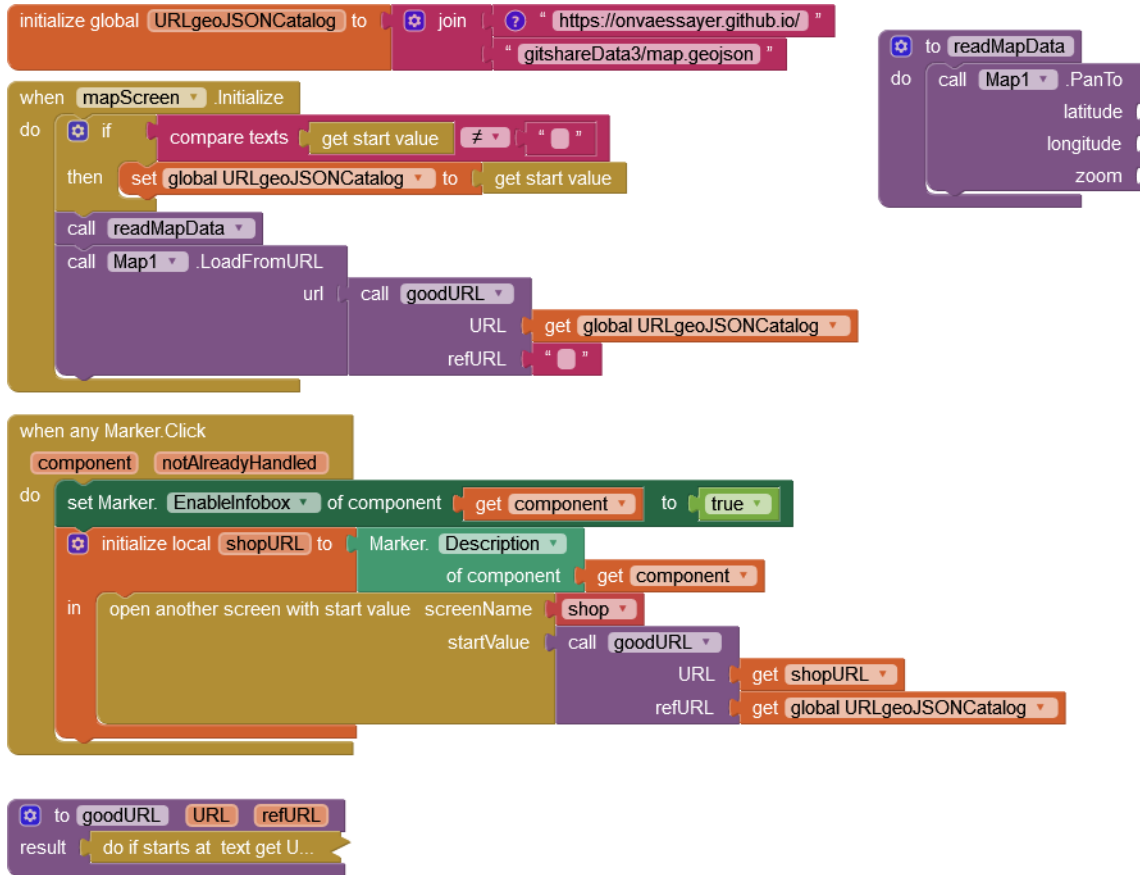
GITSHARE 3a : MAPSCREEN



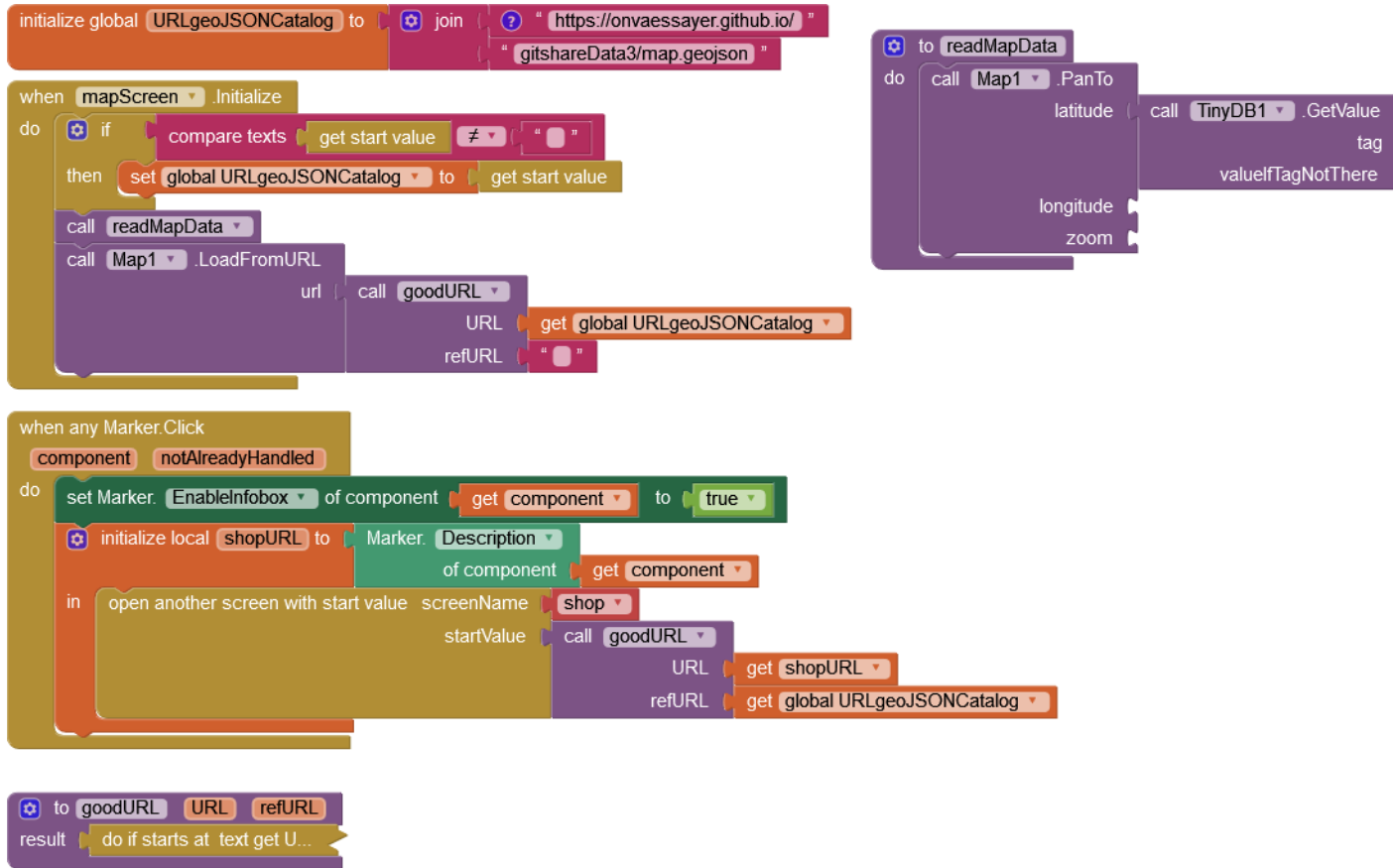
GITSHARE 3a : MAPSCREEN



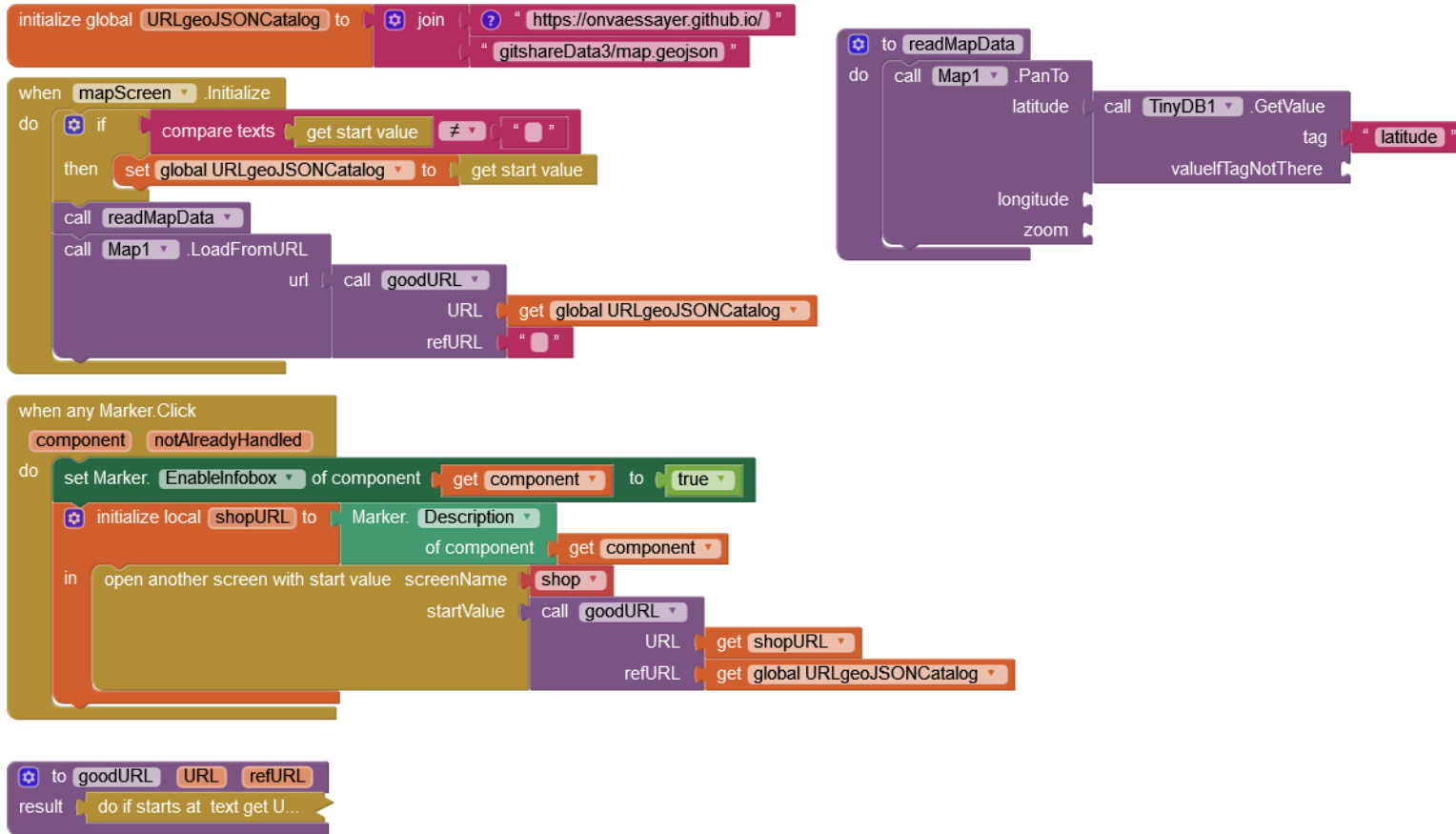
GITSHARE 3a : MAPSCREEN



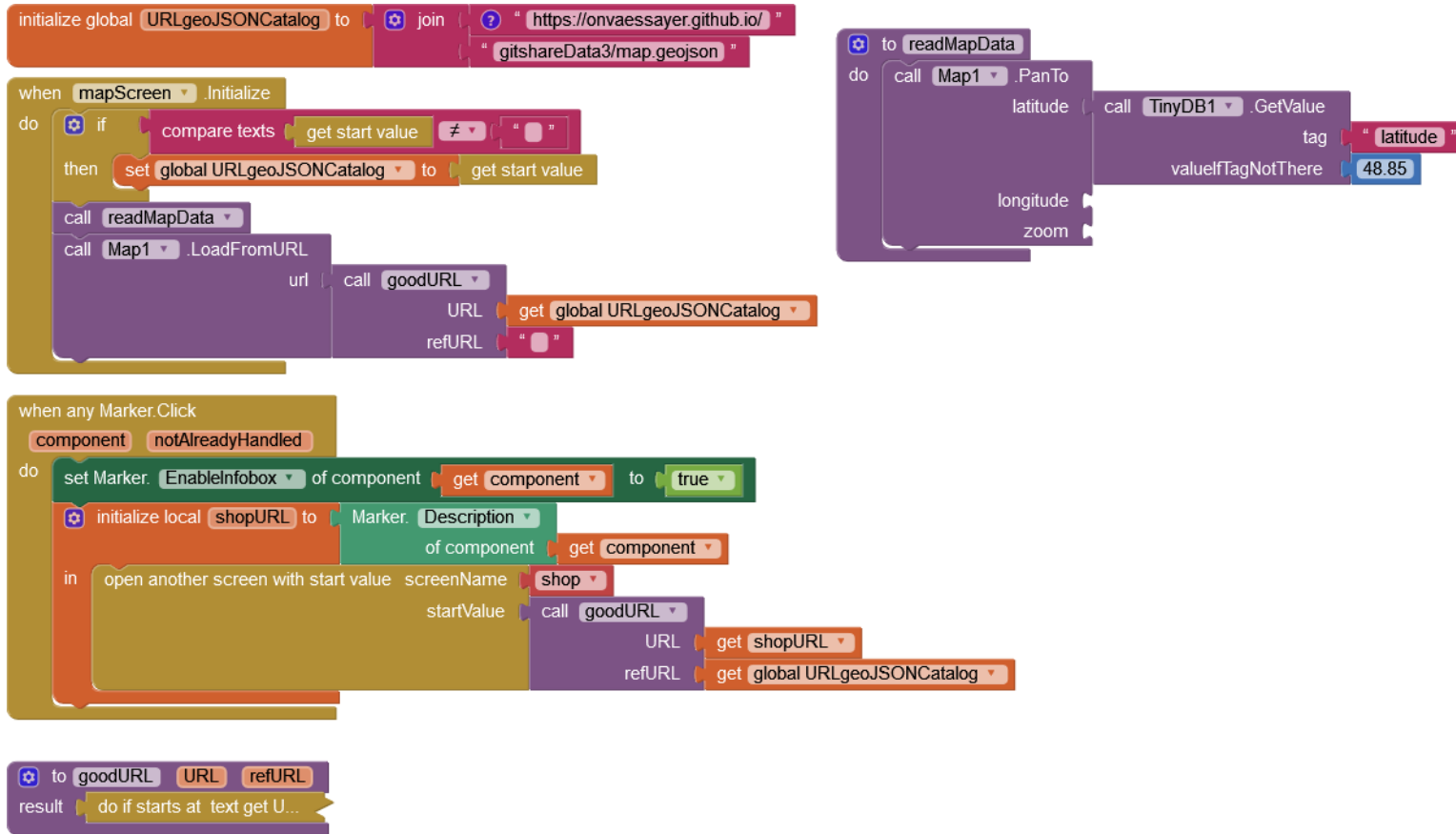
GITSHARE 3a : MAPSCREEN



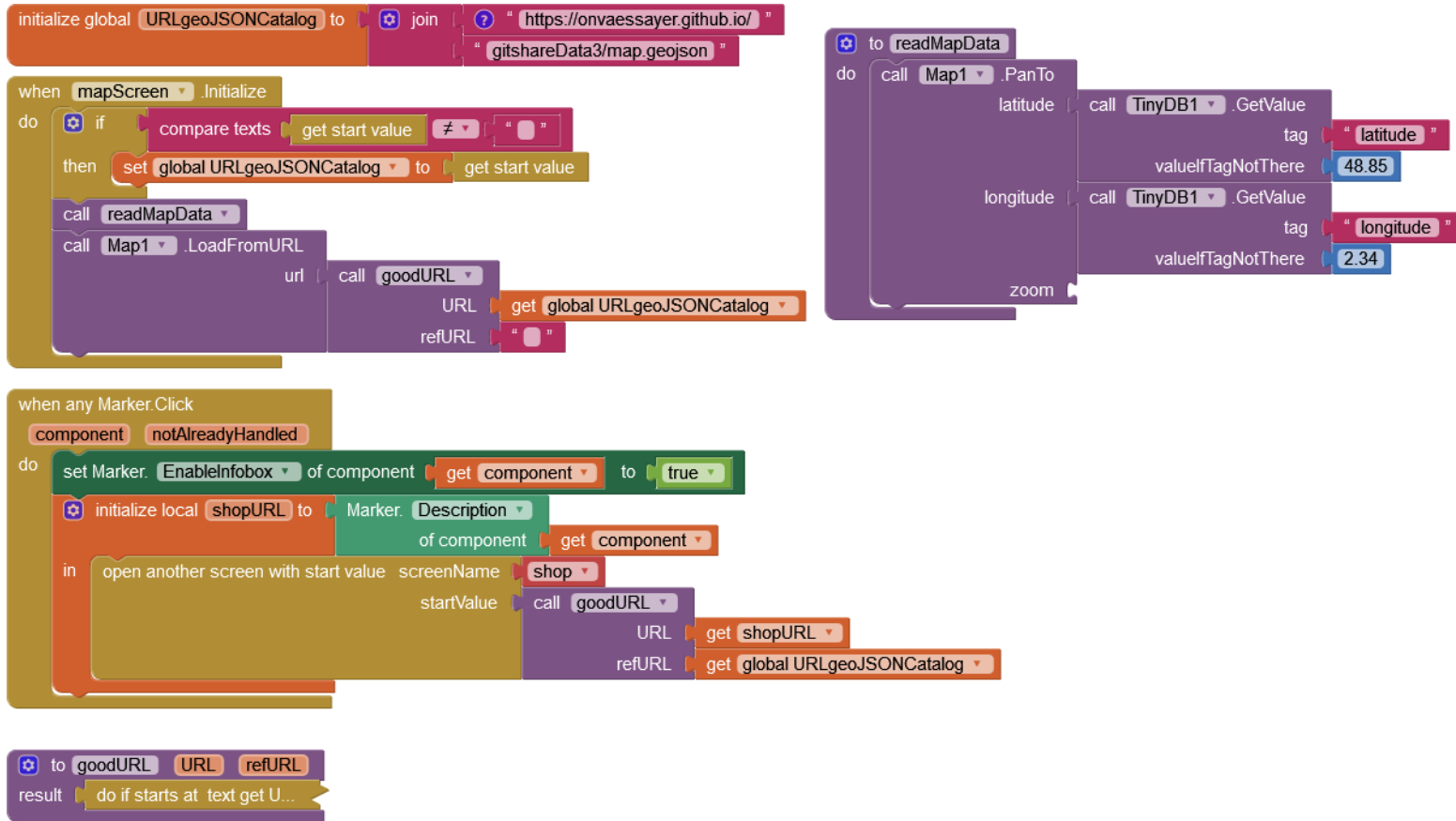
GITSHARE 3a : MAPSCREEN



GITSHARE 3a : MAPSCREEN



GITSHARE 3a : MAPSCREEN



GITSHARE 3a : MAPSCREEN

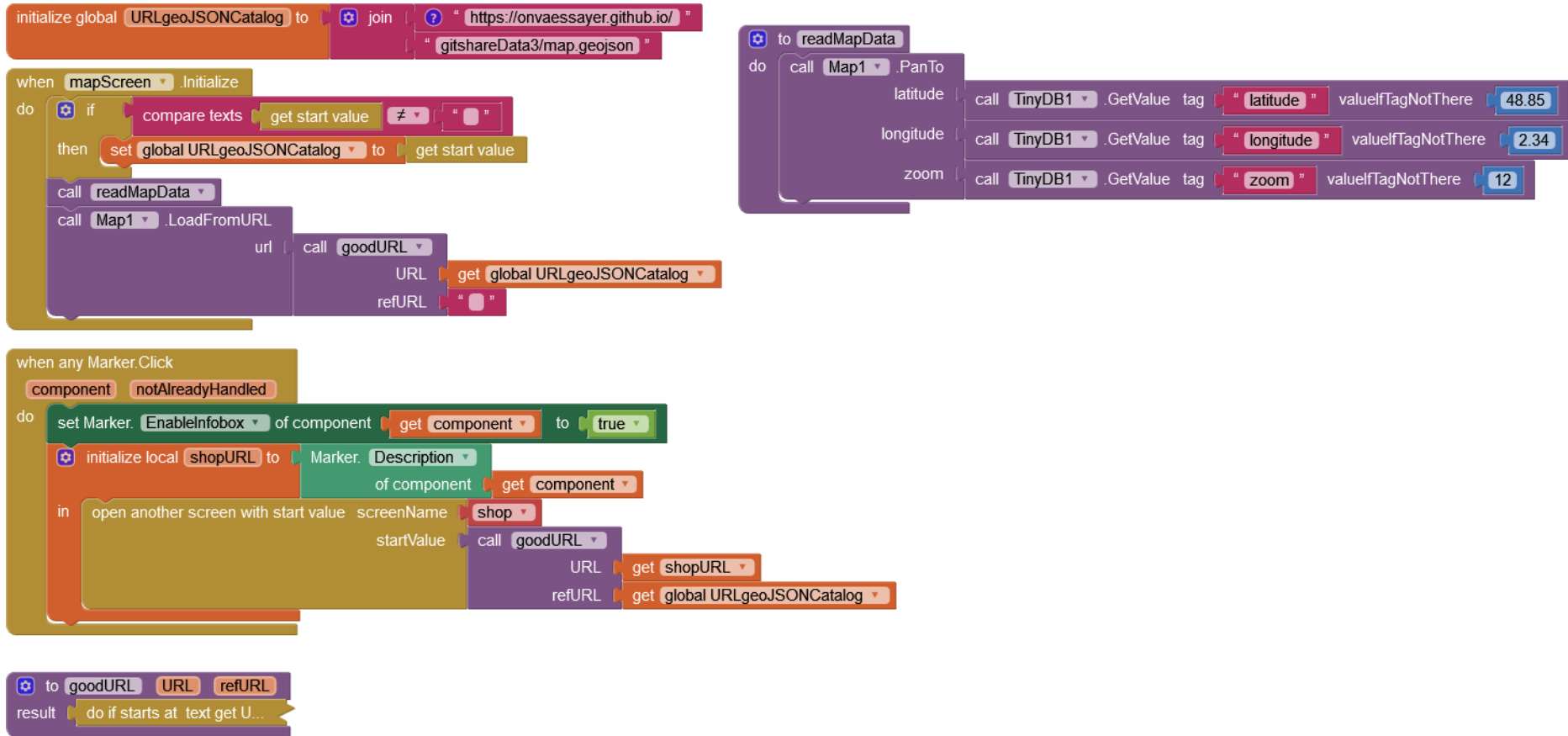
```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when mapScreen.Initialize
do
  if compare texts get start value ≠ ""
  then set global URLgeoJSONCatalog to get start value
  call readMapData
  call Map1.LoadFromURL
  url call goodURL
  URL get global URLgeoJSONCatalog
  refURL ""

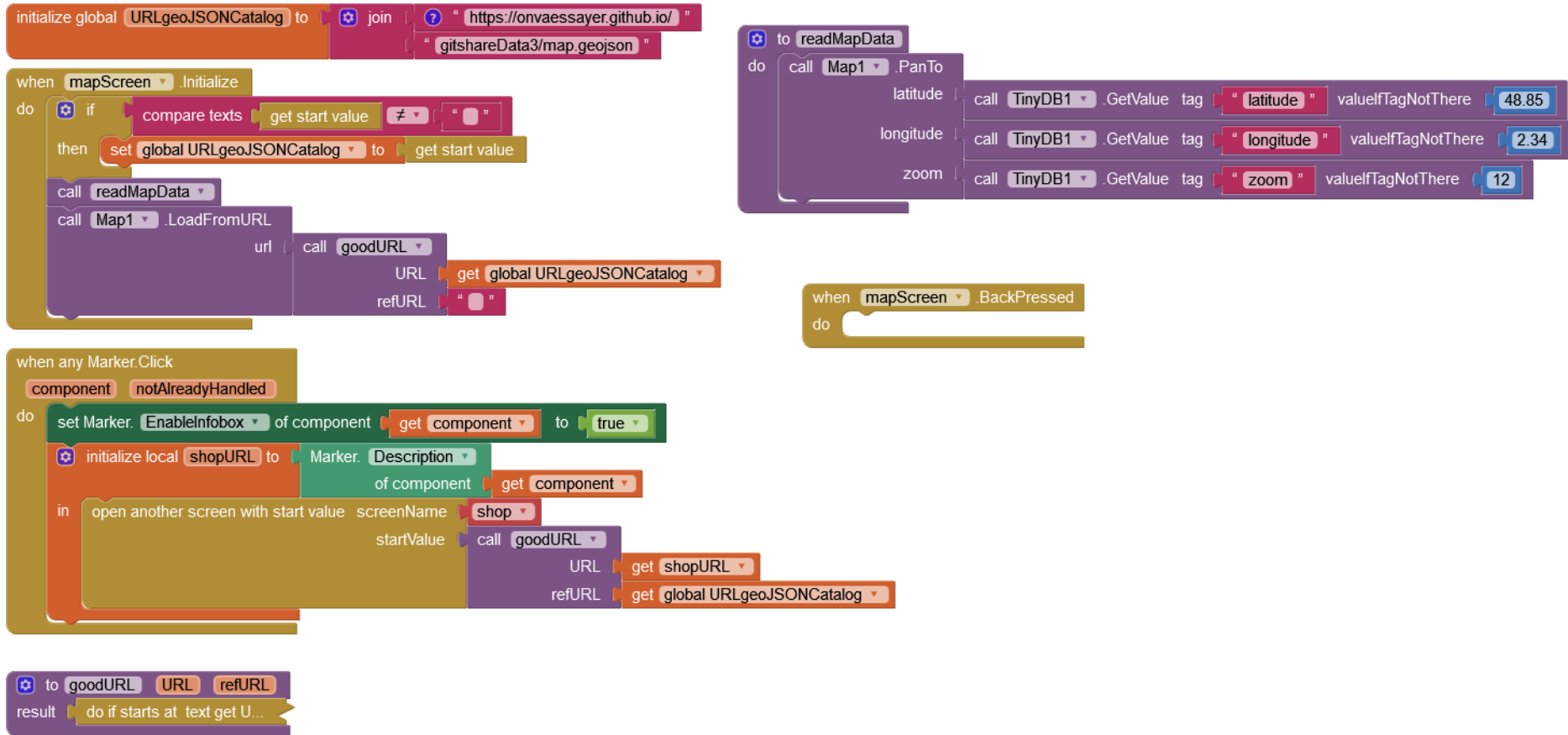
when any Marker.Click
component notAlreadyHandled
do
  set Marker.EnableInfoBox of component get component to true
  initialize local shopURL to Marker.Description of component get component
  in open another screen with start value screenName shop
  startValue call goodURL
  URL get shopURL
  refURL get global URLgeoJSONCatalog

to goodURL URL refURL
result do if starts at text get U...
```

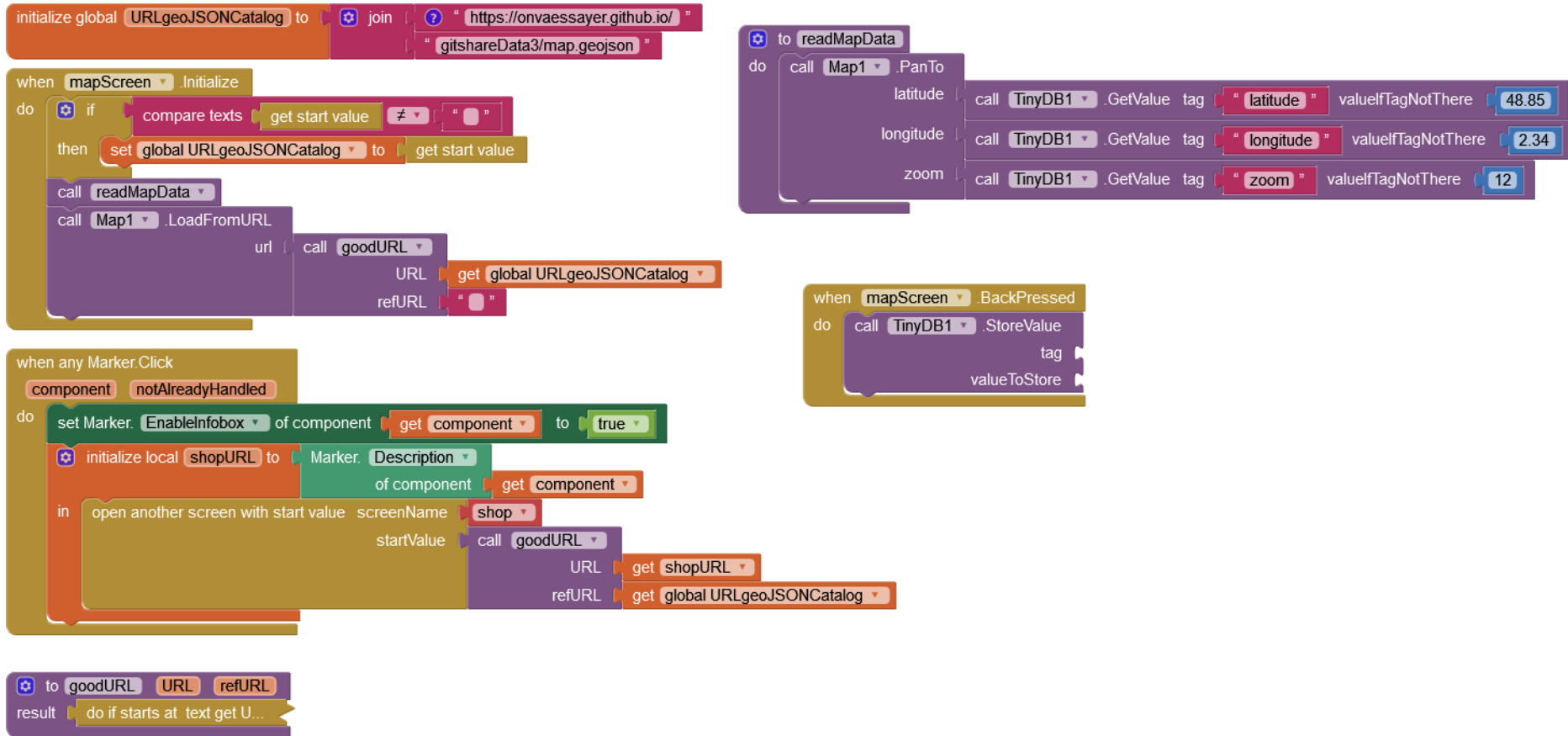
GITSHARE 3a : MAPSCREEN



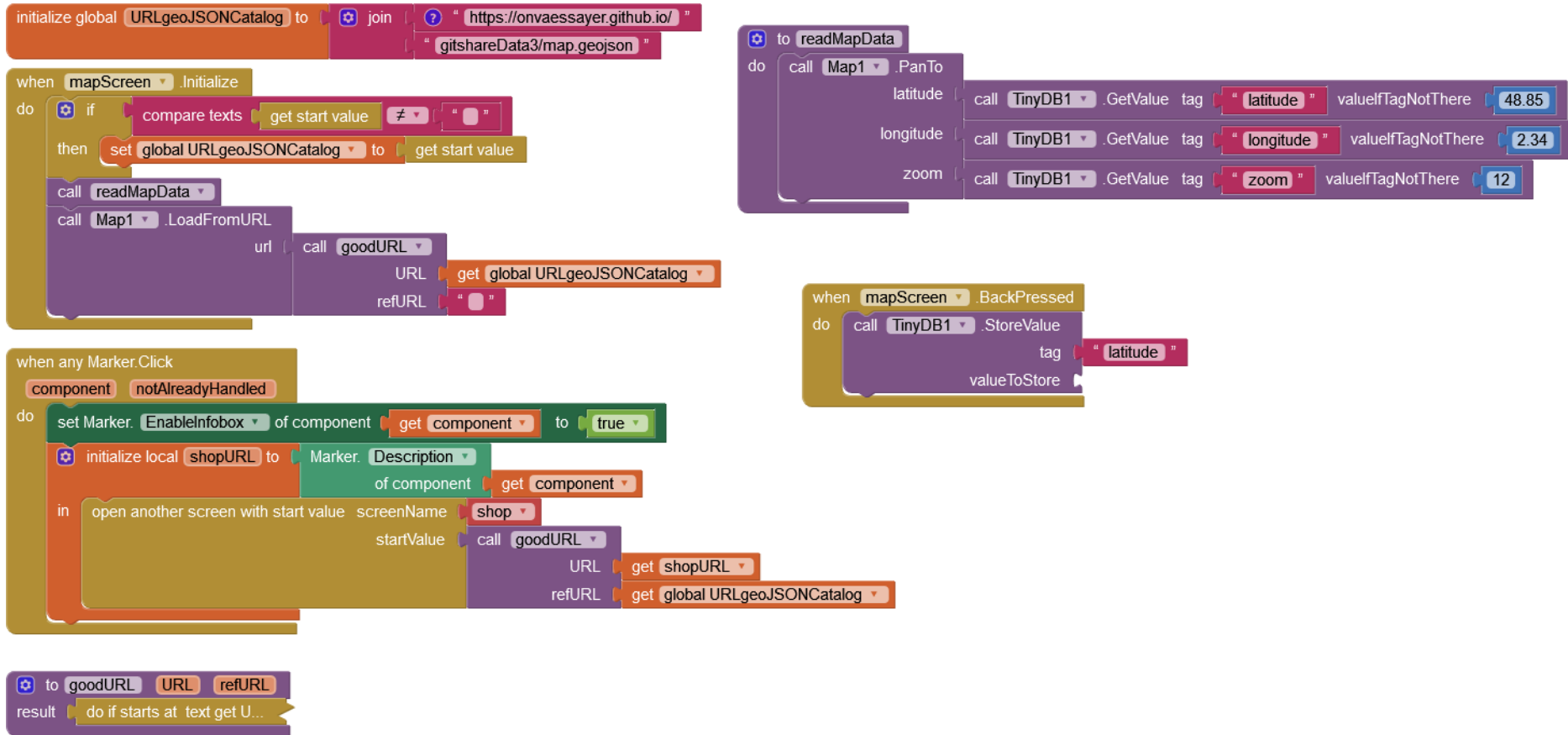
GITSHARE 3a : MAPSCREEN



GITSHARE 3a : MAPSCREEN



GITSHARE 3a : MAPSCREEN



GITSHARE 3a : MAPSCREEN

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when mapScreen.Initialize
do
  if compare texts get start value ≠ ""
  then set global URLgeoJSONCatalog to get start value
  call readMapData
  call Map1.LoadFromURL
  url call goodURL
  URL get global URLgeoJSONCatalog
  refURL ""
```

```
when any Marker.Click
component notAlreadyHandled
do
  set Marker.EnableInfoBox of component get component to true
  initialize local shopURL to Marker.Description of component
  in open another screen with start value screenName shop
  startValue call goodURL
  URL get shopURL
  refURL get global URLgeoJSONCatalog
```

```
to goodURL URL refURL
result do if starts at text get U...
```

```
to readMapData
do
  call Map1.PanTo
  latitude call TinyDB1.GetValue tag "latitude" valueIfTagNotThere 48.85
  longitude call TinyDB1.GetValue tag "longitude" valueIfTagNotThere 2.34
  zoom call TinyDB1.GetValue tag "zoom" valueIfTagNotThere 12
```

```
when mapScreen.BackPressed
do
  call TinyDB1.StoreValue
  tag "latitude"
  valueToStore Map1.Latitude
```

GITSHARE 3a : MAPSCREEN

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when mapScreen.Initialize
do
  if compare texts get start value ≠ ""
  then set global URLgeoJSONCatalog to get start value
  call readMapData
  call Map1.LoadFromURL
  url call goodURL
  URL get global URLgeoJSONCatalog
  refURL ""
```

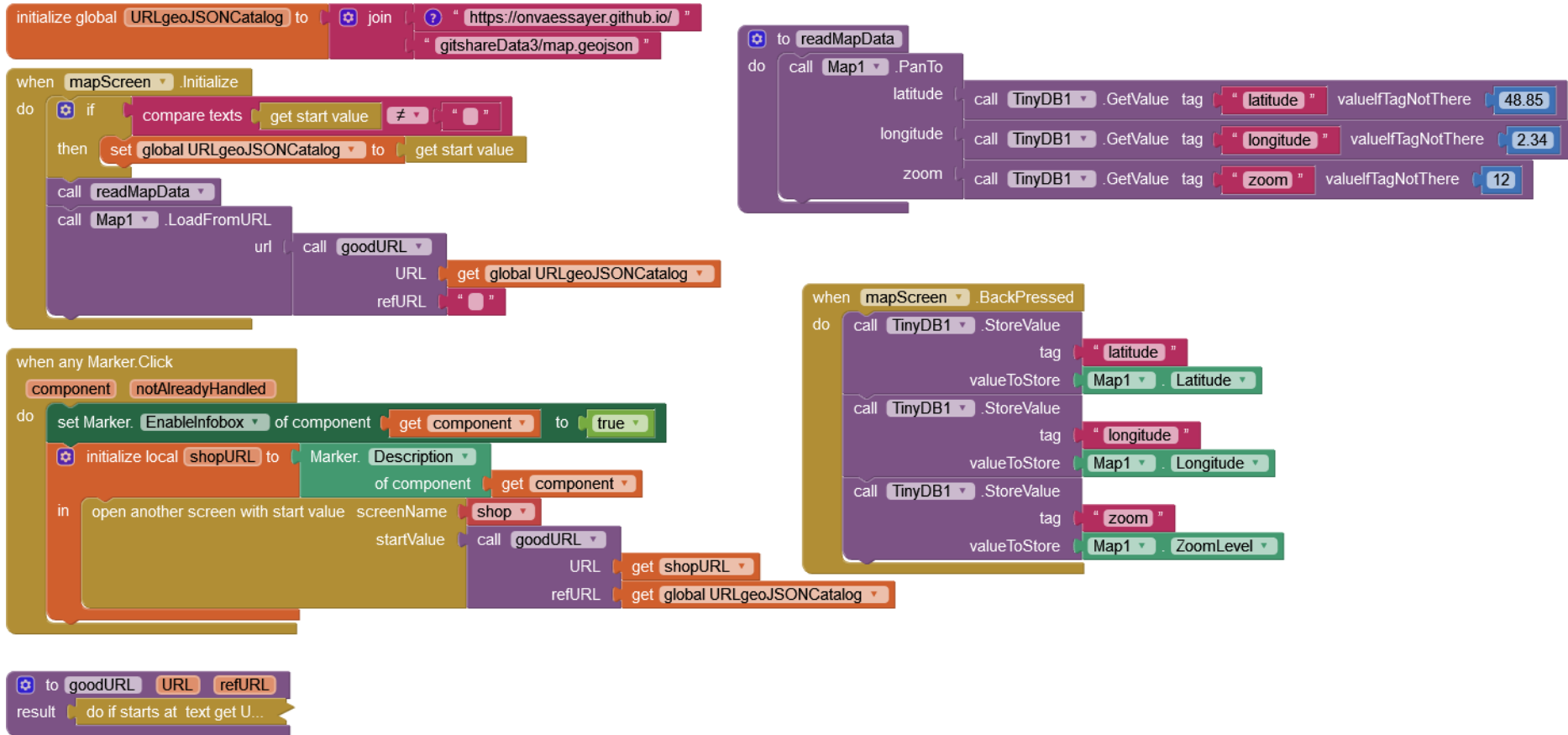
```
to readMapData
do
  call Map1.PanTo
  latitude call TinyDB1.GetValue tag "latitude" valueIfTagNotThere 48.85
  longitude call TinyDB1.GetValue tag "longitude" valueIfTagNotThere 2.34
  zoom call TinyDB1.GetValue tag "zoom" valueIfTagNotThere 12
```

```
when any Marker.Click
component notAlreadyHandled
do
  set Marker.EnableInfoBox of component get component to true
  initialize local shopURL to Marker.Description of component
  in open another screen with start value screenName shop
  startValue call goodURL
  URL get shopURL
  refURL get global URLgeoJSONCatalog
```

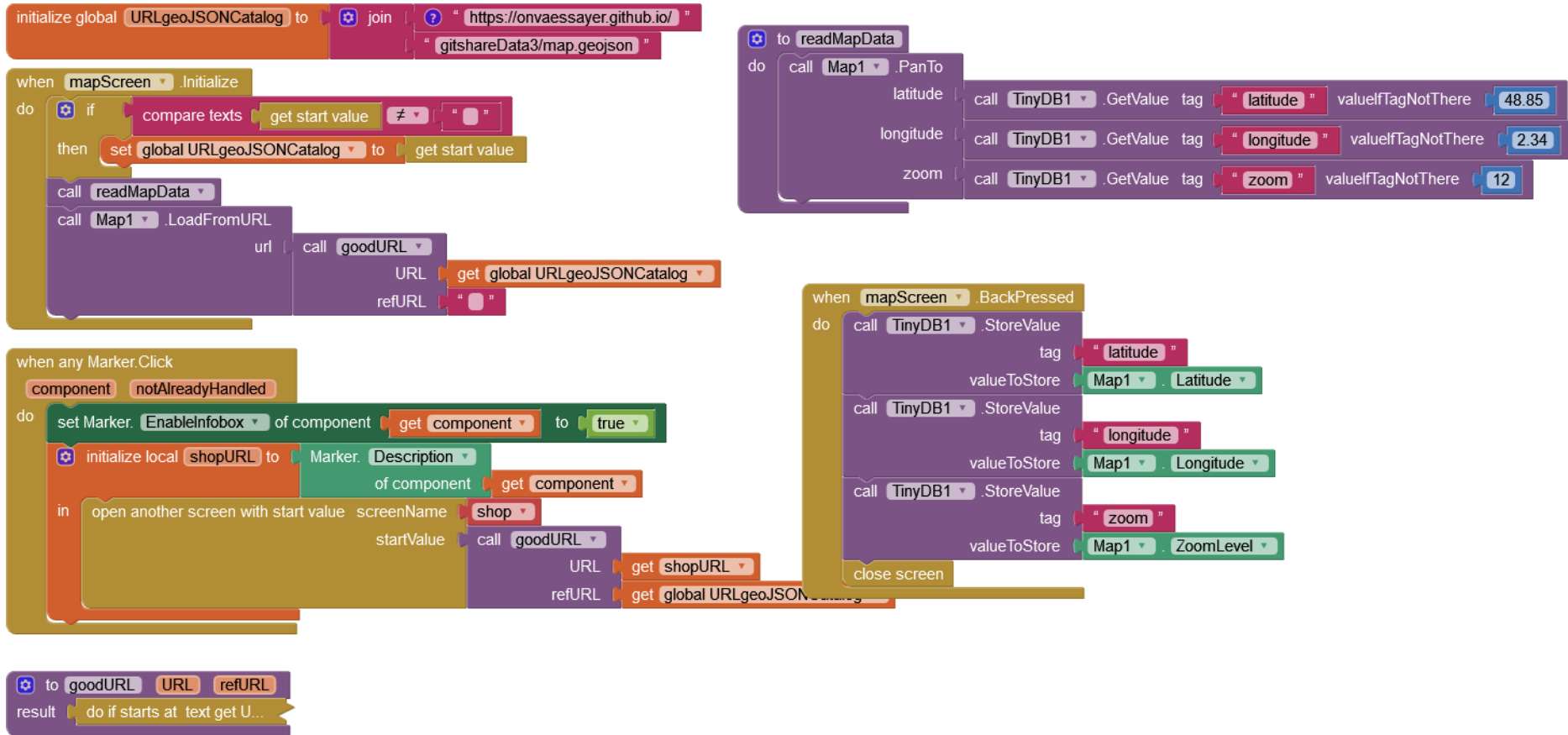
```
when mapScreen.BackPressed
do
  call TinyDB1.StoreValue
  tag "latitude"
  valueToStore Map1.Latitude
  call TinyDB1.StoreValue
  tag "longitude"
  valueToStore Map1.Longitude
```

```
to goodURL URL refURL
result do if starts at text get U...
```

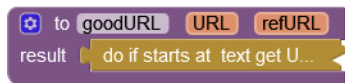
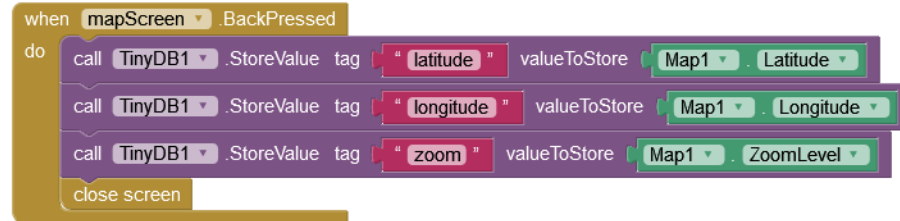
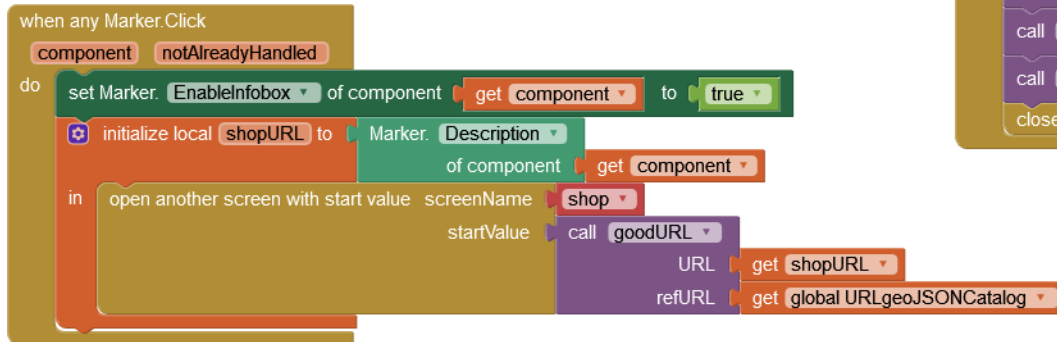
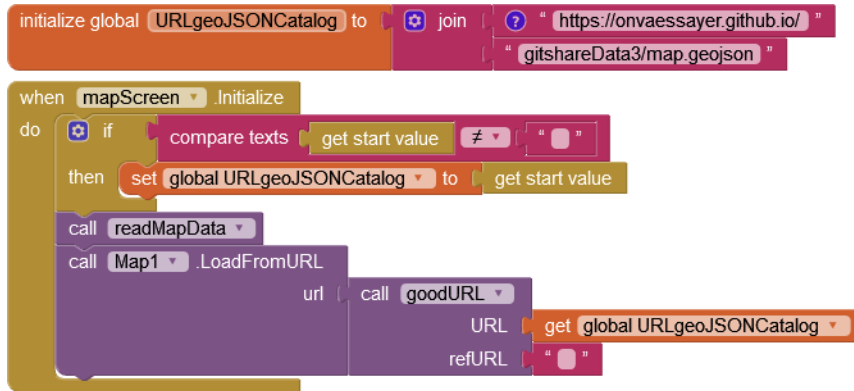
GITSHARE 3a : MAPSCREEN



GITSHARE 3a : MAPSCREEN



GITSHARE 3a : MAPSCREEN

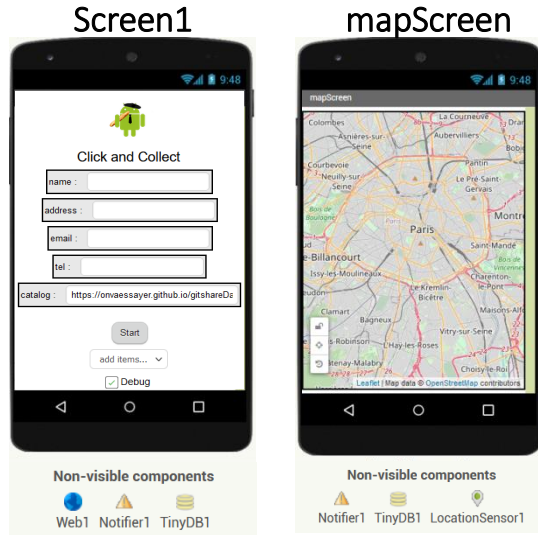




3.4.3

Centrer la carte sur la position de
l'utilisateur

GITSHARE 3a : nouvelles fonctions / mapScreen



- lire l'URL du catalogue en paramètre d'appel
- enregistrer la position de la carte (à chaque déplacement)

get start value

```
call TinyDB1 .StoreValue
tag "latitude"
valueToStore Map1 . Latitude
```

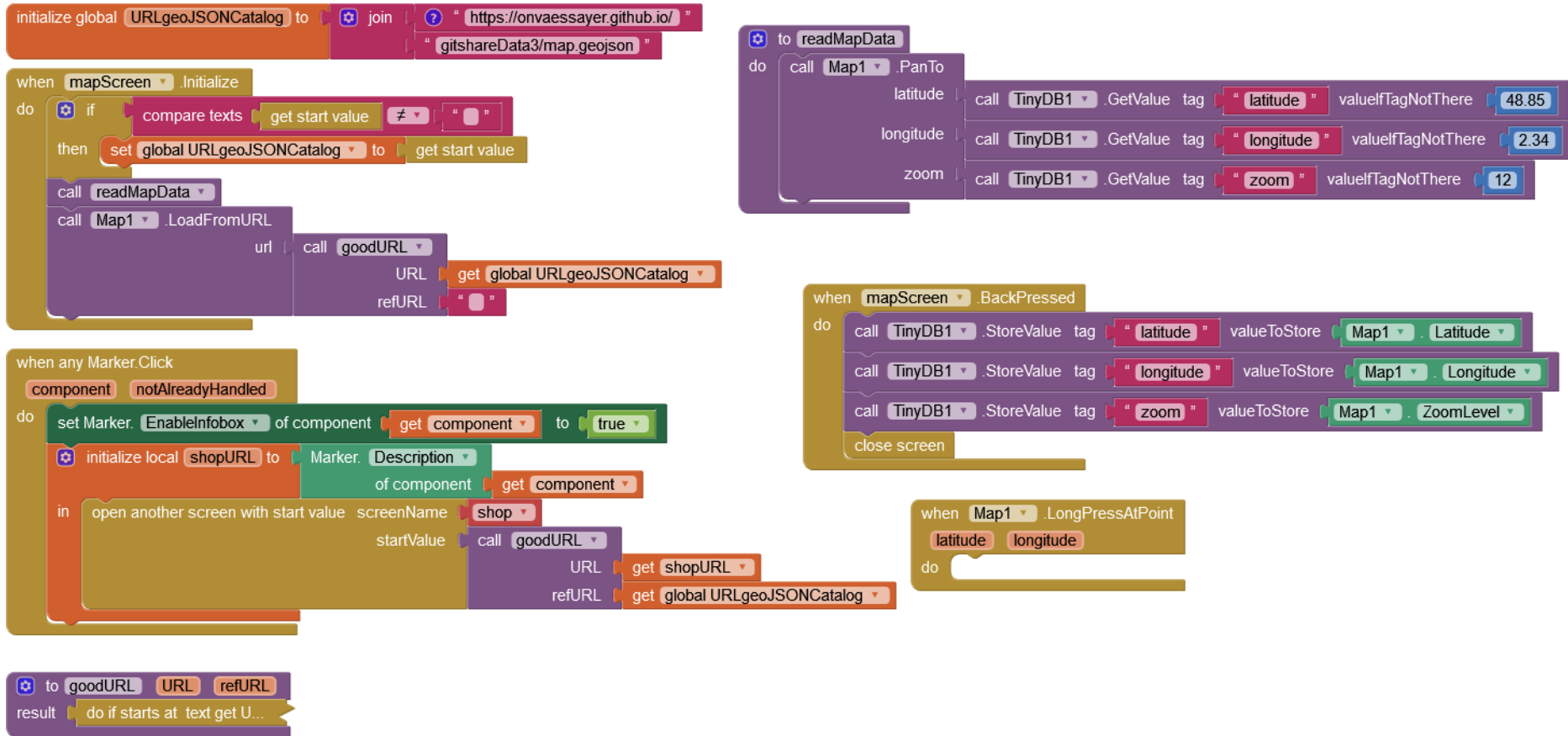
& relire sa position (au démarrage de mapScreen)

```
call TinyDB1 .GetValue
tag "latitude"
valueIfTagNotThere 48.85
```

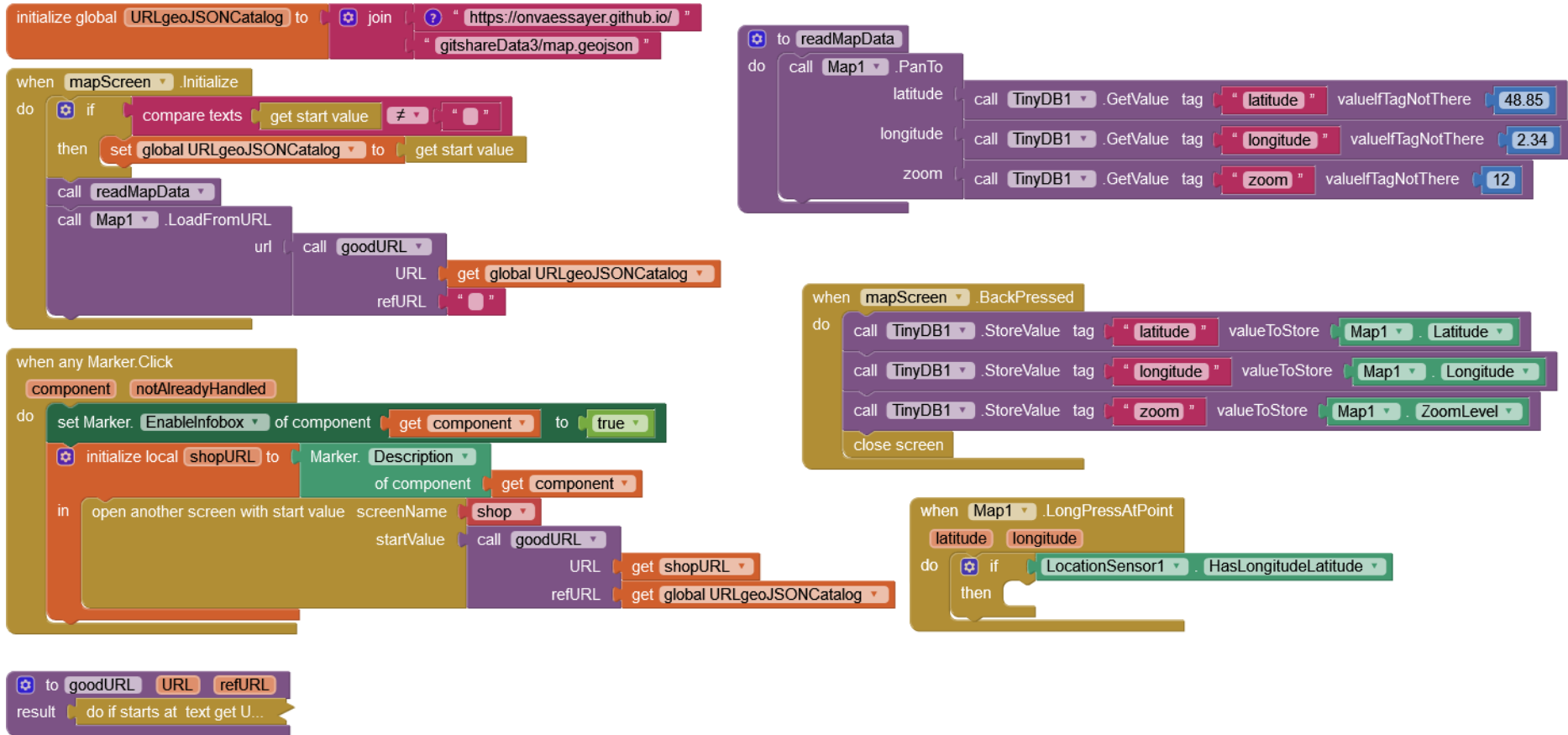
- centrer la carte sur la position de l'utilisateur

```
call Map1 .PanTo
latitude
longitude
zoom
```

GITSHARE 3a : MAPSCREEN



GITSHARE 3a : MAPSCREEN



GITSHARE 3a : MAPSCREEN

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when mapScreen.Initialize
do
  if compare texts get start value ≠ ""
  then set global URLgeoJSONCatalog to get start value
  call readMapData
  call Map1.LoadFromURL
  url call goodURL
  URL get global URLgeoJSONCatalog
  refURL ""
```

```
when any Marker.Click
component notAlreadyHandled
do
  set Marker.EnableInfoBox of component get component to true
  initialize local shopURL to Marker.Description of component
  in open another screen with start value screenName shop
  startValue call goodURL
  URL get shopURL
  refURL get global URLgeoJSONCatalog
```

```
to goodURL URL refURL
result do if starts at text get U...
```

```
to readMapData
do
  call Map1.PanTo
  latitude call TinyDB1.GetValue tag "latitude" valueIfTagNotThere 48.85
  longitude call TinyDB1.GetValue tag "longitude" valueIfTagNotThere 2.34
  zoom call TinyDB1.GetValue tag "zoom" valueIfTagNotThere 12
```

```
when mapScreen.BackPressed
do
  call TinyDB1.StoreValue tag "latitude" valueToStore Map1.Latitude
  call TinyDB1.StoreValue tag "longitude" valueToStore Map1.Longitude
  call TinyDB1.StoreValue tag "zoom" valueToStore Map1.ZoomLevel
  close screen
```

```
when Map1.LongPressAtPoint
latitude longitude
do
  if LocationSensor1.HasLongitudeLatitude
  then call Map1.PanTo
  latitude LocationSensor1.Latitude
  longitude LocationSensor1.Longitude
  zoom Map1.ZoomLevel
```

GITSHARE 3a : MAPSCREEN

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when mapScreen.Initialize
do
  if compare texts get start value ≠ ""
  then set global URLgeoJSONCatalog to get start value
  call readMapData
  call Map1.LoadFromURL
  url call goodURL
  URL get global URLgeoJSONCatalog
  refURL ""
```

```
when any Marker.Click
component notAlreadyHandled
do
  set Marker.EnableInfoBox of component get component to true
  initialize local shopURL to Marker.Description of component get component
  in open another screen with start value screenName shop
  startValue call goodURL
  URL get shopURL
  refURL get global URLgeoJSONCatalog
```

```
to goodURL URL refURL
result do if starts at text get U...
```

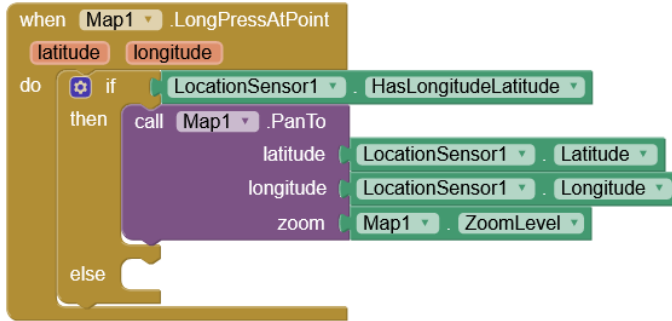
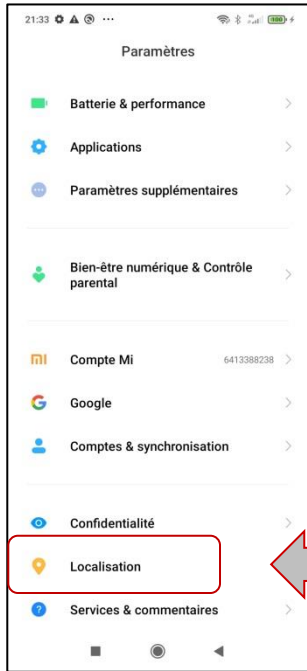
```
to readMapData
do
  call Map1.PanTo
  latitude call TinyDB1.GetValue tag "latitude" valueIfTagNotThere 48.85
  longitude call TinyDB1.GetValue tag "longitude" valueIfTagNotThere 2.34
  zoom call TinyDB1.GetValue tag "zoom" valueIfTagNotThere 12
```

```
when mapScreen.BackPressed
do
  call TinyDB1.StoreValue tag "latitude" valueToStore Map1.Latitude
  call TinyDB1.StoreValue tag "longitude" valueToStore Map1.Longitude
  call TinyDB1.StoreValue tag "zoom" valueToStore Map1.ZoomLevel
  close screen
```

```
when Map1.LongPressAtPoint
latitude longitude
do
  if LocationSensor1.HasLongitudeLatitude
  then call Map1.PanTo
  latitude LocationSensor1.Latitude
  longitude LocationSensor1.Longitude
  zoom Map1.ZoomLevel
  else
```

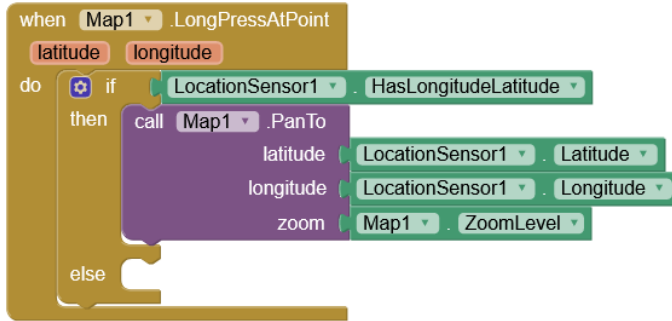
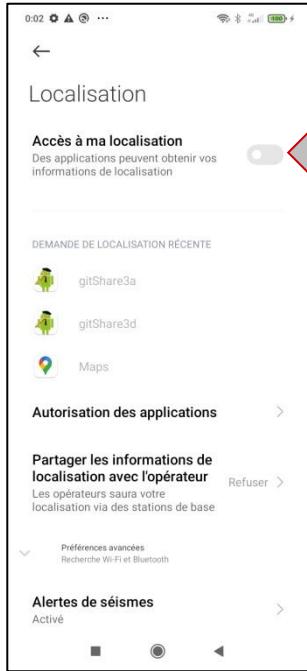
GITSHARE 3a : ACTIVER LA LOCALISATION DU SMARTPHONE

Parameters



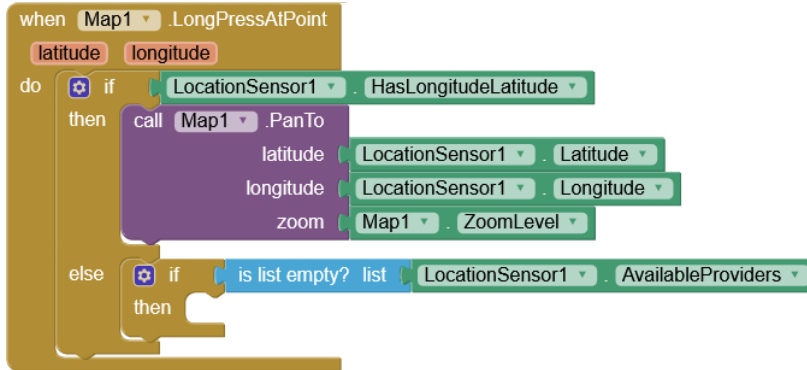
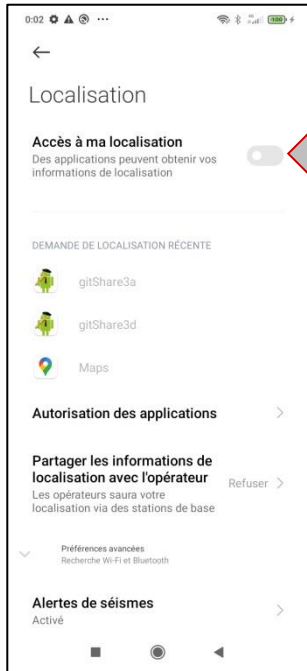
GITSHARE 3a : ACTIVER LA LOCALISATION DU SMARTPHONE

Location off



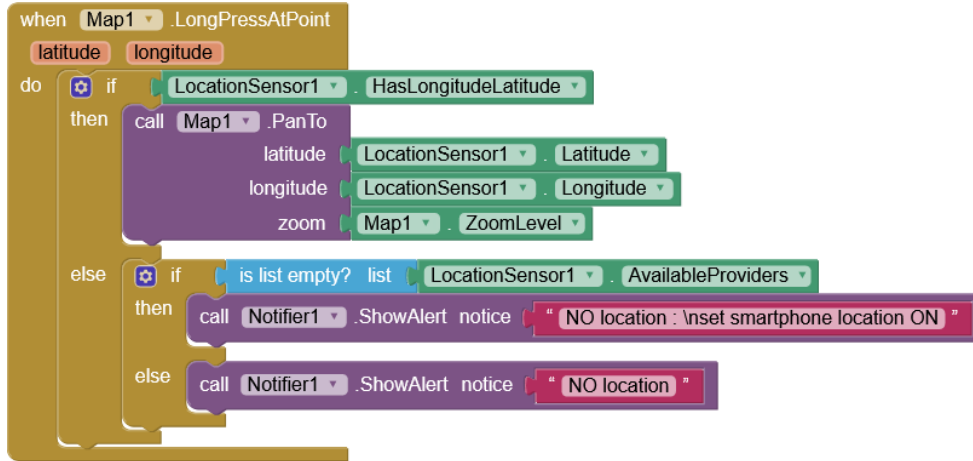
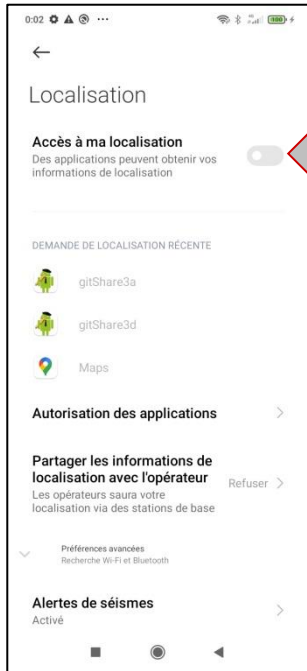
GITSHARE 3a : ACTIVER LA LOCALISATION DU SMARTPHONE

Location off



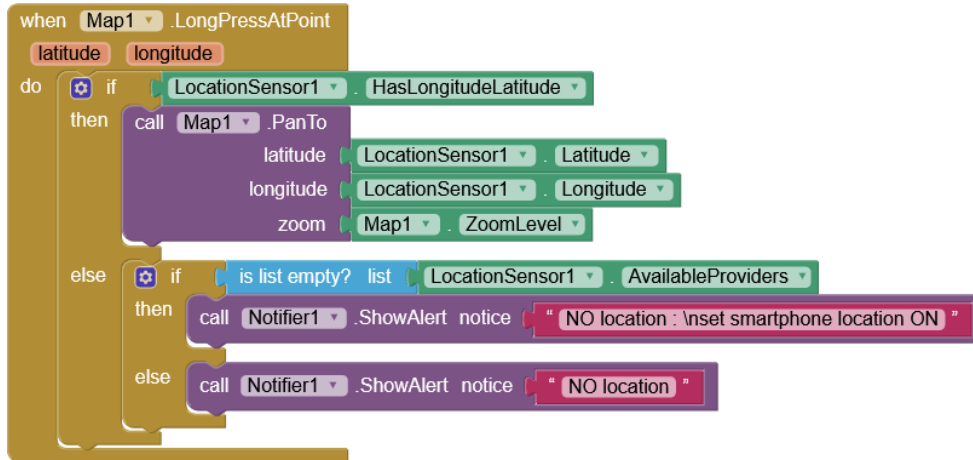
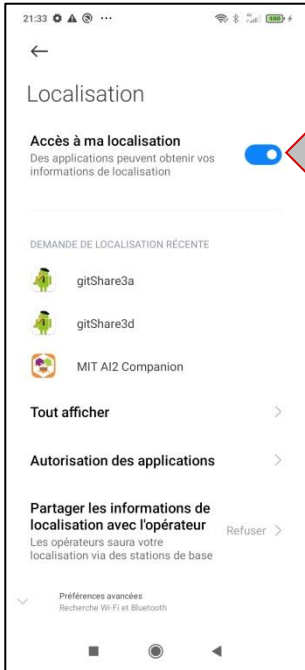
GITSHARE 3a : ACTIVER LA LOCALISATION DU SMARTPHONE

Location off



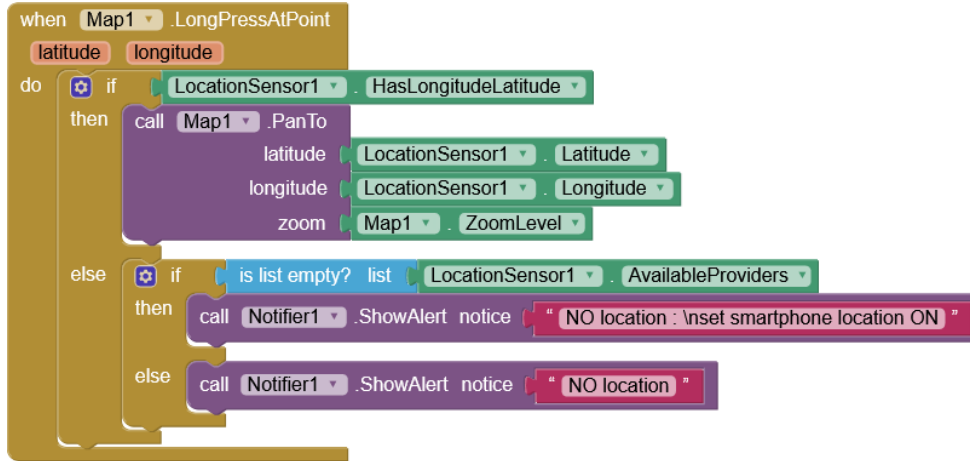
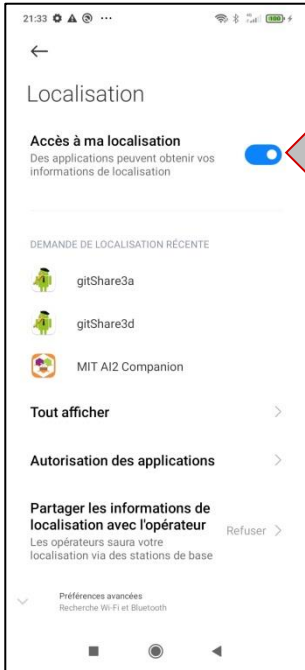
GITSHARE 3a : ACTIVER LA LOCALISATION DU SMARTPHONE

Location on

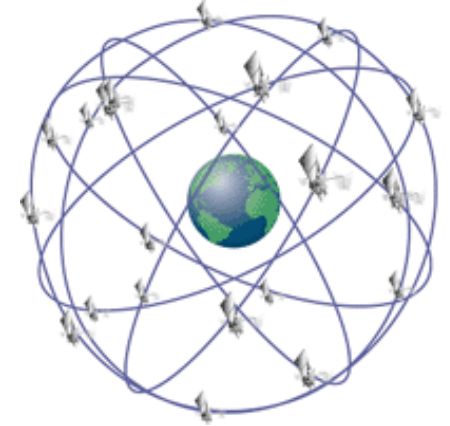


GITSHARE 3a : ACTIVER LA LOCALISATION DU SMARTPHONE

Location on

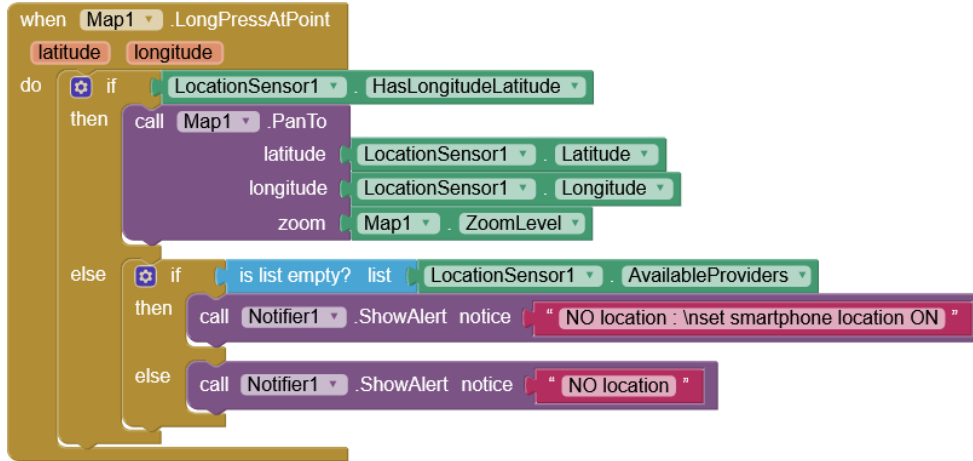
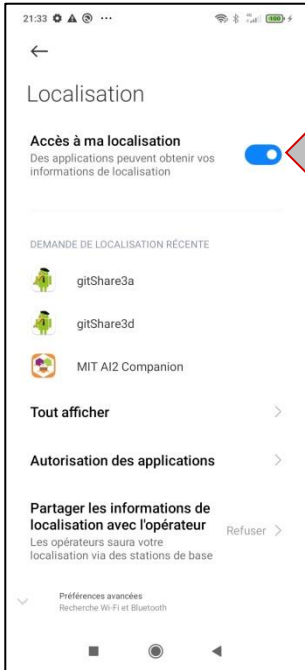


Localisation GPS

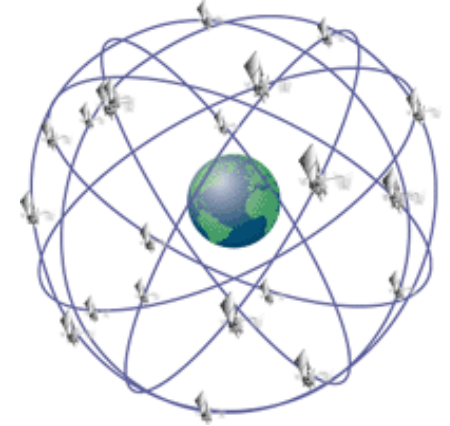


GITSHARE 3a : ACTIVER LA LOCALISATION DU SMARTPHONE

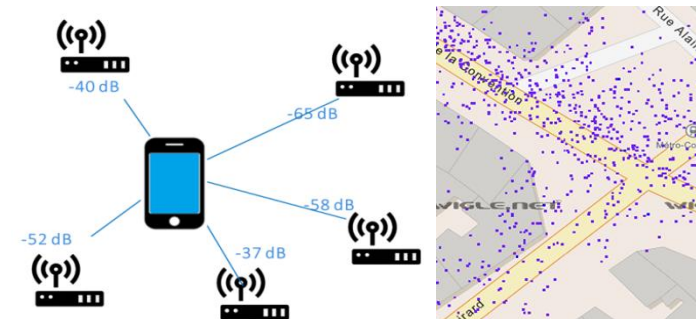
Location on



Localisation GPS

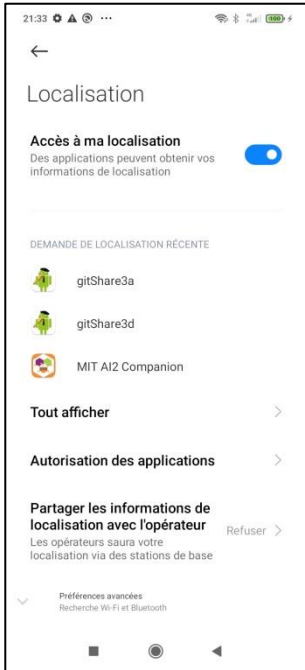


Localisation Wifi, ...
(network)



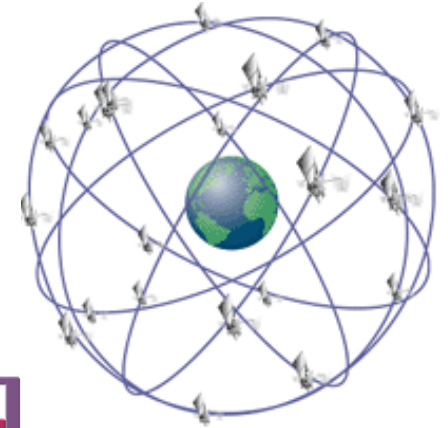
GITSHARE 3a : AFFICHER LA LISTE DES SERVICES DE LOCALISATION

Location on

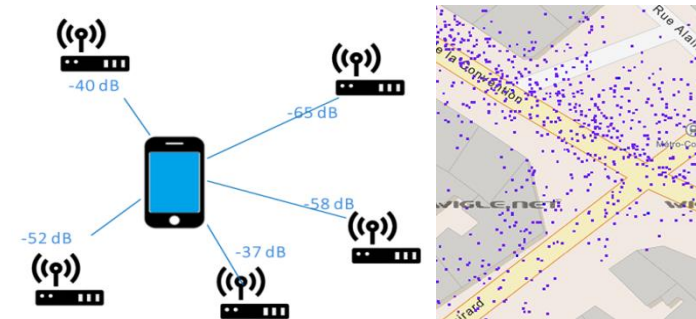


```
when Map1 .LongPressAtPoint
  latitude longitude
do
  if LocationSensor1 .HasLongitudeLatitude
  then
    call Map1 .PanTo
      latitude LocationSensor1 .Latitude
      longitude LocationSensor1 .Longitude
      zoom Map1 .ZoomLevel
  else
    if is list empty? list LocationSensor1 . AvailableProviders
    then
      call Notifier1 .ShowAlert notice " NO location : \nset smartphone location ON "
    else
      call Notifier1 .ShowAlert notice
        join " NO location : "
        join " \ncurrent mode : " LocationSensor1 . ProviderName
        join " \nin : " LocationSensor1 . AvailableProviders
        " \nactivate google precision location (for use in..."
```

Localisation GPS

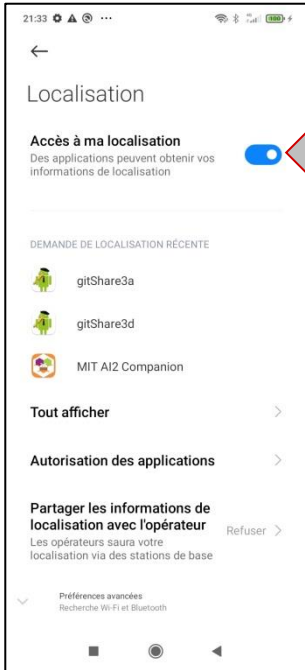


Localisation Wifi, ...
(network)



GITSHARE 3a : AFFICHER LA LISTE DES SERVICES DE LOCALISATION

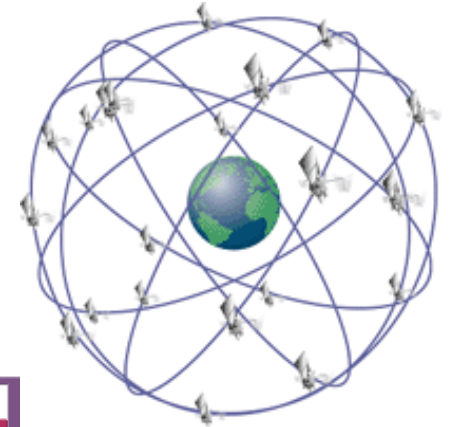
Location on



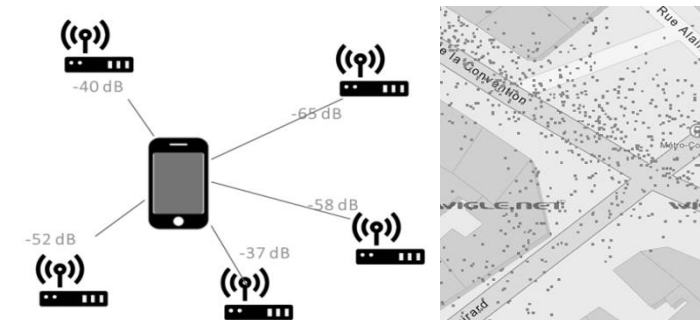
```
when Map1 .LongPressAtPoint
  latitude longitude
do
  if LocationSensor1 .HasLongitudeLatitude
  then
    call Map1 .PanTo
      latitude LocationSensor1 .Latitude
      longitude LocationSensor1 .Longitude
      zoom Map1 .ZoomLevel
  else
    if is list empty? list LocationSensor1 . AvailableProviders
    then
      call Notifier1 .ShowAlert notice " NO location : \nset smartphone location ON "
    else
      call Notifier1 .ShowAlert notice
        join " NO location : "
        join " \ncurrent mode : " LocationSensor1 . ProviderName
        join " \nin : " LocationSensor1 . AvailableProviders
        " \nactivate google precision location (for use inside buildings)"
```

NO location :
current mode : gps
in : ["gps", "passive", "gps"]
activate google precision location
(for use inside buildings)

Localisation GPS

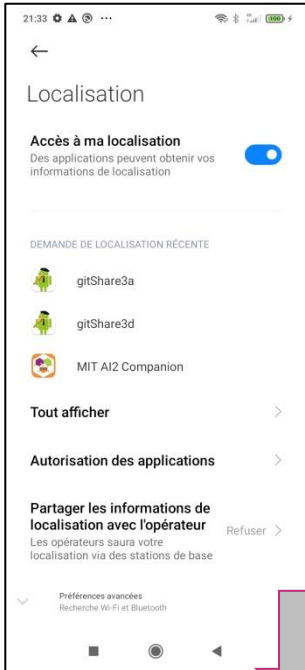


Localisation Wifi, ...
(network)



GITSHARE 3a : ACTIVER LA LOCALISATION DU SMARTPHONE

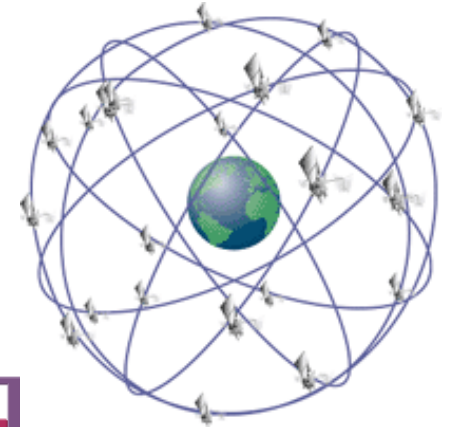
Location on



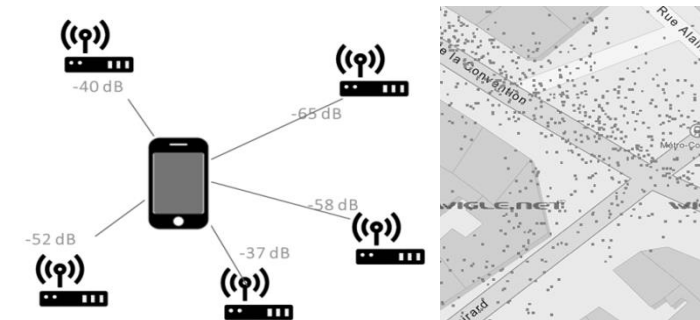
```
when Map1 .LongPressAtPoint
  latitude longitude
do
  if LocationSensor1 .HasLongitudeLatitude
  then
    call Map1 .PanTo
      latitude LocationSensor1 .Latitude
      longitude LocationSensor1 .Longitude
      zoom Map1 .ZoomLevel
  else
    if is list empty? list LocationSensor1 .AvailableProviders
    then
      call Notifier1 .ShowAlert notice " NO location : \nset smartphone location ON "
    else
      call Notifier1 .ShowAlert notice
        join " NO location : "
        join " \ncurrent mode : " LocationSensor1 .ProviderName
        join " \nin : " LocationSensor1 .AvailableProviders
        " \nactivate google precision location (for use inside buildings)"
```

NO location :
current mode : gps
in : ["gps", "passive", "gps"]
activate google precision location
(for use inside buildings)

Localisation GPS

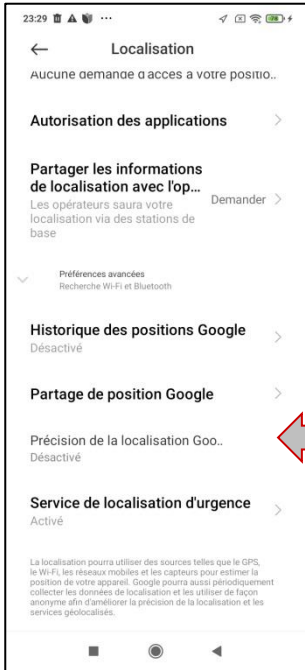


Localisation Wifi, ...
(network)



GITSHARE 3a : ACTIVER LA LOCALISATION DU SMARTPHONE

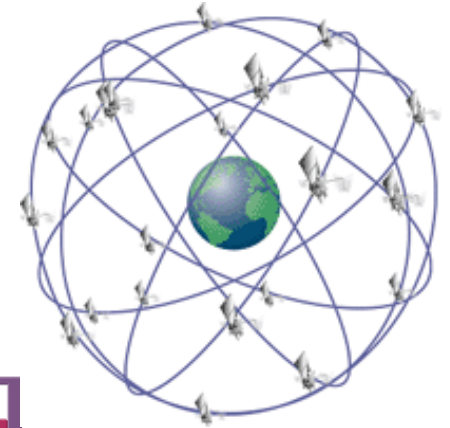
Location on



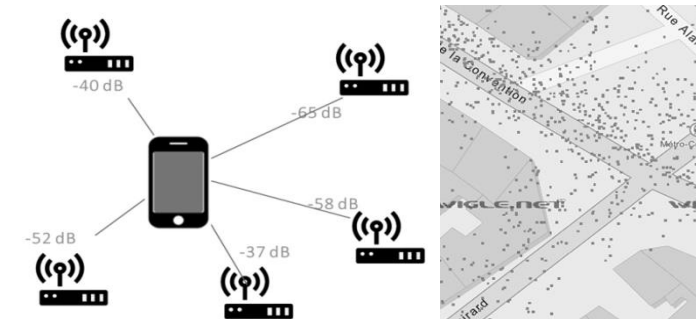
```
when Map1 .LongPressAtPoint
  latitude longitude
do
  if LocationSensor1 . HasLongitudeLatitude
  then
    call Map1 .PanTo
      latitude LocationSensor1 . Latitude
      longitude LocationSensor1 . Longitude
      zoom Map1 . ZoomLevel
  else
    if is list empty? list LocationSensor1 . AvailableProviders
    then
      call Notifier1 .ShowAlert notice " NO location : \nset smartphone location ON "
    else
      call Notifier1 .ShowAlert notice
        join " NO location : "
        join " \ncurrent mode : " LocationSensor1 . ProviderName
        join " \nin : " LocationSensor1 . AvailableProviders
        " \nactivate google precision location (for use inside buildings)"
```

NO location :
current mode : gps
in : ["gps", "passive", "gps"]
activate google precision location
(for use inside buildings)

Localisation GPS

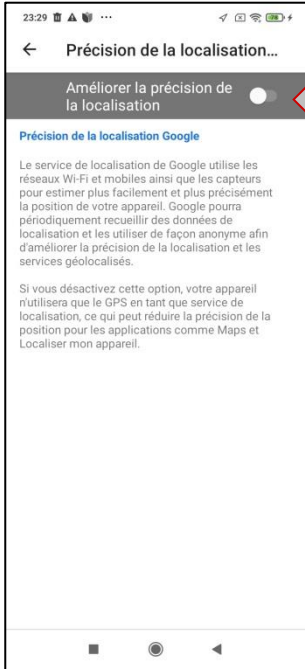


Localisation Wifi, ...
(network)



GITSHARE 3a : ACTIVER LA LOCALISATION DU SMARTPHONE

Location on

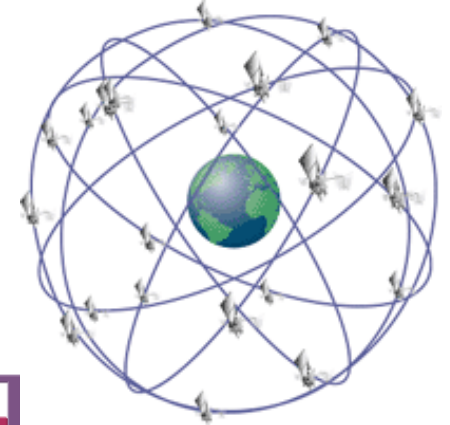


```

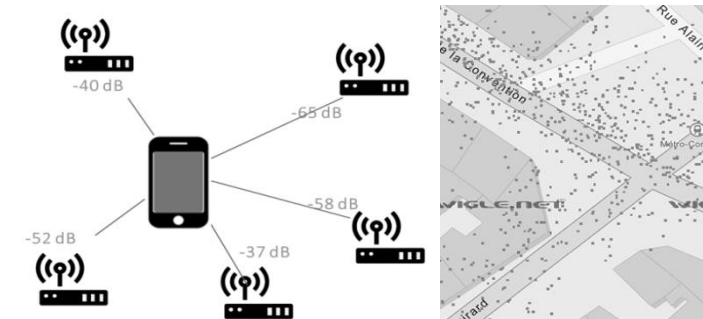
when Map1 .LongPressAtPoint
  latitude longitude
do
  if LocationSensor1 .HasLongitudeLatitude
  then
    call Map1 .PanTo
      latitude LocationSensor1 .Latitude
      longitude LocationSensor1 .Longitude
      zoom Map1 .ZoomLevel
  else
    if is list empty? list LocationSensor1 .AvailableProviders
    then
      call Notifier1 .ShowAlert notice " NO location : \nset smartphone location ON "
    else
      call Notifier1 .ShowAlert notice
        join " NO location : "
        join "\ncurrent mode : " LocationSensor1 .ProviderName
        join "\nin : " LocationSensor1 .AvailableProviders
        "\nactivate google precision location (for use inside buildings)"
  
```

NO location :
 current mode : gps
 in : ["gps", "passive", "gps"]
 activate google precision location
 (for use inside buildings)

Localisation GPS

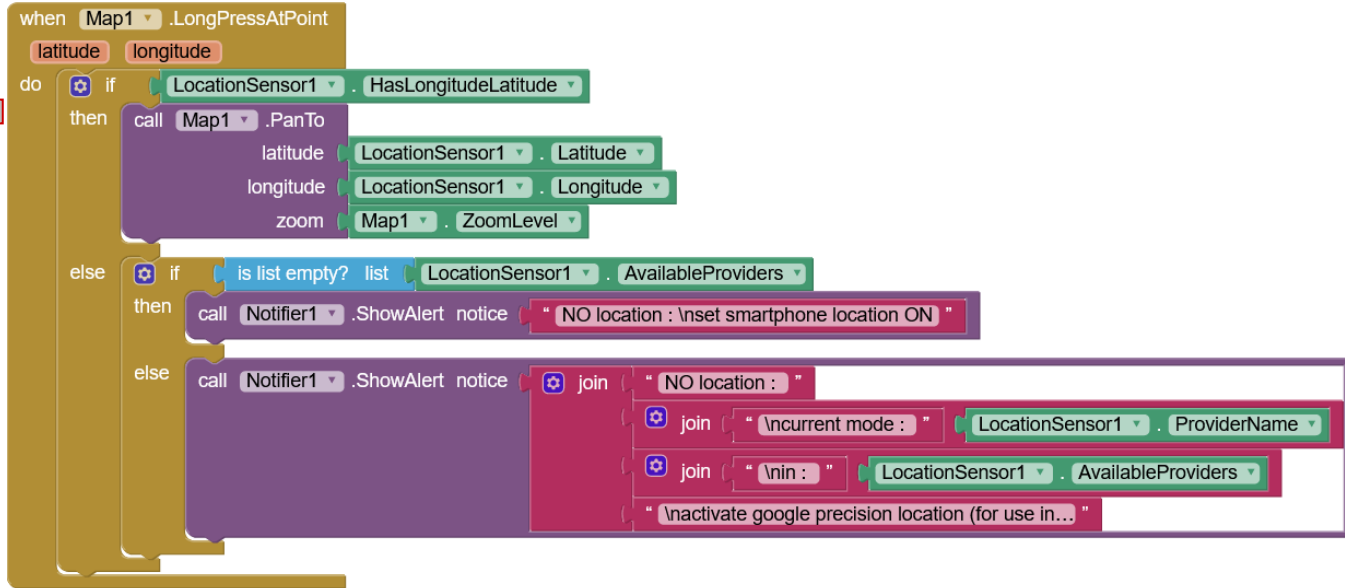
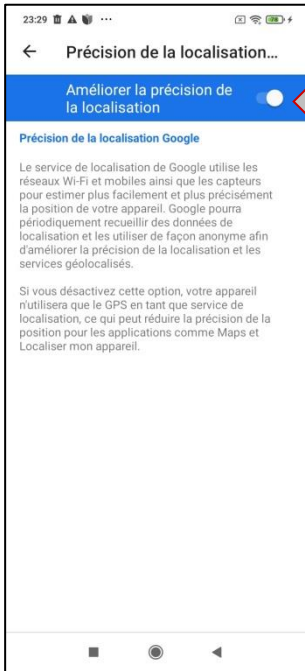


Localisation Wifi, ...
 (network)

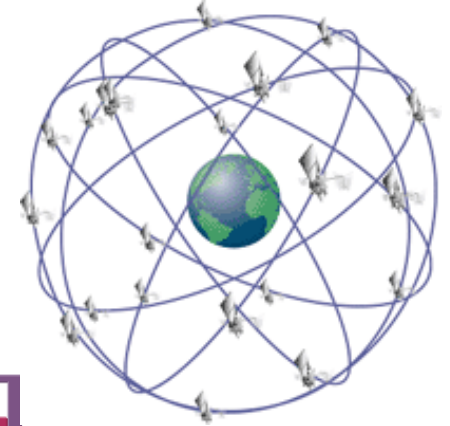


GITSHARE 3a : ACTIVER LA LOCALISATION DU SMARTPHONE

Location on

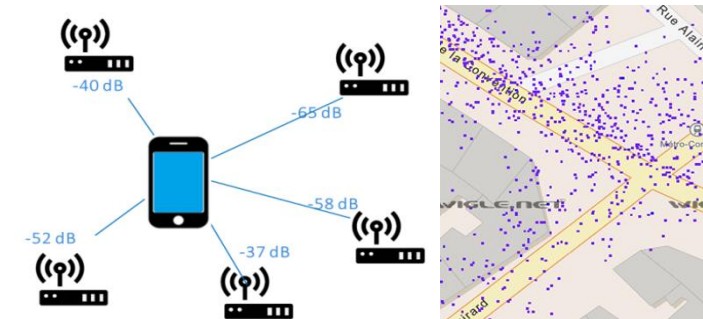


Localisation GPS



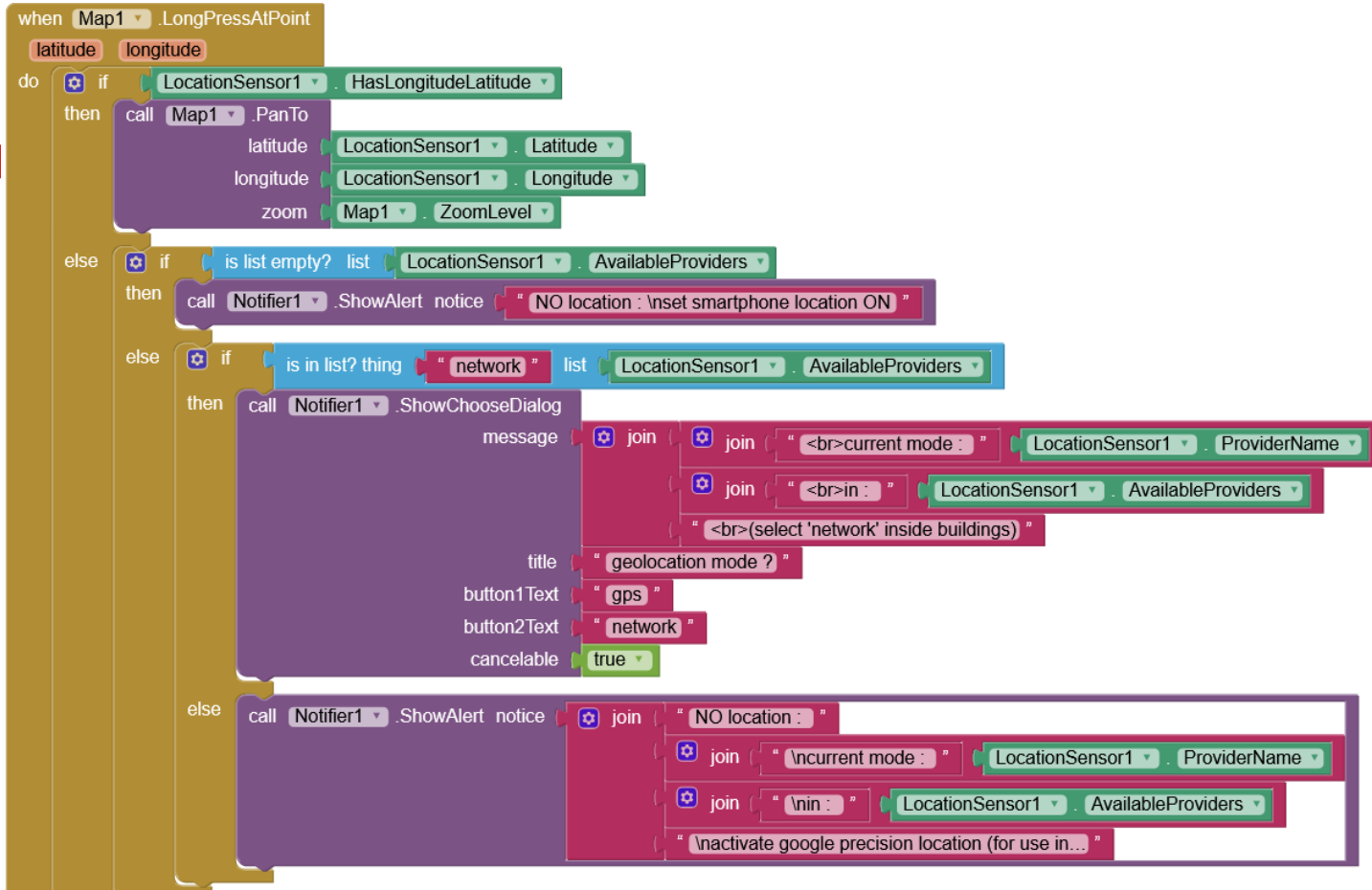
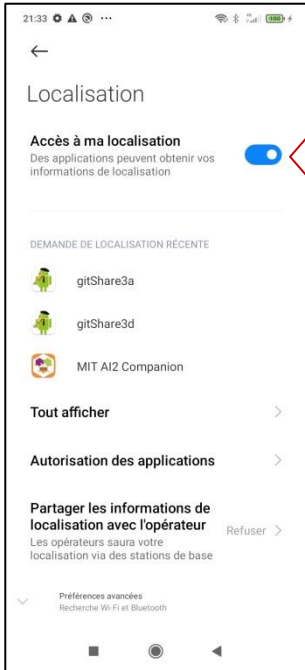
Localisation Wifi, ...
(network)

NO location :
current mode : gps
in : ["gps", "passive", "gps", "network"]
activate google precision location
(for use inside buildings)



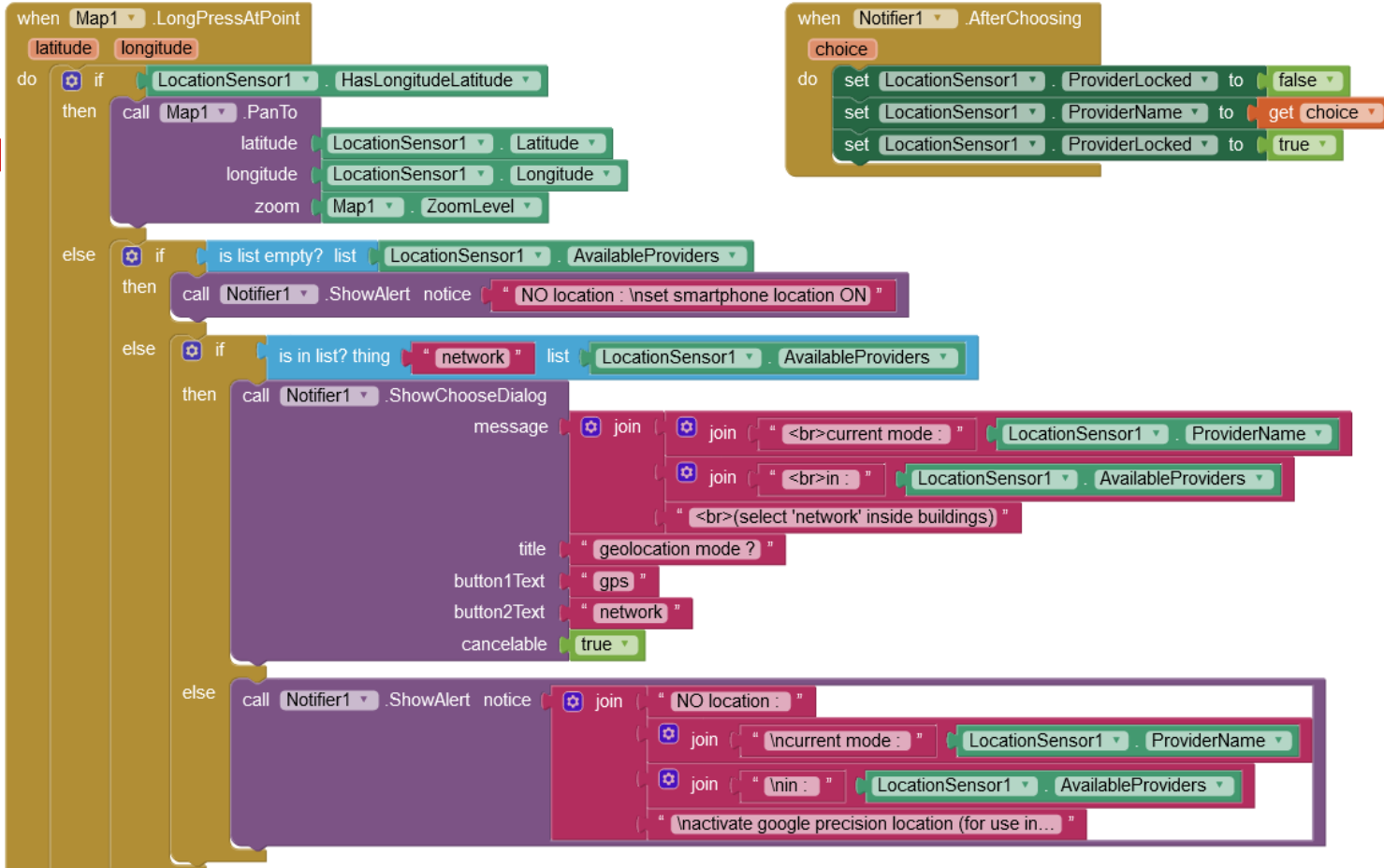
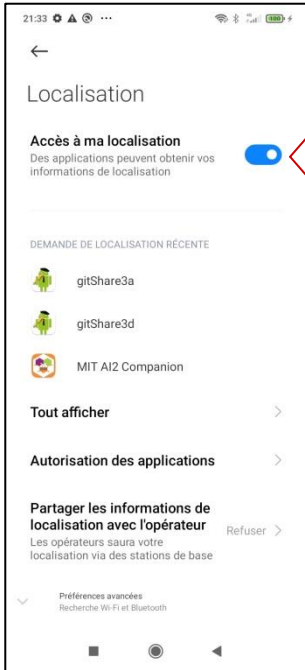
GITSHARE 3a : ACTIVER LA LOCALISATION DU SMARTPHONE

Location on



GITSHARE 3a : ACTIVER LA LOCALISATION DU SMARTPHONE

Location on



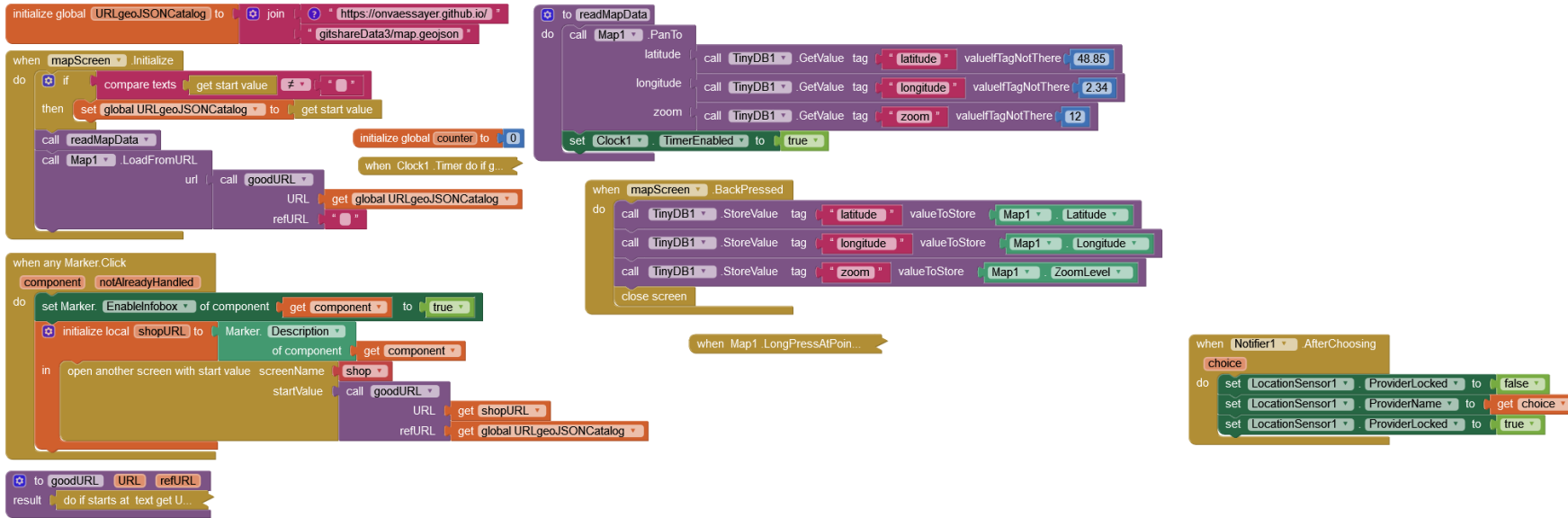
GITSHARE 3a : MAPSCREEN

GITSHARE 3a : MAPSCREEN

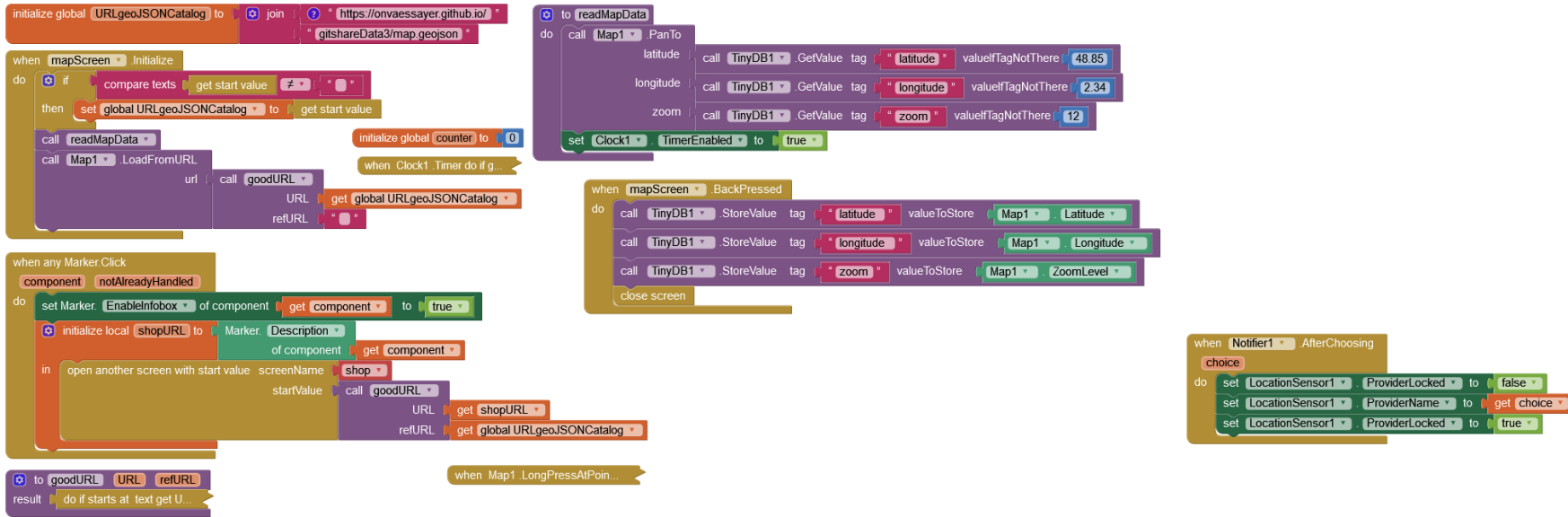
The image displays several interconnected code blocks from a visual programming environment, likely for an Android application. The blocks are color-coded and contain various logic and data handling instructions:

- Initialize global:** Sets a global variable `URLgeoJSONCatalog` to a URL: `https://onvaessayer.github.io/gitshareData3/map_geojson`.
- when mapScreen Initialize:** A conditional block that checks if the start value is not empty. If true, it sets `URLgeoJSONCatalog` to the start value, calls `readMapData`, and loads data from a URL using `goodURL`.
- readMapData:** A block that calls `Map1.PanTo` with latitude (48.85), longitude (2.34), and zoom (12). It also sets `Clock1.TimerEnabled` to true.
- when any Marker.Click:** A block that sets a marker's `EnableInfoBox` to true, initializes a `shopURL` for the marker, and opens another screen with the `shop` screen name and the `shopURL` as the start value.
- when Map1 BackPressed:** A block that stores the current map's latitude, longitude, and zoom level into `TinyDB1` and then closes the screen.
- when Map1 LongPressAtPoint:** A complex block that checks if location services are available. If not, it shows an alert. If available, it calls `Map1.PanTo` with the current location. It also shows a dialog for selecting a location provider (GPS or network) and an alert for location mode settings.
- when Notifier1 AfterChoosing:** A block that sets `LocationSensor1.ProviderLocked` to false, sets the provider name to the chosen choice, and sets `LocationSensor1.ProviderLocked` to true.
- Utility blocks:** Several smaller blocks for handling URLs and text, such as `goodURL.URL.refURL` and `do if starts at text get U...`.

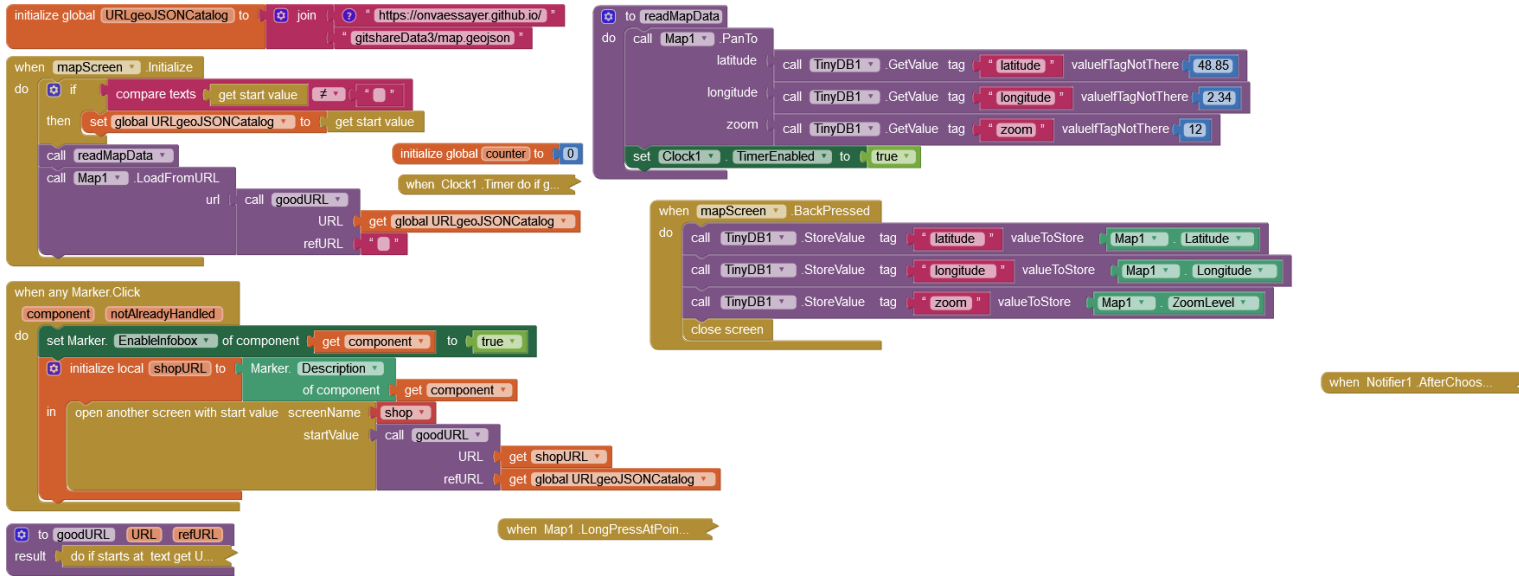
GITSHARE 3a : MAPSCREEN



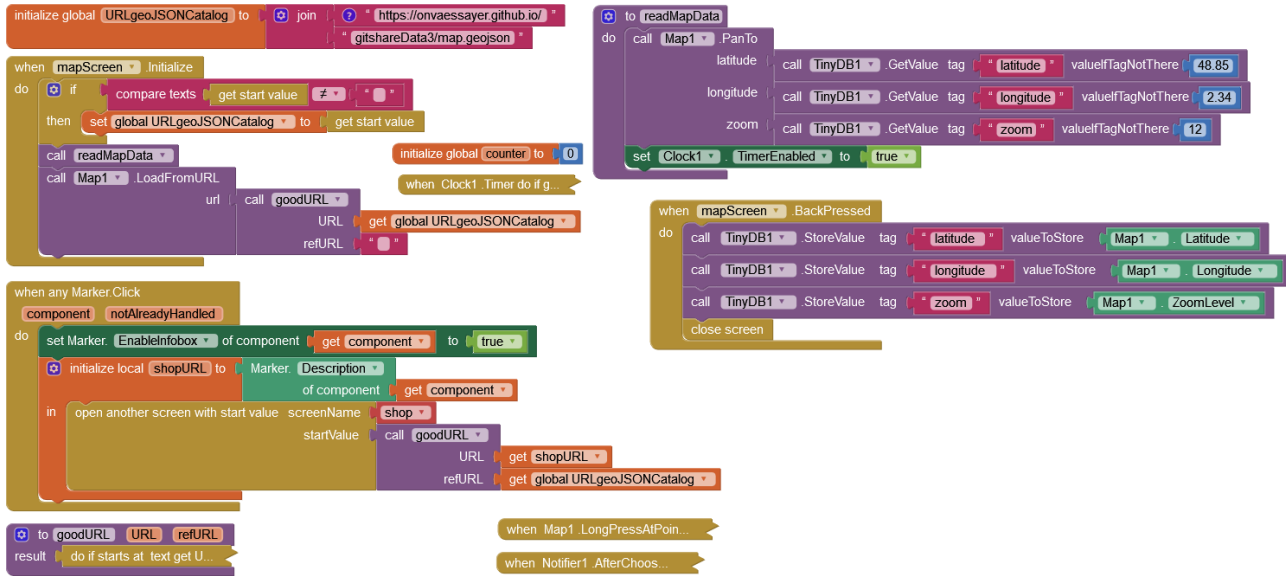
GITSHARE 3a : MAPSCREEN



GITSHARE 3a : MAPSCREEN



GITSHARE 3a : MAPSCREEN



GITSHARE 3a : MAPSCREEN

The image displays several Scratch code blocks for a map application. The blocks are organized into several functional sections:

- Initialization:** A block to initialize a global variable `URLgeoJSONCatalog` to a join of `"https://onvaessayer.github.io/"` and `"gitshareData3/map.geojson"`.
- Screen Initialization:** A `when mapScreen.Initialize` block containing:
 - An `if` block: `compare texts` (get start value) `≠` `" "`. If true, `set global URLgeoJSONCatalog` to `get start value`.
 - A `call readMapData` block.
 - A `call Map1.LoadFromURL` block with `url` set to `call goodURL` and `refURL` set to `" "`.
 - An `initialize global counter` block set to `0`.
 - A `when Clock1.Timer` block with `do if g...`.
- Map Data Retrieval:** A `to readMapData` block:
 - `call Map1.PanTo` with `latitude` (call `TinyDB1.GetValue` tag `"latitude"` value `48.85`), `longitude` (call `TinyDB1.GetValue` tag `"longitude"` value `2.34`), and `zoom` (call `TinyDB1.GetValue` tag `"zoom"` value `12`).
 - `set Clock1.TimerEnabled` to `true`.
- Marker Click:** A `when any Marker.Click` block:
 - `component notAlreadyHandled`.
 - `do` block:
 - `set Marker.EnableInfobox` of component `get component` to `true`.
 - `initialize local shopURL` to `Marker.Description` of component `get component`.
 - `in` block: `open another screen with start value` (screenName: `shop`, startValue: `call goodURL` with `URL` `get shopURL` and `refURL` `get global URLgeoJSONCatalog`).
- Good URL Function:** A `to goodURL` block with `URL` and `refURL` inputs, and a `result` output: `do if starts at text` `get U...`.
- Back Pressed:** A `when mapScreen.BackPressed` block:
 - `do` block:
 - `call TinyDB1.StoreValue` tag `"latitude"` valueToStore `Map1.Latitude`.
 - `call TinyDB1.StoreValue` tag `"longitude"` valueToStore `Map1.Longitude`.
 - `call TinyDB1.StoreValue` tag `"zoom"` valueToStore `Map1.ZoomLevel`.
 - `close screen`.
- Other Triggers:** `when Map1.LongPressAtPoin...` and `when Notifier1.AfterChoos...`.

GITSHARE 3a : MAPSCREEN

The image displays several Scratch code blocks for a map application:

- Initialize global URLgeoJSONCatalog to** join "https://onvaessayer.github.io/" and "gitshareData3/map.geojson".
- when mapScreen.Initialize** do:
 - if** compare texts **get start value** **≠** " " then **set global URLgeoJSONCatalog to** **get start value**.
 - call readMapData**.
 - call Map1.LoadFromURL** with **url** **call goodURL** (URL: **get global URLgeoJSONCatalog**, refURL: " ") and **initialize global counter to 0**.
 - when Clock1.Timer do if g...**
- when any Marker.Click** **component notAlreadyHandled** do:
 - set Marker.EnableInfoBox of component** **get component** to **true**.
 - initialize local shopURL to** **Marker.Description of component** **get component**.
 - in** **open another screen with start value** **screenName** **shop** and **startValue** **call goodURL** (URL: **get shopURL**, refURL: **get global URLgeoJSONCatalog**).
- to goodURL URL refURL** **result** **do if starts at text get U...**
- when Map1.LongPressAtPoin...**
- when Notifier1.AfterChoos...**
- to readMapData** do:
 - call Map1.PanTo** with **latitude** **call TinyDB1.GetValue tag "latitude" valueIfTagNotThere 48.85**, **longitude** **call TinyDB1.GetValue tag "longitude" valueIfTagNotThere 2.34**, and **zoom** **call TinyDB1.GetValue tag "zoom" valueIfTagNotThere 12**.
 - set Clock1.TimerEnabled to true**.
 - when mapScreen.BackPresse...**

GITSHARE 3a : MAPSCREEN

The image displays a collection of Scratch code blocks for a map application. The blocks are organized into several functional sections:

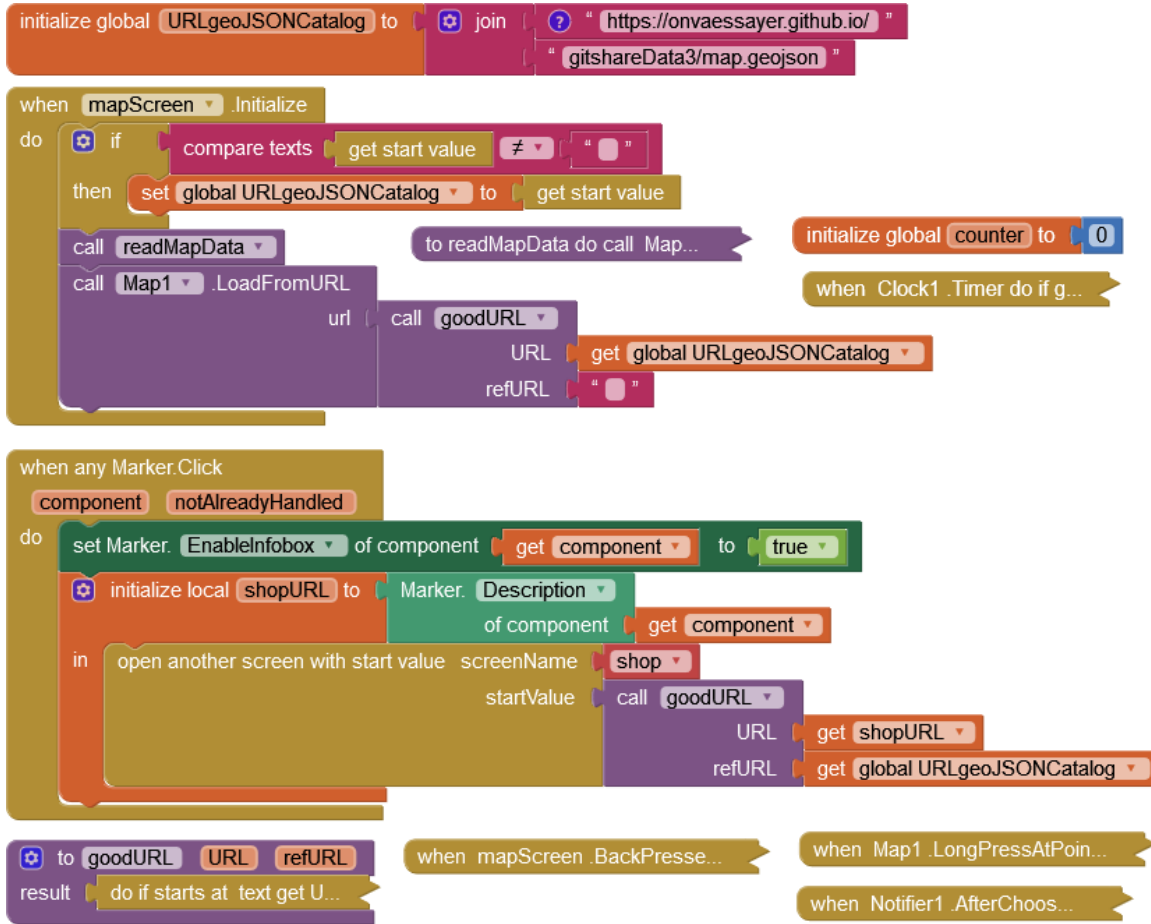
- Initialization:** A block to initialize a global variable `URLgeoJSONCatalog` to a join of `https://onvaessayer.github.io/` and `gitshareData3/map.geojson`.
- Map Screen Initialization:** A `when mapScreen.Initialize` block containing:
 - An `if` block that compares the `get start value` to `" "`. If true, it sets `global URLgeoJSONCatalog` to `get start value`.
 - A `call readMapData` block.
 - A `call Map1.LoadFromURL` block with a `url` block containing:
 - `call goodURL` block with `URL` set to `get global URLgeoJSONCatalog` and `refURL` set to `" "`.
 - An `initialize global counter to 0` block.
 - A `when Clock1.Timer do if g...` block.
- Marker Click Event:** A `when any Marker.Click` block containing:
 - `component notAlreadyHandled` block.
 - A `do` block:
 - `set Marker.EnableInfoBox of component` to `get component` to `true`.
 - `initialize local shopURL to` `Marker.Description of component` to `get component`.
 - An `in` block: `open another screen with start value` `screenName` set to `shop` and `startValue` set to `call goodURL` block with `URL` set to `get shopURL` and `refURL` set to `get global URLgeoJSONCatalog`.
- Utility and Event Blocks:**
 - `to goodURL` block with `URL` and `refURL` inputs and a `do if starts at text get U...` block.
 - `when mapScreen.BackPresse...` block.
 - `when Map1.LongPressAtPoin...` block.
 - `when Notifier1.AfterChoos...` block.
- readMapData Function:** A `to readMapData` block containing:
 - `call Map1.PanTo` block with `latitude` set to `call TinyDB1.GetValue tag "latitude" valueIfTagNotThere 48.85`, `longitude` set to `call TinyDB1.GetValue tag "longitude" valueIfTagNotThere 2.34`, and `zoom` set to `call TinyDB1.GetValue tag "zoom" valueIfTagNotThere 12`.
 - `set Clock1.TimerEnabled to true` block.

GITSHARE 3a : MAPSCREEN

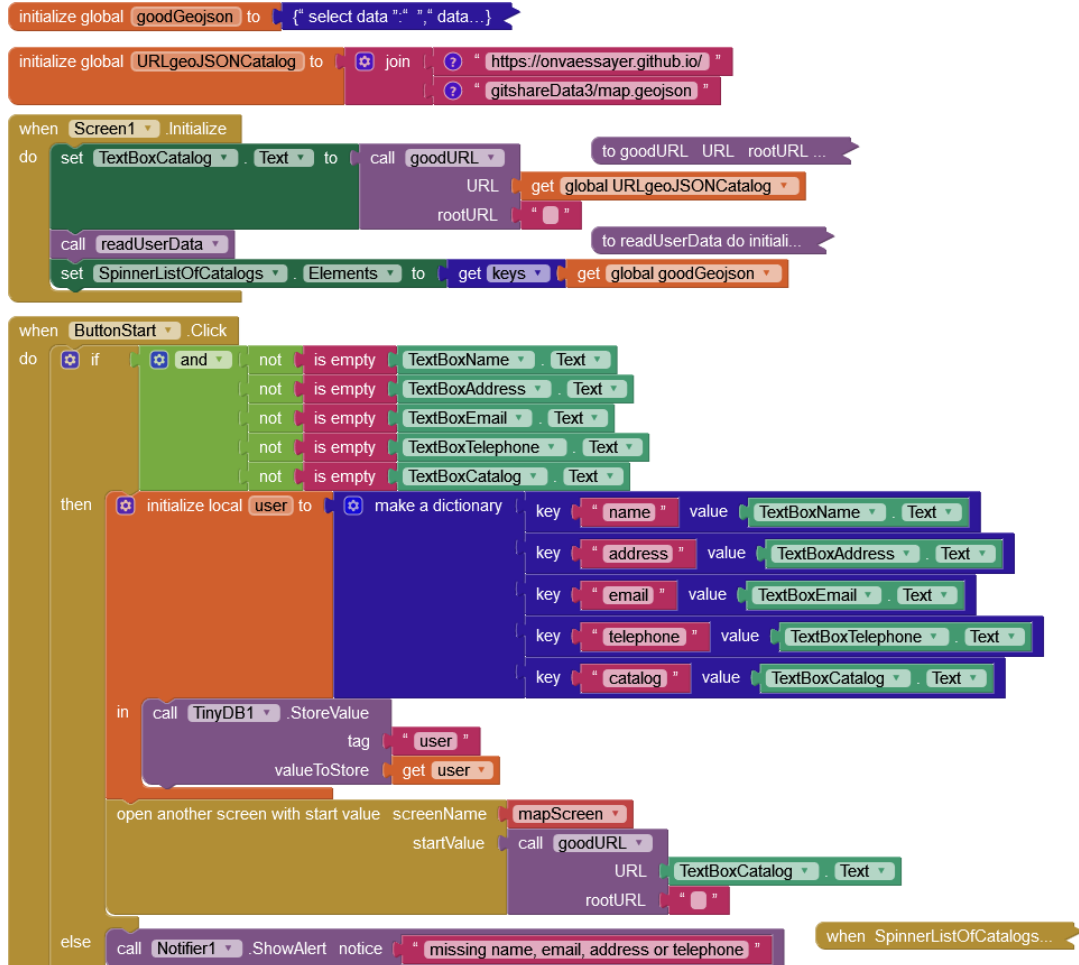
The image displays a collection of Scratch code blocks for a map application. The code is organized into several functional sections:

- Initialization:** A block to initialize a global variable `URLgeoJSONCatalog` to a join of `https://onvaessayer.github.io/` and `gitshareData3/map.geojson`. A callout bubble points to the `readMapData` block within this section.
- mapScreen.Initialize:** A 'when green flag clicked' block containing:
 - An 'if' block: if `get start value` is not equal to `"0"`, then `set global URLgeoJSONCatalog` to `get start value`.
 - A `readMapData` block.
 - A `Map1` block with `.LoadFromURL` method, where `url` is `call goodURL` and `refURL` is `"0"`.
 - A `goodURL` block with `URL` set to `get global URLgeoJSONCatalog`.
 - An `initialize global counter` block set to `0`.
 - A `when Clock1.Timer` block with `do if g...`.
- when any Marker.Click:** A block containing:
 - `component notAlreadyHandled`.
 - `set Marker.EnableInfobox` of component `get component` to `true`.
 - `initialize local shopURL` to `Marker.Description` of component `get component`.
 - An `in` block: `open another screen with start value` `screenName` `shop` and `startValue` `call goodURL` (with `URL` `get shopURL` and `refURL` `get global URLgeoJSONCatalog`).
- Event Listeners:** Three blocks at the bottom:
 - `to goodURL` with `URL` `refURL` and `do if starts at text get U...`.
 - `when mapScreen.BackPresse...`.
 - `when Map1.LongPressAtPoin...`.
 - `when Notifier1.AfterChoos...`.

GITSHARE 3a : MAPSCREEN



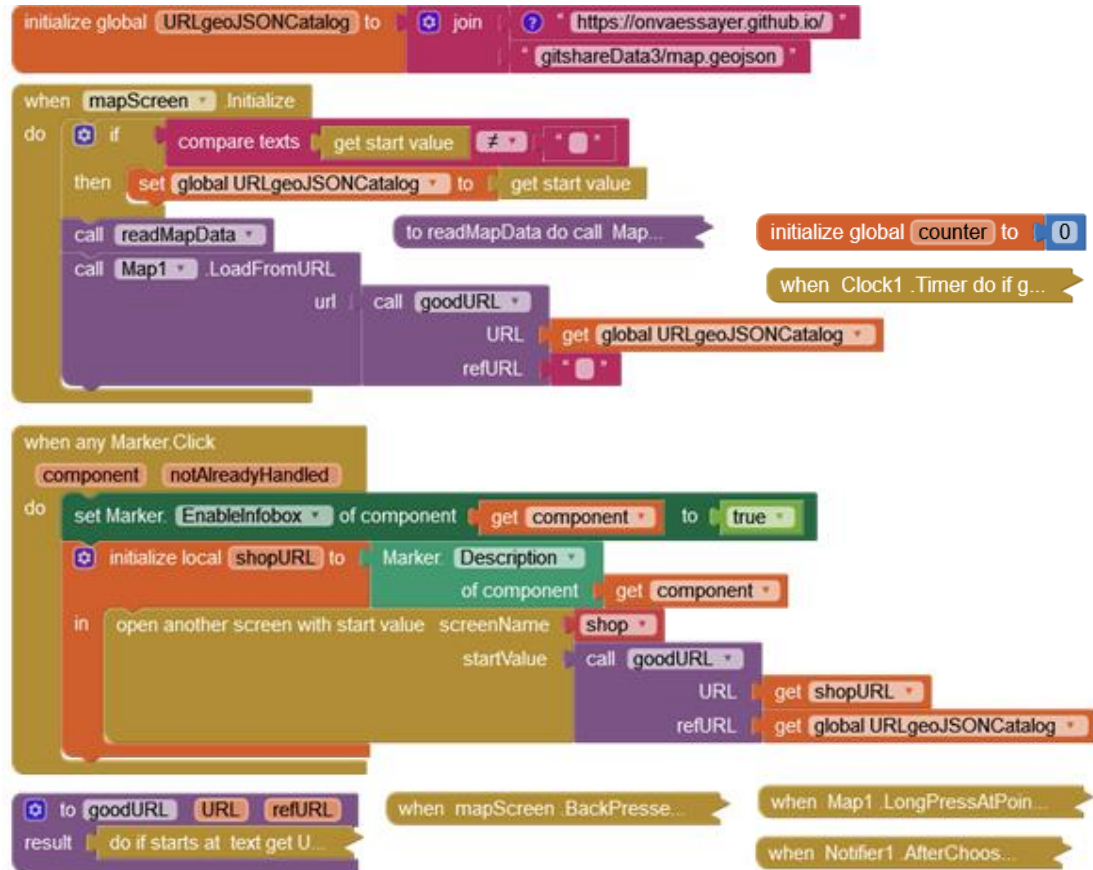
GITSHARE 3a : SCREEN1



Scratch code for SCREEN1:

- Initialize global `goodGeojson` to `["select data": ":", "data..."]`
- Initialize global `URLgeoJSONCatalog` to `join` of `["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]`
- When `Screen1` Initialize:
 - do `set TextBoxCatalog . Text` to `call goodURL` with `URL` `get global URLgeoJSONCatalog` and `rootURL` `"&"`
 - `call readUserData` with `to readUserData do initial...`
 - `set SpinnerListOfCatalogs . Elements` to `get keys` of `get global goodGeojson`
- When `ButtonStart` Click:
 - do `if` `and` of `not is empty` for `TextBoxName . Text`, `TextBoxAddress . Text`, `TextBoxEmail . Text`, `TextBoxTelephone . Text`, and `TextBoxCatalog . Text`.
 - then `initialize local user` to `make a dictionary` with keys: `"name"` value `TextBoxName . Text`, `"address"` value `TextBoxAddress . Text`, `"email"` value `TextBoxEmail . Text`, `"telephone"` value `TextBoxTelephone . Text`, and `"catalog"` value `TextBoxCatalog . Text`.
 - `in` `call TinyDB1 . StoreValue` with `tag` `"user"` and `valueToStore` `get user`.
 - `open another screen with start value` `screenName` `mapScreen` and `startValue` `call goodURL` with `URL` `TextBoxCatalog . Text` and `rootURL` `"&"`.
 - else `call Notifier1 . ShowAlert` notice `"missing name, email, address or telephone"` and `when SpinnerListOfCatalogs...`

& MAPSCREEN



Scratch code for MAPSCREEN:

- Initialize global `URLgeoJSONCatalog` to `join` of `["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]`
- When `mapScreen` Initialize:
 - do `if` `compare texts` `get start value` `≠` `"&"` then `set global URLgeoJSONCatalog` to `get start value`.
 - `call readMapData` with `to readMapData do call Map...`
 - `call Map1 . LoadFromURL` with `url` `call goodURL` and `refURL` `"&"`.
 - `initialize global counter` to `0`
 - `when Clock1 . Timer` do `if g...`
- When any `Marker` Click:
 - `component` `notAlreadyHandled`
 - do `set Marker . EnableInfoBox` of `component` to `true`.
 - `initialize local shopURL` to `Marker . Description` of `component`.
 - `in` `open another screen with start value` `screenName` `shop` and `startValue` `call goodURL` with `URL` `get shopURL` and `refURL` `get global URLgeoJSONCatalog`.
 - `to goodURL` with `URL` and `refURL` `when mapScreen BackPresse...` and `do if starts at text get U...`
 - `when Map1 . LongPressAtPoin...`
 - `when Notifier1 . AfterChoos...`

ESSAIS

GITSHARE 3a : SCREEN1

```
initialize global goodGeojson to ["select data": "data..."]
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL to goodURL URL rootURL ...
  call readUserData to readUserData do initial...
  set SpinnerListOfCatalogs.Elements to get keys get global goodGeojson

when ButtonStart.Click
do
  if and not isEmpty TextBoxName.Text
    not isEmpty TextBoxAddress.Text
    not isEmpty TextBoxEmail.Text
    not isEmpty TextBoxTelephone.Text
    not isEmpty TextBoxCatalog.Text
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName.Text
    key "address" value TextBoxAddress.Text
    key "email" value TextBoxEmail.Text
    key "telephone" value TextBoxTelephone.Text
    key "catalog" value TextBoxCatalog.Text
  in
    call TinyDB1.StoreValue tag "user" valueToStore get user
  open another screen with start value screenName mapScreen
  startValue call goodURL URL TextBoxCatalog.Text
  rootURL " "
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
  when SpinnerListOfCatalogs...
```

& MAPSCREEN

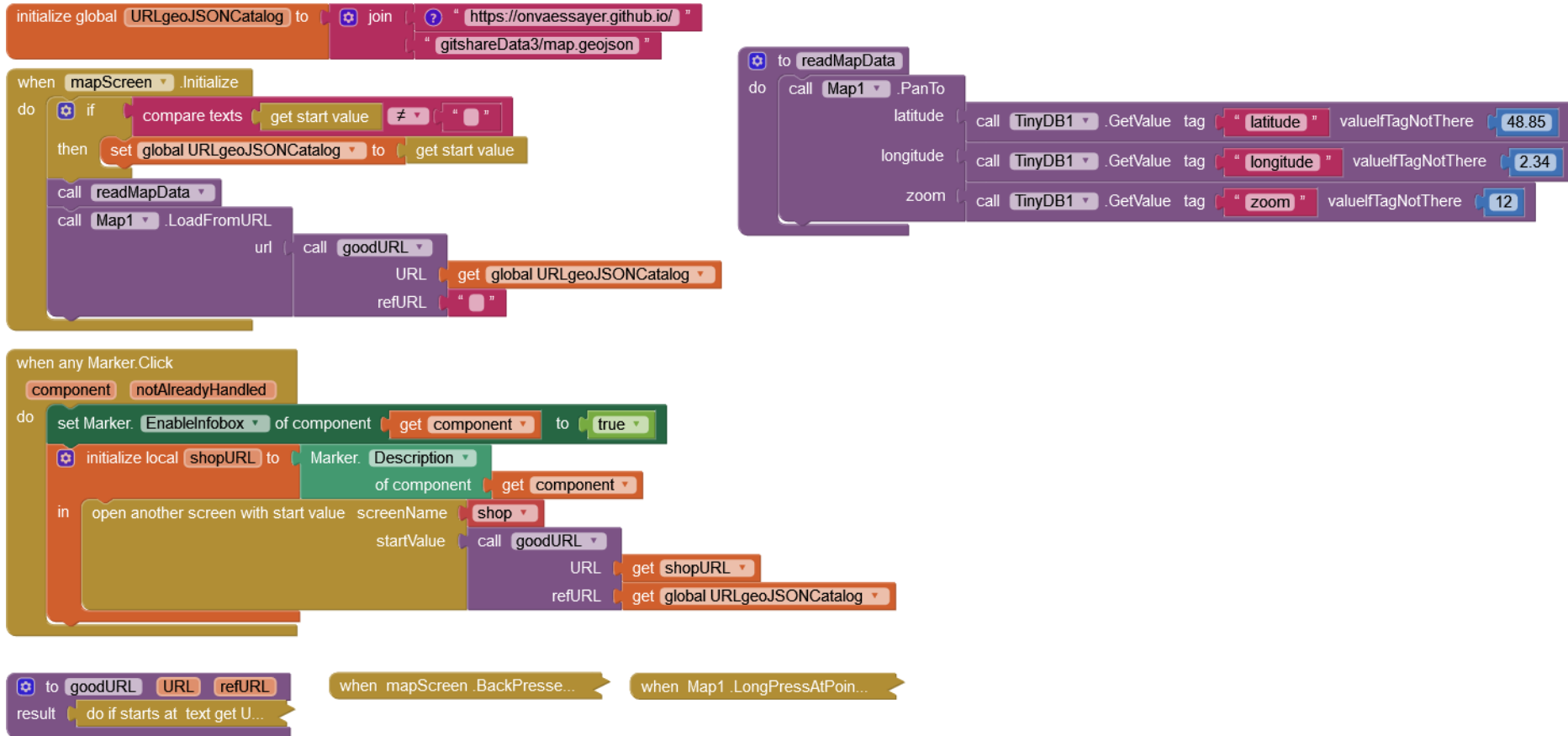
```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when mapScreen.Initialize
do
  if compare texts get start value ≠ " "
  then
    set global URLgeoJSONCatalog to get start value
  call readMapData to readMapData do call Map...
  call Map1.LoadFromURL url call goodURL URL get global URLgeoJSONCatalog
  refURL " "

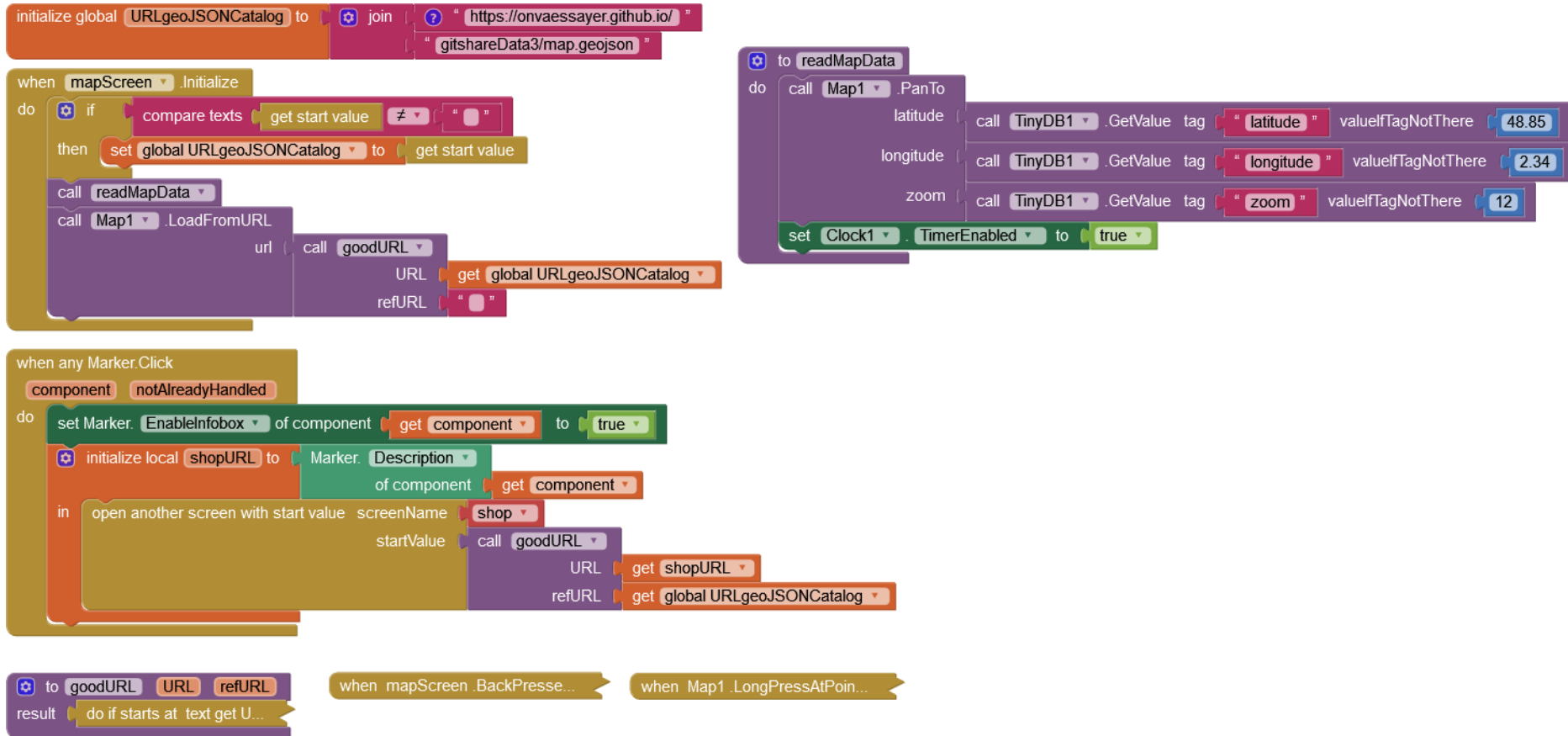
when any Marker.Click
component notAlreadyHandled
do
  set Marker.EnableInfoBox of component get component to true
  initialize local shopURL to Marker.Description of component
  in
    open another screen with start value screenName shop
    startValue call goodURL URL get shopURL
    refURL get global URLgeoJSONCatalog

to goodURL URL refURL
when mapScreen.BackPress...
when Map1.LongPressAtPoin...
when Notifier1.AfterChoo...
```


GITSHARE 3a : MAPSCREEN *(patch map.pan to)*



GITSHARE 3a : MAPSCREEN *(patch map.pan to)*

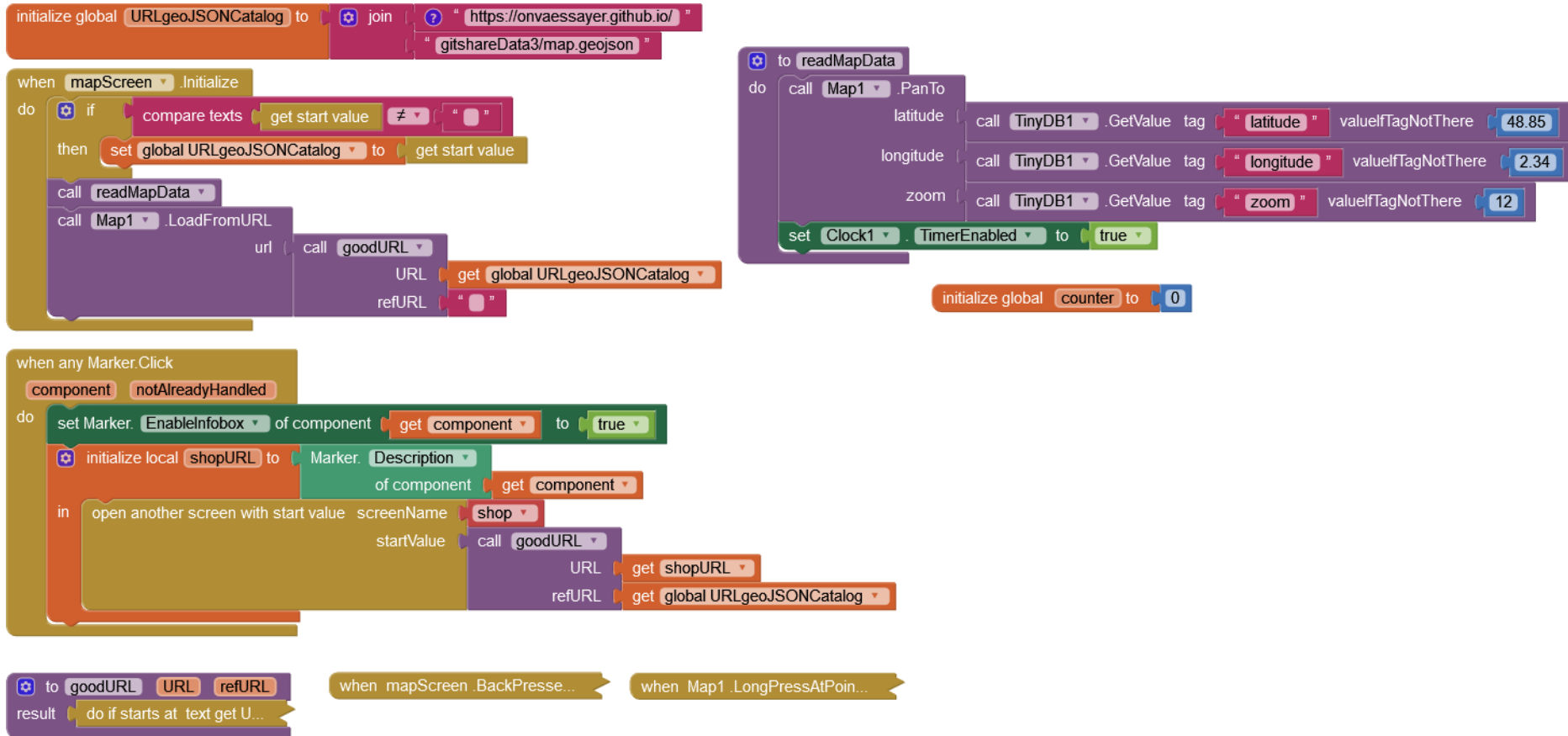


GITSHARE 3a : MAPSCREEN *(patch map.pan to)*

The code is organized into several sections:

- Initialization:** A block to initialize a global variable `URLgeoJSONCatalog` to a join of `https://onvaessayer.github.io/` and `gitshareData3/map.geojson`.
- mapScreen.Initialize:** A `when` block containing:
 - An `if` block: `compare texts` (get start value `≠` `" "`). If true, `set global URLgeoJSONCatalog to` get start value.
 - `call readMapData`
 - `call Map1 .LoadFromURL` with `url` set to `call goodURL` (URL: get global URLgeoJSONCatalog, refURL: `" "`).
- readMapData:** A `do` block:
 - `call Map1 .PanTo` with `latitude` (call TinyDB1 .GetValue tag `" latitude "` valueIfTagNotThere `48.85`), `longitude` (call TinyDB1 .GetValue tag `" longitude "` valueIfTagNotThere `2.34`), and `zoom` (call TinyDB1 .GetValue tag `" zoom "` valueIfTagNotThere `12`).
 - `set Clock1 .TimerEnabled to` `true`.
- Global Counter:** `initialize global counter to` `0`.
- when any Marker.Click:** A `do` block:
 - `component notAlreadyHandled`
 - `set Marker. EnableInfoBox of component` get component to `true`.
 - `initialize local shopURL to` Marker. Description of component get component.
 - `in open another screen with start value` screenName `shop`, startValue call goodURL (URL: get shopURL, refURL: get global URLgeoJSONCatalog).
- URL Processing:** A `to goodURL` block with `URL` and `refURL` inputs, and a `do if starts at text get U...` block.
- Navigation:** `when mapScreen.BackPresse...` and `when Map1.LongPressAtPoin...` blocks.

GITSHARE 3a : MAPSCREEN *(patch map.pan to)*



GITSHARE 3a : MAPSCREEN *(patch map.pan to)*

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]
```

```
when mapScreen.Initialize do if compare texts get start value ≠ "" then set global URLgeoJSONCatalog to get start value call readMapData call Map1.LoadFromURL url call goodURL URL get global URLgeoJSONCatalog refURL ""
```

```
to readMapData do call Map1.PanTo latitude call TinyDB1.GetValue tag "latitude" valueIfTagNotThere 48.85 longitude call TinyDB1.GetValue tag "longitude" valueIfTagNotThere 2.34 zoom call TinyDB1.GetValue tag "zoom" valueIfTagNotThere 12 set Clock1.TimerEnabled to true
```

```
when any Marker.Click component notAlreadyHandled do set Marker.EnableInfoBox of component get component to true initialize local shopURL to Marker.Description of component in open another screen with start value screenName shop startValue call goodURL URL get shopURL refURL get global URLgeoJSONCatalog
```

```
initialize global counter to 0
```

```
when Clock1.Timer do if get global counter ≤ 3 then call Map1.PanTo latitude call TinyDB1.GetValue tag "latitude" valueIfTagNotThere 48.85 longitude call TinyDB1.GetValue tag "longitude" valueIfTagNotThere 2.34 zoom call TinyDB1.GetValue tag "zoom" valueIfTagNotThere 12 set global counter to get global counter + 1 else set Clock1.TimerEnabled to false
```

```
to goodURL URL refURL result do if starts at text get U...
```

```
when mapScreen.BackPresse...
```

```
when Map1.LongPressAtPoin...
```

GITSHARE 3a : MAPSCREEN *(patch map.pan to)*

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when mapScreen.Initialize
do
  if compare texts get start value ≠ ""
  then set global URLgeoJSONCatalog to get start value
  call readMapData to readMapData do call Map1.LoadFromURL
  call Map1.LoadFromURL
  url call goodURL
  URL get global URLgeoJSONCatalog
  refURL ""

when any Marker.Click
component notAlreadyHandled
do
  set Marker.EnableInfoBox of component get component to true
  initialize local shopURL to Marker.Description of component
  in open another screen with start value screenName shop
  startValue call goodURL
  URL get shopURL
  refURL get global URLgeoJSONCatalog

to goodURL URL refURL
result do if starts at text get U...

when mapScreen.BackPresse...
when Map1.LongPressAtPoin...
```

```
initialize global counter to 0

when Clock1.Timer
do
  if get global counter ≤ 3
  then call Map1.PanTo
  latitude call TinyDB1.GetValue tag "latitude" valueIfTagNotThere 48.85
  longitude call TinyDB1.GetValue tag "longitude" valueIfTagNotThere 2.34
  zoom call TinyDB1.GetValue tag "zoom" valueIfTagNotThere 12
  set global counter to get global counter + 1
  else set Clock1.TimerEnabled to false
```

GITSHARE 3a : MAPSCREEN *(patch map.pan to)*

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map.geojson"]

when mapScreen.Initialize
do
  if compare texts get start value ≠ ""
  then set global URLgeoJSONCatalog to get start value
  call readMapData to readMapData do call Map1.LoadFromURL
  call Map1.LoadFromURL
  url call goodURL
  URL get global URLgeoJSONCatalog
  refURL ""

  initialize global counter to 0

when any Marker.Click
component notAlreadyHandled
do
  set Marker.EnableInfoBox of component get component to true
  initialize local shopURL to Marker.Description of component
  in open another screen with start value screenName shop
  startValue call goodURL
  URL get shopURL
  refURL get global URLgeoJSONCatalog

to goodURL URL refURL
result do if starts at text get U...

when mapScreen.BackPresse...
when Map1.LongPressAtPoin...
```

```
when Clock1.Timer
do
  if get global counter ≤ 3
  then call Map1.PanTo
  latitude call TinyDB1.GetValue tag "latitude" valueIfTagNotThere 48.85
  longitude call TinyDB1.GetValue tag "longitude" valueIfTagNotThere 2.34
  zoom call TinyDB1.GetValue tag "zoom" valueIfTagNotThere 12
  set global counter to get global counter + 1
  else set Clock1.TimerEnabled to false
```

GITSHARE 3a : MAPSCREEN *(patch map.pan to)*

The image displays a Scratch script for a map screen application, organized into several functional blocks:

- Initialization:** A block to initialize a global variable `URLgeoJSONCatalog` to a list containing `https://onvaessayer.github.io/` and `gitshareData3/map.geojson`.
- mapScreen.Initialize:** A `when green flag clicked` event block containing:
 - An `if` block: `compare texts` (get start value, `≠`, `" "`). If true, `set global URLgeoJSONCatalog to get start value`.
 - `call readMapData` block with a comment: `to readMapData do call Map...`
 - `initialize global counter to 0` block.
 - `call Map1.LoadFromURL` block with parameters: `url` (call `goodURL`), `URL` (get `global URLgeoJSONCatalog`), and `refURL` (`" "`).
- Marker Click:** A `when any Marker.Click` event block containing:
 - `component notAlreadyHandled` block.
 - `do` block:
 - `set Marker.EnableInfoBox of component` (get `component` to `true`).
 - `initialize local shopURL to` (Marker `Description` of `component` (get `component`)).
 - `in` block: `open another screen with start value` (screenName: `shop`, startValue: call `goodURL` (URL: get `shopURL`, refURL: get `global URLgeoJSONCatalog`)).
- Navigation:** A `to goodURL` block (URL: `refURL`, result: `do if starts at text get U...`).
- Events:** `when mapScreen.BackPresse...` and `when Map1.LongPressAtPoin...` blocks.

GITSHARE 3a : MAPSCREEN *(patch map.pan to)*

The image displays a Scratch script for a map screen application, organized into several functional sections:

- Initialization:** A block to initialize a global variable `URLgeoJSONCatalog` to a list containing `https://onvaessayer.github.io/` and `gitshareData3/map.geojson`.
- mapScreen.Initialize:** A `when green flag clicked` event block followed by an `if` statement. The condition is `compare texts` (get start value `≠` `" "`). If true, it `set global URLgeoJSONCatalog to get start value`. It then `call readMapData` (with a note `to readMapData do call Map...`) and `call Map1 .LoadFromURL`. The `url` parameter for `LoadFromURL` is a `call goodURL` block with `URL` set to `get global URLgeoJSONCatalog` and `refURL` set to `" "`. A `when Clock1 .Timer do if g...` block is also present.
- Marker Click:** A `when any Marker.Click` event block with `component notAlreadyHandled`. It `set Marker. EnableInfoBox of component get component to true`. It then `initialize local shopURL to Marker. Description of component get component`. An `in` loop `open another screen with start value` is used, with `screenName` set to `shop` and `startValue` set to `call goodURL` (with `URL` set to `get shopURL` and `refURL` set to `get global URLgeoJSONCatalog`).
- Utility and Events:** A `to goodURL` block with `URL` and `refURL` parameters, containing a `do if starts at text get U...` block. Two event blocks are shown: `when mapScreen.BackPresse...` and `when Map1.LongPressAtPoin...`.

GITSHARE 3a : SCREEN1

```
initialize global goodGeojson to ["select data ":" data..."]

initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/"
"gitshareData3/map.geojson"]

when Screen1.Initialize
do
  set TextBoxCatalog.Text to call goodURL to goodURL URL rootURL ...
  URL get global URLgeoJSONCatalog
  rootURL ""
  call readUserData to readUserData do initial...
  set SpinnerListOfCatalogs.Elements to get keys get global goodGeojson

when ButtonStart.Click
do
  if and not isEmpty TextBoxName.Text
  not isEmpty TextBoxAddress.Text
  not isEmpty TextBoxEmail.Text
  not isEmpty TextBoxTelephone.Text
  not isEmpty TextBoxCatalog.Text
  then
    initialize local user to make a dictionary
    key "name" value TextBoxName.Text
    key "address" value TextBoxAddress.Text
    key "email" value TextBoxEmail.Text
    key "telephone" value TextBoxTelephone.Text
    key "catalog" value TextBoxCatalog.Text
    in call TinyDB1.StoreValue tag "user" valueToStore get user
    open another screen with start value screenName mapScreen
    startValue call goodURL URL TextBoxCatalog.Text
    rootURL ""
  else
    call Notifier1.ShowAlert notice "missing name, email, address or telephone"
    when SpinnerListOfCatalogs...
```

& MAPSCREEN

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/"
"gitshareData3/map.geojson"]

when mapScreen.Initialize
do
  call mapScreen.AskForPermission permissionName Permission CoarseLocation
  call mapScreen.AskForPermission permissionName Permission FineLocation
  if compare texts get start value ≠ ""
  then
    set global URLgeoJSONCatalog to get start value
    when mapScreen.Permission...
  call readMapData to readMapData do call Map...
  call Map1.LoadFromURL url call goodURL URL get global URLgeoJSONCatalog
  refURL ""
  initialize global counter to 0
  when Clock1.Timer do if g...

when any Marker.Click
component notAlreadyHandled
do
  set Marker.EnableInfobox of component get component to true
  initialize local shopURL to Marker.Description
  of component get component
  in open another screen with start value screenName shop
  startValue call goodURL URL get shopURL
  refURL get global URLgeoJSONCatalog

to goodURL URL refURL
when mapScreen.BackPresse...
when Map1.LongPressAtPoin...
do if starts at text get U...
when Notifier1.AfterChoos...
```

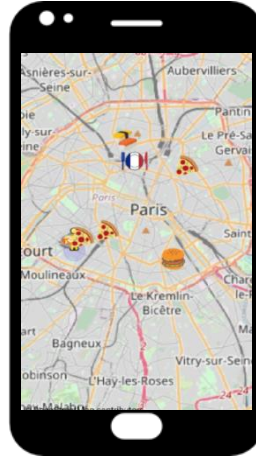
3.5

Préparer et passer
une commande

GITSHARE 3A → 3B : PRÉPARER & PASSER COMMANDE



Screen1



mapScreen



shop

```
initialize global goodGeosjon to ["dataset1" join "https://"]
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when Screen1 Initialize
do set TextBoxCatalog to Text to call goodURL refURL rootURL
URL get global URLgeoJSONCatalog
call readUserData to readUserData do initialize
set SpinnerListOfCatalogs to Elements to get keys to get global goodGeosjon

when ButtonStart Click
do if and [not is empty TextBoxName - Text -
not is empty TextBoxAddress - Text -
not is empty TextBoxEmail - Text -
not is empty TextBoxTelephone - Text -]
then initialize local user to make a dictionary key name value TextBoxName - Text -
key address value TextBoxAddress - Text -
key email value TextBoxEmail - Text -
key telephone value TextBoxTelephone - Text -
key catalog value TextBoxCatalog - Text -
in call FirebaseFirestoreValue tag user valueToStore get user -
Open another screen with start value screenName mapScreen startValue call goodURL refURL rootURL
else call Notifier1 showAlert notice "missing name, email, address or telephone"
```

```
initialize global URLgeoJSONCatalog to join ["https://onvaessayer.github.io/", "gitshareData3/map_geojson"]

when mapScreen Initialize
do if compare texts get start value != ""
then set global URLgeoJSONCatalog to get start value
call readMapData to readMapData do call Map to initialize global counter to 0
call Map1 LoadFromURL URL get global URLgeoJSONCatalog refURL
when Clock1 Timer do if g to goodURL URL refURL ...

when any Marker Click
component notAlreadyHandled
do set Marker EnableInfoBox of component get component to true
initialize local shopURL to Marker Description of component get component
in open another screen with start value screenName shop startValue call goodURL refURL get shopURL -
get global URLgeoJSONCatalog
```

```
initialize global shop to create empty dictionary
initialize global items to create empty list

when shop Initialize
do set Web1 Url to get start value
set Web1 SaveResponse to false
call Web1 Get

when Web1 GotText
url responseCode responseType responseContent
do if get responseCode == 200
then set global shop to call Web1 JsonTextDecodeWithDictionaries jsonText get responseContent -
if is a dictionary? get global shop
then set LabelTitle to Text to get value for key title in dictionary get global shop or if not found
set LabelAddress to Text to get value for key address in dictionary get global shop or if not found
set Image to Picture to call goodURL refURL get value for key image in dictionary get global shop or if not found AndroidFR.png
set global items to get value for key items in dictionary get global shop or if not found create empty list
if is a list? thing get global items
then call setListViewElements to myItems get global items to goodURL URL rootURL result do if starts at text get U
else call Notifier1 ShowMessage
else call Notifier1 ShowMessage
do to setListViewElements myItems
do initialize local listViewElements to create empty list
in for each item in list get m ...
```

PLAN

- Introduction : décomposition de l'application
- 1. Création d'un site Web / serveur de données
- 2. Définition des données et préparation d'un jeu
- 3. Création de l'application mobile avec App Inventor
 - 1. V1 : visualiser le catalogue des restaurants sur une carte
 - 2. V2a : sélectionner un restaurant et afficher ses nom, adresse, image et liste de plats
 - 3. V2b : codage défensif, modèle de données, adresses relatives, Dropbox & Google Drive
 - 4. V3a : identifier et enregistrer l'utilisateur et la carte
 - 5. V3b : préparer et passer une commande
 - 6. V3c : bonus

GITSHARE 3b : shop Screen - design

The screenshot displays the MIT App Inventor interface for a project named 'gitshare03b_E2'. The main workspace shows a mobile device preview of a 'shop' screen with fields for 'name or title' and 'address', a list of items (item1, item2, item3), and a section for 'ordered items' with '+' and '-' buttons and an 'order' button. The Components panel on the right shows a tree view of the screen's components, including a list of components that are currently hidden. A callout box provides a detailed view of these hidden components: VerticalArgtOrder, VerticalScrollArgt, LabelOrder, HorizontalArrgt, Label5, ButtonAdd, Label3, ButtonSubstract, Label4, and ButtonOrder. Another callout box, titled 'Non-visible components', lists Notifier1, Web1, TinyDB1, Texting1, and Clock1, which are components that are not visible on the screen but are present in the project.

GITSHARE 3a → 3b : ITEM MODEL – name, price, quantity

order

From :
Pierre Hugué
place d'alleray Paris 15

To :
Le Roma
61 avenue du Maine, 75014 Paris

For :
22.8 € 2 x pizza Margherita
25.8 € 2 x tagliatelles carbonara
17.5 € 1 x Chianti
66.1 € total

confirm cancel

```
{
  "title": "Le Roma",
  "address": "61 av Maine 75014 Paris",
  "image": "image.jpg",
  "telephone": "0600000004",
  "email": "ristoranteRoma@paris.fr",
  "items": [
    {
      "name": "pizza Margherita",
      "price": 11.40,
      "ingredients": ["Mozarella",
                     "champignons"],
      "image": "margherita.PNG"
    },
    {
      "name": "spaghetti carbonara",
      "price": 12.90,
      "ingredients": ["farine",
                     "boeuf"],
      "image": "carbonara.PNG"
    },
    {
      "name": "Chianti",
      "price": 17.5,
      "ingredients": ["raisin",
                     "alcool"],
      "image": "chianti.jpg"
    }
  ]
}
```

GITSHARE 3a → 3b : ITEM MODEL – name, price, quantity

order

From :
Pierre Huguet
place d'alleray Paris 15

To :
Le Roma
61 avenue du Maine, 75014 Paris

For :
22.8 € 2 x pizza Margherita
25.8 € 2 x tagliatelles carbonara
17.5 € 1 x Chianti
66.1 € total

confirm cancel

```
{
  "title": "Le Roma",
  "address": "61 av Maine 75014 Paris",
  "image": "image.jpg",
  "telephone": "0600000004",
  "email": "ristoranteRoma@paris.fr",
  "items": [
    {
      "name": "pizza Margherita",
      "price": 11.40,
      "ingredients": ["Mozarella",
                     "champignons"],
      "image": "margherita.PNG"
    },
    {
      "name": "spaghetti carbonara",
      "price": 12.90,
      "ingredients": ["farine",
                     "boeuf"],
      "image": "carbonara.PNG"
    },
    {
      "name": "Chianti",
      "price": 17.5,
      "ingredients": ["raisin",
                     "alcool"],
      "image": "chianti.jpg"
    }
  ]
}
```

```
{
  "title": "Le Roma",
  "address": "61 av Maine 75014 Paris",
  "image": "image.jpg",
  "telephone": "0600000004",
  "email": "ristoranteRoma@paris.fr",
  "items": [
    {
      "pizza Margherita, 11.40",
      "spaghetti carbonara, 12.90",
      "Chianti, 17.5"
    }
  ]
}
```

```
{
  "image": "001.png",
  "title": "Bulbasaur",
  "nom": "Bulbizarre",
  "description": "[Grass ,Poison]",
  "items": [
    "HP : 45",
    "Attack : 49",
    "Defense : 49",
    "Sp. Attack : 65",
    "Sp. Defense : 65",
    "Speed : 45"
  ]
}
```


GITSHARE 3a → 3b : ITEM MODEL – name, price, quantity

order

From :
Pierre Huguet
place d'alleray Paris 15

To :
Le Roma
61 avenue du Maine, 75014 Paris

For :
22.8 € 2 x pizza Margherita
25.8 € 2 x tagliatelles carbonara
17.5 € 1 x Chianti
66.1 € total

confirm cancel

```
{  
  "title": "Le Roma",  
  "address": "61 av Maine 75014 Paris",  
  "image": "image.jpg",  
  "telephone": "0600000004",  
  "email": "ristoranteRoma@paris.fr",  
  "items": [  
    {  
      "name": "pizza Margherita",  
      "price": 11.40,  
      "ingredients": ["Mozarella",  
                     "champignons"],  
      "image": "margherita.PNG"  
    },  
    {  
      "name": "spaghetti carbonara",  
      "price": 12.90,  
      "ingredients": ["farine",  
                     "boeuf"],  
      "image": "carbonara.PNG"  
    },  
    {  
      "name": "Chianti",  
      "price": 17.5,  
      "ingredients": ["raisin",  
                     "alcool"],  
      "image": "chianti.jpg"  
    }  
  ]  
}
```

GITSHARE 3a → 3b : ITEM MODEL – name, price, quantity

order

From :
Pierre Hugué
place d'alleray Paris 15

To :
Le Roma
61 avenue du Maine, 75014 Paris

For :
22.8 € 2 x pizza Margherita
25.8 € 2 x tagliatelles carbonara
17.5 € 1 x Chianti
66.1 € total

confirm cancel

```
{
  "title": "Le Roma",
  "address": "61 av Maine 75014 Paris",
  "image": "image.jpg",
  "telephone": "0600000004",
  "email": "ristoranteRoma@paris.fr",
  "items": [
    {
      "name": "pizza Margherita",
      "price": 11.40,
      "ingredients": ["Mozarella",
                     "champignons"],
      "image": "margherita.PNG"
    },
    {
      "name": "spaghetti carbonara",
      "price": 12.90,
      "ingredients": ["farine",
                     "boeuf"],
      "image": "carbonara.PNG"
    },
    {
      "name": "Chianti",
      "price": 17.5,
      "ingredients": ["raisin",
                     "alcool"],
      "image": "chianti.jpg"
    }
  ]
}
```

```
{
  "title": "Le Roma",
  "address": "61 av Maine 75014 Paris",
  "image": "image.jpg",
  "telephone": "0600000004",
  "email": "ristoranteRoma@paris.fr",
  "items": [
    {
      "name": "pizza Margherita",
      "price": 11.40,
      "ingredients": ["Mozarella",
                     "champignons"],
      "quantity": 2,
      "image": "margherita.PNG"
    },
    {
      "name": "spaghetti carbonara",
      "price": 12.90,
      "ingredients": ["farine",
                     "boeuf"],
      "quantity": 1,
      "image": "carbonara.PNG"
    },
    {
      "name": "Chianti",
      "price": 17.5,
      "ingredients": ["raisin",
                     "alcool"],
      "image": "chianti.jpg"
    }
  ]
}
```

GITSHARE 3a → 3b : SHOP SCREEN

```
initialize global shop to create empty dictionary
initialize global items to create empty list
initialize global shopURL to ""

when Web1 GotText
  url responseCode responseType responseContent
  do
    if get responseCode == 200
      then
        set global shop to call Web1 .JsonTextDecodeWithDictionaries
          jsonText get responseContent
        if is a dictionary? get global shop
          then
            set LabelTitle .Text to get value for key "title" in dictionary get global shop or if not found "?"
            set LabelAddress .Text to get value for key "address" in dictionary get global shop or if not found "?"
            set Image1 .Picture to call goodURL
              URL get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
              refURL get global shopURL
            set global items to get value for key "items" in dictionary get global shop or if not found create empty list
            if is a list? thing get global items
              then
                call setListViewElements myItems get global items
              else
                call Notifier1 .ShowMessag...
            else
                call Notifier1 .ShowMessag...
          else
            call Notifier1 .ShowMessag...
        else
            call Notifier1 .ShowMessag...

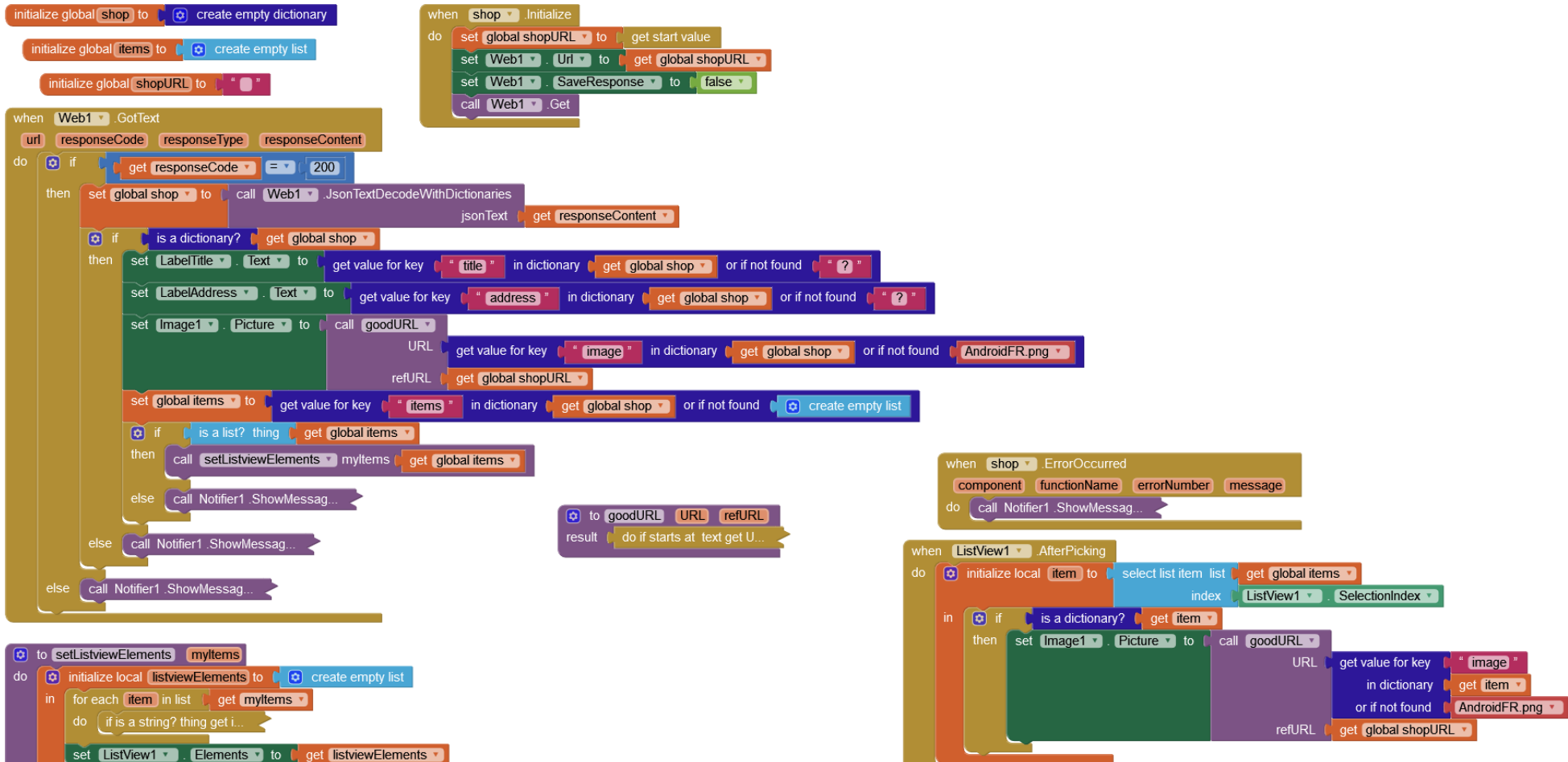
when shop .Initialize
  do
    set global shopURL to get start value
    set Web1 .Url to get global shopURL
    set Web1 .SaveResponse to false
    call Web1 .Get

when shop .ErrorOccurred
  component functionName errorNumber message
  do
    call Notifier1 .ShowMessag...

when ListView1 .AfterPicking
  do
    initialize local item to select list item list get global items
      index ListView1 .SelectionIndex
    in
      if is a dictionary? get item
        then
          set Image1 .Picture to call goodURL
            URL get value for key "image"
              in dictionary get item
            or if not found "AndroidFR.png"
            refURL get global shopURL

to setListViewElements myItems
  do
    initialize local listViewElements to create empty list
    in
      for each item in list get myItems
        do
          if is a string? thing get i...
          set ListView1 .Elements to get listViewElements
```

GITSHARE 3b : SHOP SCREEN



GITSHARE 3b : SHOP SCREEN

```
set global items to get value for key "items" in dictionary get global shop or if not found create empty list
if is a list? thing get global items
then
  call setListViewElements myItems get global items
else
  call Notifier1.ShowMessag...
else
  call Notifier1.ShowMessag...
```

```
to goodURL URL refURL
result do if starts at text get U...
```

```
to setListViewElements myItems
do
  initialize local listviewElements to create empty list
  in
    for each item in list get myItems
    do
      if is a string? thing get i...
      set ListView1.Elements to get listviewElements
```

```
when shop ErrorOccurred
component functionName errorNumber message
do
  call Notifier1.ShowMessag...
```

```
when ListView1.AfterPicking
do
  initialize local item to select list item list get global items
  index ListView1.SelectionIndex
  in
    if is a dictionary? get item
    then
      set Image1.Picture to call goodURL
      URL get value for key "image"
      in dictionary get item
      or if not found AndroidFR.png
      refURL get global shopURL
```

GITSHARE 3b : SHOP SCREEN

```
set global items to get value for key "items" in dictionary get global shop or if not found create empty list
if is a list? thing get global items
then
  call setListViewElements myItems get global items
else
  call Notifier1.ShowMessag...
else
  call Notifier1.ShowMessag...
```

```
to goodURL URL refURL
result do if starts at text get U...
```

```
when shop ErrorOccurred
component functionName errorNumber message
do call Notifier1.ShowMessag...
```

```
when ListView1 AfterPicking
do initialize local item to select list item list get global items
index ListView1.SelectionIndex
in if is a dictionary? get item
then
  set Image1.Picture to call goodURL
  URL get value for key "image"
  in dictionary get item
  or if not found AndroidFR.png
  refURL get global shopURL
```

```
to setListViewElements myItems
do initialize local listViewElements to create empty list
in for each item in list get myItems
do if is a string? thing get item
then
  add items to list list get listViewElements
  item get item
else if is a list? thing get item
then
  add items to list list get listViewElements
  item list to csv row list get item
else if is a dictionary? get item
then
  add items to list list get listViewElements
  item join
  get value for key "name" in dictionary get item or if not found "?"
  " "
  get value for key "price" in dictionary get item or if not found 999
  " €\n "
  get value for key "ingredients" in dictionary get item or if not found create empty list
else
  call Notifier1.ShowDialog message get item title "illegal item" buttonText "OK"
  break
set ListView1.Elements to get listViewElements
```

GITSHARE 3b : SHOP SCREEN

```
set global items to get value for key "items" in dictionary get global shop or if not found create empty list
if is a list? thing get global items
then
  call setListViewElements myItems get global items
else
  call Notifier1.ShowMessag...
else
  call Notifier1.ShowMessag...
```

```
to goodURL URL refURL
result do if starts at text get U...
```

```
when shop ErrorOccurred
component functionName errorNumber message
do call Notifier1.ShowMessag...
```

```
when ListView1 AfterPicking
do initialize local item to select list item list get global items
index ListView1.SelectionIndex
in if is a dictionary? get item
then
  set Image1.Picture to call goodURL
  URL get value for key "image"
  in dictionary get item
  or if not found AndroidFR.png
  refURL get global shopURL
```

```
to setListViewElements myItems
do initialize local listviewElements to create empty list
in for each item in list get myItems
do if
then
else if
then
else if is a dictionary? get item
then
  add items to list list get listviewElements
  item join
  get value for key "name" in dictionary get item or if not found "?"
  "."
  get value for key "price" in dictionary get item or if not found 999
  "€\n"
  get value for key "ingredients" in dictionary get item or if not found create empty list
else
  call Notifier1.ShowDialog message get item title "illegal item" buttonText "OK"
break
set ListView1.Elements to get listviewElements
```

GITSHARE 3b : SHOP SCREEN

The image displays several Scratch code blocks for a shop screen application. The code is organized into several functional sections:

- Global Items Initialization:** A block that sets the global variable 'items' to the value of 'items' in a dictionary, or creates an empty list if not found. It then checks if 'items' is a list. If it is, it calls 'setListViewElements' with 'myItems' and 'global items'. Otherwise, it calls 'Notifier1.ShowMessage...'.
- setListViewElements Function:** A function that initializes a local variable 'listviewElements' to an empty list. It then iterates through each 'item' in 'myItems'. For each item, it checks if it is a dictionary. If so, it adds it to 'listviewElements' after joining the 'name' (or '?'), a separator, the 'price' (or '999'), a currency symbol (€), and the 'ingredients' (or an empty list). If not a dictionary, it calls 'Notifier1.ShowDialog' with a message, the item, and a title 'illegal item' with an 'OK' button. It then sets 'ListView1.Elements' to 'listviewElements'.
- goodURL Utility:** A block that takes a 'URL' and 'refURL' and returns the result of 'do if starts at text get U...'. It is used to fetch 'image', 'URL', and 'refURL' for an item.
- Error Handling:** A 'when shop ErrorOccurred' block that calls 'Notifier1.ShowMessage...' with 'component', 'functionName', 'errorNumber', and 'message'.
- Item Selection:** A 'when ListView1.AfterPicking' block that initializes a local 'item' from 'global items' at the selected index. It then checks if 'item' is a dictionary. If so, it sets 'Image1.Picture' to the 'image' from the dictionary, and 'URL' and 'refURL' to the corresponding values from 'global shopURL'.

GITSHARE 3b : SHOP SCREEN

The image displays a collection of code blocks for a shop screen application, organized into several functional sections:

- Global Items Initialization:** A block that sets the 'global items' to the value of 'items' in a dictionary from 'global shop', or creates an empty list if not found. It then checks if 'global items' is a list. If yes, it calls 'setListViewElements' with 'myItems' and 'global items'. If no, it calls 'Notifier1.ShowMessage...'.
- URL Validation:** A block that checks if a 'URL' starts with 'goodURL'. If not, it returns 'refURL'.
- ListView Error Handling:** A 'when shop ErrorOccurred' block that calls 'Notifier1.ShowMessage...' with 'component', 'functionName', 'errorNumber', and 'message'.
- ListView Item Picking:** A 'when ListView1 AfterPicking' block that initializes a local 'item' and selects it from 'global items' based on 'ListView1.SelectionIndex'. It then checks if 'item' is a dictionary. If yes, it sets 'Image1.Picture' to the result of a 'goodURL' call with 'item' (using 'image' key, 'item', and 'global shopURL'). If no, it sets 'Image1.Picture' to 'AndroidFR.png'.
- setListViewElements Function:** A function that initializes 'listviewElements' as an empty list. It iterates through 'myItems'. For each 'item', it checks if it's a dictionary. If yes, it joins 'name' (or '?'), a colon, 'price' (or '999'), '\n', and 'ingredients' (or an empty list) into a string and adds it to 'listviewElements'. If no, it shows a dialog with 'illegal item' and 'OK' button, then breaks. Finally, it sets 'ListView1.Elements' to 'listviewElements'.

GITSHARE 3b : SHOP SCREEN

The code is organized into several interconnected blocks:

- Global Items Initialization:** A block that sets `global items` to the value of `items` in a dictionary from `global shop`, or creates an empty list if not found. It then checks if `global items` is a list. If yes, it calls `setListViewElements` with `myItems` and `global items`. If no, it calls `Notifier1.ShowMessage...`.
- URL Handling:** A block that sets `goodURL` to the value of `URL` in a dictionary from `refURL`, or does a "do if starts at text get U..." operation.
- Error Handling:** A `when shop ErrorOccurred` block that calls `Notifier1.ShowMessage...` with `component`, `functionName`, `errorNumber`, and `message`.
- Item Selection:** A `when ListView1 AfterPicking` block that initializes a local `item` from `global items` at the selected `index`. It then checks if `item` is a dictionary. If yes, it sets `Image1.Picture` to the result of a `goodURL` call with `URL` (from `image` in `item` dictionary) and `refURL` (from `global shopURL`). If no, it calls `Notifier1.ShowDialog` with a message "item is NOT a dictionary" and an "OK" button.
- setListViewElements Function:** A function that initializes a local `listviewElements` list. It iterates through `myItems`. For each `item`, if it is a dictionary, it joins the `name` (or "?"), a colon, the `price` (or "999"), and the `ingredients` (or an empty list). If not a dictionary, it shows a dialog and breaks the loop. Finally, it sets `ListView1.Elements` to `listviewElements`.

GITSHARE 3b : SHOP SCREEN

```
set global items to get value for key "items" in dictionary get global shop or if not found create empty list
if is a list? thing get global items
then call setListViewElements myItems get global items
else call Notifier1.ShowMessag...
else call Notifier1.ShowMessag...
else call Notifier1.ShowMessag...

to goodURL URL refURL
result do if starts at text get U...

to setListViewElements myItems
do initialize local listViewEL...
```

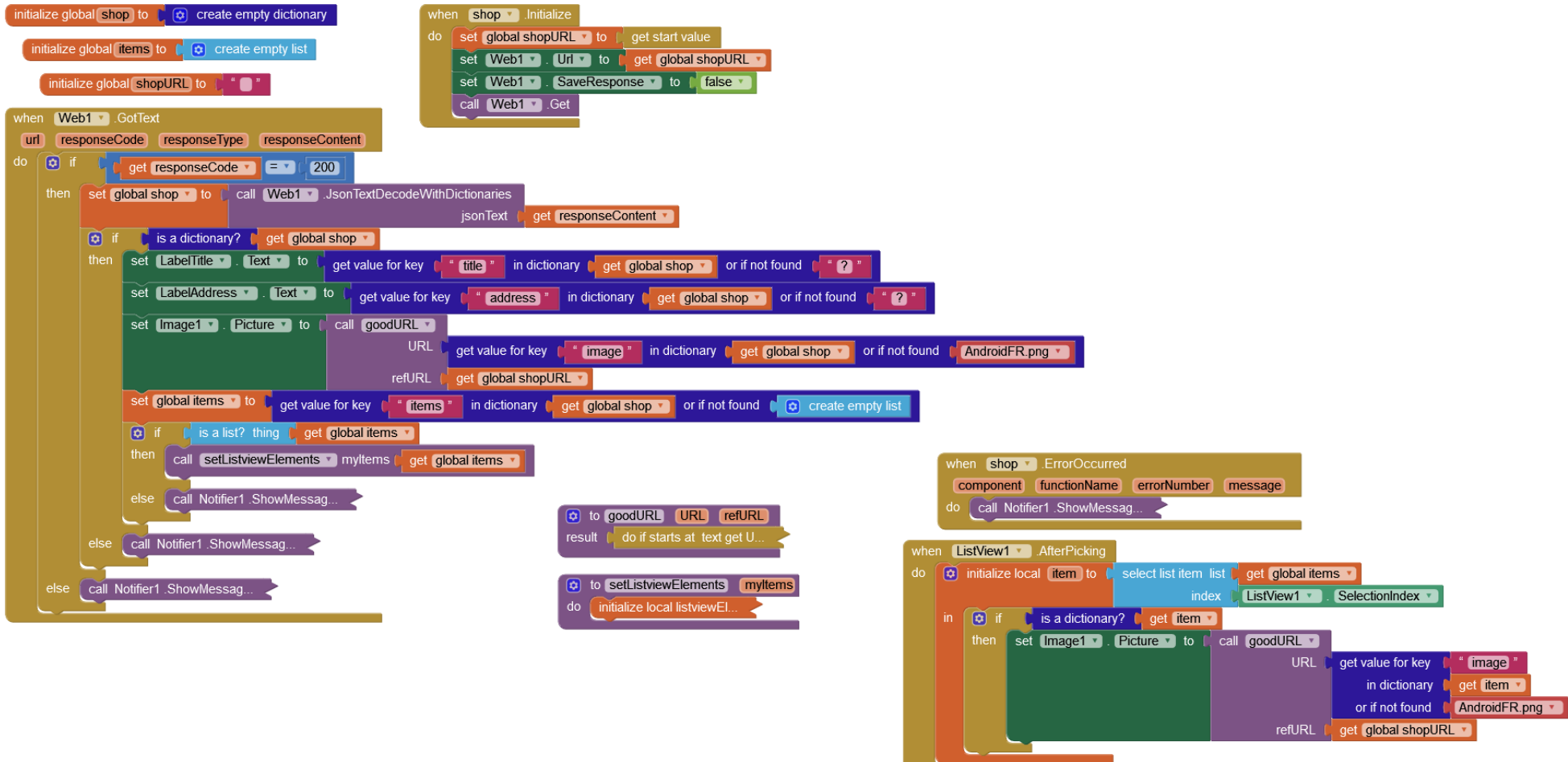
```
when shop ErrorOccurred
component functionName errorNumber message
do call Notifier1.ShowMessag...

when ListView1.AfterPicking
do initialize local item to select list item list get global items
index ListView1.SelectionIndex
in if is a dictionary? get item
then set Image1.Picture to call goodURL
URL get value for key "image"
in dictionary get item
or if not found AndroidFR.png
refURL get global shopURL
```

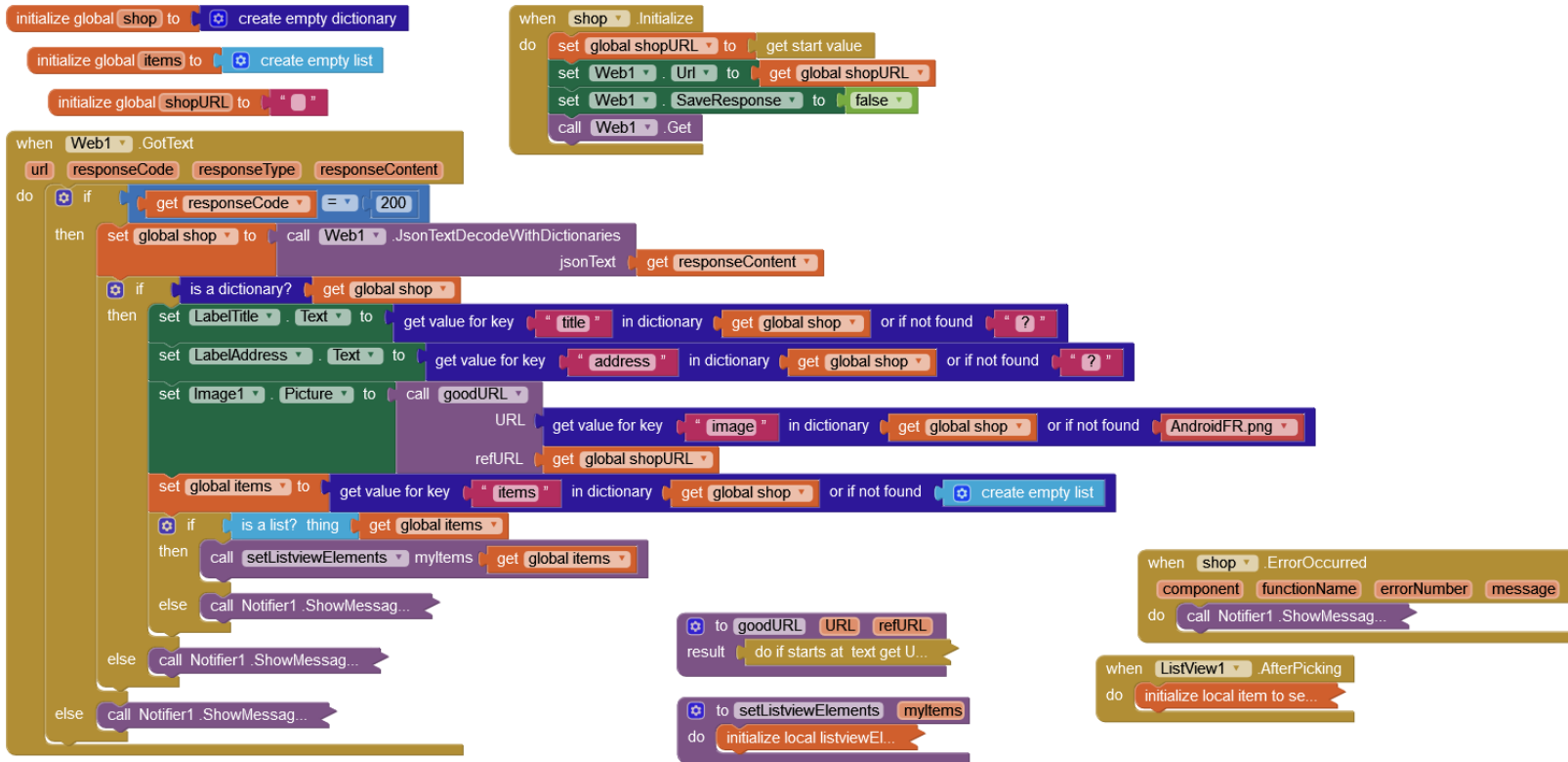
GITSHARE 3b : SHOP SCREEN



GITSHARE 3b : SHOP SCREEN



GITSHARE 3b : SHOP SCREEN



GITSHARE 3b : SHOP SCREEN

```
initialize global shop to create empty dictionary
initialize global items to create empty list
initialize global shopURL to ""

when Web1 . GotText
  url responseCode responseType responseContent
  do
    if get responseCode == 200
      then
        set global shop to call Web1 .JsonTextDecodeWithDictionaries
          jsonText get responseContent
        if is a dictionary? get global shop
          then
            set LabelTitle . Text to get value for key "title" in dictionary get global shop or if not found "?"
            set LabelAddress . Text to get value for key "address" in dictionary get global shop or if not found "?"
            set Image1 . Picture to call goodURL
              URL get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
              refURL get global shopURL
            set global items to get value for key "items" in dictionary get global shop or if not found create empty list
            if is a list? thing get global items
              then
                call setListViewElements myItems get global items
              else
                call Notifier1 .ShowMessag...
            else
              call Notifier1 .ShowMessag...
          else
            call Notifier1 .ShowMessag...

when shop . Initialize
  do
    set global shopURL to get start value
    set Web1 . Url to get global shopURL
    set Web1 . SaveResponse to false
    call Web1 .Get

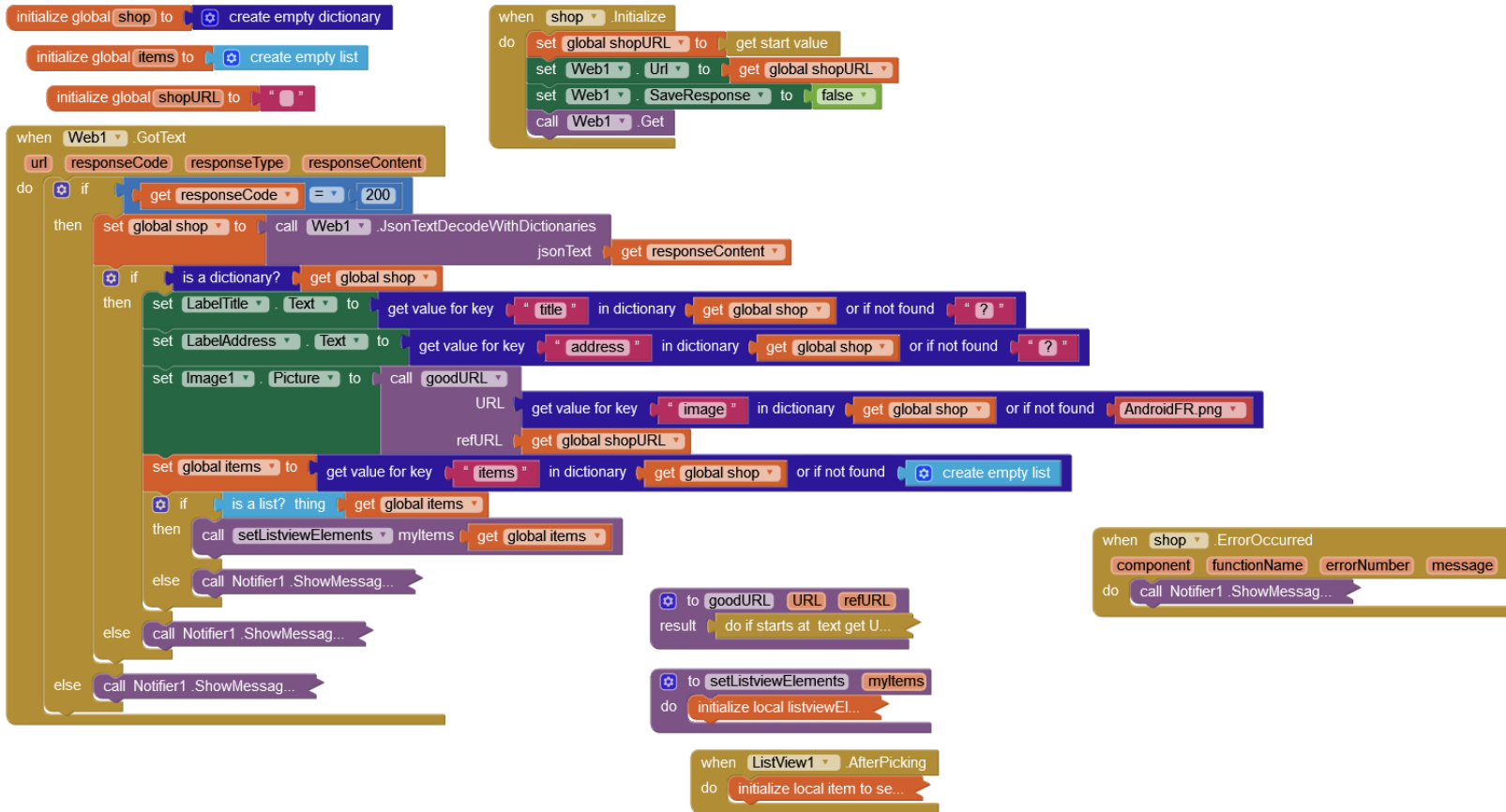
when shop . ErrorOccurred
  component functionName errorNumber message
  do
    call Notifier1 .ShowMessag...

to goodURL URL refURL
  result do if starts at text get U...

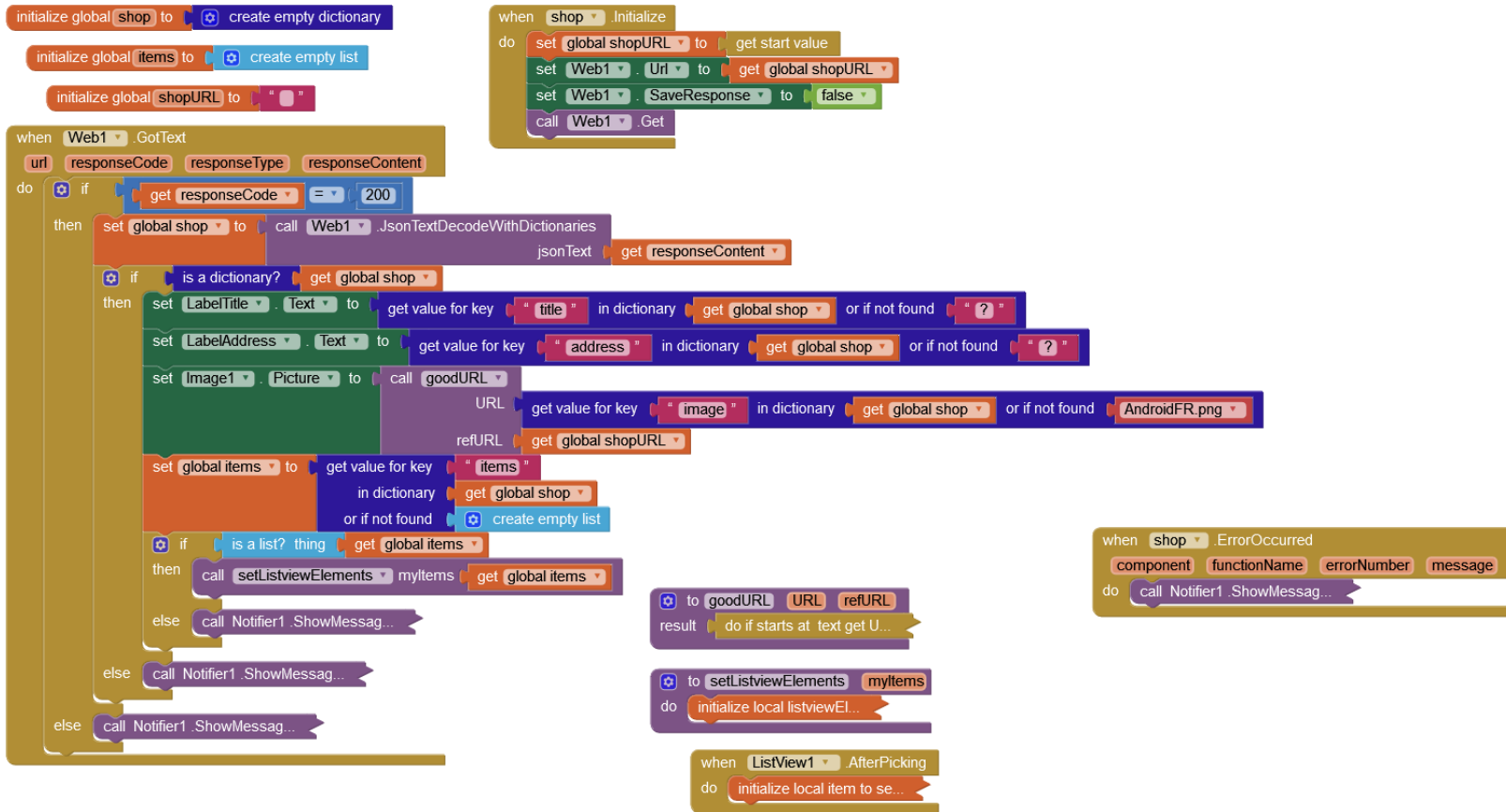
to setListViewElements myItems
  do
    initialize local listViewEl...

when ListView1 . AfterPicking
  do
    initialize local item to se...
```

GITSHARE 3b : SHOP SCREEN



GITSHARE 3b : SHOP SCREEN



GITSHARE 3b : SHOP SCREEN

```
initialize global shop to create empty dictionary
initialize global items to create empty list
initialize global shopURL to ""

when Web1 GotText
  url responseCode responseType responseContent
  do
    if responseCode == 200
      then
        set global shop to call Web1 .JsonTextDecodeWithDictionaries
          jsonText get responseContent
        if is a dictionary? get global shop
          then
            set LabelTitle . Text to get value for key "title" in dictionary get global shop or if not found "?"
            set LabelAddress . Text to get value for key "address" in dictionary get global shop or if not found "?"
            set Image1 . Picture to call goodURL
              URL get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
              refURL get global shopURL
            set global items to get value for key "items" in dictionary get global shop or if not found create empty list
            if is a list? thing get global items
              then
                call setListViewElements myItems get global items
            else
                call Notifier1 .ShowMessag...
          else
            call Notifier1 .ShowMessag...
        else
            call Notifier1 .ShowMessag...

when shop .Initialize
  do
    set global shopURL to get start value
    set Web1 . Url to get global shopURL
    set Web1 . SaveResponse to false
    call Web1 .Get

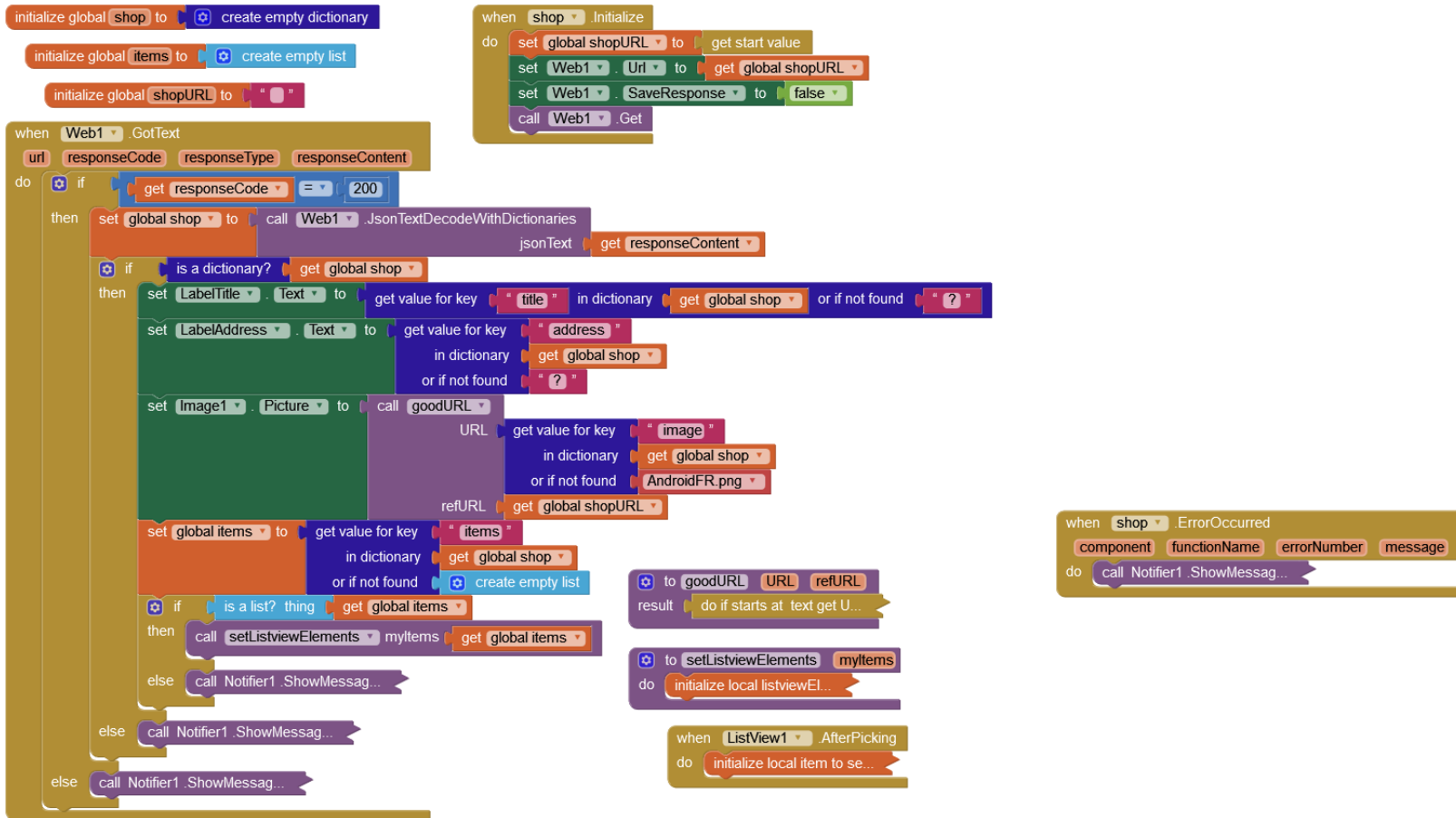
when shop .ErrorOccurred
  component functionName errorNumber message
  do
    call Notifier1 .ShowMessag...

to goodURL URL refURL
  result do if starts at text get U...

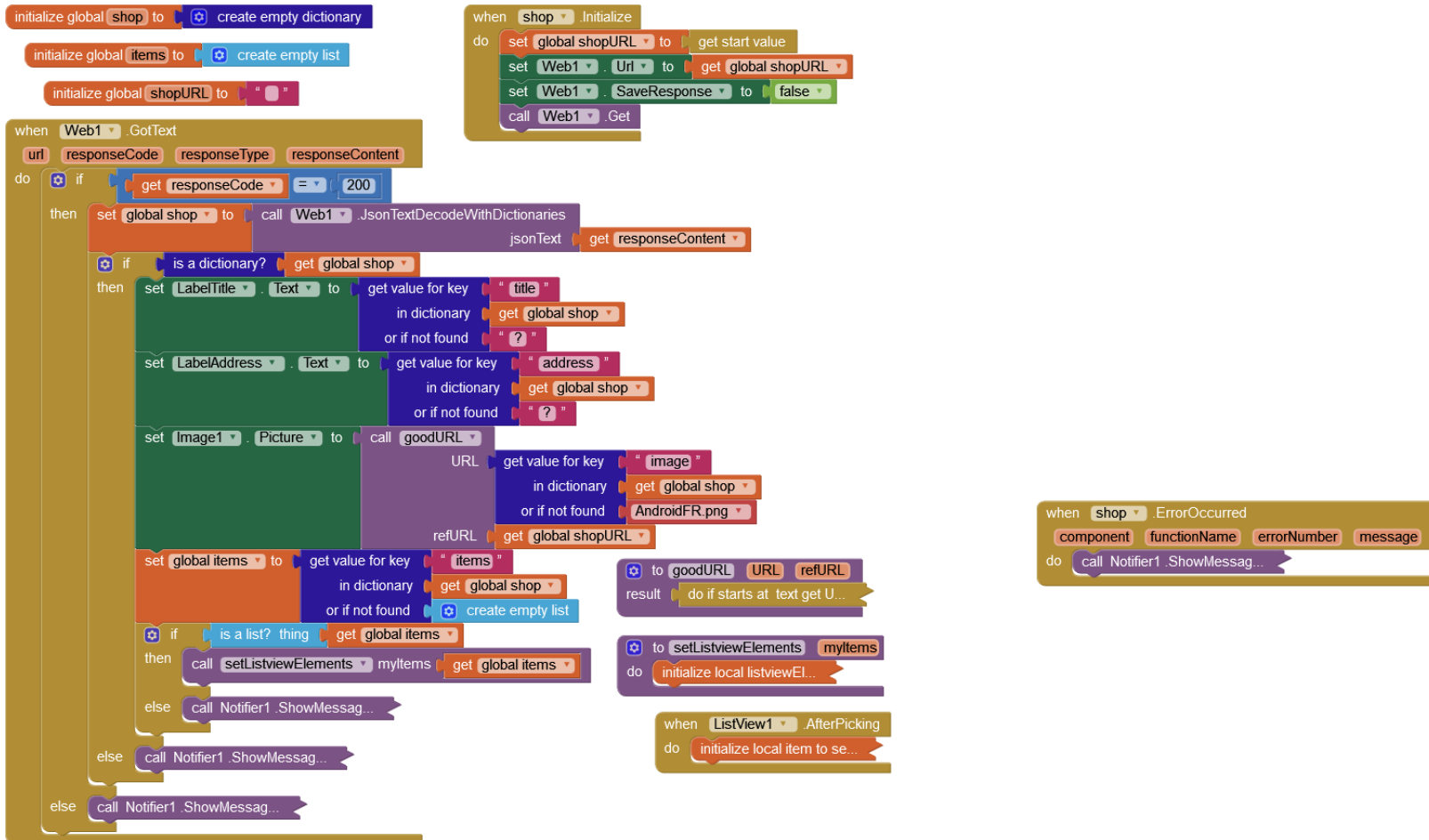
to setListViewElements myItems
  do
    initialize local listviewEl...

when ListView1 AfterPicking
  do
    initialize local item to se...
```

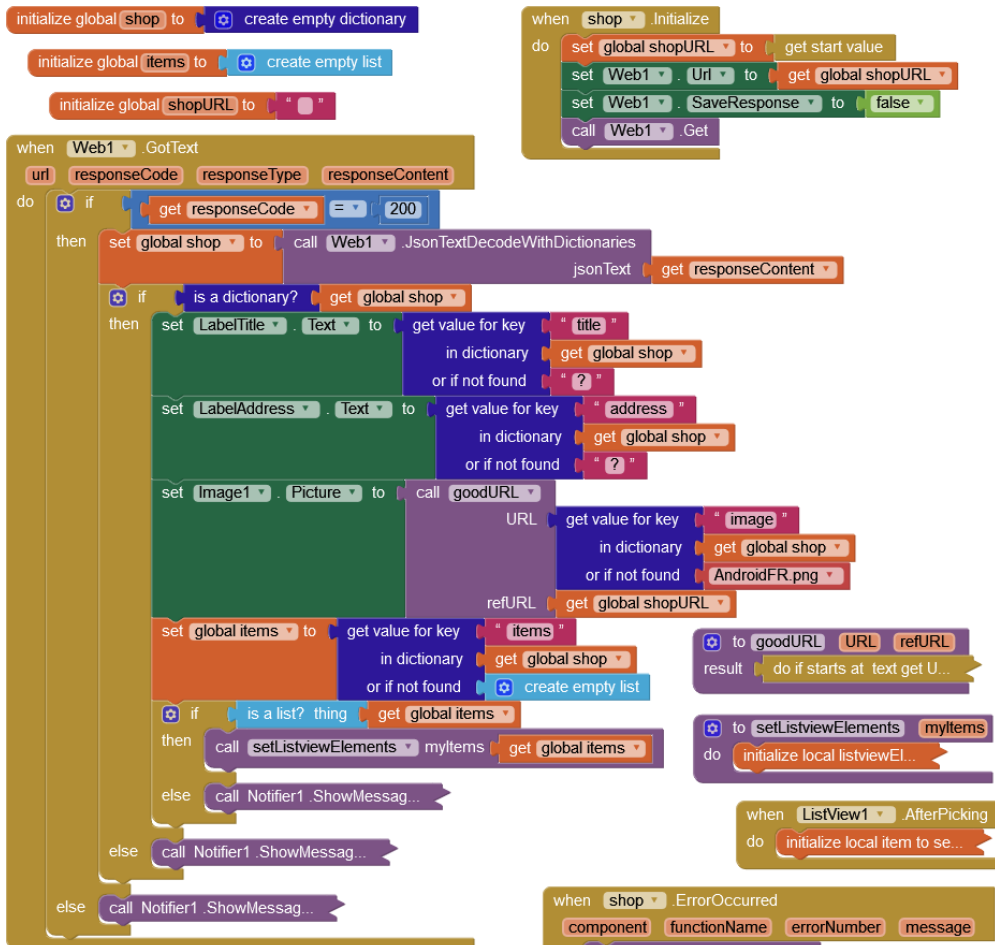
GITSHARE 3b : SHOP SCREEN



GITSHARE 3b : SHOP SCREEN



GITSHARE 3b : SHOP SCREEN



GITSHARE 3b : SHOP SCREEN

Display & select algorithm

```
initialize global shop to create empty dictionary
initialize global items to create empty list
initialize global shopURL to ""

when Web1 GotText
  url responseCode responseType responseContent
  do
    if get responseCode == 200
      then
        set global shop to call Web1 .JsonTextDecodeWithDictionaries
          jsonText get responseContent
        if is a dictionary? get global shop
          then
            set LabelTitle .Text to get value for key "title"
              in dictionary get global shop
              or if not found "?"
            set LabelAddress .Text to get value for key "address"
              in dictionary get global shop
              or if not found "?"
            set Image1 .Picture to call goodURL
              URL get value for key "image"
                in dictionary get global shop
                or if not found AndroidFR.png
              refURL get global shopURL
            set global items to get value for key "items"
              in dictionary get global shop
              or if not found create empty list
            if is a list? thing get global items
              then
                call setListViewElements myItems get global items
              else
                call Notifier1 .ShowMessag...
            else
                call Notifier1 .ShowMessag...
          else
            call Notifier1 .ShowMessag...
        else
            call Notifier1 .ShowMessag...

when shop Initialize
  do
    set global shopURL to get start value
    set Web1 .Url to get global shopURL
    set Web1 .SaveResponse to false
    call Web1 .Get

to goodURL URL refURL
  result do if starts at text get U...

to setListViewElements myItems
  do initialize local listviewEl...

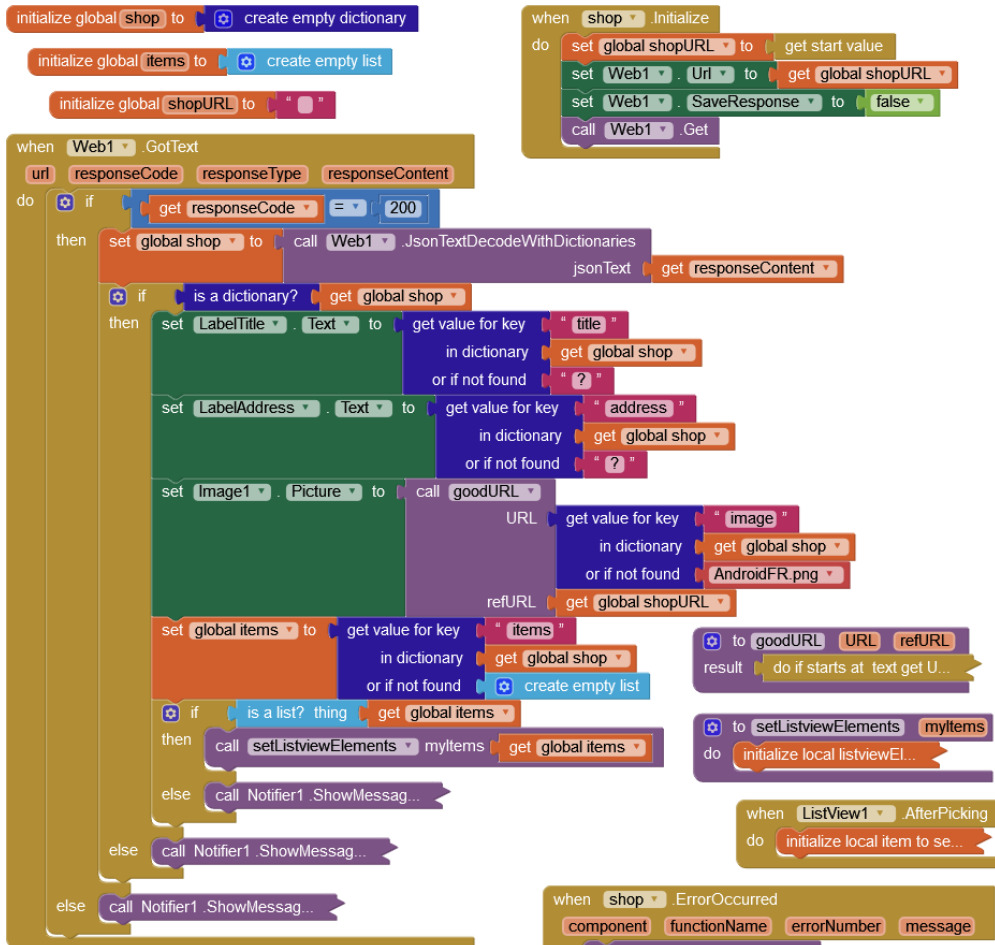
when ListView1 AfterPicking
  do initialize local item to se...

when shop ErrorOccurred
  component functionName errorNumber message
```

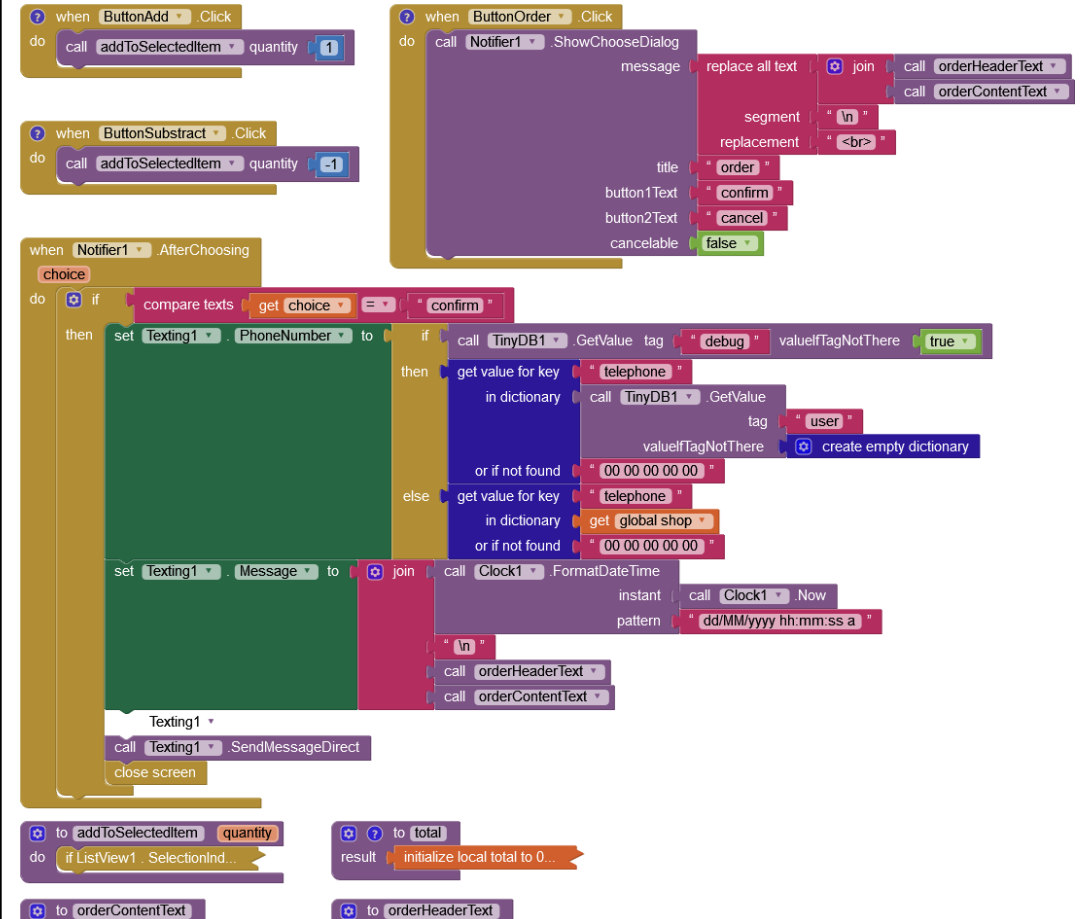
Order algorithm (3b)

GITSHARE 3b : SHOP SCREEN

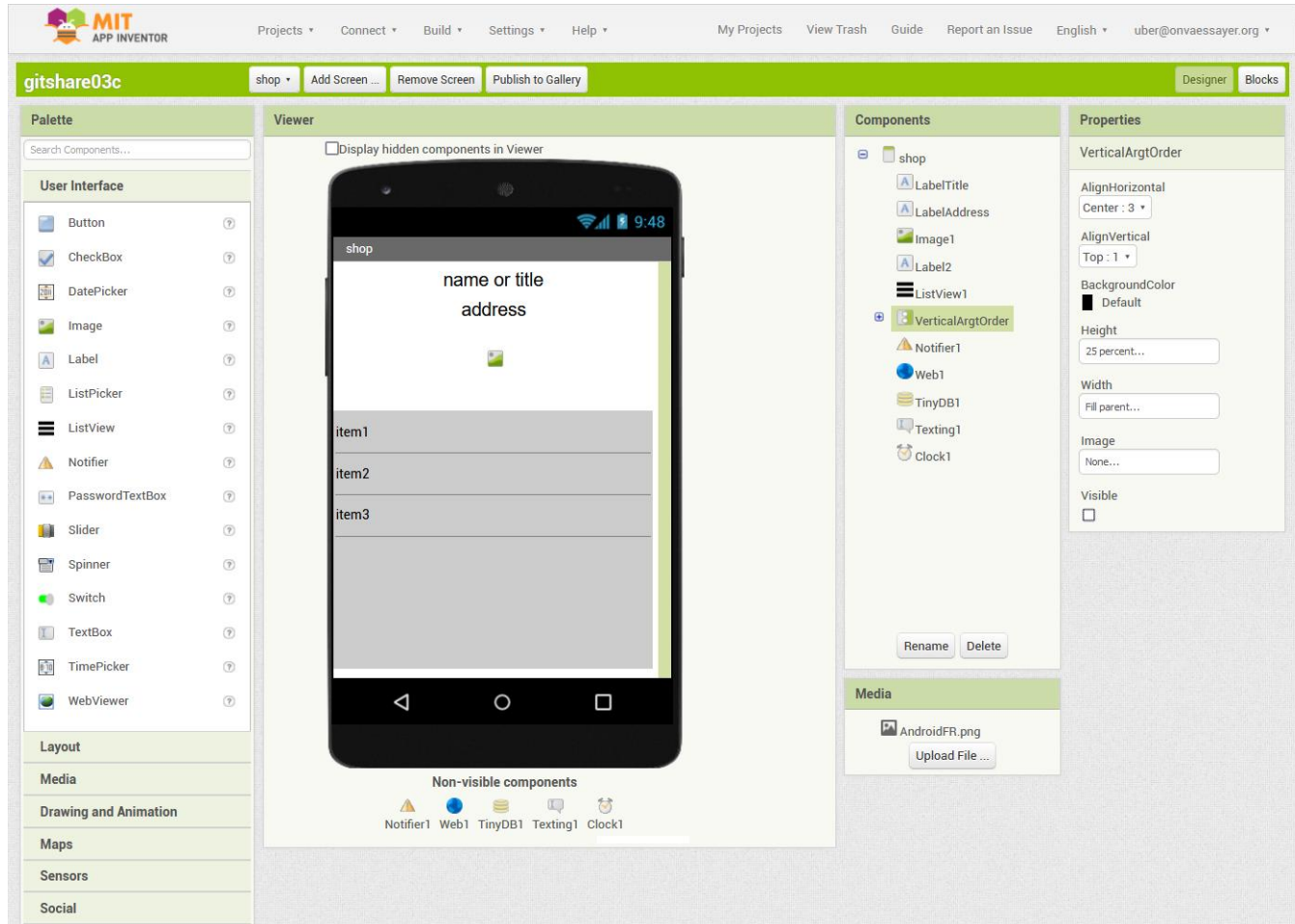
Display & select algorithm



Order algorithm (3b)



GITSHARE3b : DESIGN - SHOP SCREEN



shop

Le Roma

61 avenue du Maine, 75014 Paris



pizza Margherita : 11.4 €
["Mozarella", "champignons", "tomate"]

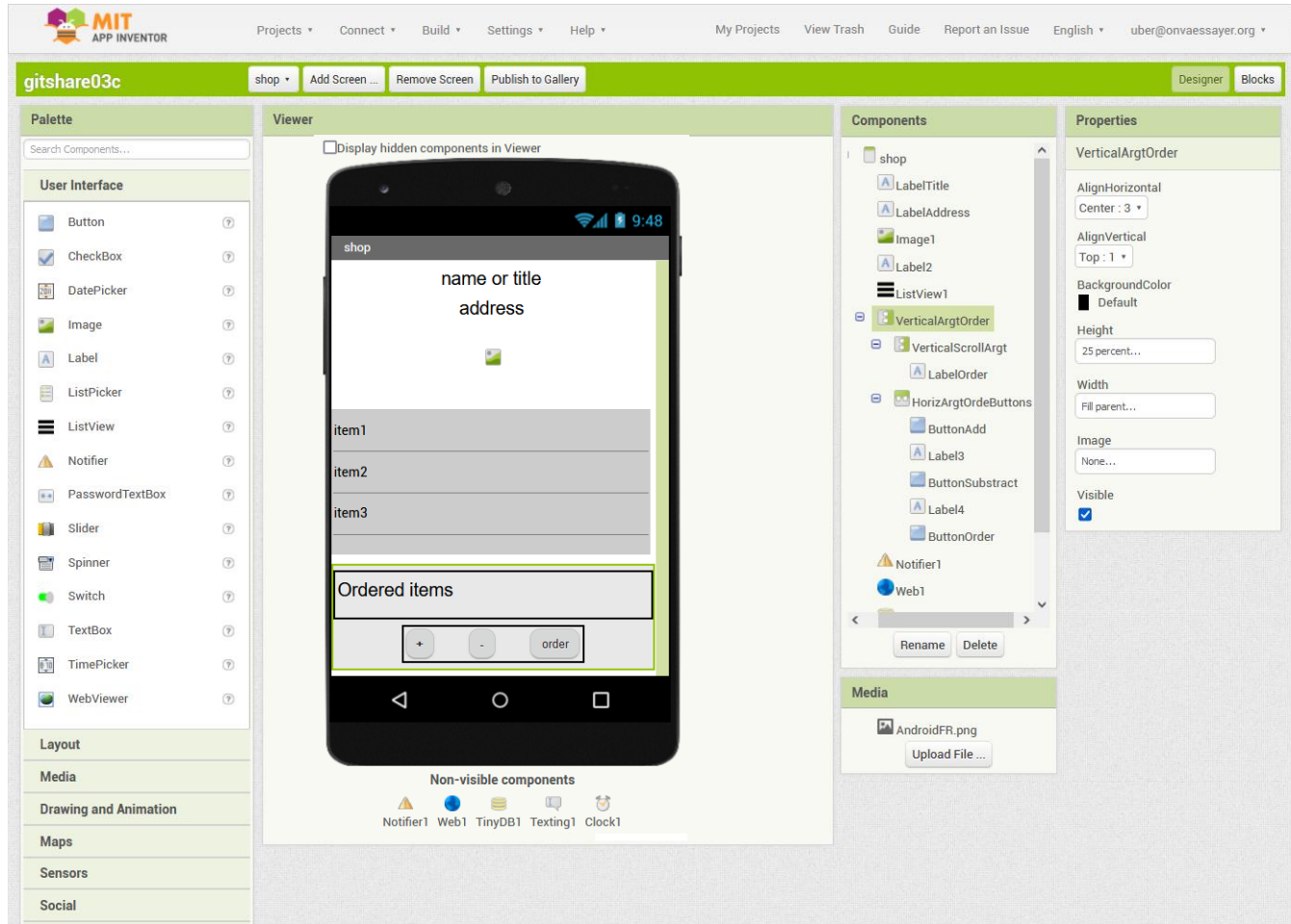
tagliatelles carbonara : 12.9 €
["farine", "viande hachée", "tomate"]

Chianti : 17.5 €
["raisin", "alcool", "sulfites"]

34.2 € 3 x pizza Margherita
17.5 € 1 x Chianti
51.7 € total

+ - order

GITSHARE3b : DESIGN - SHOP SCREEN



shop

Le Roma

61 avenue du Maine, 75014 Paris



pizza Margherita : 11.4 €
["Mozarella", "champignons", "tomate"]

tagliatelles carbonara : 12.9 €
["farine", "viande hachée", "tomate"]

Chianti : 17.5 €
["raisin", "alcool", "sulfites"]

34.2 € 3 x pizza Margherita
17.5 € 1 x Chianti
51.7 € total

+ - order

GITSHARE3b : ALGORITHM

Button add :

- ajouter 1 à la commande

Button substract :

- retirer 1 à la commande

Button Order :

- afficher header et contenu
- demander confirmation

GITSHARE3b : ALGORITHM

Button add :

- ajouter 1 à la commande

Button substract :

- retirer 1 à la commande

Button Order :

- afficher header et contenu
- demander confirmation

add to selected item (quantity)

- ajouter quantity à la commande
- mettre à jour et afficher le contenu

GITSHARE3b : ALGORITHM / DATA

```
    item
{
  "title": "Le Roma",
  "description": "plats italiens",
  "address": "avenue du Maine, Paris",
  "image": "image.jpg",
  "telephone": "0600000004",
  "email": "ristoranteRoma@paris.fr",
  "items": [
    {
      "name": "pizza Margherita",
      "price": 11.40,
      "ingredients": ["Mozarella", "tomate"],
      "image": "margherita.PNG"
    },
    {
      "name": "tagliatelles carbonara",
      "price": 12.90,
      "ingredients": ["farine", "boeuf", "tomate"],
      "image": "carbonara.PNG"
    },
    {
      "name": "Chianti",
      "price": 17.5,
      "ingredients": ["raisin", "alcool"],
      "image": "chianti.jpg"
    }
  ]
}
```

GITSHARE3b : ALGORITHM / DATA

```
    item
{
  "title": "Le Roma",
  "description": "plats italiens",
  "address": "avenue du Maine, Paris",
  "image": "image.jpg",
  "telephone": "0600000004",
  "email": "ristoranteRoma@paris.fr",
  "items": [
    {
      "quantity": 3,
      "name": "pizza Margherita",
      "price": 11.40,
      "ingredients": ["Mozarella", "tomate"],
      "image": "margherita.PNG"
    },
    {
      "name": "tagliatelles carbonara",
      "price": 12.90,
      "ingredients": ["farine", "boeuf", "tomate"],
      "image": "carbonara.PNG"
    },
    {
      "quantity": 1,
      "name": "Chianti",
      "price": 17.5,
      "ingredients": ["raisin", "alcool"],
      "image": "chianti.jpg"
    }
  ]
}
```

GITSHARE3b : ALGORITHM / DATA

```
item
{
  "title": "Le Roma",
  "description": "plats italiens",
  "address": "avenue du Maine, Paris",
  "image": "image.jpg",
  "telephone": "0600000004",
  "email": "ristoranteRoma@paris.fr",
  "items": [
    {
      "quantity": 3,
      "name": "pizza Margherita",
      "price": 11.40,
      "ingredients": ["Mozarella", "tomate"],
      "image": "margherita.PNG"
    },
    {
      "name": "tagliatelles carbonara",
      "price": 12.90,
      "ingredients": ["farine", "boeuf", "tomate"],
      "image": "carbonara.PNG"
    },
    {
      "quantity": 1,
      "name": "Chianti",
      "price": 17.5,
      "ingredients": ["raisin", "alcool"],
      "image": "chianti.jpg"
    }
  ]
}
```

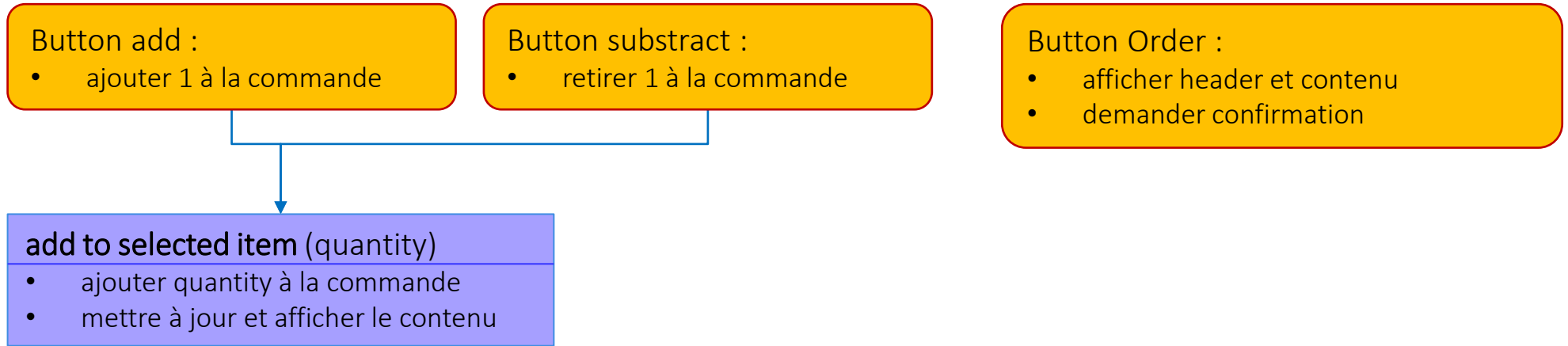
read



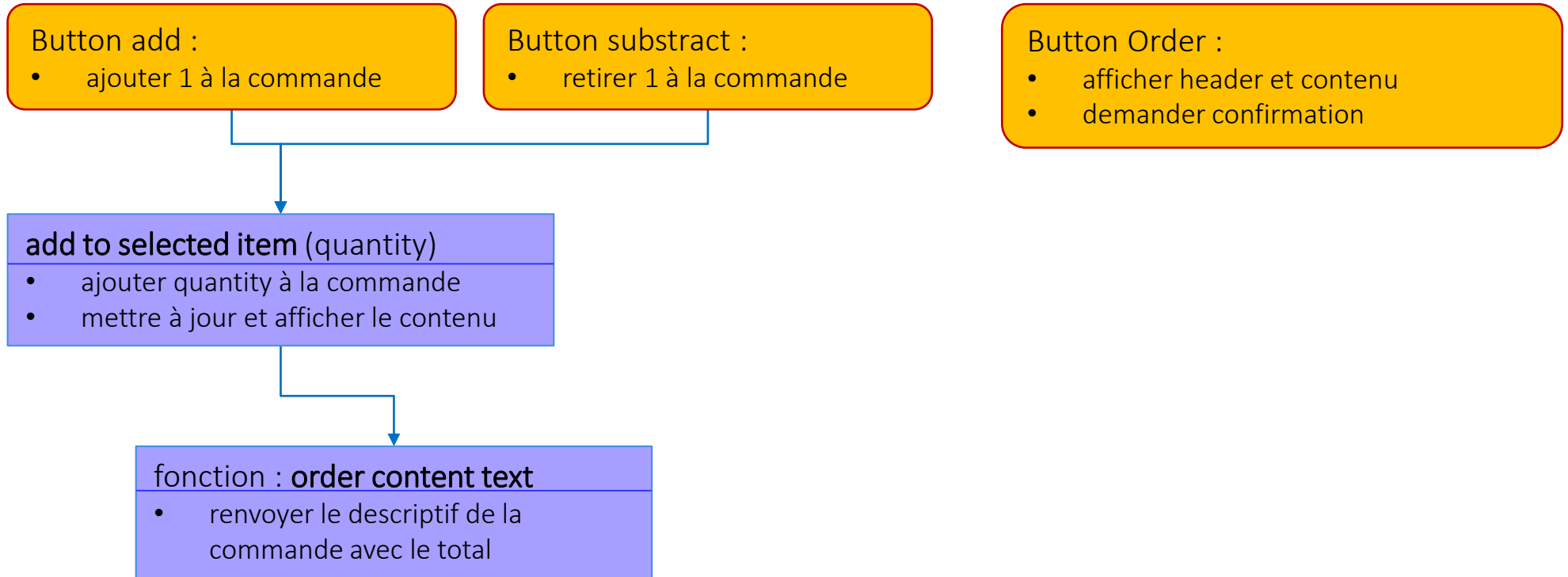
write



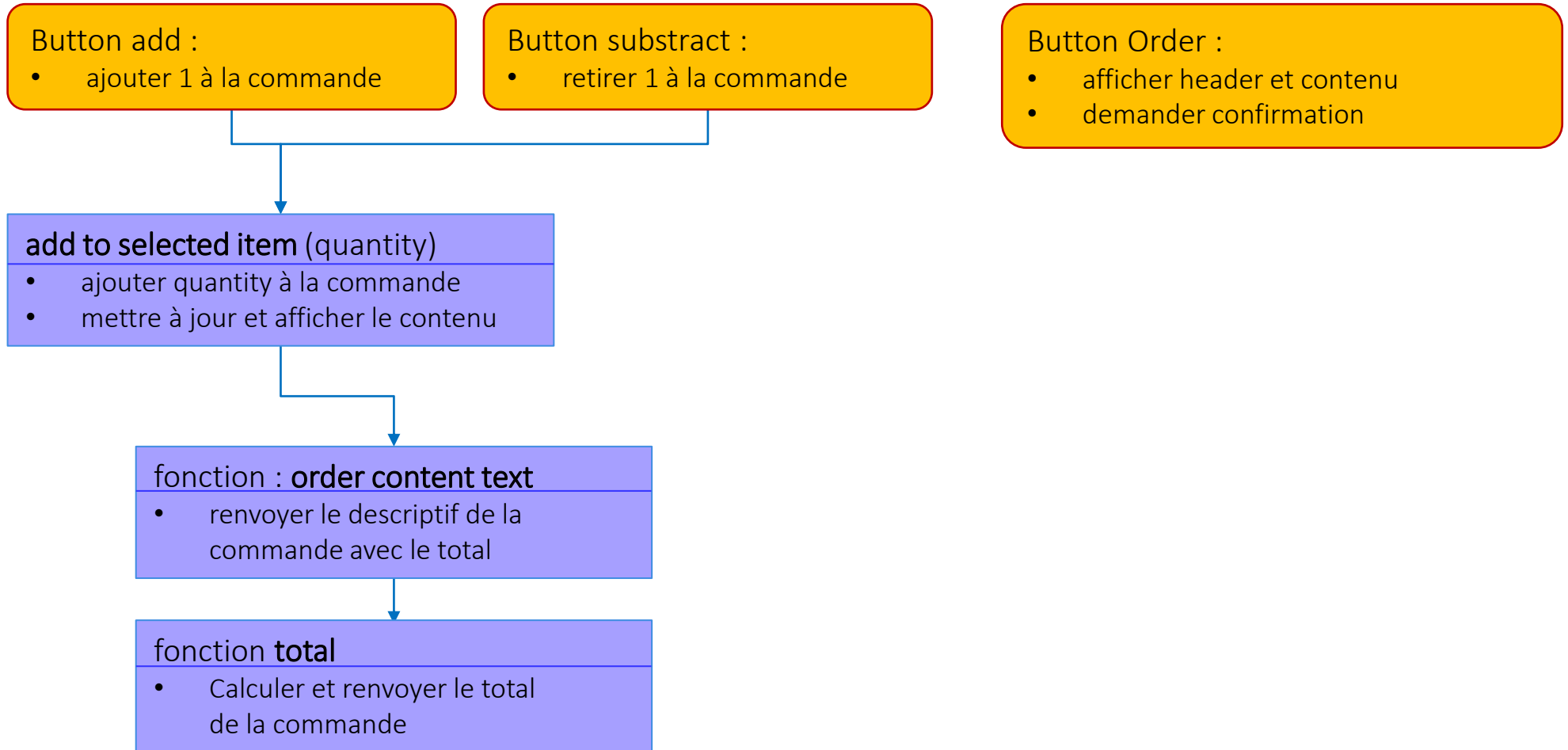
GITSHARE3b : ALGORITHM



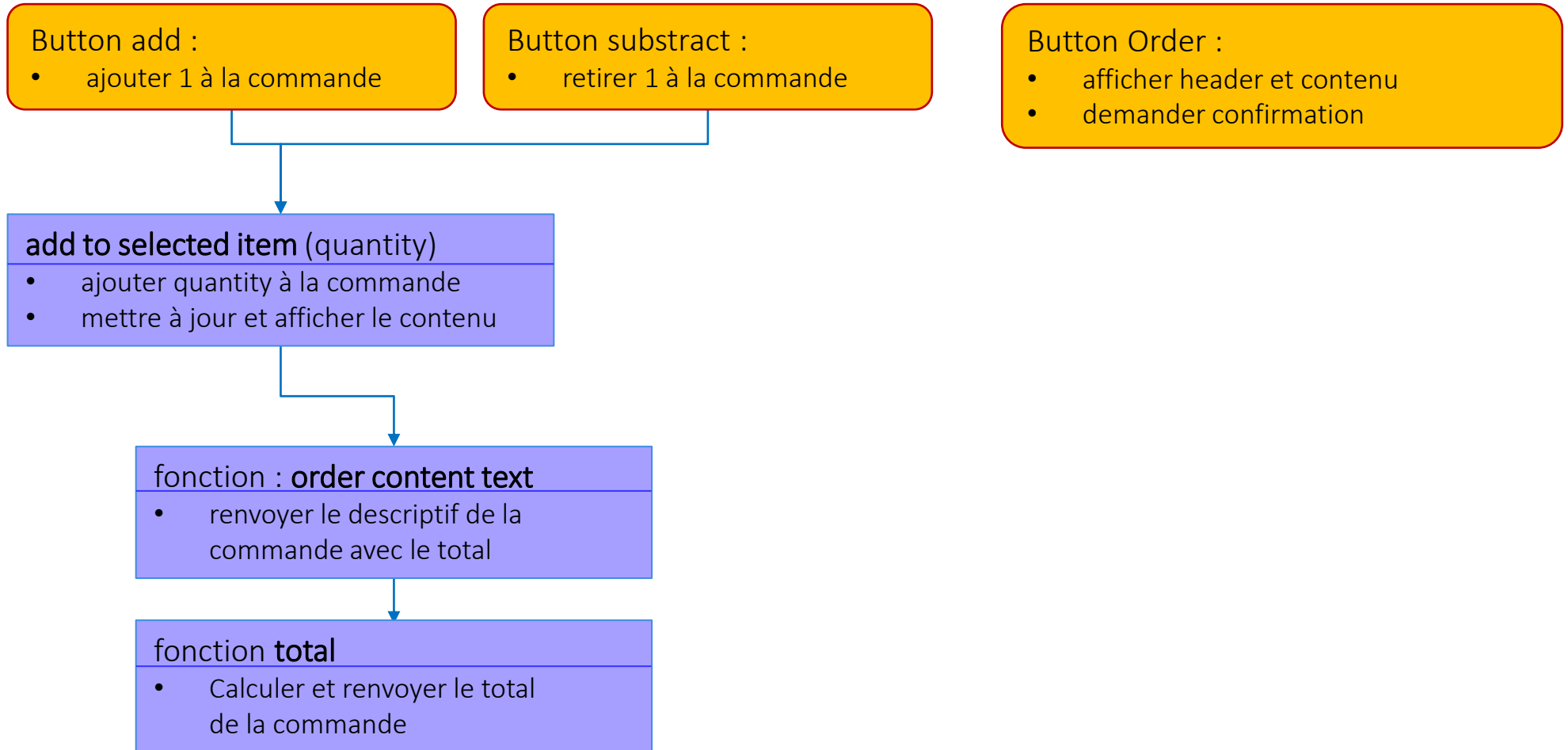
GITSHARE3b : ALGORITHM



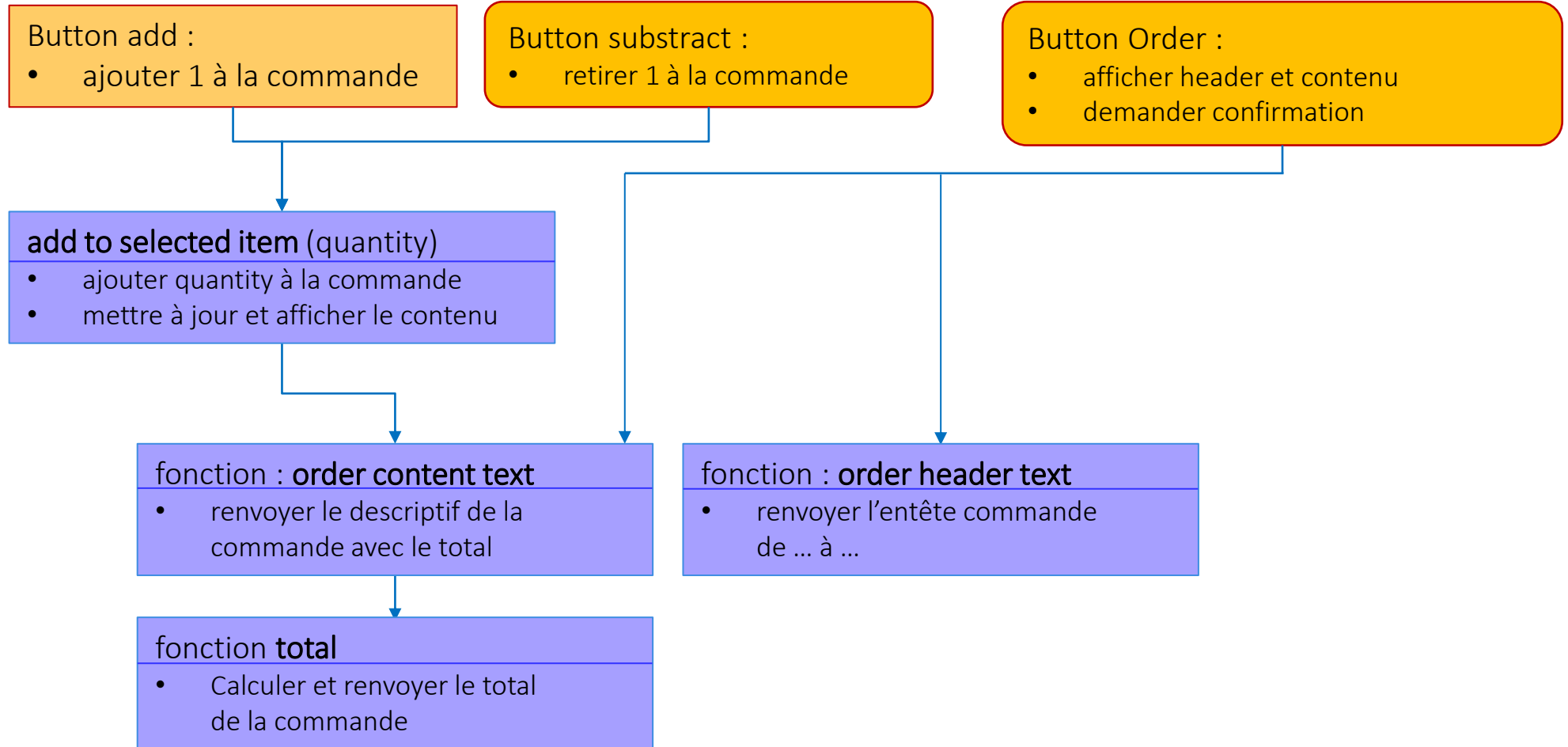
GITSHARE3b : ALGORITHM



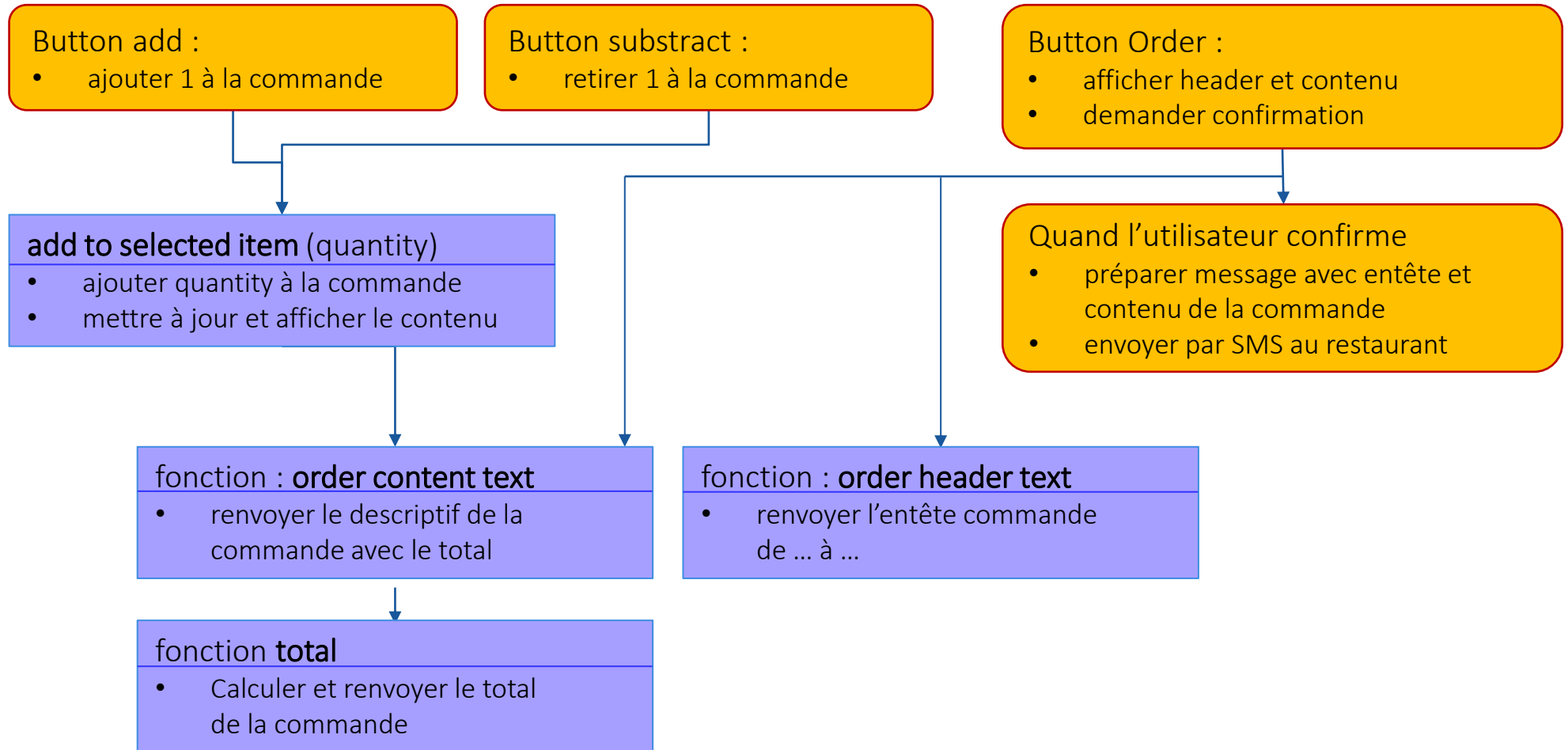
GITSHARE3b : ALGORITHM



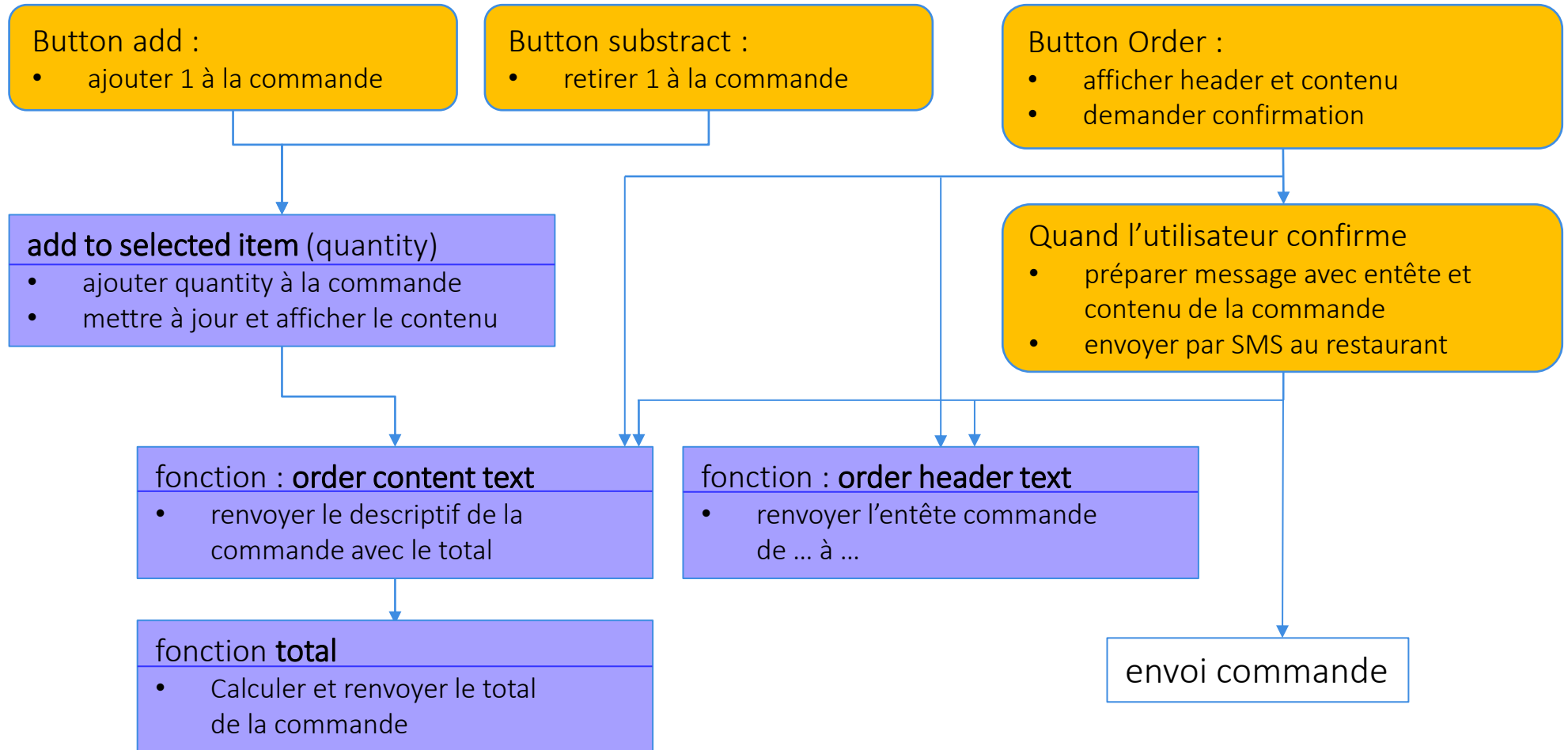
GITSHARE3b : ALGORITHM



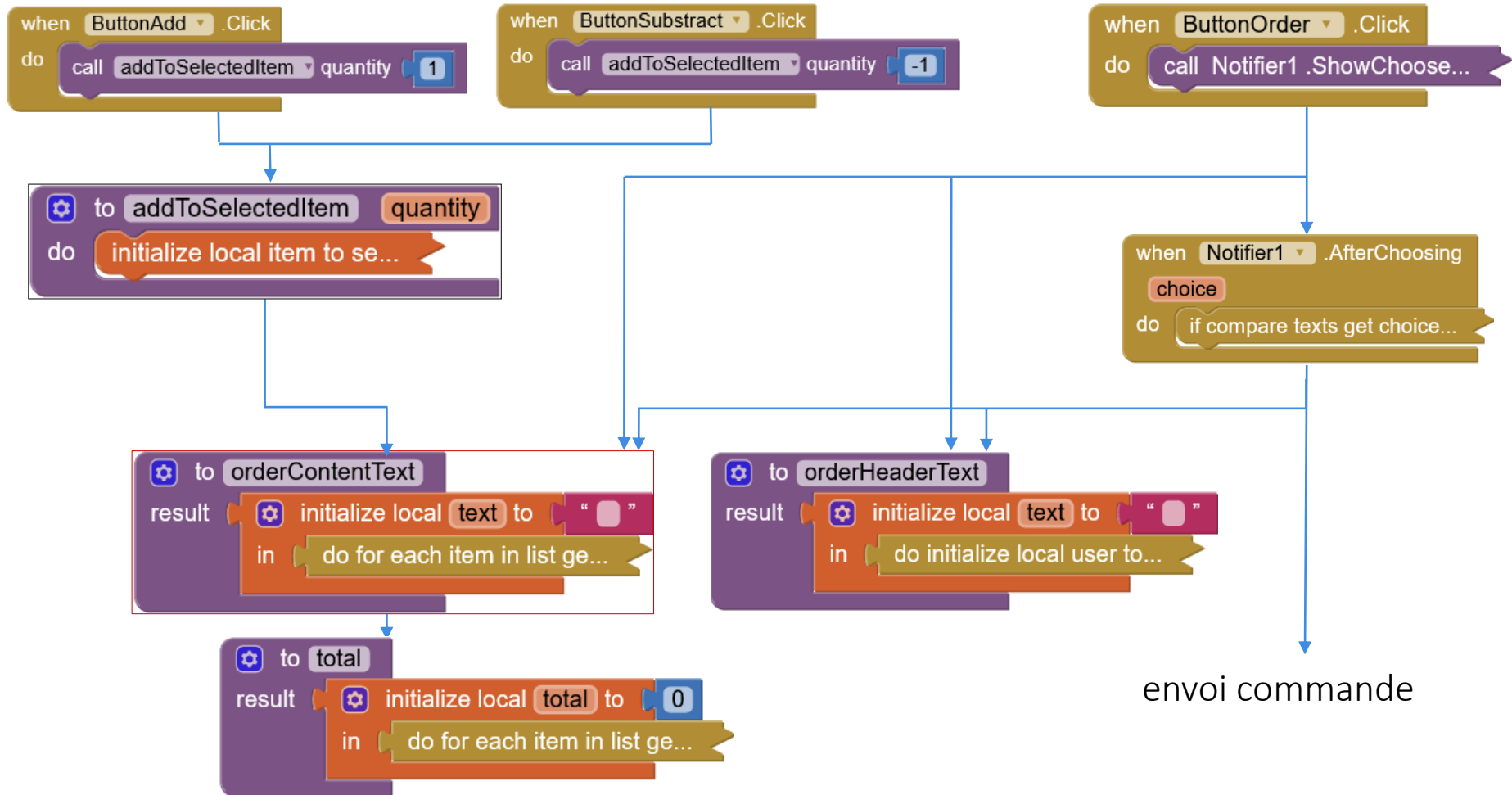
GITSHARE3b : ALGORITHM



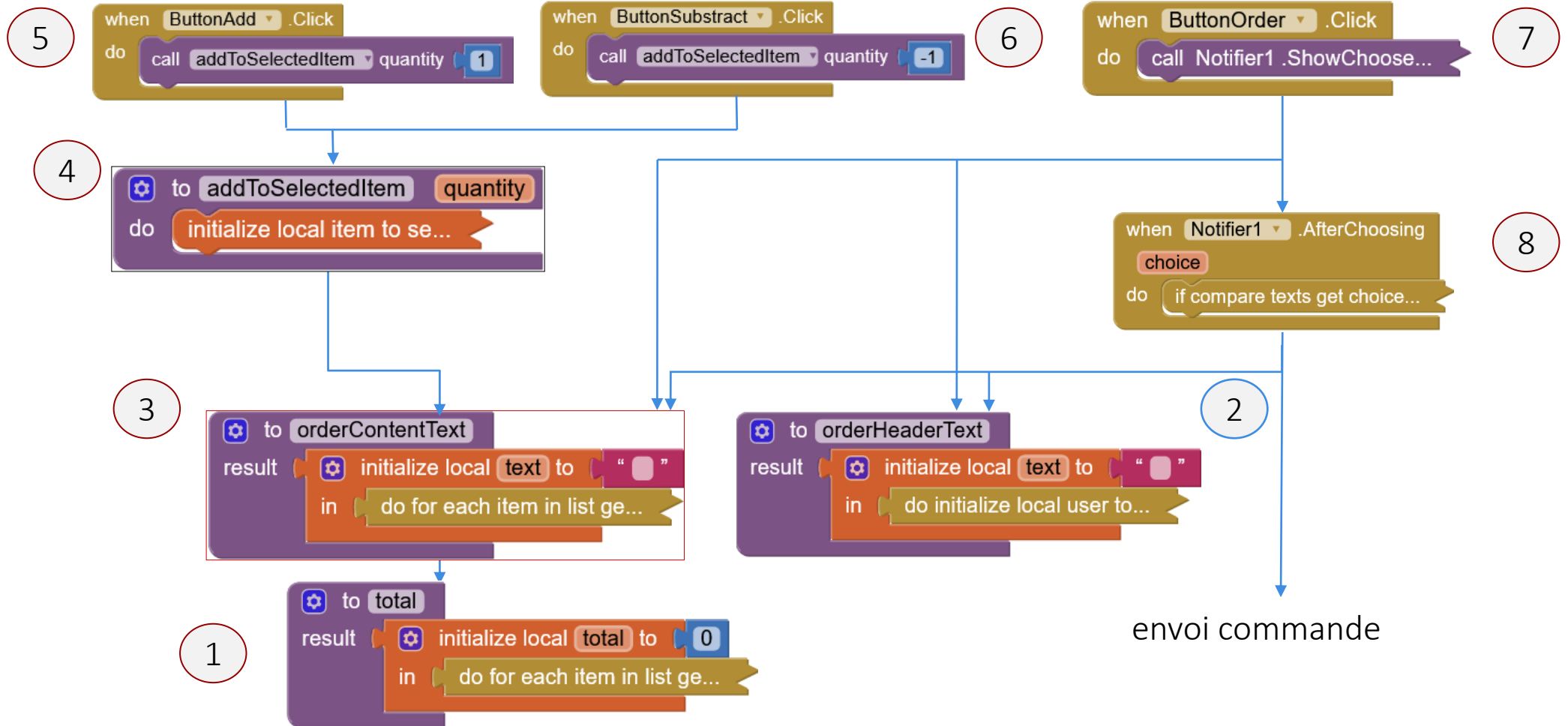
GITSHARE3b : ALGORITHM



GITSHARE3b : PRÉPARER ET PASSER UNE COMMANDE



GITSHARE3b : PRÉPARER ET PASSER UNE COMMANDE



GITSHARE3b : SHOP SCREEN BLOCKS

```
initialize global shop to create empty dictionary
initialize global items to create empty list
initialize global shopURL to ""

when shop Initialize
do
  set global shopURL to get start value
  set Web1.Uri to get global shopURL
  set Web1.SaveResponse to false
  call Web1.Get

when Web1.GetText
url responseCode responseType responseContent
do
  if get responseCode = 200
  then
    set global shop to call Web1.JsonTextDecodeWithDictionaries jsontext get responseContent
    if is a dictionary? get global shop
    then
      set LabelTitle.Text to get value for key "title" in dictionary get global shop or if not found "?"
      set LabelAddress.Text to get value for key "address" in dictionary get global shop or if not found "?"
      set Image1.Picture to call goodURL
      URL get value for key "image"
      in dictionary get global shop
      or if not found "AndroidFR.png"
      rootURL get url
      set global items to get value for key "items" in dictionary get global shop or if not found create empty list
      if is a list? thing get global items
      then
        call setListViewElements myItems get global items
      else
        call Notifier1.ShowMessage...
    else
      call Notifier1.ShowMessage...
  else
    call Notifier1.ShowMessage...

to goodURL URL rootURL
result do if starts at text get U...
```

```
to setListViewElements myItems
do
  initialize local listViewElements to create empty list
  in
  foreach item in list get myItems
  do
    if is a dictionary? get item
    then
      add items to list list get listViewElements item join
      get value for key "name"
      in dictionary get item
      or if not found "?"
      " "
      get value for key "price"
      in dictionary get item
      or if not found "9999"
      "€ln"
      get value for key "ingredients"
      in dictionary get item
      or if not found create empty list
    else
      call Notifier1.ShowDialog
      message get item
      title item is NOT a dictionary
      buttonText OK
      break
  set ListView1.Elements to get listViewElements

when ListView1.AfterPicking
do
  initialize local item to select list item list get global items
  index ListView1.SelectionIndex
  in
  set Image1.Picture to call goodURL
  URL get value for key "image"
  in dictionary get item
  or if not found ""
  rootURL get global shopURL
```

GITSHARE3b : BLOCKS

```
initialize global shop to create empty dictionary
initialize global items to create empty list
initialize global shopURL to ""

when shop Initialize
do
  set global shopURL to get start value
  set Web1.Uri to get global shopURL
  set Web1.SaveResponse to false
  call Web1.Get

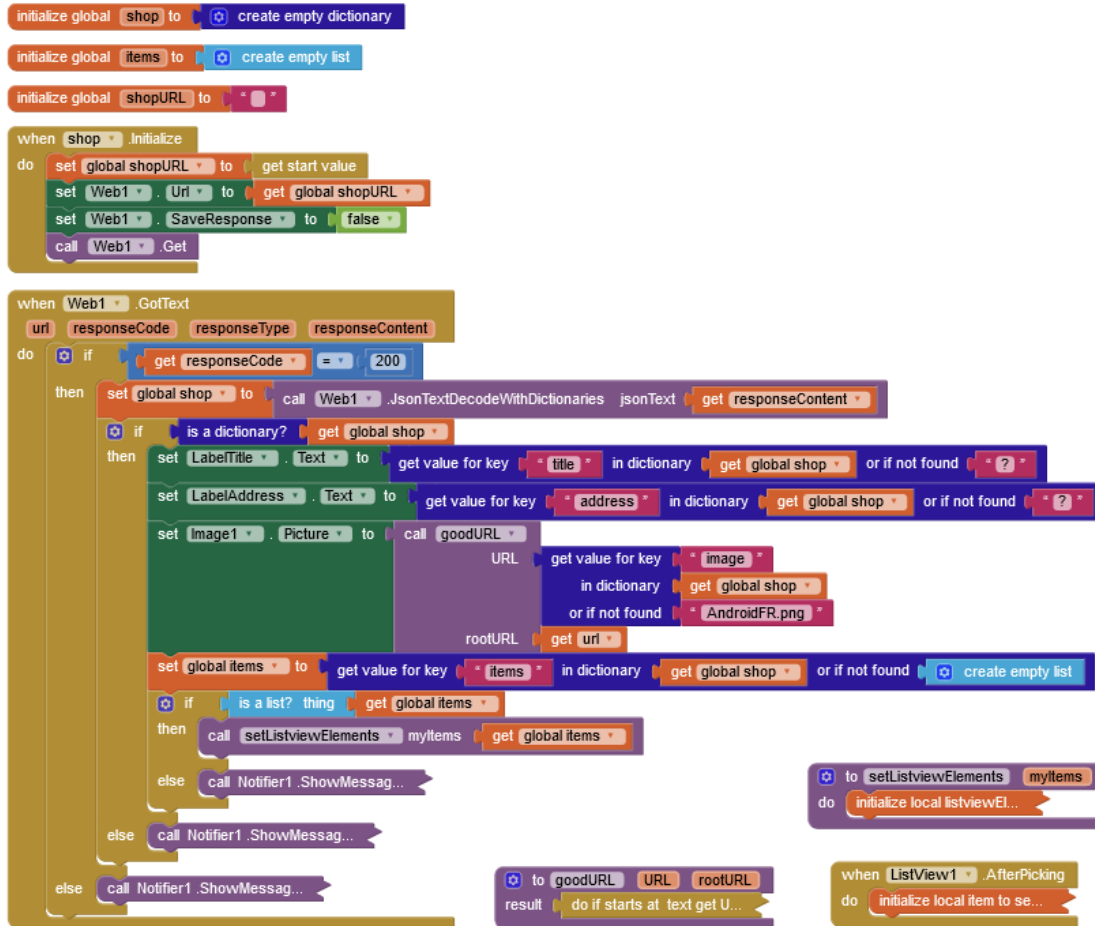
when Web1.GotText
url responseCode responseType responseContent
do
  if get responseCode = 200
  then
    set global shop to call Web1.JsonTextDecodeWithDictionaries jsonText get responseContent
    if is a dictionary? get global shop
    then
      set LabelTitle.Text to get value for key "title" in dictionary get global shop or if not found "?"
      set LabelAddress.Text to get value for key "address" in dictionary get global shop or if not found "?"
      set Image1.Picture to call goodURL
      URL get value for key "image"
      in dictionary get global shop
      or if not found "AndroidFR.png"
      rootURL get url
      set global items to get value for key "items" in dictionary get global shop or if not found create empty list
      if is a list? thing get global items
      then
        call setListViewElements myItems get global items
      else
        call Notifier1.ShowMessage...
    else
      call Notifier1.ShowMessage...
  else
    call Notifier1.ShowMessage...

to goodURL URL rootURL
result do if starts at text get U...
```

```
to setListViewElements myItems
do
  initialize local listViewElements to create empty list
  in
  foreach item in list get myItems
  do
    if is a dictionary? get item
    then
      add items to list list get listViewElements item join
      get value for key "name"
      in dictionary get item
      or if not found "?"
      " "
      get value for key "price"
      in dictionary get item
      or if not found "9999"
      "€ln"
      get value for key "ingredients"
      in dictionary get item
      or if not found create empty list
    else
      call Notifier1.ShowDialog
      message get item
      title item is NOT a dictionary
      buttonText OK
      break
  set ListView1.Elements to get listViewElements
```

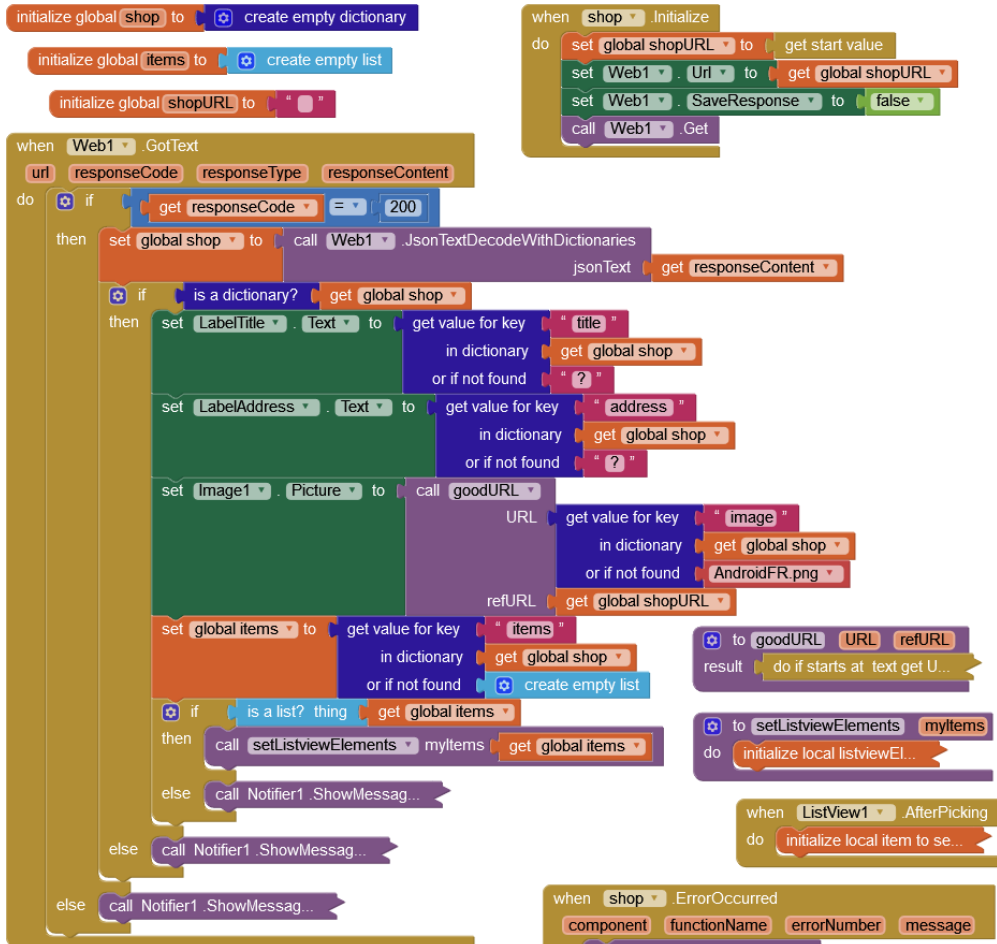
```
when ListView1.AfterPicking
do
  initialize local item to select list item list get global items
  index ListView1.SelectionIndex
  in
  set Image1.Picture to call goodURL
  URL get value for key "image"
  in dictionary get item
  or if not found ""
  rootURL get global shopURL
```

GITSHARE3b : BLOCKS

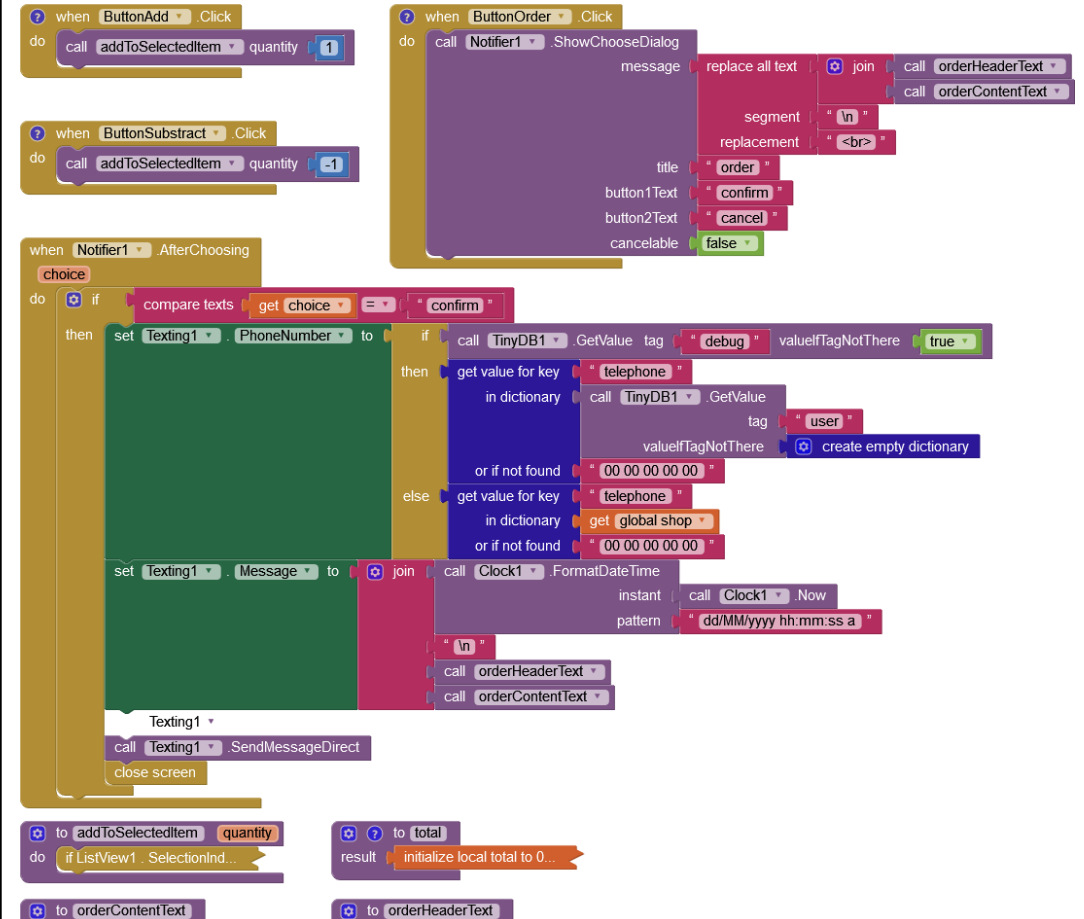


GITSHARE 3b : SHOP SCREEN

Display & select algorithm



Order algorithm (3b)



GITSHARE3b : BLOCKS

```
initialize global shop to create empty dictionary
initialize global items to create empty list
initialize global shopURL to ""

when shop.Initialize
do
  set global shopURL to get start value
  set Web1.Url to get global shopURL
  set Web1.SaveResponse to false
  call Web1.Get

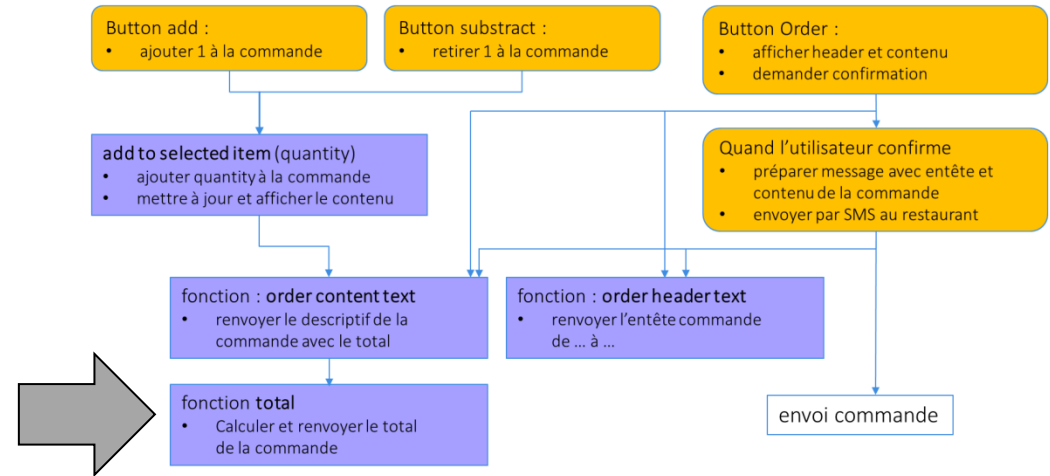
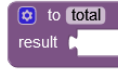
when Web1.GotText
url responseCode responseType responseContent
do
  if responseCode = 200
  then
    set global shop to call Web1.JsonTextDecodeWithDictionaries jsonText get responseContent
    if is a dictionary? get global shop
    then
      set LabelTitle.Text to get value for key "title" in dictionary get global shop or if not found "?"
      set LabelAddress.Text to get value for key "address" in dictionary get global shop or if not found "?"
      set Image1.Picture to call goodURL
      URL get value for key "image" in dictionary get global shop or if not found "AndroidFR.png"
      rootURL get url
      set global items to get value for key "items" in dictionary get global shop or if not found create empty list
      if is a list? thing get global items
      then
        call setListViewElements myitems get global items
      else
        call Notifier1.ShowMessag...
    else
      call Notifier1.ShowMessag...
  else
    call Notifier1.ShowMessag...

to goodURL URL rootURL
result do if starts at text get U...

to setListViewElements myitems
do initialize local listViewEl...

when ListView1.AfterPicking
do initialize local item to se...
```

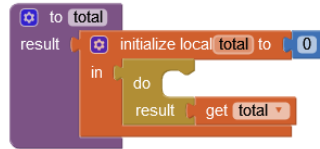
GITSHARE3b : PRÉPARER ET PASSER UNE COMMANDE



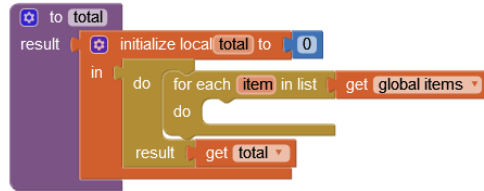
GITSHARE3b : total (FUNCTION)



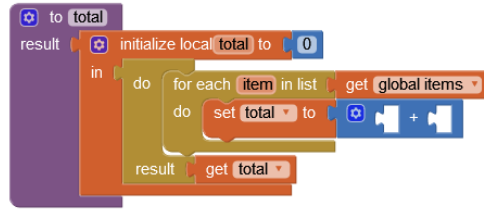
GITSHARE3b : total (FUNCTION)



GITSHARE3b : total (FUNCTION)



GITSHARE3b : total (FUNCTION)

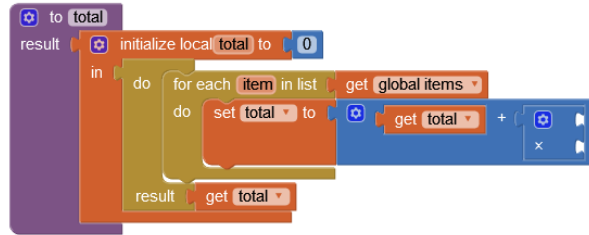


GITSHARE3b : total (FUNCTION)

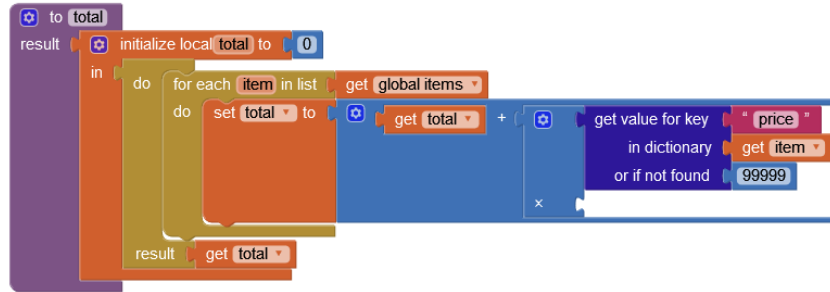
```
to total
  result ← initialize local total to 0
  in
    do
      for each item in list
        do
          set total to (get total) + (get global items)
        do
          result ← get total
    do
      result ← get total
end to total
```

The image shows a Scratch code block for a function named 'total'. The function starts with a 'result' block. It then enters an 'in' block. Inside the 'in' block, there is a 'do' block. Inside the 'do' block, there is a 'for each item in list' block. Inside the 'for each' block, there is a 'do' block. Inside the 'do' block, there is a 'set total to' block. The 'set total to' block is connected to a 'get total' block and a '+' block. The '+' block is connected to another 'get total' block. The 'for each' block is connected to a 'get global items' block. The 'do' block is connected to a 'result ← get total' block. The 'in' block is connected to a 'result ← get total' block. The 'to total' block is connected to a 'result' block.

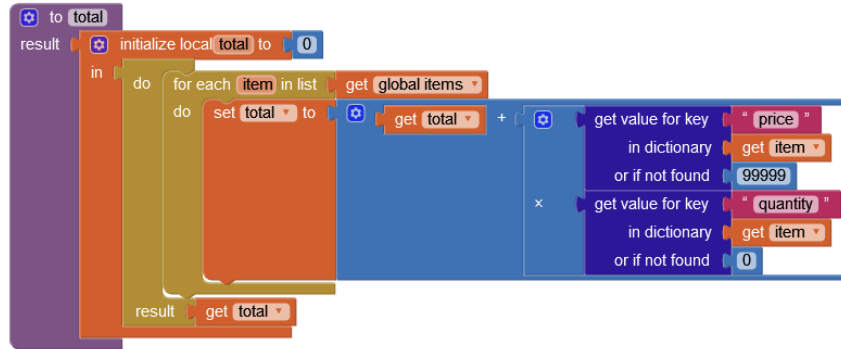
GITSHARE3b : total (FUNCTION)



GITSHARE3b : total (FUNCTION)



GITSHARE3b : total (FUNCTION)

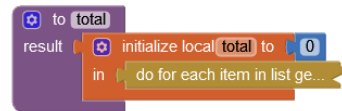


GITSHARE3b : total (FUNCTION)

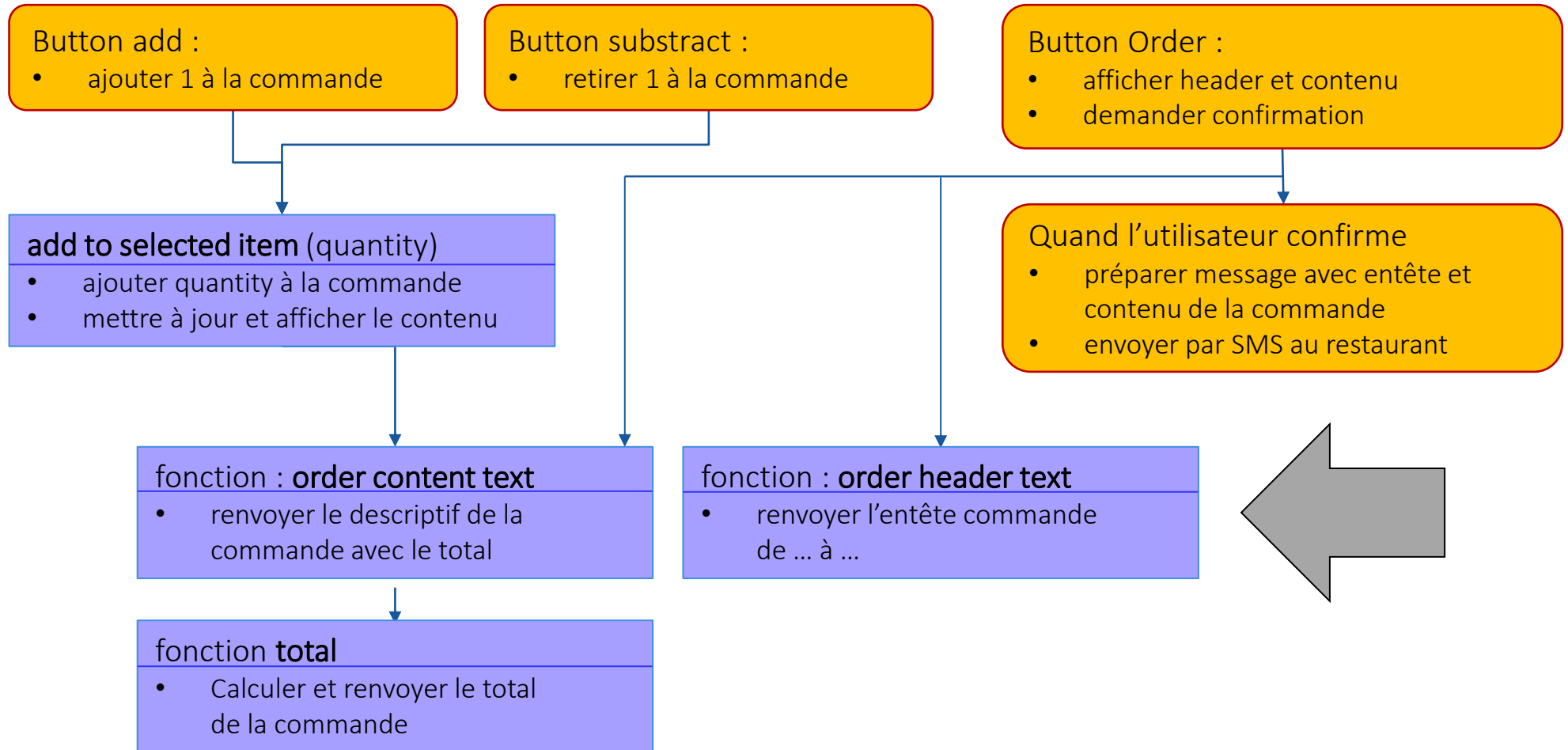
```
to total
  result
  initialize local total to 0
  in
  do
    for each item in list
      do
        set total to
          +
          get total
          x
          get value for key "price"
            in dictionary
            or if not found 99999
            get item
          get value for key "quantity"
            in dictionary
            or if not found 0
            get item
        result get total
```

```
to total
  result
  initialize local total to 0
  in
  do for each item in list ge...
```

GITSHARE3b : total (FUNCTION)



GITSHARE3b : ALGORITHM



GITSHARE3b : header text (function)

The image shows a Scratch script for initializing a user dictionary and retrieving data from it. The script consists of the following blocks:

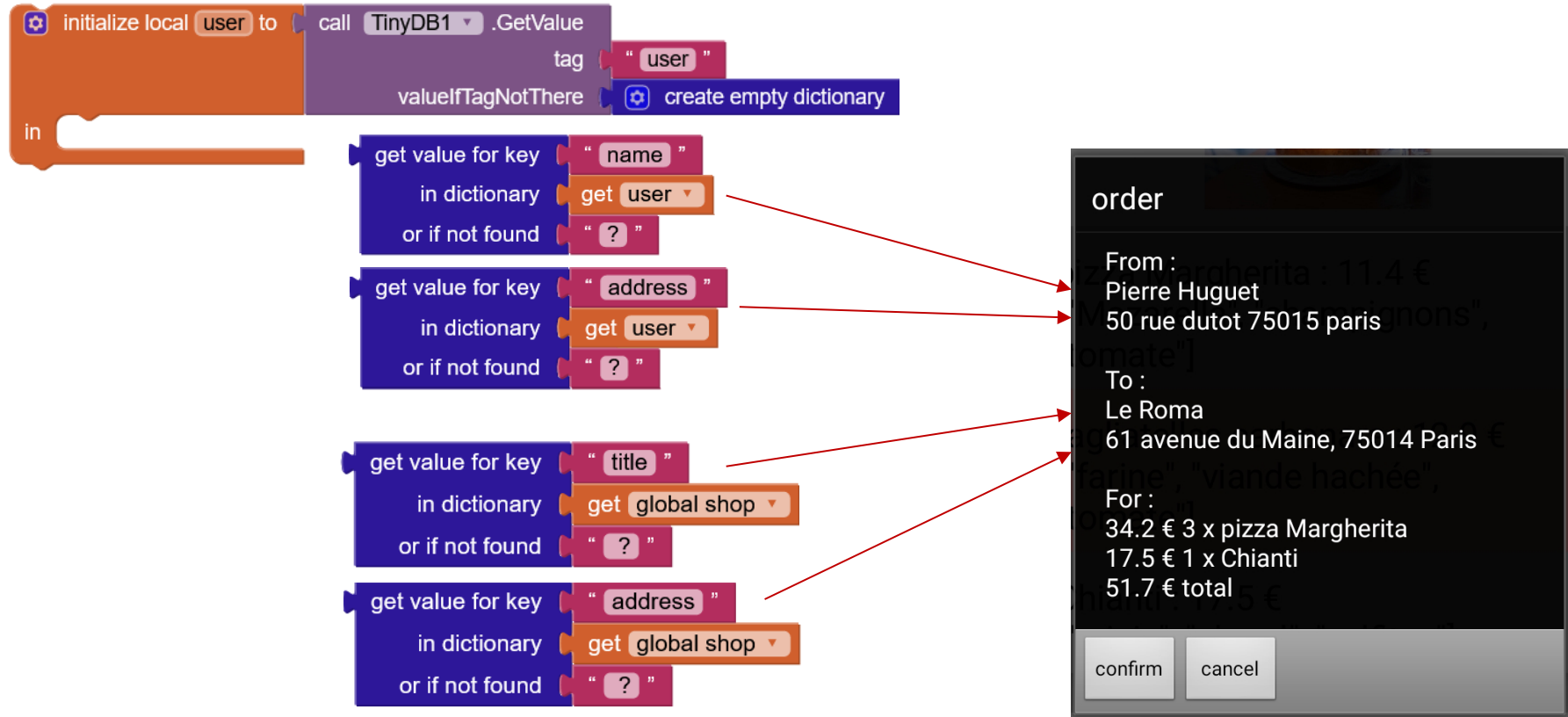
- Initialize local variable:** A block labeled "initialize local" with a gear icon, containing the variable "user".
- Call TinyDB1 .GetValue:** A purple block with a gear icon, containing "TinyDB1", ".GetValue", a tag field with "user", and a "valueIfTagNotThere" field with a "create empty dictionary" block.
- Get value for key:** A blue block with a gear icon, containing "name", "in dictionary" with a "get user" dropdown, and "or if not found" with "?".
- Get value for key:** A blue block with a gear icon, containing "address", "in dictionary" with a "get user" dropdown, and "or if not found" with "?".

Two red arrows point from the "get user" dropdowns in the second and third "Get value for key" blocks to a screenshot of a dark-themed dialog box titled "order". The dialog box contains the following text:

```
order
From :
Pierre Huguet
50 rue dutot 75015 paris
To :
Le Roma
61 avenue du Maine, 75014 Paris
For :
34.2 € 3 x pizza Margherita
17.5 € 1 x Chianti
51.7 € total
```

At the bottom of the dialog box are two buttons: "confirm" and "cancel".

GITSHARE3b : header text (function)



GITSHARE3b : header text (function)

```
to orderHeaderText
result
```

```
to total
result
  initialize local total to 0
  in do for each item in list ge...
```

GITSHARE3b : header text (function)

```
to orderHeaderText
result
  initialize local text to ""
in
```

```
to total
result
  initialize local total to 0
  in do for each item in list ge...
```

GITSHARE3b : header text (function)

```
to orderHeaderText
result initialize local text to ""
in do
result get text
```

```
to total
result initialize local total to 0
in do for each item in list ge...
```

GITSHARE3b: header text (function)

```
to orderHeaderText
result
  initialize local text to ""
  in
    do
      initialize local user to
      in
      result get text
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```

GITSHARE3b : header text (function)

```
to orderHeaderText
result
  initialize local text to ""
  in
    do
      initialize local user to call TinyDB1 .GetValue
      tag "user"
      valueIfTagNotThere
    in
      result get text
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```


GITSHARE3b : header text (function)

```
to orderHeaderText
result
  initialize local text to ""
  in
    do
      initialize local user to call TinyDB1 .GetValue
      tag "user"
      valueIfTagNotThere create empty dictionary
    in
      result get text
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```

GITSHARE3b : header text (function)

```
to orderHeaderText
result
  initialize local text to ""
  in
    do
      initialize local user to call TinyDB1 .GetValue
      tag "user"
      valueIfTagNotThere create empty dictionary
    in
      set text to
  result
  get text
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```

GITSHARE3b : header text (function)

```
to orderHeaderText
result initialize local text to ""
in do initialize local user to call TinyDB1 .GetValue
tag "user "
valueIfTagNotThere create empty dictionary
in set text to join
result get text
```

```
to total
result initialize local total to 0
in do for each item in list ge...
```

GITSHARE3b : header text (function)

```
to orderHeaderText
result initialize local text to ""
in do initialize local user to call TinyDB1 .GetValue
tag "user "
valueIfTagNotThere create empty dictionary
in set text to join "From \n "
result get text
```

```
to total
result initialize local total to 0
in do for each item in list ge...
```

GITSHARE3b : header text (function)

```
to orderHeaderText
result
  initialize local text to ""
  in
    do
      initialize local user to call TinyDB1 .GetValue
      tag "user"
      valueIfTagNotThere
      create empty dictionary
      in
        set text to join "From \n"
        join
      result
    get text
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```

GITSHARE3b : header text (function)

```
to orderHeaderText
result
  initialize local text to ""
  in
    do
      initialize local user to call TinyDB1 .GetValue
      tag "user"
      valueIfTagNotThere create empty dictionary
      in
        set text to join "From \n"
        join
          get value for key "name"
          in dictionary get user
          or if not found "?"
      result
    get text
  end
end
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
  end
end
```

GITSHARE3b : header text (function)

```
to orderHeaderText
result
  initialize local text to ""
  in
    do
      initialize local user to call TinyDB1 .GetValue
      tag "user"
      valueIfTagNotThere create empty dictionary
      in
        set text to join "From \n"
        join
          get value for key "name"
          in dictionary get user
          or if not found "?"
          "\n"
      result get text
    end
  end
end
```

```
to total
result
  initialize local total to 0
  in do for each item in list ge...
```

GITSHARE3b : header text (function)

```
to orderHeaderText
result initialize local text to ""
in do initialize local user to call TinyDB1 .GetValue
tag "user "
valueIfTagNotThere create empty dictionary
in set text to join "From \n "
join get value for key "name "
in dictionary get user
or if not found "?"
"\n "
get value for key "address "
in dictionary get user
or if not found "?"
"\n\n "
result get text
end
```

```
to total
result initialize local total to 0
in do for each item in list ge...
```


GITSHARE3B : header text (function)

```
to orderHeaderText
result
  initialize local text to ""
  in
    do
      initialize local user to
        call TinyDB1 GetValue
          tag "user"
          valueIfTagNotThere create empty dictionary
      in
        set text to
          join
            "From \n"
            join
              get value for key "name" in dictionary get user or if not found "?"
              "\n"
              get value for key "address" in dictionary get user or if not found "?"
              "\n\n"
          result
    get text
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```

GITSHARE3B : header text (function)

```
to orderHeaderText
  result initialize local text to ""
  in do
    initialize local user to call TinyDB1 GetValue
    tag "user"
    valueIfTagNotThere create empty dictionary
    in set text to join
      "From \n"
      join
        get value for key "name" in dictionary get user or if not found "?"
        "\n"
        get value for key "address" in dictionary get user or if not found "?"
        "\n\n"
      "To \n"
    result get text
  end
end

to total
  result initialize local total to 0
  in do for each item in list ge...
```

GITSHARE3B : header text (function)

```
to orderHeaderText
result initialize local text to ""
in do initialize local user to call TinyDB1 GetValue
tag "user"
valueIfTagNotThere create empty dictionary
in set text to join "From:\n"
join
get value for key "name" in dictionary get user or if not found "?"
"\n"
get value for key "address" in dictionary get user or if not found "?"
"\n\n"
join "To:\n"
join
result get text
end

to total
result initialize local total to 0
in do for each item in list ge...
```

GITSHARE3B : header text (function)

```
to orderHeaderText
result
  initialize local text to ""
  in
    do
      initialize local user to
        call TinyDB1 GetValue
          tag "user"
          valueIfTagNotThere create empty dictionary
      in
        set text to
          join
            "From \n"
            join
              get value for key "name" in dictionary get user or if not found "?"
              "\n"
              get value for key "address" in dictionary get user or if not found "?"
              "\n\n"
            "To:\n"
            join
              get value for key "title" in dictionary get global shop or if not found "?"
              "\n"
          result
    get text
end
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```

GITSHARE3B : header text (function)

```
to orderHeaderText
result initialize local text to ""
in do initialize local user to call TinyDB1 GetValue
tag "user"
valueIfTagNotThere create empty dictionary
in set text to join "From \n"
join
get value for key "name" in dictionary get user or if not found "?"
"\n"
get value for key "address" in dictionary get user or if not found "?"
"\n\n"
" To \n"
join
get value for key "title" in dictionary get global shop or if not found "?"
"\n"
get value for key "address" in dictionary get global shop or if not found "?"
"\n\n"
" For \n"
result get text
end

to total
result initialize local total to 0
in do for each item in list ge...
```

GITSHARE3B : header text (function)

```
to orderHeaderText
result initialize local text to ""
in do initialize local user to call TinyDB1 GetValue
tag "user"
valueIfTagNotThere create empty dictionary
in set text to join "From \n"
join get value for key "name" in dictionary get user or if not found "?"
"\n"
get value for key "address" in dictionary get user or if not found "?"
"\n\n"
" To \n"
join get value for key "title" in dictionary get global shop or if not found "?"
"\n"
get value for key "address" in dictionary get global shop or if not found "?"
"\n\n"
" For \n"
result get text
end

to total
result initialize local total to 0
in do for each item in list ge...
```

GITSHARE3B : header text (function)

```
to orderHeaderText
result
  initialize local text to ""
  in
    do
      initialize local user to
        call TinyDB1 GetValue
          tag "user"
          valueIfTagNotThere create empty dictionary
      in
        set text to
          join
            "From \n"
            join
              get value for key "name" in dictionary get user or if not found "?"
              "\n"
              get value for key "address" in dictionary get user or if not found "?"
              "\n\n"
            "To \n"
            join
              get value for key "title" in dictionary get global shop or if not found "?"
              "\n"
              get value for key "address" in dictionary get global shop or if not found "?"
              "\n\n"
            "For \n"
          result
        get text
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...

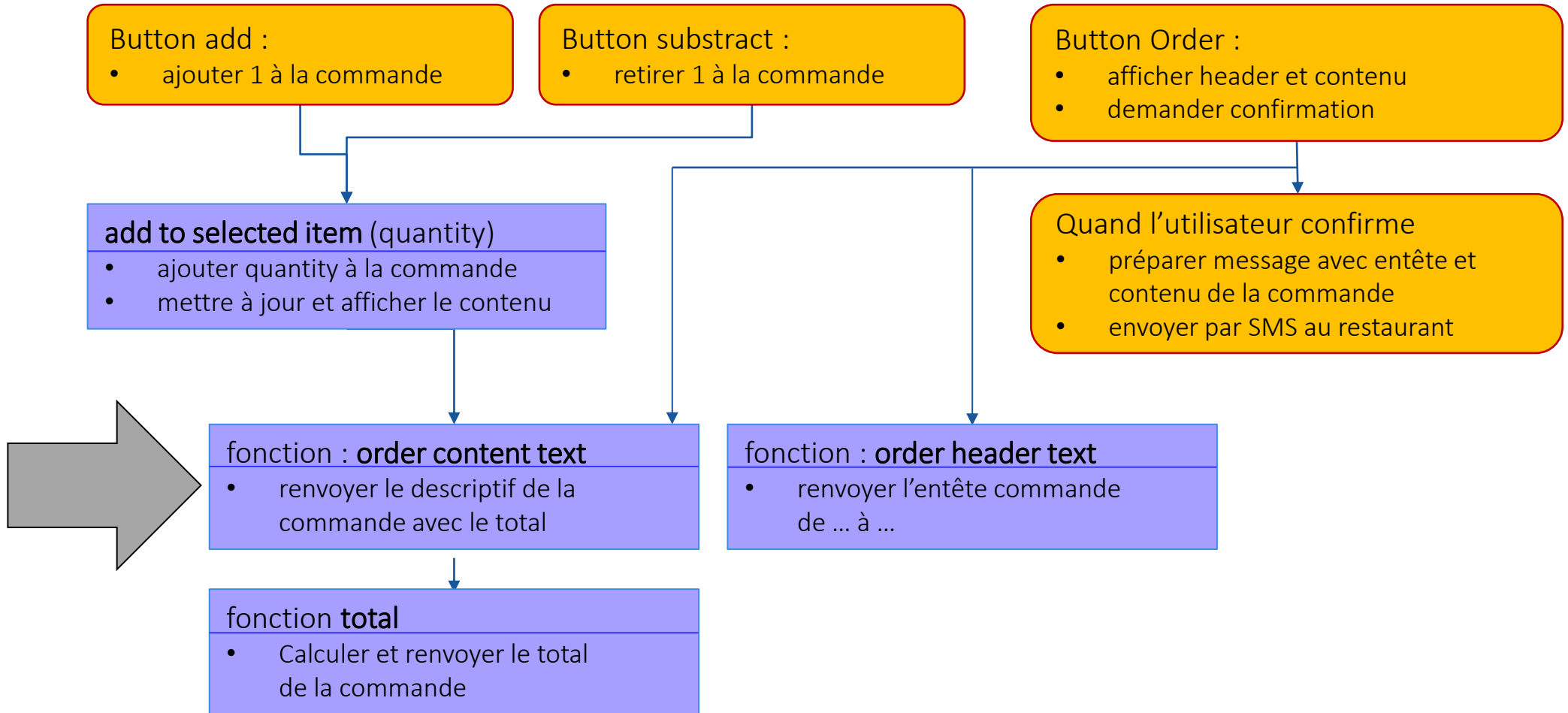
to orderHeaderText
result
  initialize local text to ""
  in
    do initialize local user to ...
```

GITSHARE3B : header text (function)

```
to total
result
  initialize local total to 0
  in do for each item in list ge...
end

to orderHeaderText
result
  initialize local text to ""
  in do initialize local user to...
```


GITSHARE3B : ALGORITHM



GITSHARE3B : order content (function)

```
to orderContentText
result
```

```
34.2 € 3 x pizza Margherita
17.5 € 1 x Chianti
51.7 € total
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...

to orderHeaderText
result
  initialize local text to ""
  in
    do initialize local user to ...
```

GITSHARE3B : order content (function)

```
to orderContentText
result initialize local text to ""
in do
result get text
```

```
to total
result initialize local total to 0
in do for each item in list ge...
```

```
to orderHeaderText
result initialize local text to ""
in do initialize local user to ...
```

GITSHARE3B : order content (function)

```
to orderContentText
result
  initialize local text to ""
  in
    do
      for each item in list
        do
          if
            then
          result
            get text
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```

```
to orderHeaderText
result
  initialize local text to ""
  in
    do initialize local user to...
```

GITSHARE3B : order content (function)

```
to orderContentText
result
  initialize local text to ""
  in
    do
      for each item in list get global items
      do
        if
        then
      result get text
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```

```
to orderHeaderText
result
  initialize local text to ""
  in
    do initialize local user to ...
```

GITSHARE3B : order content (function)

```
to orderContentText
result
  initialize local text to ""
  in
    do
      for each item in list
        get global items
        do
          if
            get value for key "quantity"
            in dictionary
            get item
            or if not found 0
          then
            result
            get text
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```

```
to orderHeaderText
result
  initialize local text to ""
  in
    do initialize local user to...
```

GITSHARE3B : order content (function)

```
to orderContentText
result
  initialize local text to ""
  in
    do
      for each item in list
        get global items
        do
          if
            get value for key "quantity"
            in dictionary
            or if not found 0
            then
              > 0
          result get text
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```

```
to orderHeaderText
result
  initialize local text to ""
  in
    do initialize local user to...
```

GITSHARE3B : order content (function)

```
to orderContentText
result initialize local text to ""
in do
  for each item in list get global items
  do
    if get value for key "quantity"
      in dictionary get item
      or if not found 0
    then set text to
  result get text
```

```
to total
result initialize local total to 0
in do for each item in list ge...
```

```
to orderHeaderText
result initialize local text to ""
in do initialize local user to ...
```


GITSHARE3B : order content (function)

```
to orderContentText
result initialize local text to ""
in do
  for each item in list
    get global items
    do
      if
        get value for key "quantity"
        in dictionary
        or if not found 0
      then
        set text to join
result get text
```

```
to total
result initialize local total to 0
in do for each item in list ge...
```

```
to orderHeaderText
result initialize local text to ""
in do initialize local user to ...
```

GITSHARE3B : order content (function)

```
to orderContentText
result initialize local text to ""
in do
  for each item in list
    get global items
    do
      if
        get value for key "quantity"
        in dictionary
        or if not found 0
      then
        set text to join get text
```

```
to total
result initialize local total to 0
in do for each item in list ge...
```

```
to orderHeaderText
result initialize local text to ""
in do initialize local user to ...
```

GITSHARE3B : order content (function)

```
to orderContentText
result initialize local text to ""
in do
  for each item in list
    get global items
    do
      if
        get value for key "quantity"
        in dictionary
        or if not found 0
      then
        set text to join
        get text
        ""
    result get text
```

```
to total
result initialize local total to 0
in do for each item in list ge...
```

```
to orderHeaderText
result initialize local text to ""
in do initialize local user to ...
```

GITSHARE3B : order content (function)

```
to orderContentText
result initialize local text to ""
in do
  for each item in list
    get global items
    do
      if
        get value for key "quantity"
        in dictionary
        or if not found 0
      then
        set text to join
          get text
          " "
          ×
    result get text
```

```
to total
result initialize local total to 0
in do for each item in list ge...
```

```
to orderHeaderText
result initialize local text to ""
in do initialize local user to ...
```

GITSHARE3B : order content (function)

```
to orderContentText
result initialize local text to ""
in do
  for each item in list get global items
  do
    if
      get value for key "quantity"
      in dictionary get item
      or if not found 0
      > 0
    then
      set text to join
      get text
      +
      get value for key "quantity"
      in dictionary get item
      or if not found
      x
  result get text
```

```
to total
result initialize local total to 0
in do for each item in list ge...
```

```
to orderHeaderText
result initialize local text to ""
in do initialize local user to...
```

GITSHARE3B : order content (function)

```
to orderContentText
result initialize local text to ""
in do
  for each item in list get global items
  do
    if
      get value for key "quantity"
      in dictionary get item
      or if not found 0
    then
      set text to join
      get text
      " "
      get value for key "quantity"
      in dictionary get item
      or if not found 0
      ×
      get value for key "price"
      in dictionary get item
      or if not found 0
  result get text
end
```

```
to total
result initialize local total to 0
in do for each item in list ge...
```

```
to orderHeaderText
result initialize local text to ""
in do initialize local user to ...
```

GITSHARE3B : order content (function)

```
to orderContentText
result initialize local text to ""
in do
  for each item in list get global items
  do
    if get value for key "quantity"
      in dictionary get item
      or if not found 0
    > 0
    then set text to join
      get text
      " "
      get value for key "quantity"
      in dictionary get item
      or if not found 0
      ×
      get value for key "price"
      in dictionary get item
      or if not found 9999
  result get text
end
```

```
to total
result initialize local total to 0
in do for each item in list ge...
```

```
to orderHeaderText
result initialize local text to ""
in do initialize local user to ...
```

GITSHARE3B : order content (function)

```
to orderContentText
result
  initialize local text to ""
  in
    do
      for each item in list
        get global items
        do
          if
            get value for key "quantity"
            in dictionary
            get item
            or if not found 0
            > 0
          then
            set text to join
              get text
              " "
              get value for key "quantity" in dictionary
              get item
              or if not found 0
              " x "
              get value for key "price" in dictionary
              get item
              or if not found 9999
            result
            get text
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```

```
to orderHeaderText
result
  initialize local text to ""
  in
    do initialize local user to...
```


GITSHARE3B : order content (function)

```
to orderContentText
result
  initialize local text to ""
  in
    do
      for each item in list
        get global items
        do
          if
            get value for key "quantity"
            in dictionary
            get item
            or if not found 0
            > 0
          then
            set text to
              join
                get text
                " "
                get value for key "quantity" in dictionary
                get item
                or if not found 0
                ×
                get value for key "price" in dictionary
                get item
                or if not found 9999
                " € "
            result
              get text
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```

```
to orderHeaderText
result
  initialize local text to ""
  in
    do initialize local user to...
```

GITSHARE3B : order content (function)

```
to orderContentText
result
  initialize local text to ""
  in
    do
      for each item in list
        get global items
        do
          if
            get value for key "quantity"
            in dictionary
            get item
            or if not found 0
            > 0
          then
            set text to join
              get text
              " "
              get value for key "quantity" in dictionary
              get item
              or if not found 0
              " x "
              get value for key "price" in dictionary
              get item
              or if not found 9999
              " € "
              get value for key "quantity" in dictionary
              get item
              or if not found 0
            result
            get text
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```

```
to orderHeaderText
result
  initialize local text to ""
  in
    do initialize local user to...
```

GITSHARE3B : order content (function)

```
to orderContentText
result
  initialize local text to ""
  in
    do
      for each item in list
        get global items
        do
          if
            get value for key "quantity"
            in dictionary
            get item
            or if not found 0
            > 0
          then
            set text to join
              get text
              " "
              get value for key "quantity" in dictionary
              get item
              or if not found 0
              " x "
              get value for key "price" in dictionary
              get item
              or if not found 9999
              " € "
              get value for key "quantity" in dictionary
              get item
              or if not found 0
              " x "
            result
            get text
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```

```
to orderHeaderText
result
  initialize local text to ""
  in
    do initialize local user to...
```

GITSHARE3B : order content (function)

```
to orderContentText
result
  initialize local text to ""
  in
    do
      for each item in list
        get global items
        do
          if
            get value for key "quantity"
            in dictionary
            get item
            or if not found 0
            > 0
          then
            set text to
              join
                get text
                " "
                get value for key "quantity" in dictionary
                get item
                or if not found 0
                " x "
                get value for key "price" in dictionary
                get item
                or if not found 9999
                " € "
                get value for key "quantity" in dictionary
                get item
                or if not found 0
                " x "
                get value for key "name" in dictionary
                get item
                or if not found "?"
            result
            get text
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```

```
to orderHeaderText
result
  initialize local text to ""
  in
    do initialize local user to...
```

GITSHARE3B : order content (function)

```
to orderContentText
result
  initialize local text to ""
  in
    do
      for each item in list
        get global items
        do
          if
            get value for key "quantity"
            in dictionary
            get item
            or if not found 0
            > 0
          then
            set text to
              join
                get text
                " "
                get value for key "quantity" in dictionary
                get item
                or if not found 0
                " x "
                get value for key "price" in dictionary
                get item
                or if not found 9999
                " € "
                get value for key "quantity" in dictionary
                get item
                or if not found 0
                " x "
                get value for key "name" in dictionary
                get item
                or if not found "?"
                "\n"
            result
              get text
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```

```
to orderHeaderText
result
  initialize local text to ""
  in
    do initialize local user to...
```

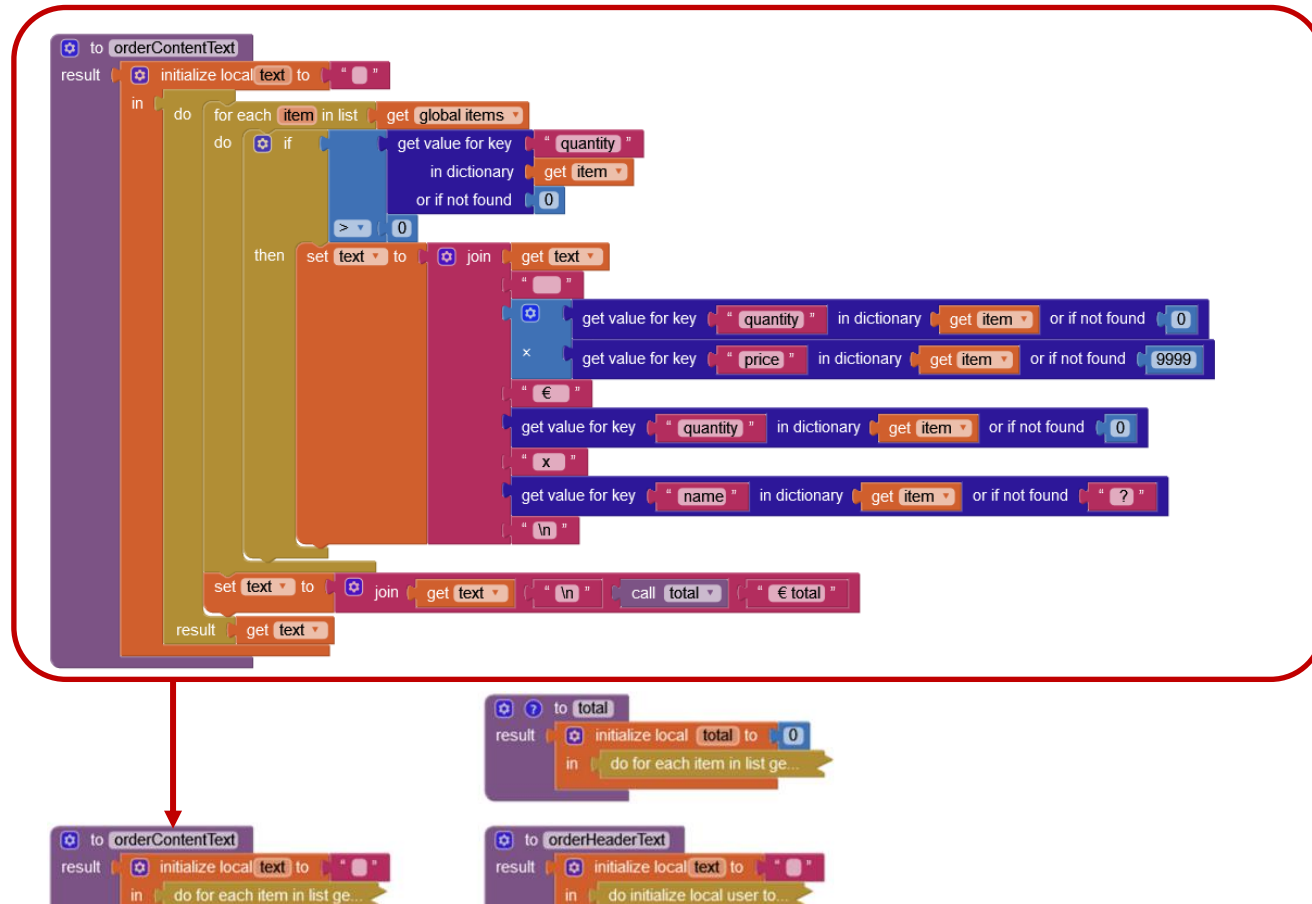
GITSHARE3B : order content (function)

```
to orderContentText
result initialize local text to ""
in do
  for each item in list
    get global items
    do
      if
        get value for key "quantity"
        in dictionary
        get item
        or if not found 0
        > 0
      then
        set text to join
          get text
          " "
          get value for key "quantity" in dictionary
          get item
          or if not found 0
          " x "
          get value for key "price" in dictionary
          get item
          or if not found 9999
          " € "
          get value for key "quantity" in dictionary
          get item
          or if not found 0
          " x "
          get value for key "name" in dictionary
          get item
          or if not found "?"
          "\n"
        set text to join
          get text
          "\n"
          call total
          " € total "
    result get text
end
```

```
to total
result initialize local total to 0
in do for each item in list ge...
```

```
to orderHeaderText
result initialize local text to ""
in do initialize local user to...
```

GITSHARE3B : order content (function)



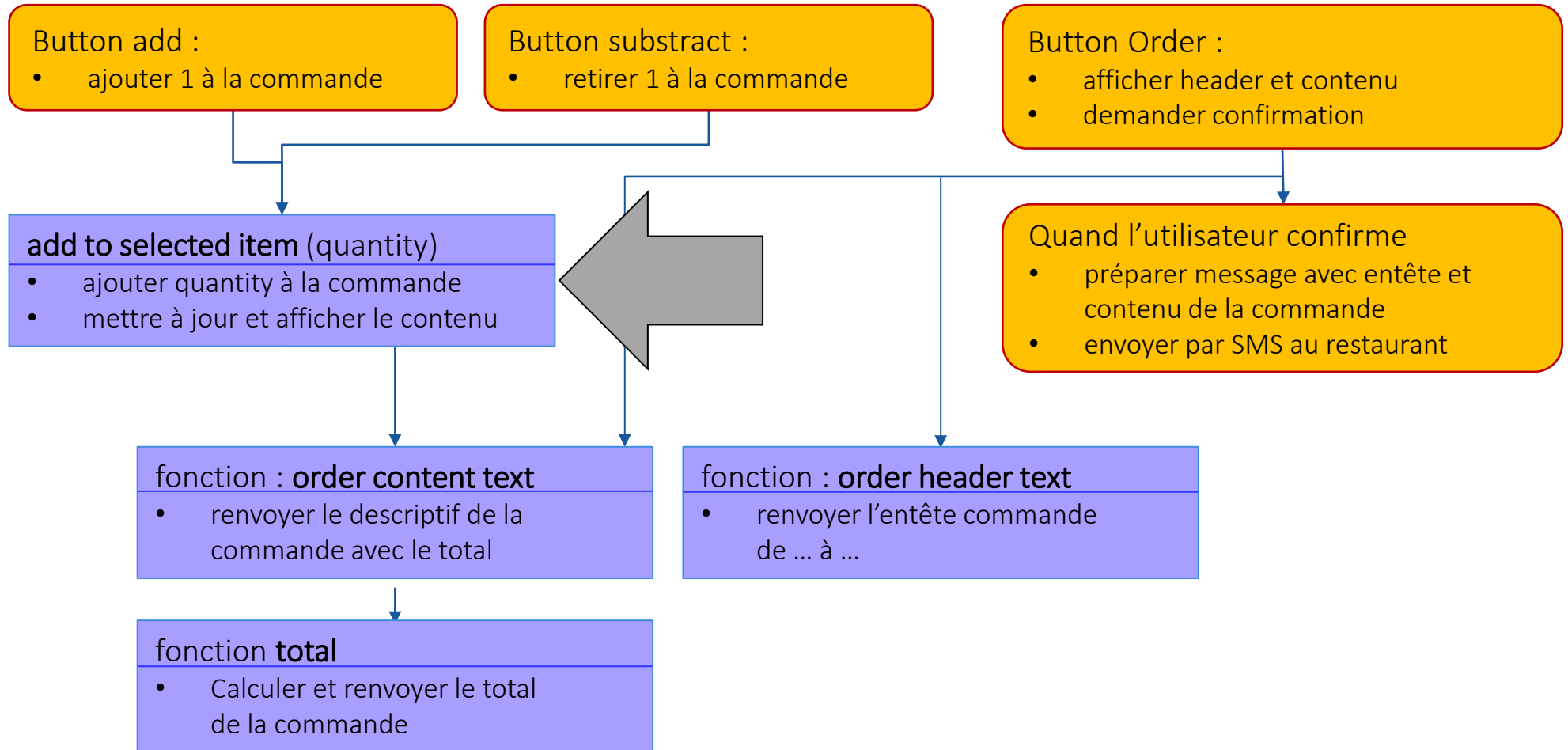
GITSHARE3B : order content (function)

```
to orderContentText
  result initialize local text to ""
  in do for each item in list ge...
end

to total
  result initialize local total to 0
  in do for each item in list ge...
end

to orderHeaderText
  result initialize local text to ""
  in do initialize local user to ...
end
```


GITSHARE3B : ALGORITHM



GITSHARE3B : addToSelectedItem (quantity)

```
to addToSelectedItem quantity
do
```

```
to total
result
  initialize local total to 0
  in do for each item in list ge...

to orderContentText
result
  initialize local text to ""
  in do for each item in list ge...

to orderHeaderText
result
  initialize local text to ""
  in do initialize local user to...
```

GITSHARE3B : addToSelectedItem (quantity)

```
to addToSelectedItem quantity
do
  if ListView1 . SelectionIndex ≠ 0
  then
```

```
to total
result
  initialize local total to 0
  in do for each item in list ge...

to orderContentText
result
  initialize local text to ""
  in do for each item in list ge...

to orderHeaderText
result
  initialize local text to ""
  in do initialize local user to...
```

GITSHARE3B : addToSelectedItem (quantity)

```
to addToSelectedItem quantity
do
  if ListView1 . SelectionIndex ≠ 0
  then
    initialize local item to
      select list item list
      index ListView1 . SelectionIndex
    in
```

```
to total
result
  initialize local total to 0
  in do for each item in list ge...
```

```
to orderContentText
result
  initialize local text to ""
  in do for each item in list ge...
```

```
to orderHeaderText
result
  initialize local text to ""
  in do initialize local user to...
```

GITSHARE3B : addToSelectedItem (quantity)

```
to addToSelectedItem quantity
do
  if ListView1 . SelectionIndex ≠ 0
  then
    initialize local item to select list item list get global items
    index ListView1 . SelectionIndex
    in
      set value for key "quantity"
      in dictionary get item
      to
```

```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```

```
to orderContentText
result
  initialize local text to ""
  in
    do for each item in list ge...
```

```
to orderHeaderText
result
  initialize local text to ""
  in
    do initialize local user to...
```

GITSHARE3B : addToSelectedItem (quantity)

```
to addToSelectedItem quantity
do
  if [ListView1 . SelectionIndex] ≠ 0
  then
    initialize local item to select list item list
    index [ListView1 . SelectionIndex]
    in
      set value for key "quantity"
      in dictionary get item
      to +
```

```
to total
result
  initialize local total to 0
  in do for each item in list ge...
```

```
to orderContentText
result
  initialize local text to ""
  in do for each item in list ge...
```

```
to orderHeaderText
result
  initialize local text to ""
  in do initialize local user to...
```

GITSHARE3B : addToSelectedItem (quantity)

```
to addToSelectedItem quantity
do
  if ListView1 . SelectionIndex ≠ 0
  then
    initialize local item to select list item list
    index ListView1 . SelectionIndex
    in
      set value for key "quantity"
      in dictionary get item
      to
        get value for key "quantity" +
        in dictionary get item
        or if not found 0
```

```
to total
result initialize local total to 0
in do for each item in list ge...
```

```
to orderContentText
result initialize local text to ""
in do for each item in list ge...
```

```
to orderHeaderText
result initialize local text to ""
in do initialize local user to...
```

GITSHARE3B : addToSelectedItem (quantity)

```
to addToSelectedItem quantity
do
  if ListView1 . SelectionIndex ≠ 0
  then
    initialize local item to select list item list
    index ListView1 . SelectionIndex
    in
      set value for key "quantity"
      in dictionary get item
      to
        get value for key "quantity" + get quantity
        in dictionary get item
        or if not found 0
```

```
to total
result
  initialize local total to 0
  in do for each item in list ge...
```

```
to orderContentText
result
  initialize local text to ""
  in do for each item in list ge...
```

```
to orderHeaderText
result
  initialize local text to ""
  in do initialize local user to...
```


GITSHARE3B : addToSelectedItem (quantity)

```
to addToSelectedItem quantity
do
  if [ListView1 . SelectionIndex] ≠ 0
  then
    initialize local item to select list item list
    index [ListView1 . SelectionIndex]
    in
      set value for key "quantity"
      in dictionary
      to
        get value for key "quantity"
        in dictionary
        or if not found 0
        + get quantity
    if
    then
```

```
to total
result initialize local total to 0
in do for each item in list ge...
```

```
to orderContentText
result initialize local text to ""
in do for each item in list ge...
```

```
to orderHeaderText
result initialize local text to ""
in do initialize local user to...
```

GITSHARE3B : addToSelectedItem (quantity)

```
to addToSelectedItem quantity
do
  if [ListView1 . SelectionIndex] ≠ 0
  then
    initialize local item to select list item list
    index [ListView1 . SelectionIndex]
    in
      set value for key "quantity"
      in dictionary get item
      to
        get value for key "quantity" + get quantity
        in dictionary get item
        or if not found 0
      if
        get value for key "quantity" < 0
        in dictionary get item
        or if not found 0
      then
```

```
to total
result
  initialize local total to 0
  in do for each item in list ge...
```

```
to orderContentText
result
  initialize local text to ""
  in do for each item in list ge...
```

```
to orderHeaderText
result
  initialize local text to ""
  in do initialize local user to...
```

GITSHARE3B : addToSelectedItem (quantity)

```
when ButtonAdd .Click
do
  to addToSelectedItem quantity
  do
    if ListView1 .SelectionIndex ≠ 0
    then
      initialize local item to select list item list
      index ListView1 .SelectionIndex
      in
        set value for key "quantity"
        in dictionary get item
        to
          get value for key "quantity" + get quantity
          in dictionary get item
          or if not found 0
        if
          get value for key "quantity" < 0
          in dictionary get item
          or if not found 0
        then
          set value for key "quantity"
          in dictionary get item
          to 0
    end
  end
end
```

```
to total
result
  initialize local total to 0
  in do for each item in list ge...
```

```
to orderContentText
result
  initialize local text to ""
  in do for each item in list ge...
```

```
to orderHeaderText
result
  initialize local text to ""
  in do initialize local user to...
```

GITSHARE3B : addToSelectedItem (quantity)

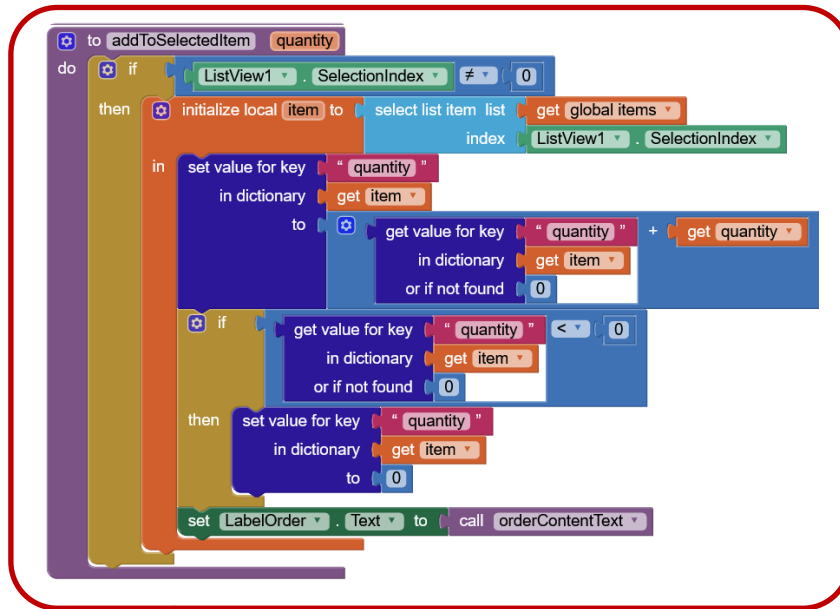
```
to addToSelectedItem quantity
do
  if [ListView1 . SelectionIndex] ≠ 0
  then
    initialize local item to select list item list
    index [ListView1 . SelectionIndex]
    in
      set value for key "quantity"
      in dictionary get item
      to
        get value for key "quantity" + get quantity
        in dictionary get item
        or if not found 0
      if
        get value for key "quantity" < 0
        in dictionary get item
        or if not found 0
      then
        set value for key "quantity"
        in dictionary get item
        to 0
    set LabelOrder . Text to call orderContentText
```

```
to total
result
  initialize local total to 0
  in do for each item in list ge...
```

```
to orderContentText
result
  initialize local text to ""
  in do for each item in list ge...
```

```
to orderHeaderText
result
  initialize local text to ""
  in do initialize local user to...
```

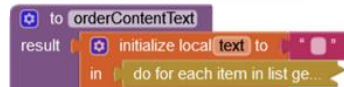
GITSHARE3B : addToSelectedItem (quantity)



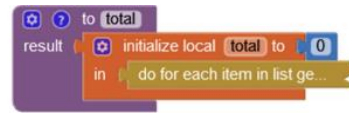
```
to addToSelectedItem quantity
do
  if (ListView1.SelectionIndex <> 0)
  then
    initialize local item to select list item list
    index ListView1.SelectionIndex
    in
      set value for key "quantity"
      in dictionary get item
      to
        get value for key "quantity" + get quantity
        in dictionary get item
        or if not found 0
      if
        get value for key "quantity" < 0
        in dictionary get item
        or if not found 0
      then
        set value for key "quantity"
        in dictionary get item
        to 0
    set LabelOrder.Text to call orderContentText
```



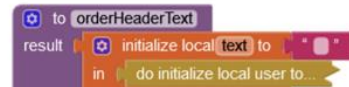
```
to addToSelectedItem quantity
do
  initialize local item to se...
```



```
to orderContentText
result
  initialize local text to " "
  in
    do for each item in list ge...
```



```
to total
result
  initialize local total to 0
  in
    do for each item in list ge...
```



```
to orderHeaderText
result
  initialize local text to " "
  in
    do initialize local user to...
```

GITSHARE3B : addToSelectedItem (quantity)

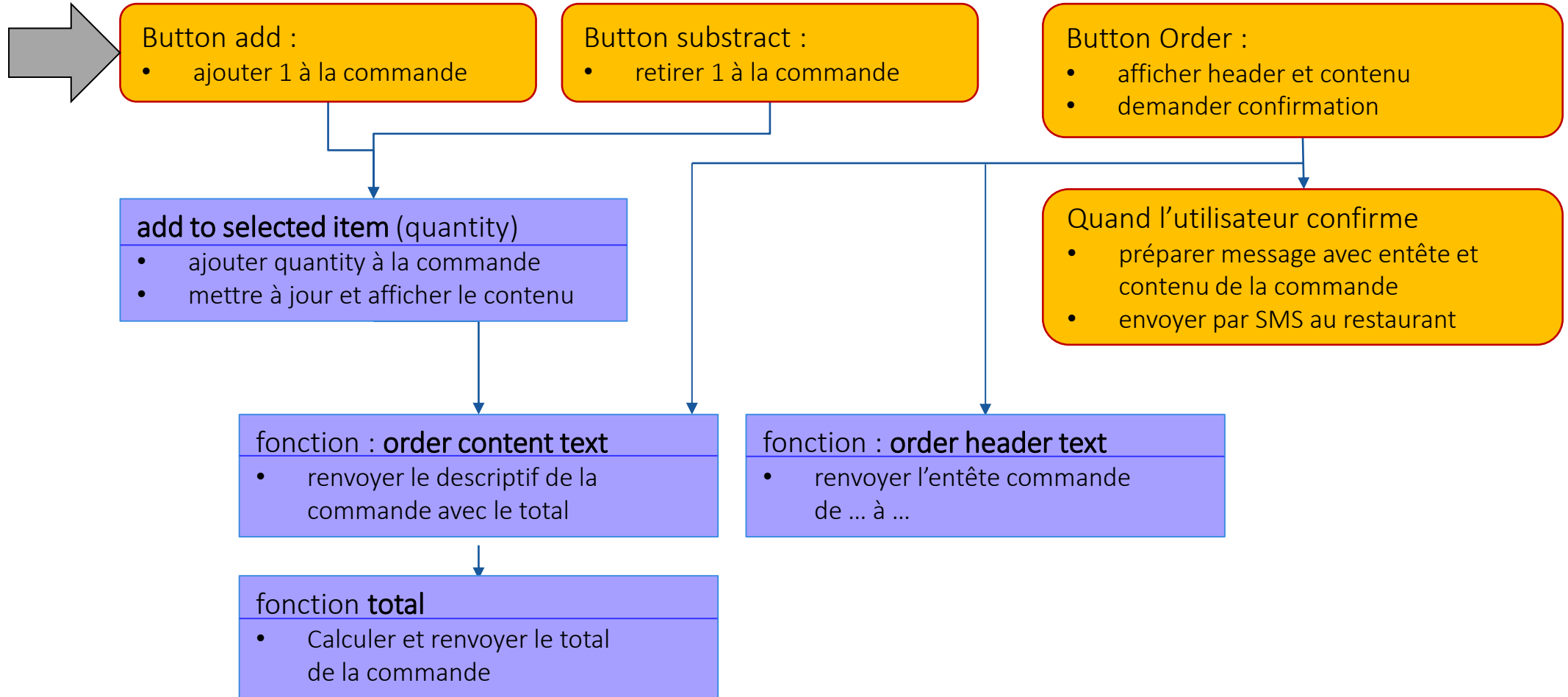
```
to addToSelectedItem quantity
do
  initialize local item to se...

to total
result
  initialize local total to 0
  in
    do for each item in list ge...

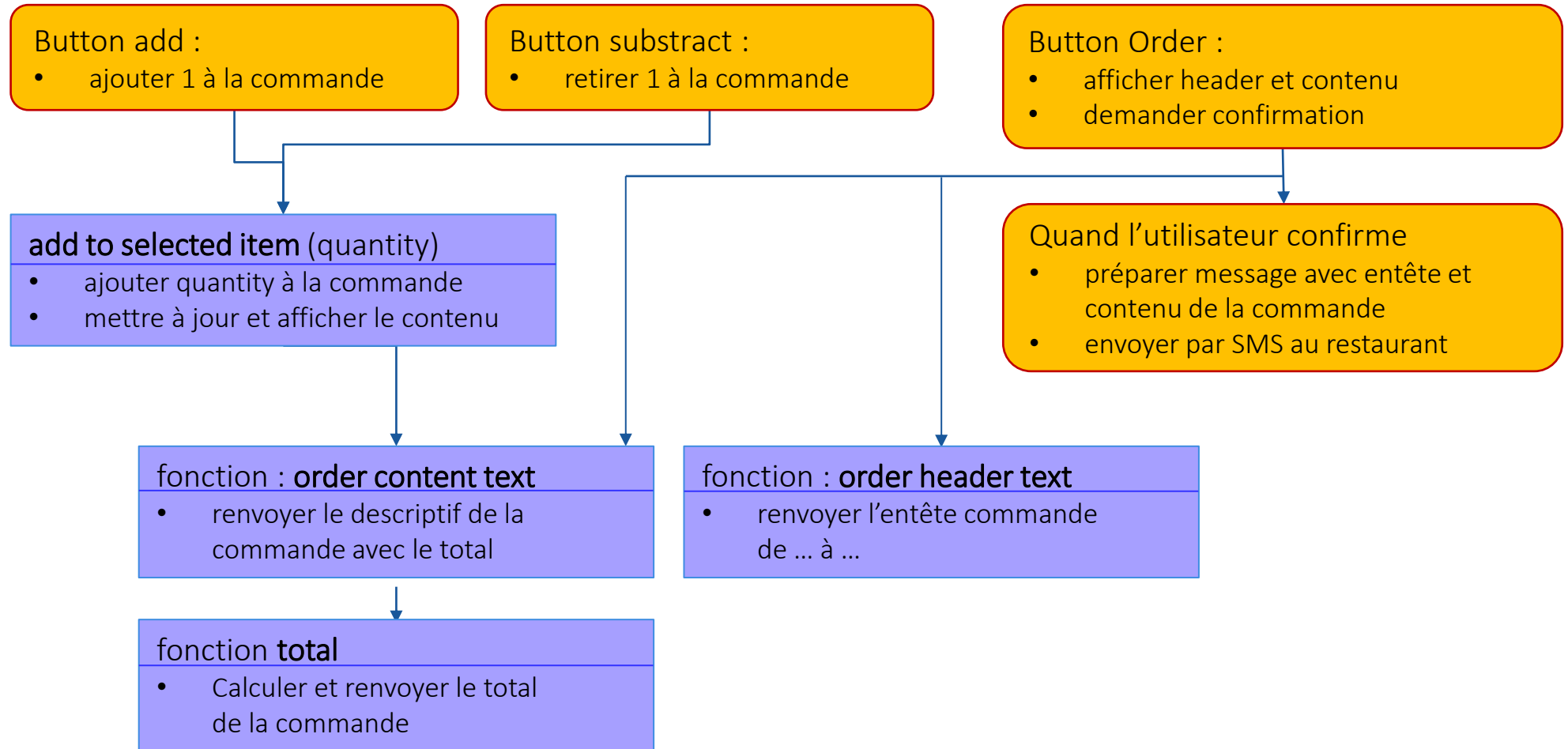
to orderContentText
result
  initialize local text to ""
  in
    do for each item in list ge...

to orderHeaderText
result
  initialize local text to ""
  in
    do initialize local user to...
```

GITSHARE3b : ALGORITHM



GITSHARE3b : ALGORITHM



GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

```
to addToSelectedItem quantity
do
  if ListView1_SelectionInd...

to total
result initialize local total to 0...

to orderContentText
result initialize local text to *

to orderHeaderText
result initialize local text to *
```

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

```
? when ButtonAdd .Click  
do
```

```
+ to addToSelectedItem quantity  
do if ListView1.SelectionInd...
```

```
+ ? to total  
result initialize local total to 0...
```

```
+ to orderContentText  
result initialize local text to "
```

```
+ to orderHeaderText  
result initialize local text to "
```

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

```
when ButtonAdd Click
do
  call addToSelectedItem quantity 1
```

```
to addToSelectedItem quantity
do
  if ListView1_SelectionInd...
```

```
to total
result initialize local total to 0...
```

```
to orderContentText
result initialize local text to "
```

```
to orderHeaderText
result initialize local text to "
```

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

```
? when ButtonAdd Click
do
  call addToSelectedItem quantity 1
```

```
? when ButtonSubtract Click
do
```

```
+ to addToSelectedItem quantity
do
  if ListView1_SelectionInd...
```

```
+ to total
result initialize local total to 0...
```

```
+ to orderContentText
result initialize local text to "
```

```
+ to orderHeaderText
result initialize local text to "
```

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

```
? when ButtonAdd Click
do
  call addToSelectedItem quantity 1
```

```
? when ButtonSubstract Click
do
  call addToSelectedItem quantity -1
```

```
+ to addToSelectedItem quantity
do
  if ListView1_SelectionInd...
```

```
+ to total
result initialize local total to 0...
```

```
+ to orderContentText
result initialize local text to *
```

```
+ to orderHeaderText
result initialize local text to *
```

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

```
? when ButtonAdd .Click  
do  
  call addToSelectedItem quantity 1
```

```
? when ButtonOrder .Click  
do
```

```
? when ButtonSubtract .Click  
do  
  call addToSelectedItem quantity -1
```

```
+ to addToSelectedItem quantity  
do  
  if ListView1.SelectionInd...
```

```
+ to total  
result initialize local total to 0...
```

```
+ to orderContentText  
result initialize local text to *
```

```
+ to orderHeaderText  
result initialize local text to *
```

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

```
? when ButtonAdd .Click
do
  call addToSelectedItem quantity 1

? when ButtonSubtract .Click
do
  call addToSelectedItem quantity 1

? when ButtonOrder .Click
do
  call Notifier1 .ShowChooseDialog
  message
  title
  button1Text
  button2Text
  cancelable false
```

```
+ to addToSelectedItem quantity
do
  if ListView1 .SelectionInd...

+ to total
result initialize local total to 0...

+ to orderContentText
result initialize local text to "

+ to orderHeaderText
result initialize local text to "
```

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

```
? when ButtonAdd Click
do
  call addToSelectedItem quantity 1

? when ButtonSubtract Click
do
  call addToSelectedItem quantity -1

? when ButtonOrder Click
do
  call Notifier1 ShowChooseDialog
  message
  title
  button1Text
  button2Text "cancel"
  cancelable false
```

```
+ to addToSelectedItem quantity
do
  if ListView1.SelectionInd...

+ to total
result initialize local total to 0...

+ to orderContentText
result initialize local text to "

+ to orderHeaderText
result initialize local text to "
```


GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

```
when ButtonAdd Click
do
  call addToSelectedItem quantity 1

when ButtonSubtract Click
do
  call addToSelectedItem quantity -1

when ButtonOrder Click
do
  call Notifier1 ShowChooseDialog
  message
  title
  button1Text confirm
  button2Text cancel
  cancelable false
```

```
to addToSelectedItem quantity
do
  if ListView1 SelectionInd...

to total
result initialize local total to 0...

to orderContentText
result initialize local text to "

to orderHeaderText
result initialize local text to "
```

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

The image shows three Scratch code blocks. The first block is a 'when ButtonAdd Click' block with a 'do' block containing 'call addToSelectedItem quantity' and a '1' block. The second block is a 'when ButtonSubtract Click' block with a 'do' block containing 'call addToSelectedItem quantity' and a '-1' block. The third block is a 'when ButtonOrder Click' block with a 'do' block containing 'call Notifier1 ShowChooseDialog' and several property blocks: 'message' (empty), 'title' ('order'), 'button1Text' ('confirm'), 'button2Text' ('cancel'), and 'cancelable' (false).

The image shows four Scratch code blocks. The first block is a 'to addToSelectedItem quantity' block with a 'do' block containing 'if ListView1 SelectionInd...'. The second block is a 'to total' block with a 'result' block containing 'initialize local total to 0...'. The third block is a 'to orderContentText' block with a 'result' block containing 'initialize local text to...'. The fourth block is a 'to orderHeaderText' block with a 'result' block containing 'initialize local text to...'. The text in the last two blocks is partially obscured by a red bar at the bottom of the slide.

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

The image shows three Scratch code blocks. The first block is a 'when ButtonAdd Click' block containing a 'do' block with 'call addToSelectedItem quantity' and a '1' block. The second block is a 'when ButtonSubtract Click' block containing a 'do' block with 'call addToSelectedItem quantity' and a '-1' block. The third block is a 'when ButtonOrder Click' block containing a 'do' block with 'call Notifier1 ShowChooseDialog'. This block has several properties: 'message' is 'join', 'title' is 'order', 'button1Text' is 'confirm', 'button2Text' is 'cancel', and 'cancelable' is 'false'.

The image shows four Scratch code blocks. The first block is a 'to addToSelectedItem quantity' block containing a 'do' block with 'if ListView1 SelectionInd...'. The second block is a 'to total' block containing a 'result' block with 'initialize local total to 0...'. The third block is a 'to orderContentText' block containing a 'result' block with 'initialize local text to...'. The fourth block is a 'to orderHeaderText' block containing a 'result' block with 'initialize local text to...'. The text in the last two blocks is partially obscured by the red footer bar.

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

```
? when ButtonAdd Click
do
  call addToSelectedItem quantity 1
```

```
? when ButtonSubtract Click
do
  call addToSelectedItem quantity -1
```

```
? when ButtonOrder Click
do
  call Notifier1 ShowChooseDialog
  message join call orderHeaderText
  title "order"
  button1Text "confirm"
  button2Text "cancel"
  cancelable false
```

```
+ to addToSelectedItem quantity
do
  if ListView1_SelectionInd...
```

```
+ to total
result initialize local total to 0...
```

```
+ to orderContentText
result initialize local text to "
```

```
+ to orderHeaderText
result initialize local text to "
```

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

```
? when ButtonAdd Click
do
  call addToSelectedItem quantity 1
```

```
? when ButtonSubtract Click
do
  call addToSelectedItem quantity -1
```

```
? when ButtonOrder Click
do
  call Notifier1 ShowChooseDialog
  message join call orderHeaderText
              call orderContentText
  title "order"
  button1Text "confirm"
  button2Text "cancel"
  cancelable false
```

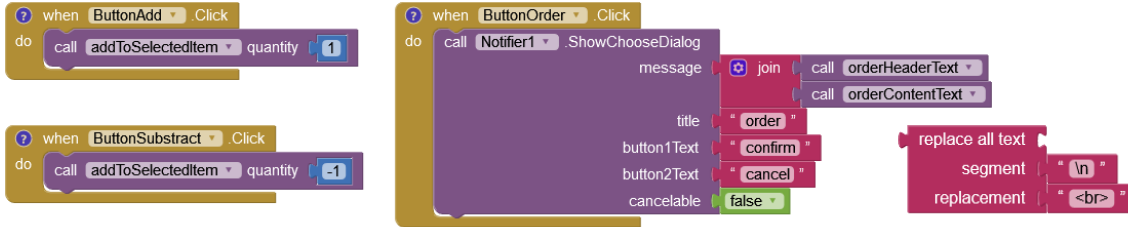
```
+ to addToSelectedItem quantity
do
  if ListView1 SelectionInd...
```

```
+ to total
result initialize local total to 0...
```

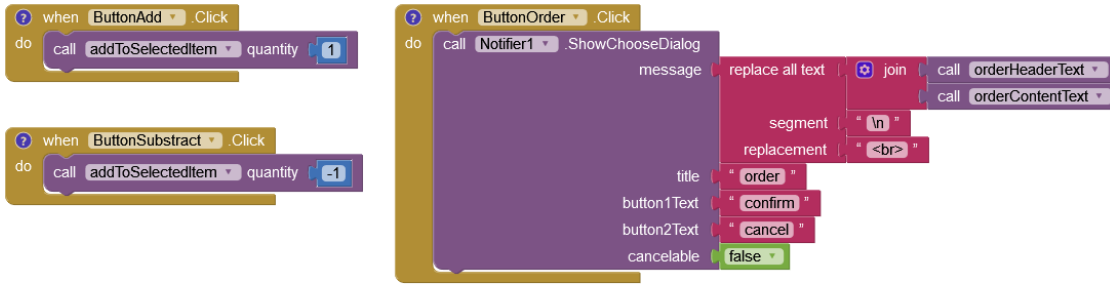
```
+ to orderContentText
result initialize local text to "
```

```
+ to orderHeaderText
result initialize local text to "
```

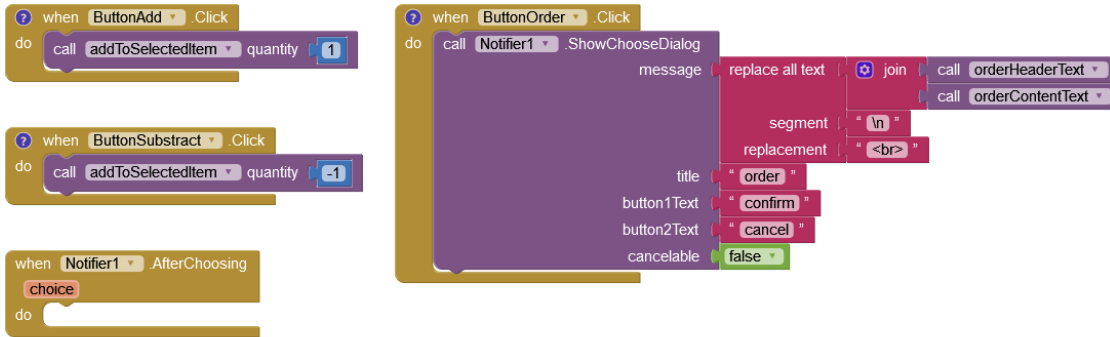
GITSHARE3b : BUTTONS, SUBSTRACT, ORDER



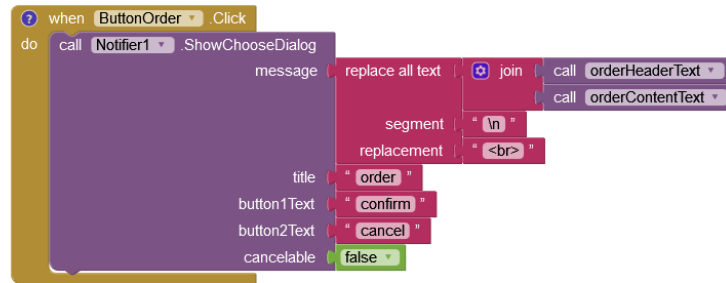
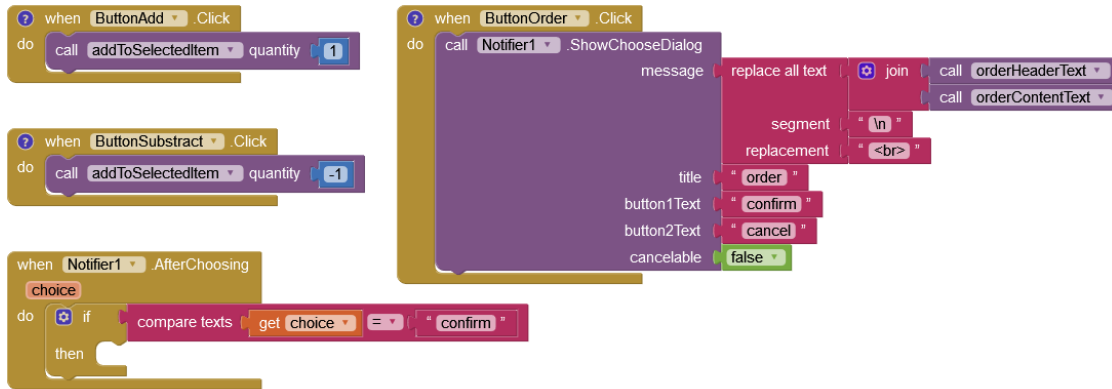
GITSHARE3b : BUTTONS, SUBSTRACT, ORDER



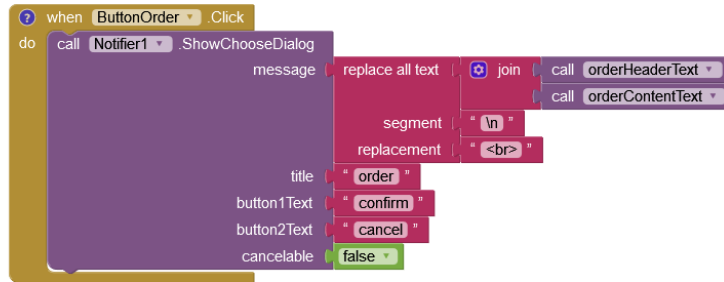
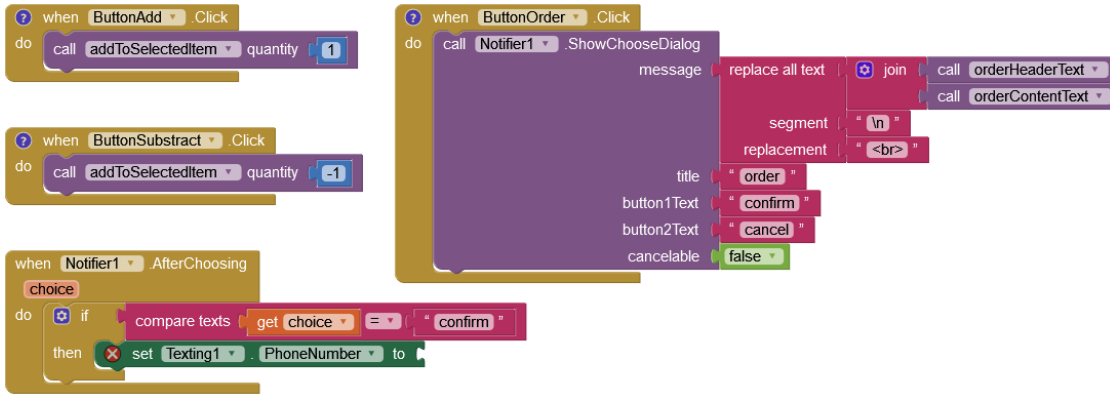
GITSHARE3b : BUTTONS, SUBSTRACT, ORDER



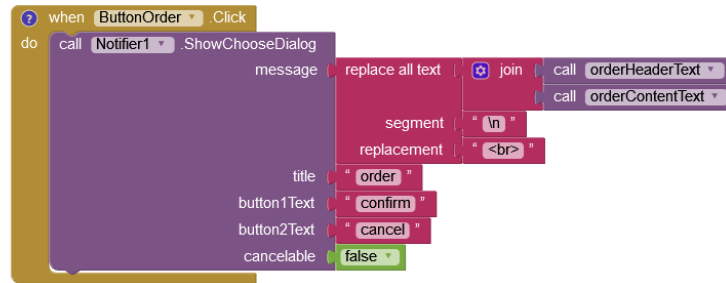
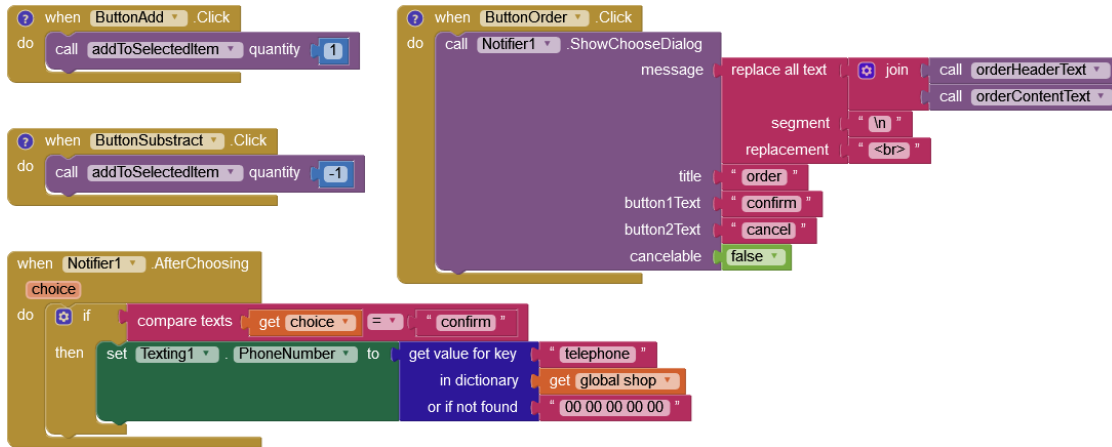
GITSHARE3b : BUTTONS, SUBSTRACT, ORDER



GITSHARE3b : BUTTONS, SUBSTRACT, ORDER



GITSHARE3b : BUTTONS, SUBSTRACT, ORDER



GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

```
when ButtonAdd Click
do
  call addToSelectedItem quantity 1

when ButtonSubtract Click
do
  call addToSelectedItem quantity -1

when Notifier1 AfterChoosing
choice
do
  if compare texts get choice = confirm
  then
    set Texting1 PhoneNumber to

when ButtonOrder Click
do
  call Notifier1.ShowChooseDialog
  message replace all text join call orderHeaderText
  segment "
  replacement "<br>"
  title "order"
  button1Text "confirm"
  button2Text "cancel"
  cancelable false
```

```
get value for key "telephone"
in dictionary get global shop
or if not found "00 00 00 00 00"
```

```
to addToSelectedItem quantity
do
  if ListView1.SelectionInd...

to total
result initialize local total to 0...

to orderContentText
result initialize local text to "

to orderHeaderText
result initialize local text to "
```

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER



GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

The image displays several Scratch code blocks for a shopping application:

- ButtonAdd Click:** A 'when ButtonAdd Click' block containing a 'do' block with 'call addToSelectedItem' and 'quantity' set to '1'.
- ButtonSubtract Click:** A 'when ButtonSubtract Click' block containing a 'do' block with 'call addToSelectedItem' and 'quantity' set to '-1'.
- Notifier1 AfterChoosing:** A 'when Notifier1 AfterChoosing' block with a 'choice' block. It contains an 'if' block comparing 'get choice' to 'confirm'. If true, it sets 'Texting1 . PhoneNumber' to a value. It also includes an 'if' block calling 'TinyDB1 . GetValue' with tag 'debug' and 'valueIfTagNotThere' set to 'true'.
- ButtonOrder Click:** A 'when ButtonOrder Click' block containing a 'do' block with 'call Notifier1 . ShowChooseDialog'. The dialog box configuration includes:
 - message: 'replace all text' joined with 'call orderHeaderText' and 'call orderContentText', with a segment of '\n' and replacement of '
'.
 - title: 'order'
 - button1Text: 'confirm'
 - button2Text: 'cancel'
 - cancelable: 'false'
- Dictionary Lookup:** A block to 'get value for key' 'telephone' in dictionary 'global shop', with 'or if not found' set to '00 00 00 00 00'.
- Local Variables:** Four 'to' blocks for 'addToSelectedItem', 'total', 'orderContentText', and 'orderHeaderText', each with an 'initialize local text to' block.

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

The image shows three event-driven code blocks:

- when ButtonAdd Click**: do call addToSelectedItem quantity 1
- when ButtonSubtract Click**: do call addToSelectedItem quantity -1
- when Notifier1 AfterChoosing**: choice if compare texts get choice = confirm then set Texting1.PhoneNumber to [if call TinyDB1.GetValue tag debug valueIfTagNotThere true] then else

when ButtonOrder Click: do call Notifier1.ShowChooseDialog message replace all text join call orderHeaderText call orderContentText segment "\n" replacement "
" title "order" button1Text "confirm" button2Text "cancel" cancelable false

get value for key "telephone" in dictionary get global shop or if not found "00 00 00 00 00"

to addToSelectedItem quantity do if ListView1.SelectionInd...
to total result initialize local total to 0...
to orderContentText result initialize local text to "
to orderHeaderText result initialize local text to "

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

The image displays several Scratch code blocks for a shopping application:

- ButtonAdd Click:** A 'when ButtonAdd Click' block containing a 'do' block with 'call addToSelectedItem quantity 1'.
- ButtonSubtract Click:** A 'when ButtonSubtract Click' block containing a 'do' block with 'call addToSelectedItem quantity -1'.
- Notifier1 AfterChoosing:** A 'when Notifier1 AfterChoosing' block with a 'choice' block. Inside, an 'if' block compares 'get choice' to 'confirm'. If true, it sets 'Texting1 . PhoneNumber' to a value. If false, it calls 'TinyDB1 . GetValue' with tag 'debug' and 'valueIfTagNotThere true'. Below this, another 'call TinyDB1 . GetValue' block with tag 'user' and 'valueIfTagNotThere create empty dictionary'.
- ButtonOrder Click:** A 'when ButtonOrder Click' block with a 'do' block. It calls 'Notifier1 . ShowChooseDialog' with a 'message' block (joining 'orderHeaderText' and 'orderContentText' with a '\n' segment and '
' replacement), a 'title' of 'order', 'button1Text' of 'confirm', 'button2Text' of 'cancel', and 'cancelable' set to 'false'.
- Database Lookups:** A 'get value for key "telephone" in dictionary get global shop or if not found "00 00 00 00 00"' block.
- Initialization:** Four 'to' blocks: 'addToSelectedItem quantity', 'total', 'orderContentText', and 'orderHeaderText', each with a 'do' block 'initialize local text to...'.

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER



GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

The image displays several Scratch code blocks for a shopping application:

- when ButtonAdd Click:** A 'do' block containing 'call addToSelectedItem quantity 1'.
- when ButtonSubtract Click:** A 'do' block containing 'call addToSelectedItem quantity -1'.
- when Notifier1 AfterChoosing:** A 'choice' block with an 'if' condition 'compare texts get choice = confirm'. The 'then' branch contains a 'set Texting1 PhoneNumber to' block. The 'else' branch contains an 'if' condition 'call TinyDB1 GetValue tag debug valueIfTagNotThere true'. Inside this 'if' block, there is a 'then' branch with 'get value for key telephone in dictionary' and 'call TinyDB1 GetValue tag user valueIfTagNotThere create empty dictionary'. The 'or if not found' block contains '00 00 00 00 00'. The 'else' branch of the outer 'if' contains 'get value for key telephone in dictionary get global shop or if not found 00 00 00 00 00'.
- when ButtonOrder Click:** A 'do' block containing 'call Notifier1 ShowChooseDialog'. The dialog box configuration includes:
 - message: 'replace all text join call orderHeaderText call orderContentText segment "\n" replacement "
"'
 - title: 'order'
 - button1Text: 'confirm'
 - button2Text: 'cancel'
 - cancelable: 'false'
- to addToSelectedItem quantity:** A 'do' block with 'if ListView1 SelectionInd...'.
- to total:** A 'result' block with 'initialize local total to 0...'.
- to orderContentText:** A 'result' block with 'initialize local text to ...'.
- to orderHeaderText:** A 'result' block with 'initialize local text to ...'.

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

The image displays several Scratch code blocks for a shopping application:

- when ButtonAdd Click:** A 'do' block containing a 'call addToSelectedItem' block with 'quantity' set to 1.
- when ButtonSubtract Click:** A 'do' block containing a 'call addToSelectedItem' block with 'quantity' set to -1.
- when Notifier1 AfterChoosing:** A 'choice' block with an 'if' condition 'compare texts' set to 'confirm'.
 - then:** A 'set Texting1.PhoneNumber to' block followed by a large green 'do' block.
 - else:** A 'get value for key' block for 'telephone' in a dictionary, with an 'or if not found' block set to '00 00 00 00 00'.
- when ButtonOrder Click:** A 'do' block containing a 'call Notifier1.ShowChooseDialog' block with the following properties:
 - message: 'replace all text' joined with 'call orderHeaderText' and 'call orderContentText'.
 - segment: '\n'
 - replacement: '
'
 - title: 'order'
 - button1Text: 'confirm'
 - button2Text: 'cancel'
 - cancelable: 'false'
- if TinyDB1.GetValue tag 'debug' valueIfTagNotThere true:** A block containing:
 - then: 'get value for key' 'telephone' in dictionary, followed by 'call TinyDB1.GetValue' with tag 'user' and 'valueIfTagNotThere' 'create empty dictionary'.
 - or if not found: '00 00 00 00 00'
- else:** 'get value for key' 'telephone' in dictionary, followed by 'get global shop' and 'or if not found' '00 00 00 00 00'.

Additional blocks at the bottom include:

- to addToSelectedItem quantity:** A 'do' block with 'if ListView1.SelectionInd...'.
- to total:** A 'result' block with 'initialize local total to 0...'.
- to orderContentText:** A 'result' block with 'initialize local text to ...'.
- to orderHeaderText:** A 'result' block with 'initialize local text to ...'.

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

The image displays several Scratch code blocks for a shopping application:

- ButtonAdd Click:** A 'when ButtonAdd Click' block containing a 'do' block with 'call addToSelectedItem quantity' and a value of '1'.
- ButtonSubtract Click:** A 'when ButtonSubtract Click' block containing a 'do' block with 'call addToSelectedItem quantity' and a value of '-1'.
- Notifier1 AfterChoosing:** A 'when Notifier1 AfterChoosing' block with a 'choice' block. Inside, an 'if' block compares 'get choice' to 'confirm'.
 - Then branch:** 'set Texting1 . PhoneNumber to' followed by a large green block.
 - Else branch:** 'set Texting1 . Message to' followed by a 'join' block.
- Notifier1 ShowChooseDialog:** A 'when Notifier1 ShowChooseDialog' block with a 'do' block containing:
 - 'message' block: 'replace all text' (join), 'call orderHeaderText', 'call orderContentText', 'segment' ('\n'), 'replacement' ('
').
 - 'title' block: 'order'.
 - 'button1Text' block: 'confirm'.
 - 'button2Text' block: 'cancel'.
 - 'cancelable' block: 'false'.
- Logic for 'confirm' choice:** An 'if' block with 'call TinyDB1 . GetValue tag "debug" valueIfTagNotThere true'.
 - Then:** 'get value for key "telephone" in dictionary', 'call TinyDB1 . GetValue tag "user" valueIfTagNotThere create empty dictionary', 'or if not found "00 00 00 00 00"'. This is followed by 'get value for key "telephone" in dictionary', 'get global shop', and 'or if not found "00 00 00 00 00"'. The result is joined to the message.
 - Else:** 'get value for key "telephone" in dictionary', 'get global shop', and 'or if not found "00 00 00 00 00"'. The result is joined to the message.
- Global Variables:** 'addToSelectedItem quantity' (if ListView1_SelectionInd...), 'total' (initialize local total to 0...), 'orderContentText' (initialize local text to "..."), and 'orderHeaderText' (initialize local text to "...").

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

The image displays several Scratch code blocks for a shopping application:

- ButtonAdd Click:** A 'when ButtonAdd Click' block containing a 'do' block with 'call addToSelectedItem quantity' and a value of '1'.
- ButtonSubtract Click:** A 'when ButtonSubtract Click' block containing a 'do' block with 'call addToSelectedItem quantity' and a value of '-1'.
- Notifier1 ShowChooseDialog:** A 'when Notifier1 ShowChooseDialog' block with a 'do' block containing:
 - 'message' block: 'replace all text' (with a 'join' block), 'call orderHeaderText', and 'call orderContentText'.
 - 'segment' block: '"\n"'
 - 'replacement' block: '"
"'
 - 'title' block: '"order"'
 - 'button1Text' block: '"confirm"'
 - 'button2Text' block: '"cancel"'
 - 'cancelable' block: 'false'.
- Notifier1 AfterChoosing:** A 'when Notifier1 AfterChoosing' block with a 'choice' block containing:
 - 'do' block: 'if' block with 'compare texts' (get choice, '=', 'confirm').
 - 'then' block: 'set Texting1 PhoneNumber to' block.
 - 'else' block: 'if' block with 'call TinyDB1 GetValue tag "debug" valueIfTagNotThere true'.
 - 'then' block: 'get value for key "telephone" in dictionary' (TinyDB1) and 'call TinyDB1 GetValue tag "user" valueIfTagNotThere create empty dictionary'.
 - 'or if not found' block: '"00 00 00 00 00"'
 - 'else' block: 'get value for key "telephone" in dictionary' (global shop) and 'get global shop'.
 - 'or if not found' block: '"00 00 00 00 00"'
 - 'set Texting1 Message to' block: 'join' block with 'call Clock1 FormatDateTime' (instant: 'call Clock1 Now', pattern: '"dd/MM/yyyy hh:mm:ss a"').

- Global Variables:** Four 'to' blocks at the bottom: 'addToSelectedItem quantity', 'total', 'orderContentText', and 'orderHeaderText', each with an 'initialize local text to' block.

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

The image displays several Scratch code blocks for a Click&Collect application:

- when ButtonAdd Click**: A 'do' block containing a 'call addToSelectedItem quantity' block with a value of '1'.
- when ButtonSubtract Click**: A 'do' block containing a 'call addToSelectedItem quantity' block with a value of '-1'.
- when Notifier1 AfterChoosing**: A 'choice' block containing:
 - An 'if' block with 'compare texts' set to 'confirm'.
 - 'then' branch: 'set Texting1 . PhoneNumber to' followed by a large green block.
 - 'else' branch: 'set Texting1 . Message to' followed by a 'join' block and a 'call Clock1 . FormatDateTime' block (instant: 'call Clock1 . Now', pattern: 'dd/MM/yyyy hh:mm:ss a', separator: '\n').
- A 'call TinyDB1 . GetValue' block (tag: 'debug', valueIfTagNotThere: 'true').
 - 'then' branch: 'get value for key telephone in dictionary' (TinyDB1) and 'call TinyDB1 . GetValue' (tag: 'user', valueIfTagNotThere: 'create empty dictionary').
 - 'or if not found': '00 00 00 00 00'.
 - 'else' branch: 'get value for key telephone in dictionary' (global shop) and 'or if not found': '00 00 00 00 00'.

- when ButtonOrder Click**: A 'do' block containing a 'call Notifier1 . ShowChooseDialog' block with:
- 'message': 'replace all text' (join), 'call orderHeaderText', 'call orderContentText', 'segment': '\n', 'replacement': '
'.
- 'title': 'order'.
- 'button1Text': 'confirm'.
- 'button2Text': 'cancel'.
- 'cancelable': 'false'.
- to addToSelectedItem quantity**: A 'do' block with 'if ListView1 . SelectionInd...'.
- to total**: A 'result' block with 'initialize local total to 0...'.
- to orderContentText**: A 'result' block with 'initialize local text to '...'.
- to orderHeaderText**: A 'result' block with 'initialize local text to '...'.

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

when ButtonAdd Click
do
call addToSelectedItem quantity 1

when ButtonSubtract Click
do
call addToSelectedItem quantity -1

when ButtonOrder Click
do
call Notifier1.ShowChooseDialog
message replace all text join call orderHeaderText call orderContentText
segment "\n"
replacement "
"
title "order"
button1Text "confirm"
button2Text "cancel"
cancelable false

when Notifier1 AfterChoosing
choice
do
if compare texts get choice == "confirm"
then
set Texting1.PhoneNumber to
if TinyDB1.GetValue tag "debug" valselfTagNotThere true
then
get value for key "telephone"
in dictionary
call TinyDB1.GetValue
tag "user"
valselfTagNotThere create empty dictionary
or if not found "00 00 00 00 00"
else
get value for key "telephone"
in dictionary
get global shop
or if not found "00 00 00 00 00"
then
set Texting1.Message to
join
call Clock1.FormatDateTime
instant call Clock1.Now
pattern "dd/MM/yyyy hh:mm:ss a"
"\n"
call orderHeaderText

to addToSelectedItem quantity
do
if ListView1.SelectionInd...

to total
result initialize local total to 0...

to orderContentText
result initialize local text to "

to orderHeaderText
result initialize local text to "

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

The image displays several Scratch code blocks for a shopping application:

- ButtonAdd Click:** A 'when ButtonAdd Click' block containing a 'do' block with 'call addToSelectedItem quantity' and a value of '1'.
- ButtonSubtract Click:** A 'when ButtonSubtract Click' block containing a 'do' block with 'call addToSelectedItem quantity' and a value of '-1'.
- Notifier1 ShowChooseDialog:** A 'when Notifier1 ShowChooseDialog' block with a 'do' block containing:
 - 'message' block: 'replace all text' (with a 'join' block), 'call orderHeaderText', and 'call orderContentText'.
 - 'segment' block: '"\n"'
 - 'replacement' block: '"
"'
 - 'title' block: '"order"'
 - 'button1Text' block: '"confirm"'
 - 'button2Text' block: '"cancel"'
 - 'cancelable' block: 'false'.
- Notifier1 AfterChoosing:** A 'when Notifier1 AfterChoosing' block with a 'choice' block and a 'do' block:
 - 'if' block: 'compare texts' (get choice, '=', 'confirm').
 - 'then' block: 'set Texting1 PhoneNumber to' followed by a large green block.
 - 'else' block: 'set Texting1 Message to' followed by a 'join' block containing:
 - 'call TinyDB1 GetValue tag "debug" valueIfTagNotThere true'.
 - 'if' block: 'get value for key "telephone" in dictionary' (call TinyDB1 GetValue, tag "user", valueIfTagNotThere create empty dictionary) or 'or if not found "00 00 00 00 00"'. 'get value for key "telephone" in dictionary' (get global shop) or 'or if not found "00 00 00 00 00"'. 'call Clock1 FormatDateTime' (instant call Clock1 Now, pattern "dd/MM/yyyy hh:mm:ss a").
 - '\n' block.
 - 'call orderHeaderText'.
 - 'call orderContentText'.
- Global Variables:** Four 'to' blocks at the bottom: 'addToSelectedItem quantity', 'total', 'orderContentText', and 'orderHeaderText', each with an 'initialize local text to' block.

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

The image displays several Scratch code blocks for a shopping application:

- when ButtonAdd Click**: A 'do' block containing 'call addToSelectedItem quantity 1'.
- when ButtonSubtract Click**: A 'do' block containing 'call addToSelectedItem quantity -1'.
- when Notifier1 AfterChoosing**: A 'choice' block with an 'if' condition 'compare texts get choice == confirm'.
 - then**: 'set Texting1 PhoneNumber to' followed by a large green block.
 - else**: 'set Texting1 Message to' followed by a 'join' block containing:
 - 'call TinyDB1 GetValue tag debug valuelTagNotThere true'.
 - 'then' block: 'get value for key telephone in dictionary TinyDB1 GetValue tag user valuelTagNotThere create empty dictionary'.
 - 'or if not found' block: '00 00 00 00 00'.
 - 'else' block: 'get value for key telephone in dictionary get global shop'.
 - 'or if not found' block: '00 00 00 00 00'.
- call Texting1 SendMessage** block.

when ButtonOrder Click: A 'do' block containing:

- 'call Notifier1 ShowChooseDialog'.
- 'message' block: 'replace all text' (join), 'segment' ('\n'), 'replacement' ('
').
- 'title' block: 'order'.
- 'button1Text' block: 'confirm'.
- 'button2Text' block: 'cancel'.
- 'cancelable' block: 'false'.

to addToSelectedItem quantity: A 'do' block with 'if ListView1 SelectionInd...'.

to total: A 'result' block with 'initialize local total to 0...'.

to orderContentText: A 'result' block with 'initialize local text to...'.

to orderHeaderText: A 'result' block with 'initialize local text to...'.

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

The image displays several Scratch code blocks for a Click&Collect application:

- when ButtonAdd Click:** A 'do' block containing 'call addToSelectedItem quantity' with a value of '1'.
- when ButtonSubtract Click:** A 'do' block containing 'call addToSelectedItem quantity' with a value of '-1'.
- when Notifier1 AfterChoosing:** A 'choice' block with an 'if' condition 'compare texts' set to 'confirm'.
 - then:** 'set Texting1 . PhoneNumber to' followed by a large green block.
 - else:** 'set Texting1 . Message to' followed by a 'join' block containing:
 - 'call TinyDB1 . GetValue tag "debug" valueIfTagNotThere true'.
 - 'then' block: 'get value for key "telephone" in dictionary' (TinyDB1) and 'call TinyDB1 . GetValue tag "user" valueIfTagNotThere create empty dictionary'.
 - 'or if not found' block: '"00 00 00 00 00"'.
 - 'else' block: 'get value for key "telephone" in dictionary' (global shop) and 'get global shop'.
 - 'or if not found' block: '"00 00 00 00 00"'.
 - 'call Clock1 . FormatDateTime' with 'instant' as 'call Clock1 . Now' and 'pattern' as '"dd/MM/yyyy hh:mm:ss a"'.
 - '\n' block.
 - 'call orderHeaderText'.
 - 'call orderContentText'.
- when Notifier1 ShowChooseDialog:** A 'message' block with:
 - 'replace all text' block joined with 'call orderHeaderText' and 'call orderContentText'.
 - 'segment' block: '"\n"'.
 - 'replacement' block: '"
"'.
 - 'title' block: '"order"'.
 - 'button1Text' block: '"confirm"'.
 - 'button2Text' block: '"cancel"'.
 - 'cancelable' block: 'false'.

- to addToSelectedItem quantity:** A 'do' block with 'if ListView1 . SelectionInd...'.
- to total:** A 'result' block with 'initialize local total to 0...'.
- to orderContentText:** A 'result' block with 'initialize local text to...'.
- to orderHeaderText:** A 'result' block with 'initialize local text to...'.

GITSHARE3b : BUTTONS, SUBSTRACT, ORDER

The image displays several Scratch code blocks for a Click&Collect application:

- when ButtonAdd Click**: A block that calls `addToSelectedItem` with the parameter `quantity` set to `1`.
- when ButtonSubtract Click**: A block that calls `addToSelectedItem` with the parameter `quantity` set to `-1`.
- when Notifier1 AfterChoosing**: A large block containing a `choice` block and an `if` block. The `if` block compares the `choice` to `"confirm"`. If true, it sets `Texting1` `PhoneNumber` to a value from `TinyDB1` (tag `debug`, `valueIfTagNotThere` `true`), then gets `telephone` from the dictionary. If false, it gets `telephone` from the `global shop` dictionary. Both paths then format a date and time using `Clock1` and join it with `orderHeaderText` and `orderContentText` before sending a message to `Texting1` and closing the screen.
- when ButtonOrder Click**: A block that calls `Notifier1` `ShowChooseDialog` with a `message` block (joining `orderHeaderText` and `orderContentText` with a `"\n"` segment and replacing `"\n"` with `"
"`), a `title` of `"order"`, `button1Text` of `"confirm"`, `button2Text` of `"cancel"`, and `cancelable` set to `false`.
- to addToSelectedItem quantity**: A block that checks `if ListView1` `SelectionInd...`.
- to total**: A block that initializes a local `total` to `0...`.
- to orderContentText**: A block that initializes local `text` to `"`.
- to orderHeaderText**: A block that initializes local `text` to `"`.

GITSHARE 3b : SHOP SCREEN

Display & select algorithm

The code for the 'Display & select algorithm' is organized into several sections:

- Initialization:** Three blocks at the top initialize global variables: 'shop' to an empty dictionary, 'items' to an empty list, and 'shopURL' to a string.
- shop.Initialize:** A 'when' block that sets 'global shopURL' to 'get start value', 'Web1.Uri' to 'get global shopURL', 'Web1.SaveResponse' to 'false', and calls 'Web1.Get'.
- Web1.GotText:** A large 'when' block that checks 'responseCode' for '200'. If successful, it decodes the response into a dictionary, extracts 'title', 'address', and 'image' (using 'goodURL'), and retrieves 'items' from the dictionary. It then sets 'global items' and calls 'setListViewElements' with 'myItems'. If there's an error, it shows a message.
- Web1.ErrorOccurred:** A 'when' block that handles errors by logging the component, function name, error number, and message.

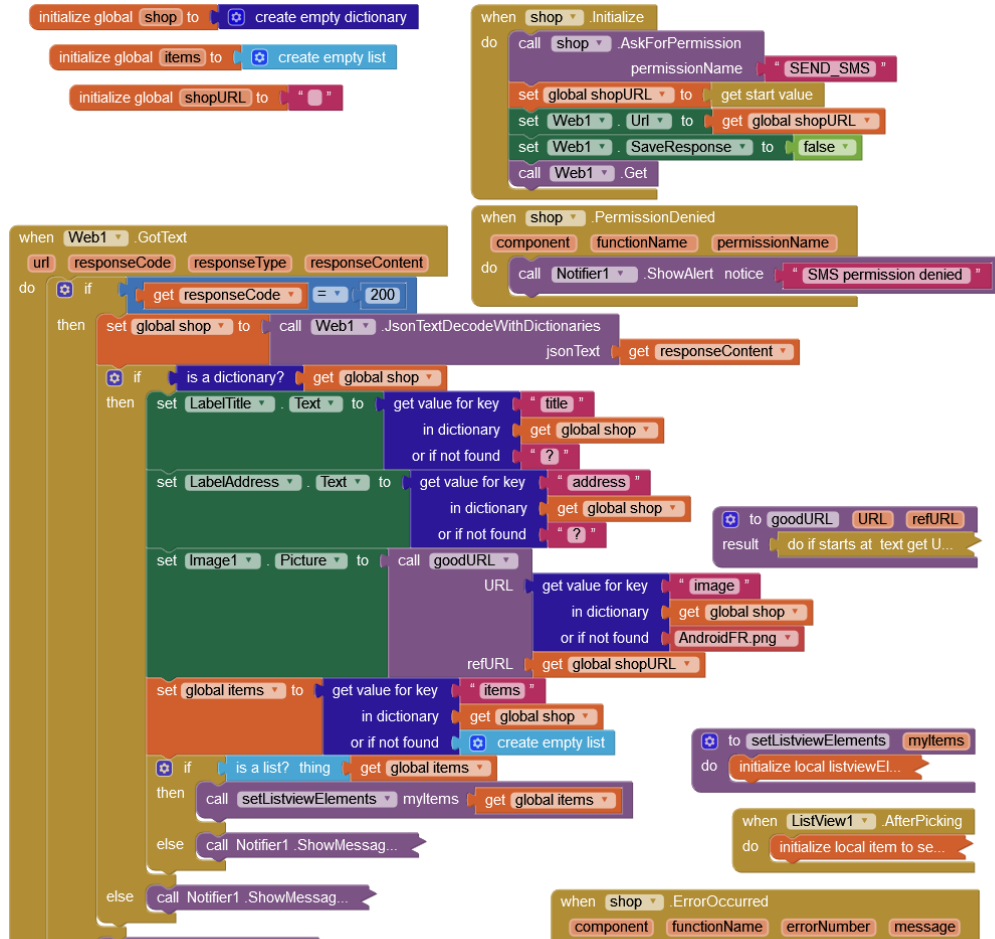
Order algorithm (3b)

The code for the 'Order algorithm (3b)' is organized into several sections:

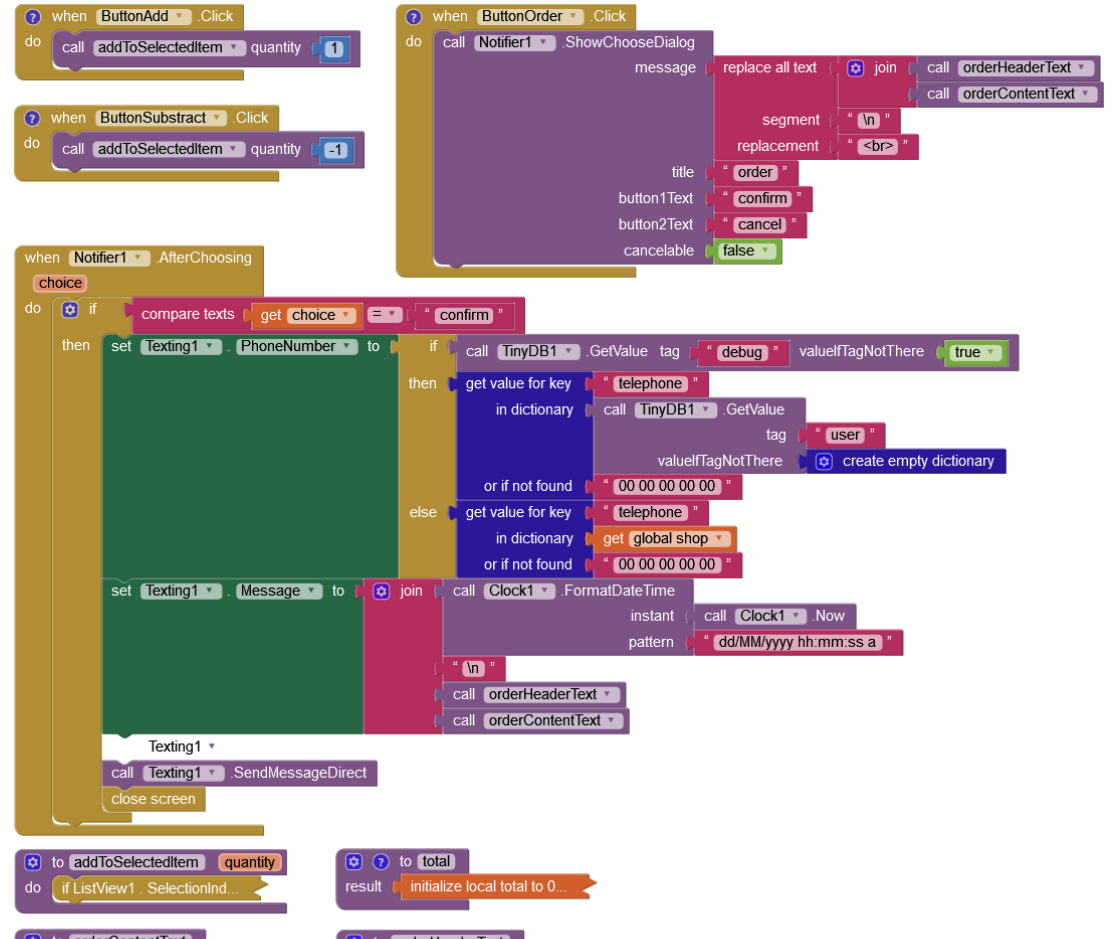
- ButtonAdd.Click:** A 'when' block that calls 'addToSelectedItem' with a quantity of 1.
- ButtonSubtract.Click:** A 'when' block that calls 'addToSelectedItem' with a quantity of -1.
- ButtonOrder.Click:** A 'when' block that calls 'Notifier1.ShowChooseDialog' with a message containing 'orderHeaderText' and 'orderContentText', a title of 'order', and buttons for 'confirm' and 'cancel'.
- Notifier1.AfterChoosing:** A 'when' block that checks if the user chose 'confirm'. It then sets 'Texting1.PhoneNumber' based on a 'TinyDB1' value for 'telephone'. It also sets 'Texting1.Message' with a date and the order content. Finally, it calls 'Texting1.SendMessageDirect' and 'close screen'.
- addToSelectedItem:** A block that updates the 'total' variable based on the selected item's quantity.

GITSHARE 3b : SHOP SCREEN

Display & select algorithm



Order algorithm (3b)

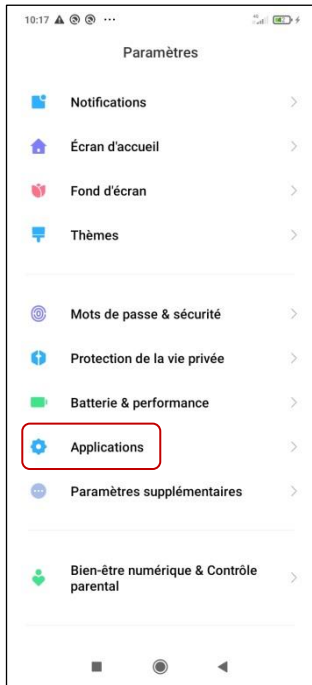




GITSHARE 3B SETUP : AUTHORIZE SMS/TEXTING

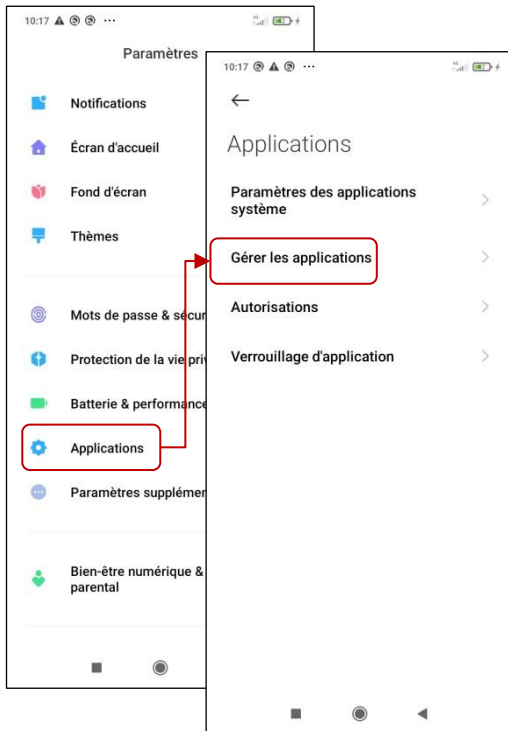


GITSHARE 3B SETUP : AUTHORIZE SMS/TEXTING



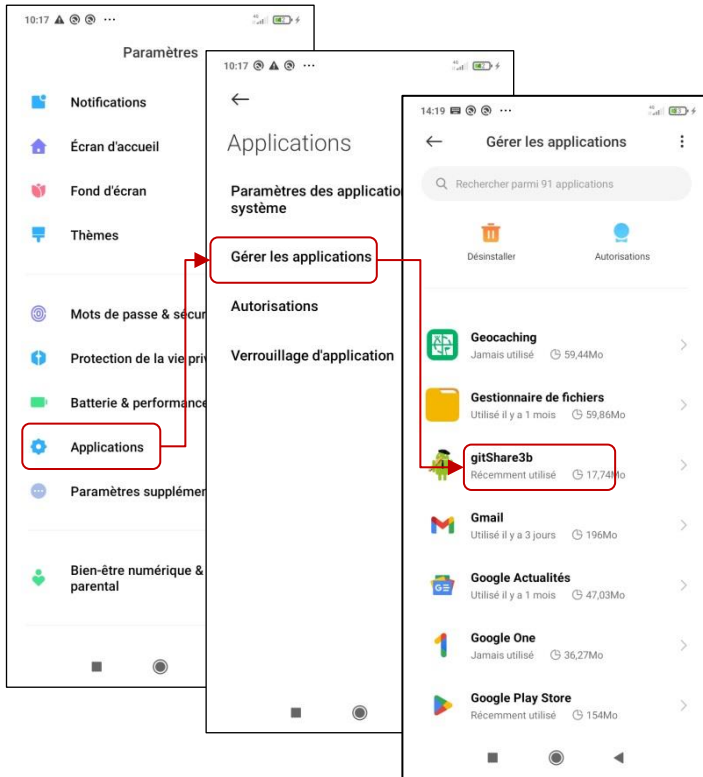


GITSHARE 3B SETUP : AUTHORIZE SMS/TEXTING



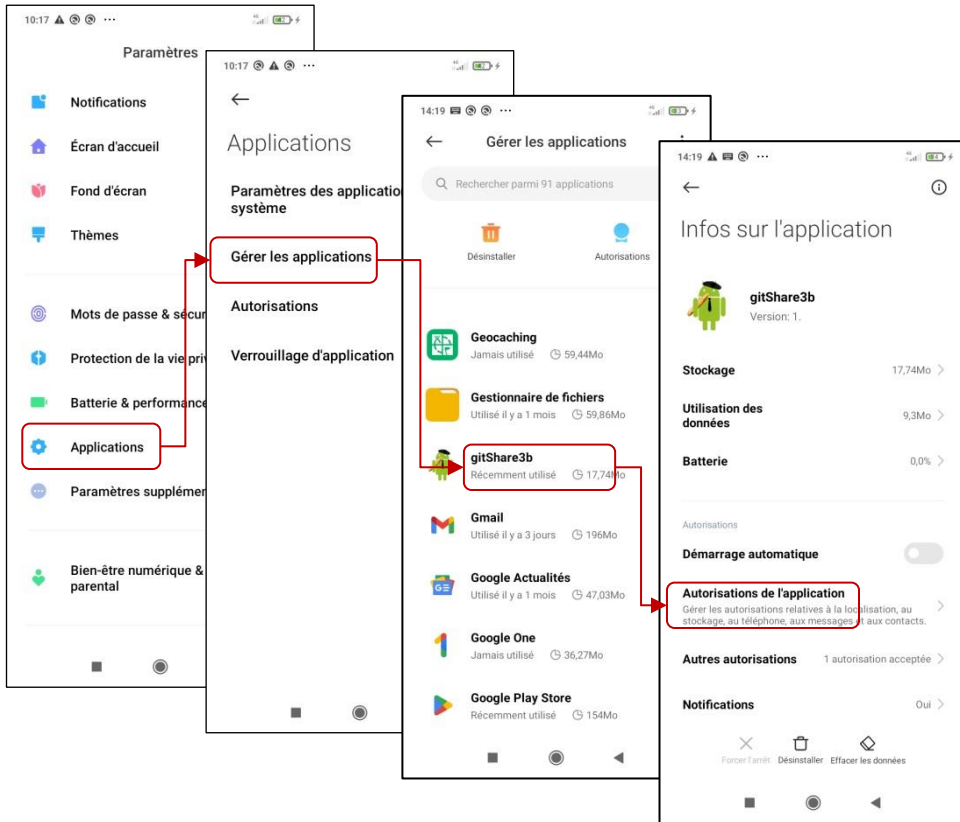


GITSHARE 3B SETUP : AUTHORIZE SMS/TEXTING



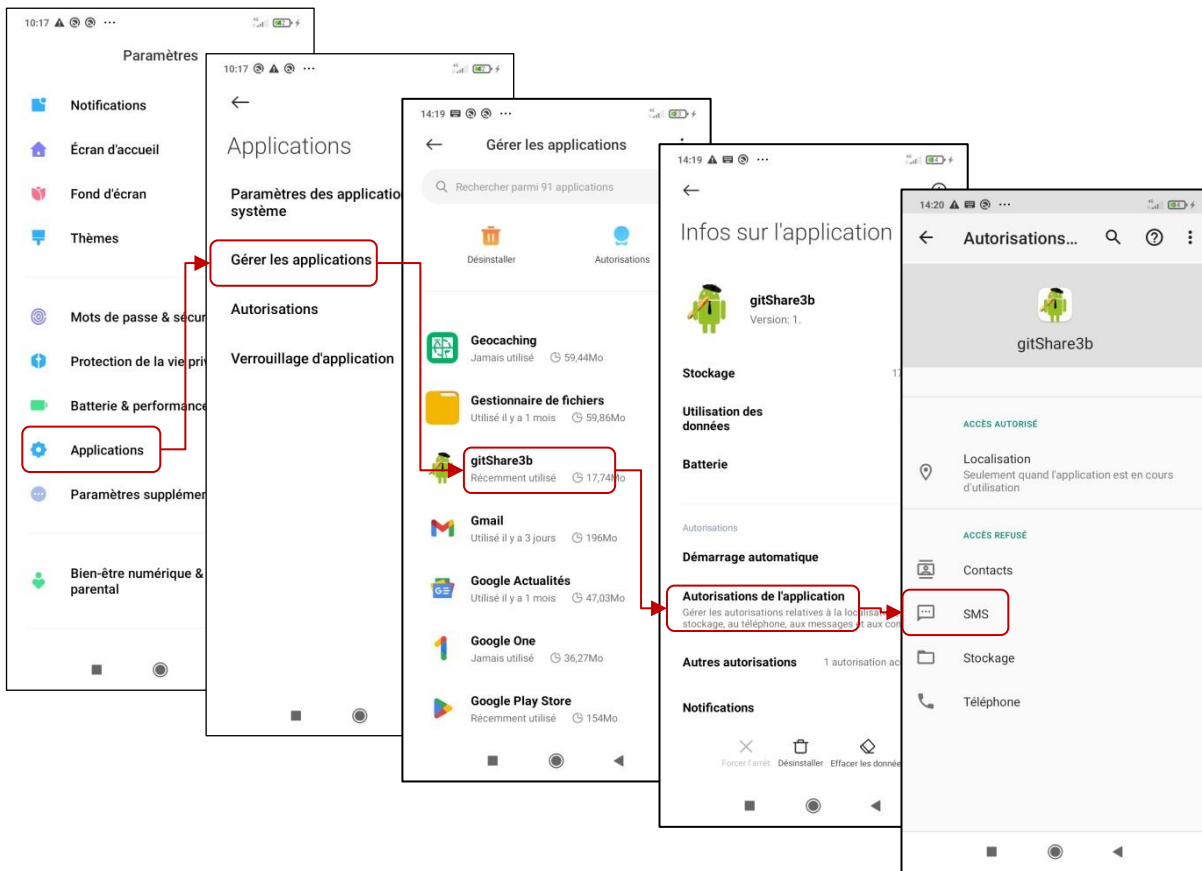


GITSHARE 3B SETUP : AUTHORIZE SMS/TEXTING



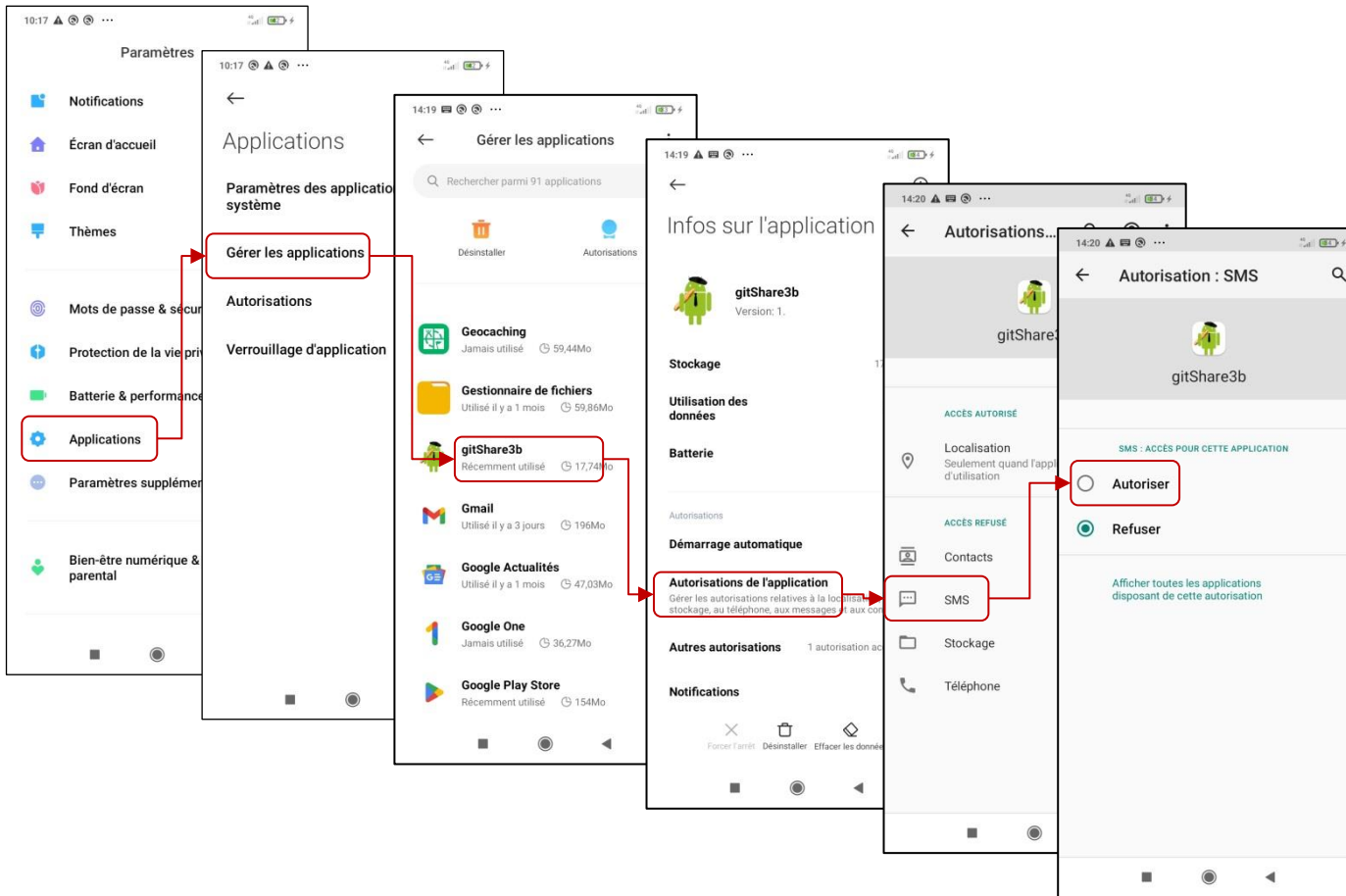


GITSHARE 3B SETUP : AUTHORIZE SMS/TEXTING



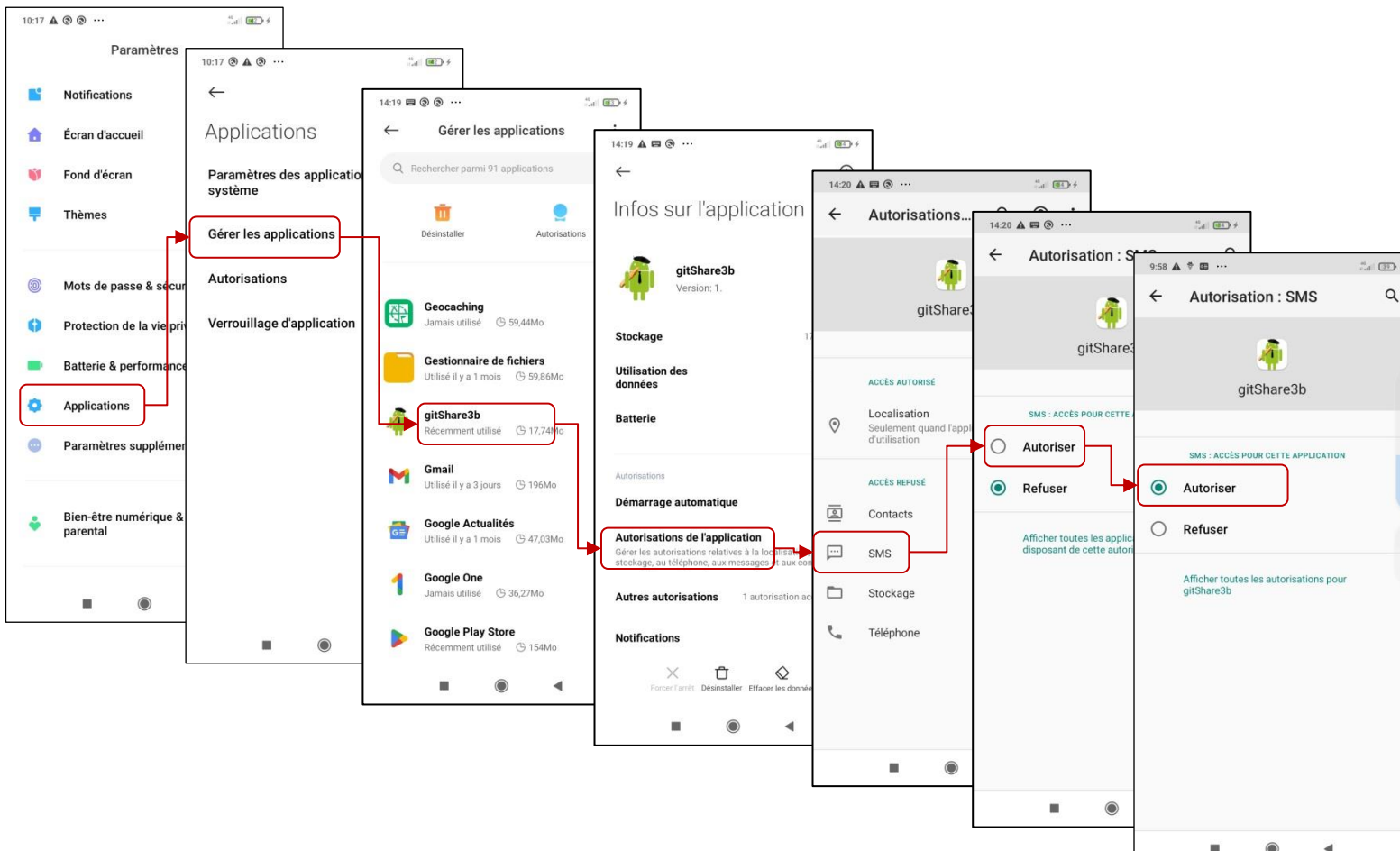


GITSHARE 3B SETUP : AUTHORIZE SMS/TEXTING





GITSHARE 3B SETUP : AUTHORIZE SMS/TEXTING





GITSHARE 3B SETUP : CHECK TELEPHONE NUMBER

```
when ButtonStart .Click
do
  if
    and
      not is empty TextBoxName .Text
      not is empty TextBoxAddress .Text
      not is empty TextBoxEmail .Text
      not is empty TextBoxTelephone .Text
      not is empty TextBoxCatalog .Text
  then
    if
      and
        compare texts ButtonStart .Text = " Start "
        compare texts TextBoxTelephone .Text =
          get global checkNumber
    then
      initialize local user to
        make a dictionary
          key " name "
          value TextBoxName .Text
          key " address "
          value TextBoxAddress .Text
          key " email "
          value TextBoxEmail .Text
          key " telephone "
          value TextBoxTelephone .Text
          key " catalog "
          value TextBoxCatalog .Text
      in
        call TinyDB1 .StoreValue tag " user " valueToStore get user
      open another screen with start valuescreenName mapScreen
        startValue
          call goodURL
            URL TextBoxCatalog .Text
            rootURL ""
    else
      call checkTelephone
  else
    call Notifier1 .ShowAlert notice " missing name, email, address, telephone or catalog "
```



GITSHARE 3B SETUP : CHECK TELEPHONE NUMBER

The image displays two Scratch code blocks. The left block is a 'when ButtonStart . Click' event handler. It contains an 'if' block with an 'and' condition checking if five text boxes (Name, Address, Email, Telephone, Catalog) are not empty. If all are filled, it checks if the ButtonStart text is 'Start' and if the Telephone text matches a global variable 'checkNumber'. If both are true, it initializes a local 'user' dictionary with fields for name, address, email, telephone, and catalog, stores it in TinyDB1, and maps to a 'mapScreen' screen. If the conditions are not met, it calls a 'checkTelephone' block. The right block is the 'checkTelephone' block, which sets ButtonStart text to 'check telephone', changes its background color to yellow, shows an alert 'téléphone SMS check', sets Texting1's phone number to the Telephone text, sets a global 'secret' to a random integer from 1 to 9999, sets Texting1's message to the global secret, sets its receiving state to 'Foreground', and finally calls Texting1's 'SendMessageDirect' block. A red box highlights the 'SendMessageDirect' block in the right code block, and a red arrow points from this box to the 'call checkTelephone' block in the left code block.



GITSHARE 3B SETUP : CHECK TELEPHONE NUMBER

```
when ButtonStart . Click
do
  if
    and
      not is empty TextBoxName . Text
      not is empty TextBoxAddress . Text
      not is empty TextBoxEmail . Text
      not is empty TextBoxTelephone . Text
      not is empty TextBoxCatalog . Text
  then
    if
      and
        compare texts ButtonStart . Text = " Start "
        compare texts TextBoxTelephone . Text = get global checkNumber
    then
      initialize local user to
        make a dictionary
          key " name " value TextBoxName . Text
          key " address " value TextBoxAddress . Text
          key " email " value TextBoxEmail . Text
          key " telephone " value TextBoxTelephone . Text
          key " catalog " value TextBoxCatalog . Text
      in call TinyDB1 . StoreValue tag " user " valueToStore get user
      open another screen with start valuescreenName mapScreen
        startValue call goodURL
          URL TextBoxCatalog . Text
          rootURL " "
    else
      call checkTelephone
  else
    call Notifier1 . ShowAlert notice " missing name, email, address, telephone or catalog "
```

```
initialize global secret to 1
initialize global checkNumber to " "

to checkTelephone
do
  set ButtonStart . Text to " check telephone "
  set ButtonStart . BackgroundColor to yellow
  call Notifier1 . ShowAlert notice " téléphone SMS check "
  set Texting1 . PhoneNumber to TextBoxTelephone . Text
  set global secret to random integer from 1 to 9999
  set Texting1 . Message to get global secret
  set Texting1 . ReceivingEnabled to ReceivingState Foreground
  call Texting1 . SendMessageDirect
```

```
when Texting1 . MessageReceived
number messageText
do
  set Texting1 . ReceivingEnabled to ReceivingState Off
  if
    compare texts get messageText = get global secret
  then
    set TextBoxTelephone . Text to get number
    set global checkNumber to TextBoxTelephone . Text
    set ButtonStart . Text to " Start "
    set ButtonStart . BackgroundColor to green
  else
    call Notifier1 . ShowAlert notice " telephone check KO "
```

GITSHARE : NOTIONS MISES EN OEUVRE

- analyse et définition de données structurées avec JSON

```
{
  "title": "Café Luna",
  "description": "Pizzeria",
  "address": "612 Main St, Cambridge, MA 02139, États-Unis",
  "telephone": "0600000002",
  "image": "image.jpg",
  "items": [
    {
      "name": "wings and waffles",
      "price": 16,
      "description": "a double waffle with marinated chicken wings tossed in mango habanero",
      "ingredients": ["chicken wings", "Waffles"],
      "image": "wingsAndWaffles.PNG"
    },
    {
      "name": "grilled honey butter biscuits",
      "price": 15,
      "description": "house made fig jam, Lemon curd or mapple-bacon honey butter",
      "ingredients": ["jam", "mapple bacon", "honey"],
      "image": "grilledHoneyButter.PNG"
    }
  ]
}
```

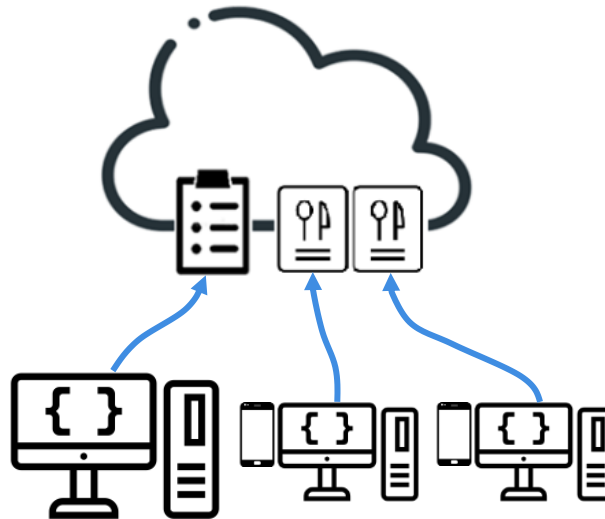
GITSHARE : NOTIONS MISES EN OEUVRE

- analyse et définition de données structurées avec JSON et de données géo-localisées avec geoJSON,

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "title": "Songsan",
        "description": "https://onvaessayer.github.io/gitshareData2/songSan/restaurant.json",
        "fill": "#00FF00",
        "image": "korean.png"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [2.29935, 48.836747]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "title": "le relais",
        "description": "https://onvaessayer.github.io/gitshareData2/relaisDeLaPlace/restaurant.json",
        "fill": "#0000FF",
        "image": "frenchFood.png"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [2.306638, 48.836606]
      }
    }
  ]
}
```

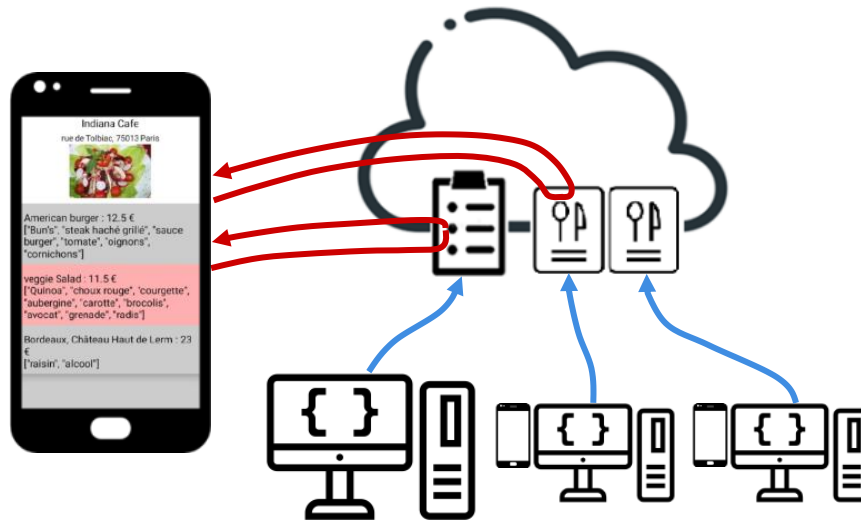
GITSHARE : NOTIONS MISES EN OEUVRE

- analyse et définition de données structurées avec JSON et de données géo-localisées avec geoJSON,
- partage de données sur Internet avec github, dropbox ou drive,



GITSHARE : NOTIONS MISES EN OEUVRE

- analyse et définition de données structurées avec JSON et de données géo-localisées avec geoJSON,
- partage de données sur Internet avec github, dropbox ou drive, lecture et décodage depuis un smartphone,



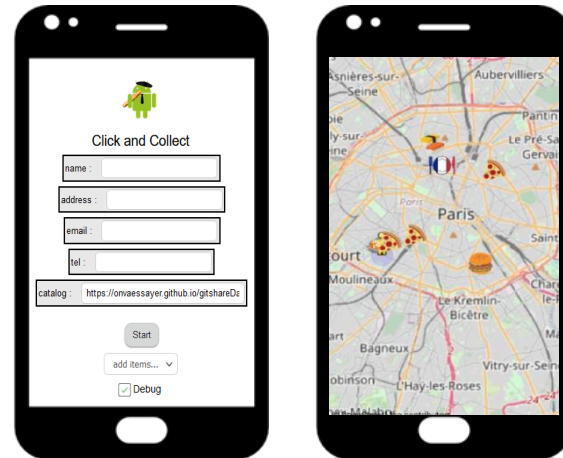
GITSHARE : NOTIONS MISES EN OEUVRE

- analyse et définition de données structurées avec JSON et de données géo-localisées avec geoJSON,
- partage de données sur Internet avec github, dropbox ou drive, lecture et décodage depuis un smartphone,
- définition et l'enregistrement du profil utilisateur,



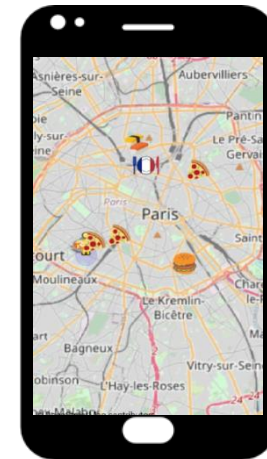
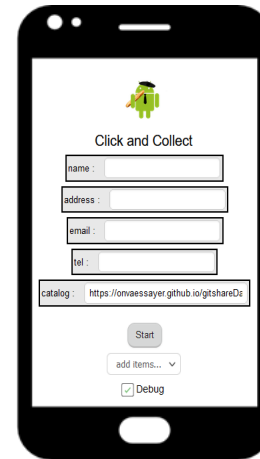
GITSHARE : NOTIONS MISES EN OEUVRE

- analyse et définition de données structurées avec JSON et de données géo-localisées avec geoJSON,
- partage de données sur Internet avec github, dropbox ou drive, lecture et décodage depuis un smartphone,
- définition et l'enregistrement du profil utilisateur,
- affichage et sélection de données sur une carte,



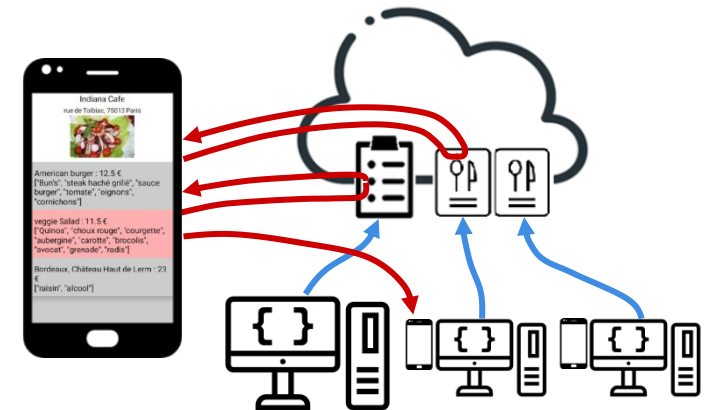
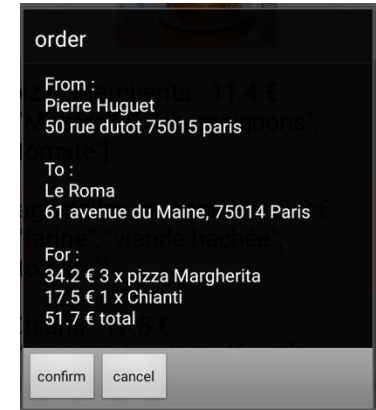
GITSHARE : NOTIONS MISES EN OEUVRE

- analyse et définition de données structurées avec JSON et de données géo-localisées avec geoJSON,
- partage de données sur Internet avec github, dropbox ou drive, lecture et décodage depuis un smartphone,
- définition et l'enregistrement du profil utilisateur,
- affichage et sélection de données sur une carte,
- affichage des produits proposés par un commerçant,



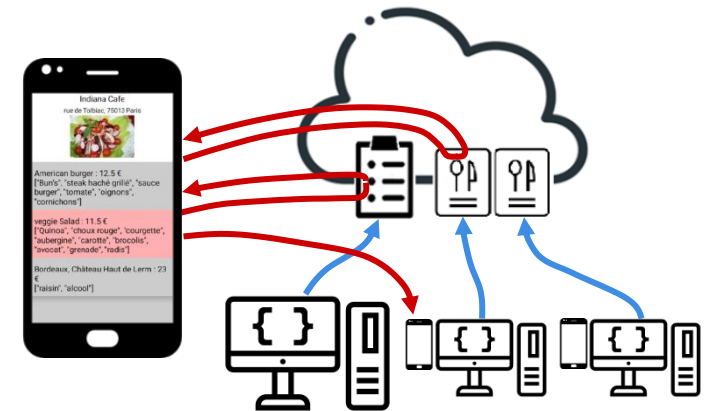
GITSHARE : NOTIONS MISES EN OEUVRE

- analyse et définition de données structurées avec JSON et de données géo-localisées avec geoJSON,
- partage de données sur Internet avec github, dropbox ou drive, lecture et décodage depuis un smartphone,
- définition et l'enregistrement du profil utilisateur,
- affichage et sélection de données sur une carte,
- affichage des produits proposés par un commerçant,
- Préparation et envoi de commandes par SMS



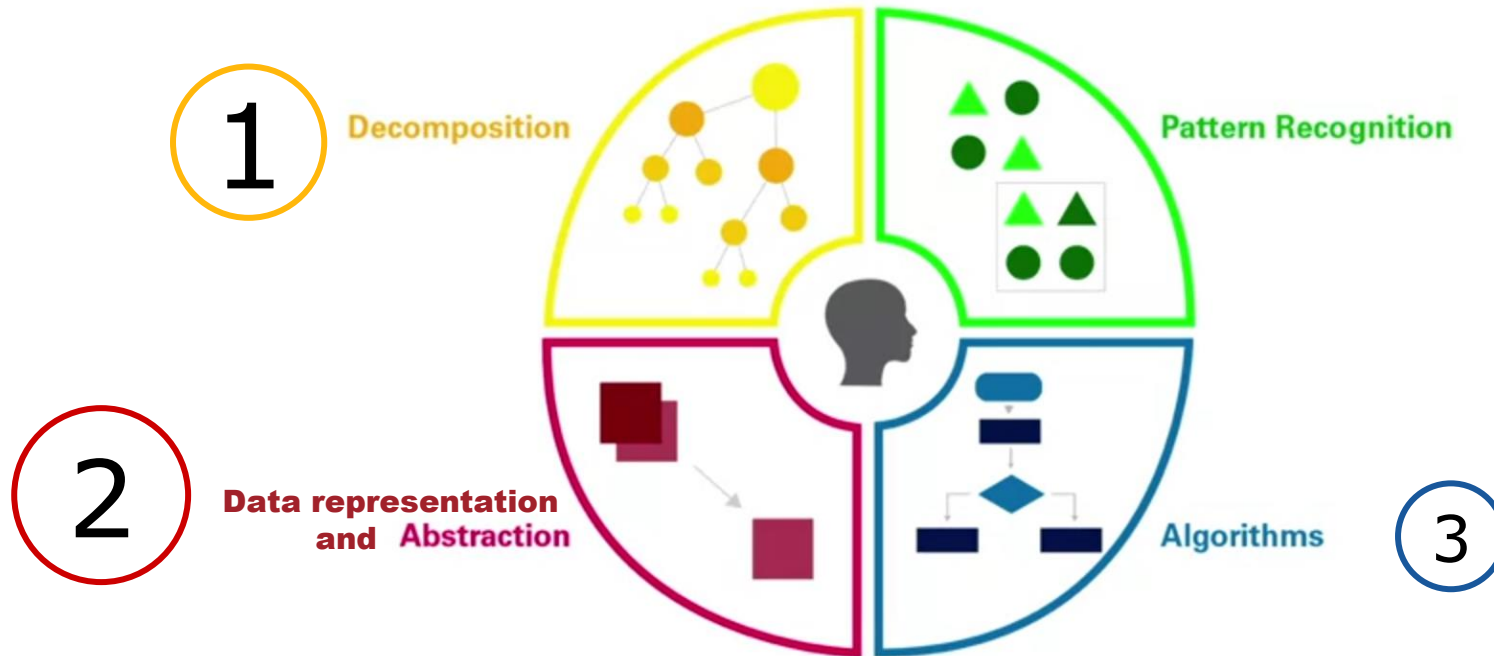
GITSHARE : NOTIONS MISES EN OEUVRE

- analyse et définition de données structurées avec JSON et de données géo-localisées avec geoJSON,
- partage de données sur Internet avec github, dropbox ou drive, lecture et décodage depuis un smartphone,
- définition et l'enregistrement du profil utilisateur,
- affichage et sélection de données sur une carte,
- affichage des produits proposés par un commerçant,
- Préparation et envoi de commandes par SMS



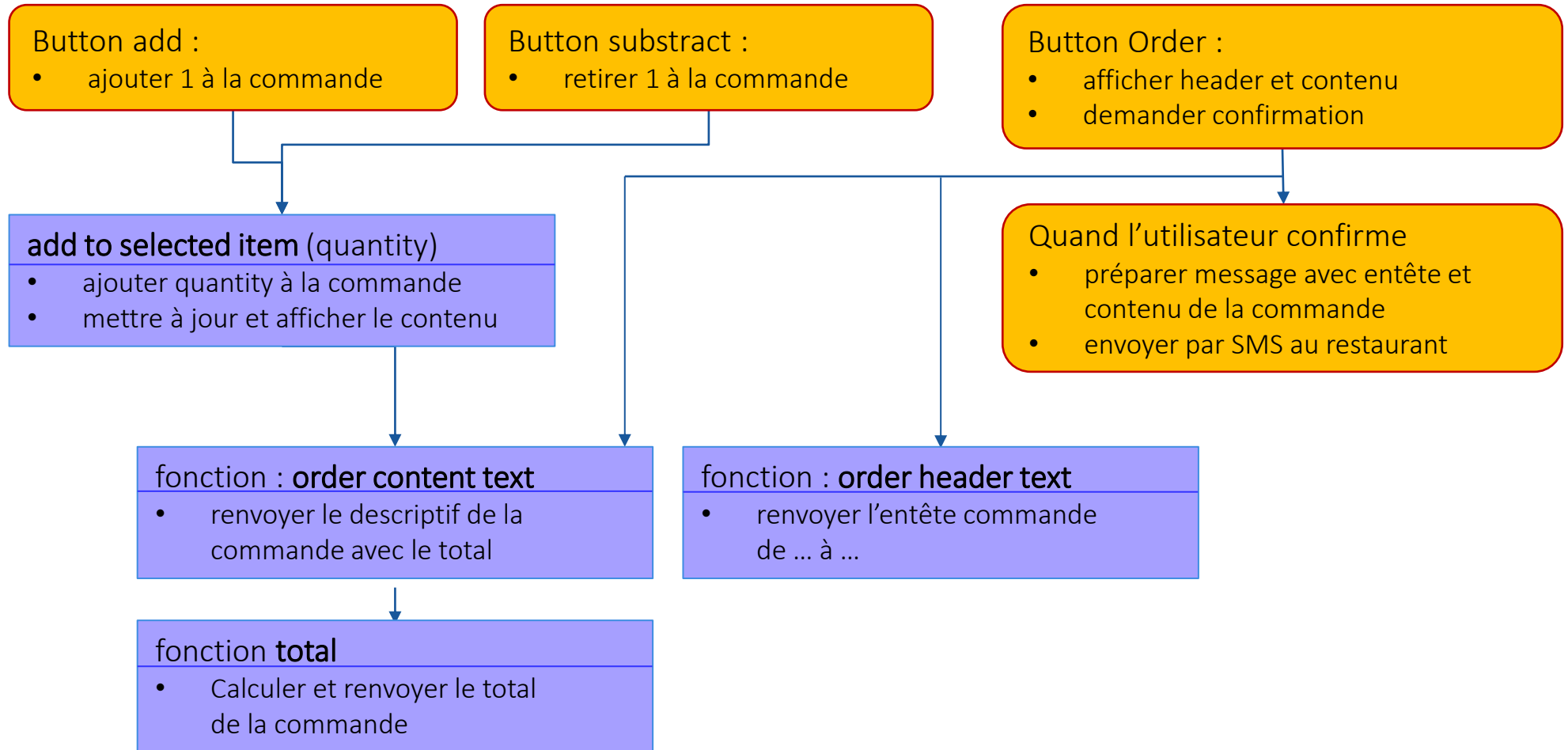
PRINCIPALES ACTIVITÉS PROPOSÉES DANS CE COURS

Pillars of Computational Thinking



Jeannette M. Wing: Computational thinking. Commun. ACM 49(3) [1]: 33-35 (2006)
Property of Penn Engineering

GITSHARE : NOTIONS MISES EN OEUVRE




3.6

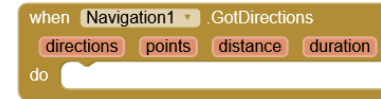
Bonus & extensions

PLAN

- Introduction : décomposition de l'application
- 1. Création d'un site Web / serveur de données
- 2. Définition des données et préparation d'un jeu
- 3. Création de l'application mobile avec App Inventor
 - 1. V1 : visualiser le catalogue des restaurants sur une carte
 - 2. V2a : sélectionner un restaurant et afficher ses nom, adresse, image et liste de plats
 - 3. V2b : codage défensif, modèle de données, adresses relatives, Dropbox & Google Drive
 - 4. V3a : identifier et enregistrer l'utilisateur et la carte
 - 5. V3b : préparer et passer une commande
 - 6. V3c : bonus
 - paiements,
 - partage de l'application, vérification du n° de téléphone ,
 - partage de catalogues, affichage amélioré

GITSHARE 3C : BONUS & EXTENSIONS

- extensions
 - publish / playstore : <https://appinventor.mit.edu/explore/ai2/google-play.html>
 - route/navigation components : 
cf. : <https://www.youtube.com/watch?v=dy54OixThW4>
 - delivery : see next video with firebase
 - payment
- bonus :
 - Screen1 :
 - application flashcode
 - list of catalogs
 - telephone : authentication
 - Shop screen
 - improved display





3.6.1

Paievements

GITSHARE 3C : BONUS & EXTENSIONS / PAYMENTS

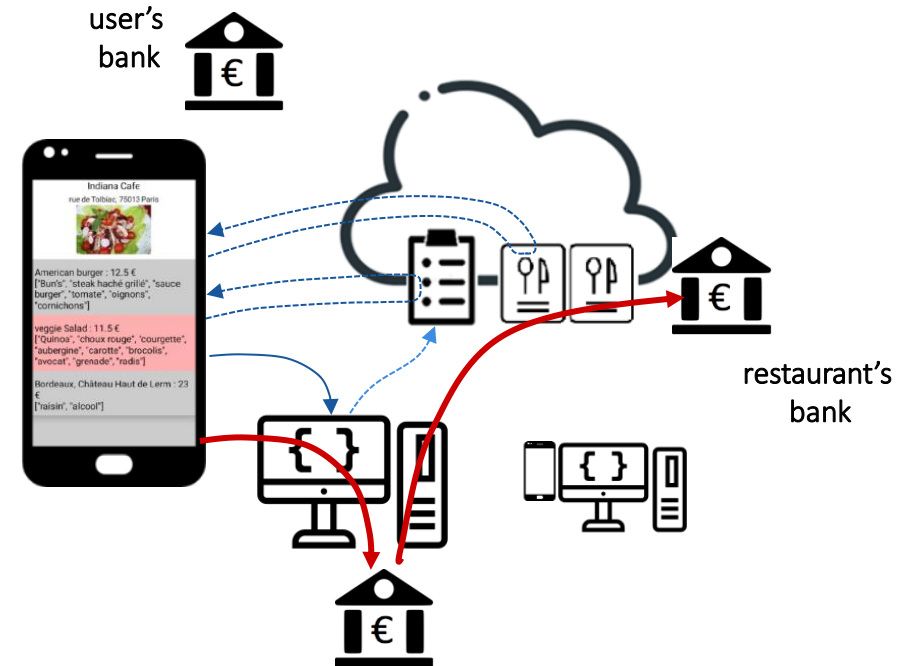
- "Quel type de solutions de paiement en ligne choisir pour son site e-commerce ? "

<https://www.francenum.gouv.fr/guides-et-conseils/developpement-commercial/solutions-de-paiement/quel-type-de-solutions-de>

- solutions des banques : contrat vente à distance
- prestataires indépendants
- portefeuilles de paiement en ligne
 - **Paylib / Lyf** : banques françaises,
*** même prix que carte bancaire, mais encore assez peu d'adhérents
 - Lydia : intéressante pour les petits commerçants,
sans frais de démarrage ou d'abonnement.
Commission par transaction.
 - **Paypal** : 1° solution mondiale, commission élevée
- " in app billing extension " Google play
<https://puravidaapps.com/billing.php> mal adapté aux produits physiques (non numériques)
- " Uber " ...

GITSHARE 3C : BONUS & EXTENSIONS / PAYMENTS

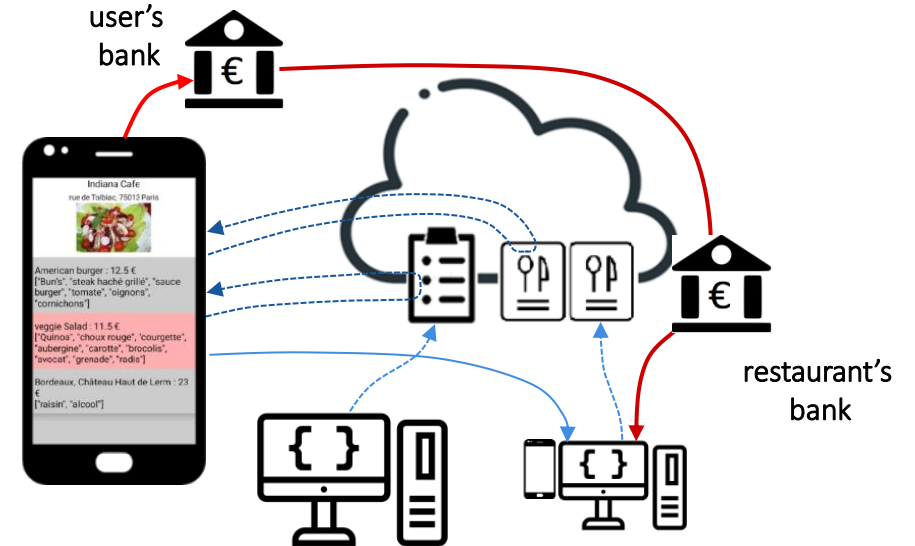
- Modes opératoires
 - Uber : le client paie Uber à la commande, et Uber paie le vendeur



GITSHARE 3C : BONUS & EXTENSIONS / PAYMENTS

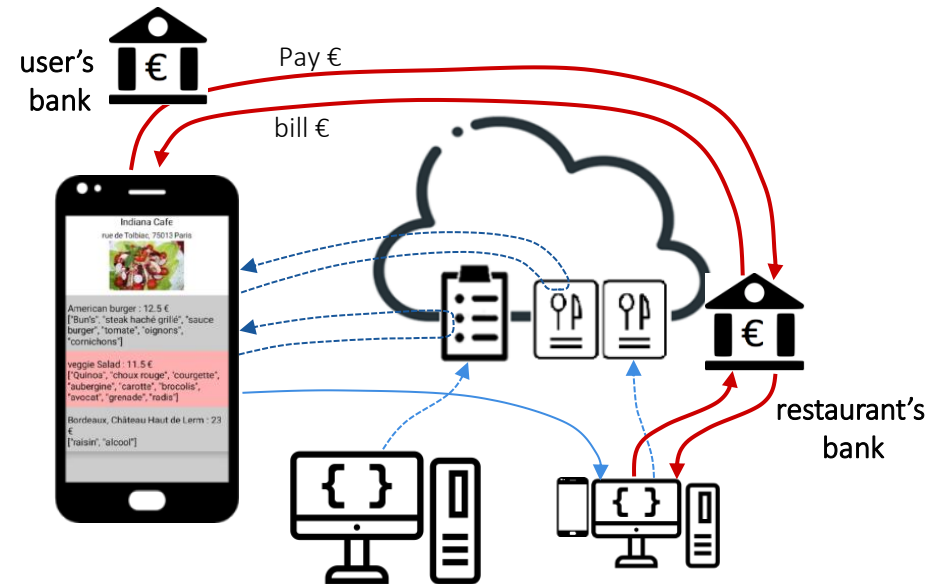
- Modes opératoires
 - Uber : le client paie Uber à la commande, et Uber paie le vendeur
 - Paiement à la commande
 1. le client envoie sa commande par SMS
 2. et déclenche le paiement correspondant

- Inconvénient :
pas d'accord ou de validation vendeur



GITSHARE 3C : BONUS & EXTENSIONS / PAYMENTS

- Modes opératoires
 - Uber : le client paie Uber à la commande, et Uber paie le vendeur
 - Paiement à la commande
 1. le client envoie sa commande par SMS
 2. et déclenche le paiement correspondant
 - Inconvénient : pas d'accord ou de validation vendeur
 - Paiement sur facture du vendeur
 1. le client envoie sa comamnde par SMS
 2. le vendeur envoie sa facture
 3. le client paie,
 4. le vendeur est informé



GITSHARE 3C : BONUS & EXTENSIONS / PAYMENTS

Liens rapides

- App Center
- Facturation
- Demander de l'argent
- Envoyer de l'argent
- PayPal.Me
- PayPal Checkout

Actions

Modifier Masquer

Créer une facture

Adresse email du client
pierre.huguet50@gmail.com

Nom de l'objet
commande du 21/11/2022

Montant de l'objet 58,5 € Devise EUR

[Envoyer](#) [Ajouter plus de détails](#)

Ajouter une deuxième action

- + Créer un lien de don rapide
- + QR code
- Afficher toutes les actions

m... vous a envoyé
1,00 € EUR.

Remarque de m... :

test app inventor

Détails de la transaction

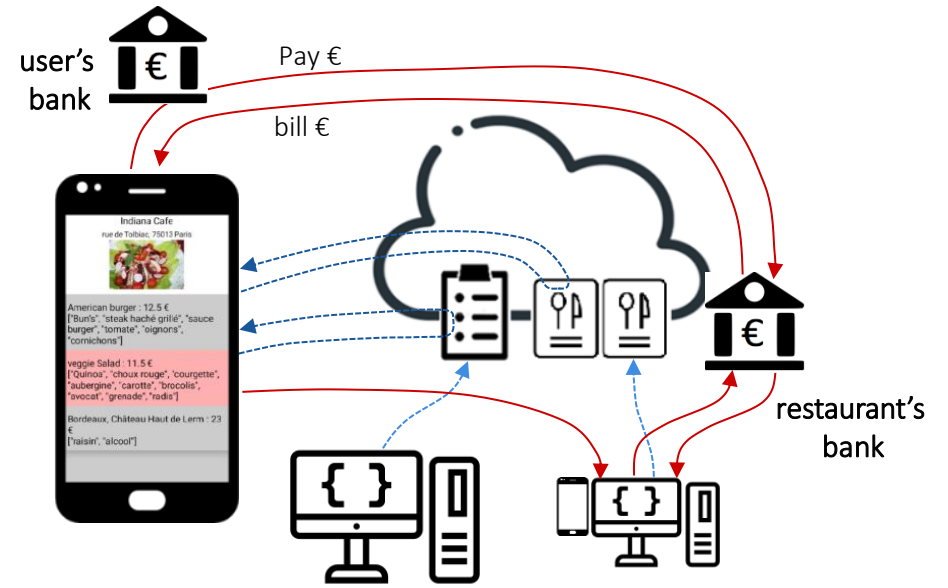
Numéro de transaction 7HW01...	Date de la transaction 21 novembre 2022
Montant reçu	1,00 € EUR
Frais	0,38 € EUR
Total	0,62 € EUR

Adresse de livraison

...ray
75015 Paris
France

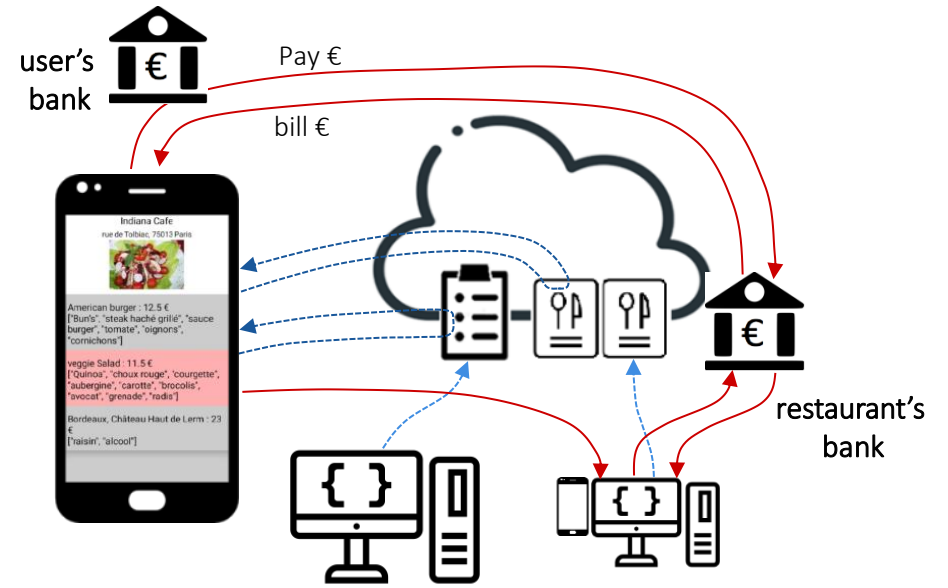
GITSHARE 3C : BONUS & EXTENSIONS / PAYMENTS

- Solutions candidates
 - " in app billing "



GITSHARE 3C : BONUS & EXTENSIONS / PAYMENTS

- Solutions candidates
 - " in app billing "




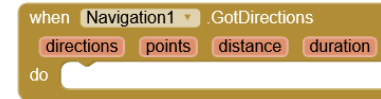


3.6.2


Flashcode
for application download

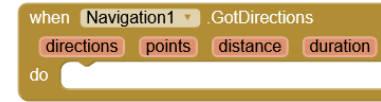
GITSHARE 3C : BONUS & EXTENSIONS

- extensions
 - publish / playstore : <https://appinventor.mit.edu/explore/ai2/google-play.html>
 - route/navigation components : 
cf. : <https://www.youtube.com/watch?v=dy54OixThW4>
 - delivery : see next video with firebase
 - payment
- bonus :
 - Screen1 :
 - application flashcode
 - list of catalogs
 - telephone : authentication
 - Shop screen
 - improved display



GITSHARE 3C : BONUS & EXTENSIONS

- extensions
 - publish / playstore : <https://appinventor.mit.edu/explore/ai2/google-play.html>
 - route/navigation components : 
cf. : <https://www.youtube.com/watch?v=dy54OixThW4>
 - delivery : see next video with firebase
 - payment
- bonus :
 - Screen1 :
 - application flashcode
 - list of catalogs
 - telephone : authentication
 - Shop screen
 - improved display



```
when Navigation1 .GotDirections
  directions points distance duration
do
```



GITSHARE 3C : application flashcode (download)

1. Upload application apk :

<https://onvaessayer.github.io/gitshare.apk>

GITSHARE 3C : application flashcode (download)

1. Upload application apk :

<https://onvaessayer.github.io/gitshare.apk>

2. generate flashcode :

- <https://chart.googleapis.com/chart?cht=qr&chs=320x320&chl=>

GITSHARE 3C : application flashcode (download)

1. Upload application apk :

<https://onvaessayer.github.io/gitshare.apk>

2. generate flashcode :

- <https://chart.googleapis.com/chart?cht=qr&chs=320x320&chl=https://onvaessayer.github.io/gitshare.apk>

GITSHARE 3C : application flashcode (download)

1. Upload application apk :

<https://onvaessayer.github.io/gitshare.apk>

2. generate flashcode :

- <https://chart.googleapis.com/chart?cht=qr&chs=320x320&chl=https://onvaessayer.github.io/gitshare.apk>



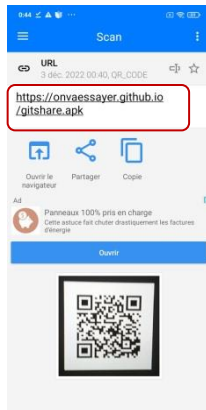
GITSHARE 3C : application flashcode (download)

1. Upload application apk :

<https://onvaessayer.github.io/gitshare.apk>

2. generate flashcode & test app download, with Android Scanner

- <https://chart.googleapis.com/chart?cht=qr&chs=320x320&chl=https://onvaessayer.github.io/gitshare.apk>
- Test App download (use off the shelf Scanner)



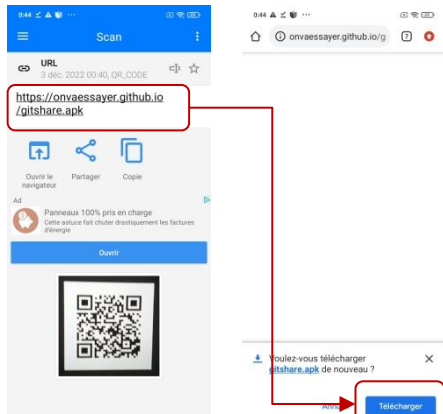
GITSHARE 3C : application flashcode (download)

1. Upload application apk :

<https://onvaessayer.github.io/gitshare.apk>

2. generate flashcode & test app download, with Android Scanner

- <https://chart.googleapis.com/chart?cht=qr&chs=320x320&chl=https://onvaessayer.github.io/gitshare.apk>
- Test App download (use off the shelf Scanner)



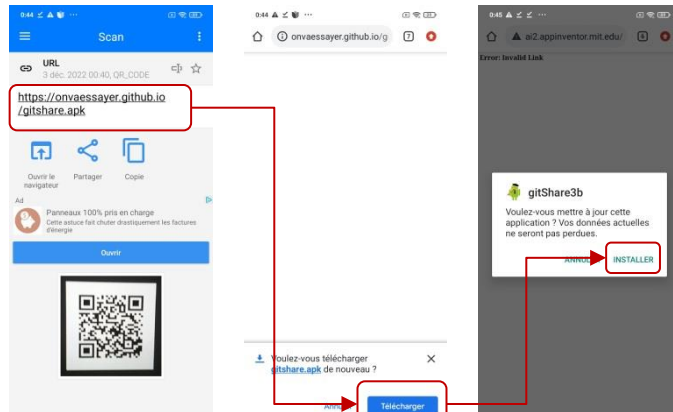
GITSHARE 3c : application flashcode (download)

1. Upload application apk :

<https://onvaessayer.github.io/gitshare.apk>

2. generate flashcode & test app download, with Android Scanner

- <https://chart.googleapis.com/chart?cht=qr&chs=320x320&chl=https://onvaessayer.github.io/gitshare.apk>
- Test App download (use off the shelf Scanner)



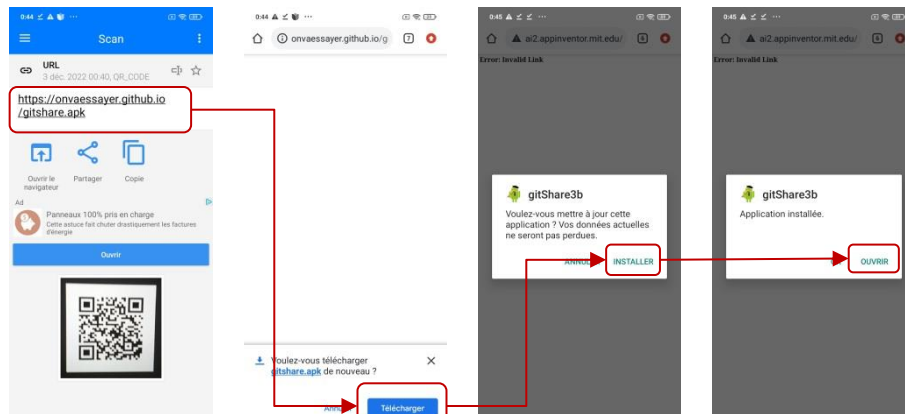
GITSHARE 3c : application flashcode (download)

1. Upload application apk :

<https://onvaessayer.github.io/gitshare.apk>

2. generate flashcode & test app download, with Android Scanner

- <https://chart.googleapis.com/chart?cht=qr&chs=320x320&chl=https://onvaessayer.github.io/gitshare.apk>
- Test App download (use off the shelf Scanner)



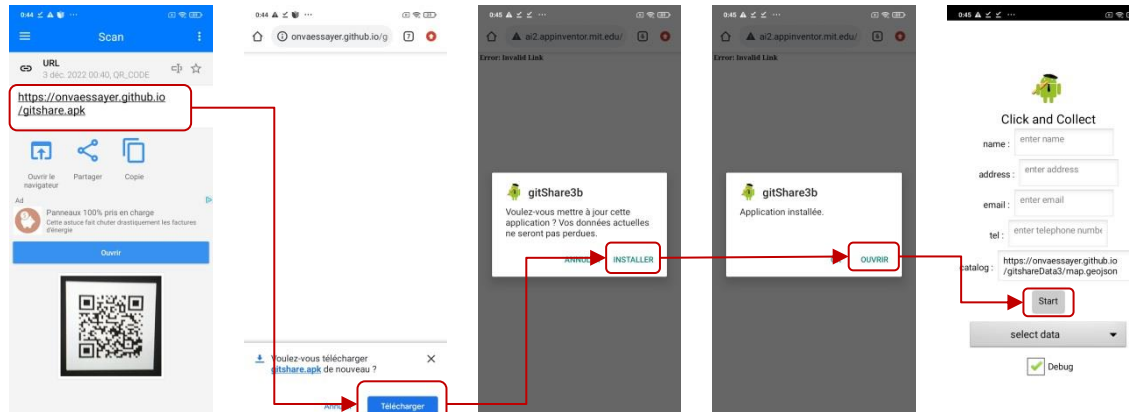
GITSHARE 3c : application flashcode (download)

1. Upload application apk :

<https://onvaessayer.github.io/gitshare.apk>

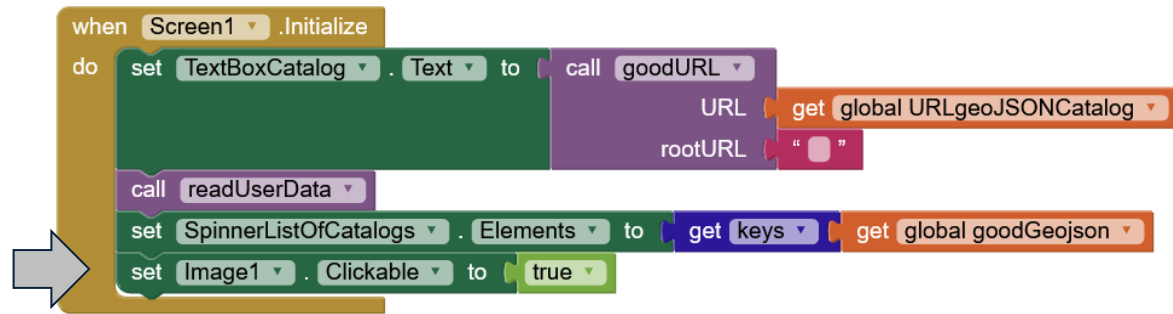
2. generate flashcode & test app download, with Android Scanner

- <https://chart.googleapis.com/chart?cht=qr&chs=320x320&chl=https://onvaessayer.github.io/gitshare.apk>
- Test App download (use off the shelf Scanner)



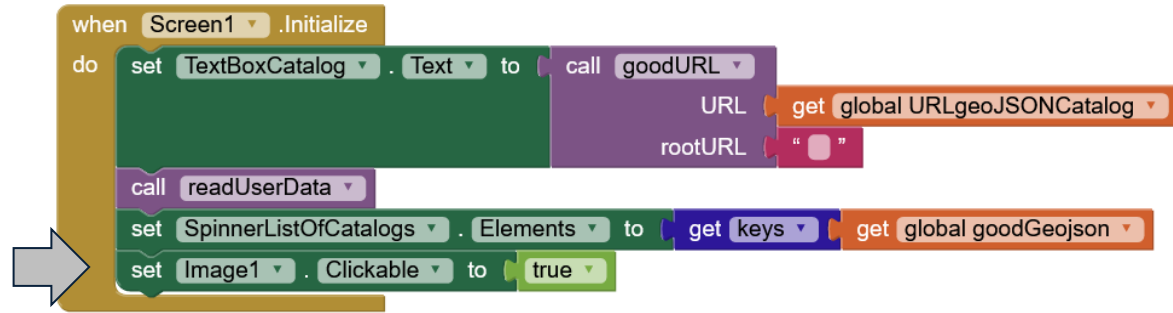
GITSHARE 3c : application flashcode (download)

1. Image clickable



GITSHARE 3c : application flashcode (download)

1. Image clickable



```
when Screen1.Initialize do
  set TextBoxCatalog.Text to call goodURL
  call readUserData
  set SpinnerListOfCatalogs.Elements to get keys
  set Image1.Clickable to true
```

The code block is a 'when Screen1.Initialize' block. It contains a 'do' block with four sub-blocks: 'set TextBoxCatalog.Text to call goodURL', 'call readUserData', 'set SpinnerListOfCatalogs.Elements to get keys', and 'set Image1.Clickable to true'. A grey arrow points to the 'set Image1.Clickable to true' block.

2. Flashcode URL :

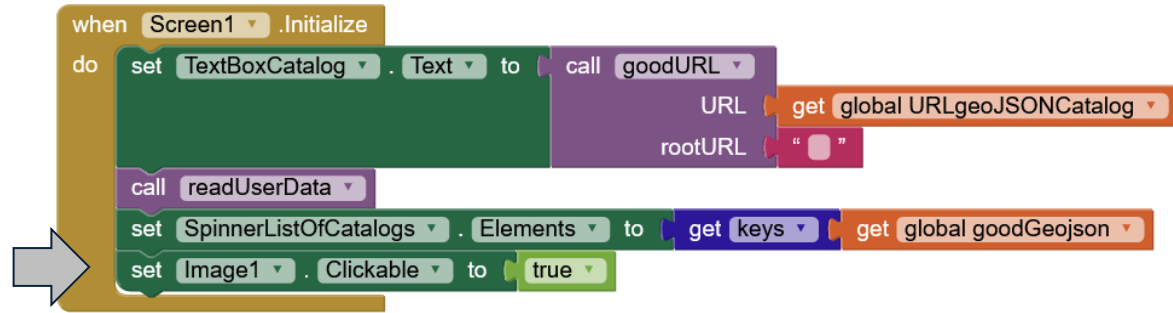


```
initialize global flaschode to join
  "https://chart.googleapis.com/chart?"
  "cht=qr&chs=320x320&chl="
  "https://onvaessayer.github.io/gitshare.apk"
```

The code block is an 'initialize global flaschode to join' block. It contains three string blocks: 'https://chart.googleapis.com/chart?', 'cht=qr&chs=320x320&chl=', and 'https://onvaessayer.github.io/gitshare.apk'.

GITSHARE 3C : application flashcode (download)

1. Image clickable



```
when Screen1.Initialize do
  set TextBoxCatalog.Text to call goodURL
  call readUserData
  set SpinnerListOfCatalogs.Elements to get keys
  set Image1.Clickable to true
```

The code block is titled "when Screen1.Initialize". It contains a "do" loop with four blocks: "set TextBoxCatalog.Text to call goodURL" (with sub-blocks "URL" and "rootURL" pointing to "get global URLgeoJSONCatalog" and "get global goodGeojson" respectively), "call readUserData", "set SpinnerListOfCatalogs.Elements to get keys" (with sub-block "get global goodGeojson"), and "set Image1.Clickable to true". A grey arrow points to the "set Image1.Clickable to true" block.

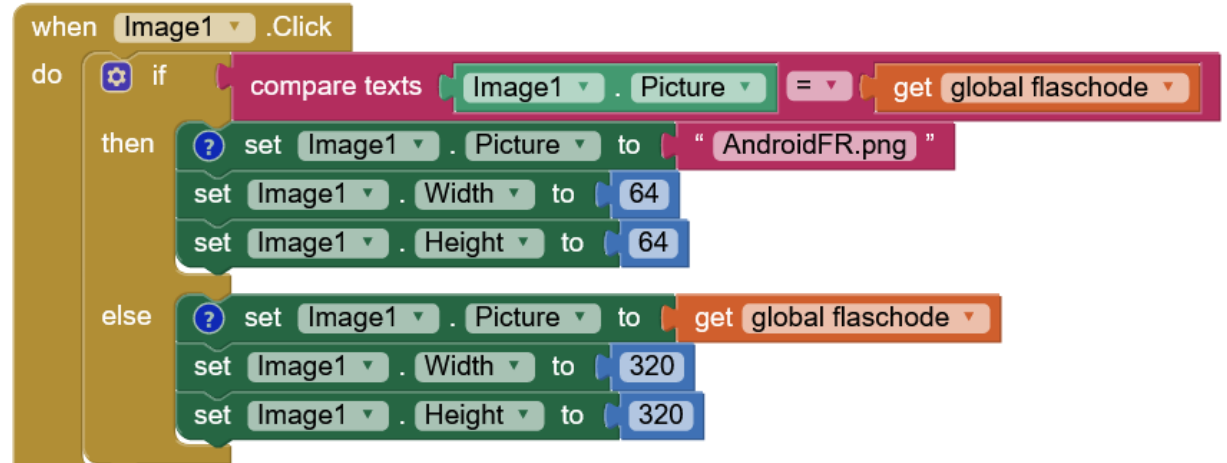
2. Flashcode URL :



```
initialize global flaschode to join
  "https://chart.googleapis.com/chart?"
  "cht=qr&chs=320x320&chl="
  "https://onvaessayer.github.io/gitshare.apk"
```

The code block is titled "initialize global flaschode to". It contains a "join" block with three string blocks: "https://chart.googleapis.com/chart?", "cht=qr&chs=320x320&chl=", and "https://onvaessayer.github.io/gitshare.apk".

3. Display flashcode :



```
when Image1.Click do
  if compare texts Image1.Picture = get global flaschode
  then
    set Image1.Picture to "AndroidFR.png"
    set Image1.Width to 64
    set Image1.Height to 64
  else
    set Image1.Picture to get global flaschode
    set Image1.Width to 320
    set Image1.Height to 320
```

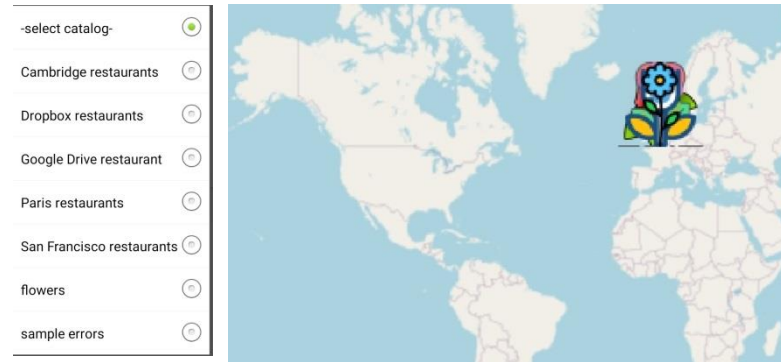
The code block is titled "when Image1.Click". It contains a "do" loop with an "if" block. The "if" block has a "compare texts" block with "Image1.Picture" and "get global flaschode". The "then" branch has three "set" blocks: "set Image1.Picture to 'AndroidFR.png'", "set Image1.Width to 64", and "set Image1.Height to 64". The "else" branch has three "set" blocks: "set Image1.Picture to get global flaschode", "set Image1.Width to 320", and "set Image1.Height to 320".

A large red circle with a thin outline, centered on the page. Inside the circle, the text '3.6.3' is written in a bold, black, sans-serif font.

3.6.3

Share
Catalog of catalogs

GITSHARE 3C : WEB catalog of catalogs



GITSHARE 3C : WEB catalog of catalogs

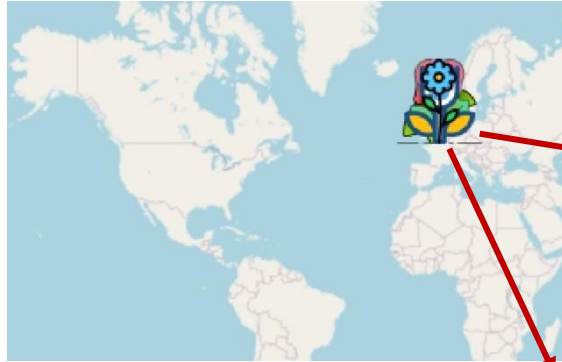
- select catalog-
- Cambridge restaurants
- Dropbox restaurants
- Google Drive restaurant
- Paris restaurants
- San Francisco restaurants
- flowers
- sample errors



A detailed map of Paris, France, showing various districts like Clichy, Clignancourt, 18e Arrondissement, 17e Arrondissement, Monceau, Faubourg Saint-Honoré, Opéra, 1er Arrondissement, Faubourg Saint-Germain, Paris, Faubourg Saint-Jacques, Plaisance, Quartier de la Glacière, Maison-Blanche, and Montrouge. Several restaurant icons are overlaid on the map. To the right, a vertical stack of restaurant detail cards is shown. The top card is for 'Le Relais' in Allerey, Paris 15, featuring a photo of the restaurant and a menu item 'Pizza Roma : 13.5 €' with ingredients 'farine', 'tomate', 'jambon'. Below it is a card for 'Tagliatelle : 12.5 €' with ingredients 'pates', 'crevettes', 'persil'. The next card is for 'Indiana Cafe' at 'rue de Tolbiac, 75013 Paris', showing a photo of the cafe and a list of items: '13.50 € 1 x', '37.50 € 3 x', '6.50 € 1 x', '23.00 € 1 x', and '80.50 € total'. Below that is a card for 'American burger : 12.5 €' with ingredients 'Buns', 'steak haché grillé', 'sauce burger', 'tomate', 'oignons', 'cornichons'. The next card is for 'veggie Salad : 11.5 €' with ingredients 'Quinoa', 'choux rouge', 'courgette', 'aubergine', 'carotte', 'brocolis', 'avocat', 'grenade', 'radis'. The bottom card is for 'Bordeaux, Château Haut de Lerm : 23 €' and shows a list of items: '12.50 € 1 x American burger', '23.00 € 2 x veggie Salad', '23.00 € 1 x Bordeaux, Château Haut de Lerm', and '58.50 € total'. At the bottom of the cards are '+' and '-' buttons and an 'order' button.

GITSHARE 3C : WEB catalog of catalogs

- select catalog-
- Cambridge restaurants
- Dropbox restaurants
- Google Drive restaurant
- Paris restaurants
- San Francisco restaurants
- flowers
- sample errors



shop 13:50

Le Relais
Alleray, Paris 15

Pizza Roma : 13.5 €
[farine, "tomate", "jambon"]

Tagliatelle : 12.5 €
[pates, "crevettes", "persil"]

shop 13:51

Indiana Cafe
rue de Tolbiac, 75013 Paris

13.50 € 1 x
37.50 € 3 x
6.50 € 1 x
23.00 € 1 x
80.50 € total

American burger : 12.5 €
[Buns, "steak haché grillé",
sauce burger, "tomate",
"oignons", "cornichons"]

veggie Salad : 11.5 €
[Quinoa, "choux rouge",
"courgette", "aubergine",
"carotte", "brocolis", "avocat",
"grenade", "radis"]

Bordeaux, Château Haut de Lerm : 23 €

12.50 € 1 x American burger
23.00 € 2 x veggie Salad
23.00 € 1 x Bordeaux, Château
Haut de Lerm
58.50 € total

+ - order

shop 13:52

palais Des Roses
40 Rue Mathurin Régnier, 75015 Paris

happy fleurs
53 Rue de Vouillé, 75015 Paris

Gysophiles : 19 €
[roses, "branches"]

bouquet 5 : 17.5 €
[roses, "branches"]

bouquet 6 : 17.5 €
[tulipes, "mousse"]

bouquet de lys : 18 €
[roses]

17.50 € 1 x bouquet 6
35.00 € total

GITSHARE 3C : WEB catalog of catalogs

The image displays the Click&collect app interface, which allows users to browse and order from various catalogs. The main screen shows a list of catalogs on the left, a world map in the center, and a detailed view of a selected catalog on the right. Red arrows indicate the flow from the catalog list to the world map, and from the world map to the detailed catalog views.

-select catalog-

- Cambridge restaurants
- Dropbox restaurants
- Google Drive restaurant
- Paris restaurants
- San Francisco restaurants
- flowers
- sample errors

World Map: A world map with a red flower icon over Europe. Red arrows point from the map to the detailed catalog views.

Paris Catalog (Le Relais):

- Le Relais, Allerey, Paris 15
- Pizza Roma : 13.5 € [farine, "tomate", "jambon"]
- Tagliatelle : 12.5 € [pates, "crevettes", "persil"]
- Indiana Cafe, rue de Tolbiac, 75013 Paris
- American burger : 12.5 € ["Buns", "steak haché grillé", "sauce burger", "tomate", "oignons", "cornichons"]
- veggie Salad : 11.5 € ["Quinoa", "choux rouge", "courgette", "aubergine", "carotte", "brocolis", "avocat", "grenade", "radis"]
- Bordeaux, Château Haut de Lerm : 23 €

Boston Catalog (Café Luna):

- cafe Luna, cafeLuna/restaurant.json
- Flour Bakery + Cafe, 190 Massachusetts Ave, Cambridge, MA 02139, États-Unis
- Café Luna, 612 Main St, Cambridge, MA 02139, États-Unis
- soup of the day : 5 € [fennel, "tomato"]

Paris Catalog (Happy fleurs):

- Happy fleurs, happyFleurs/flowers.json
- palais Des Roses, 40 Rue Mathurin Régnier, 75015 Paris
- happy fleurs, 53 Rue de Vouillé, 75015 Paris
- Gysophiles : 19 € ["roses", "branches"]
- bouquet 5 : 17.5 € ["roses", "branches"]
- bouquet 6 : 17.5 € ["tulipes", "mousse"]
- bouquet de lys : 18 € ["roses"]
- grilled honey butter
- 17.50 € 1 x bouquet 6, 35.00 € total

GITSHARE 3C : WEB catalog of catalogs

The image displays the GitShare 3C application interface. At the top, a world map shows a flower icon with red arrows pointing to detailed views of two catalogs: 'cafe Luna' in Somerville, Boston, and 'Happy fleurs' in Paris. A central menu lists various catalog categories: -select catalog-, Cambridge restaurants, Dropbox restaurants, Google Drive restaurant, Paris restaurants, San Francisco restaurants, flowers, and sample errors. The 'cafe Luna' view shows a map of Boston with a pizza icon and a list of items like 'Flour Bakery + Cafe' and 'Café Luna'. The 'Happy fleurs' view shows a map of Paris with a flower icon and a list of items like 'palais Des Roses', 'bouquet 5', and 'Gysophiles'. Both views include detailed menu items with prices and descriptions.

cafe Luna
cafeLuna/restaurant.json

Happy fleurs
happyFleurs/flowers.json

cafe Luna Menu:

- Flour Bakery + Cafe: 190 Massachusetts Ave, Cambridge, MA 02139, États-Unis
- Café Luna: 612 Main St, Cambridge, MA 02139, États-Unis
- soup of the day : 5 € [fennel, "tomato"]
- 7.20 € 1 x French Onion Soup
- 43.20 € 3 x House Salad
- 21.54 € 1 x Chicken Pesto Crepe
- 14.40 € 1 x Sweet As Apple Pie
- 86.34 € total

Happy fleurs Menu:

- palais Des Roses: 40 Rue Mathurin Régnier, 75015 Paris
- happy fleurs: 53 Rue de Vouillé, 75015 Paris
- Gysophiles : 19 € ["roses", "branches"]
- bouquet 5 : 17.5 € ["roses", "branches"]
- bouquet 6 : 17.5 € ["tulipes", "mousse"]
- Bouquet de lys : 18 € ["roses"]
- 17.50 € 1 x bouquet 6
- 35.00 € total

GITSHARE 3C : WEB catalog of catalogs

1. Create & upload catalog of catalogs (github, dropbox, ...)

```
{  
  " select catalog ":"",  
  "flowers": "https://onvaessayer.github.io/flowers/map.geojson",  
  "Paris restaurants": "https://onvaessayer.github.io/gitshareData3/map.geojson",  
  "Dropbox restaurants": "https://www.dropbox.com/s/p8oizwazpme7xpq/map.json?dl=1",  
  "Google Drive restaurant": "https://drive.google.com/file/d/lim0UjN6umweZR7nJlegobKk9edIGO5gK/view",  
  "Cambridge restaurants": "https://onvaessayer.github.io/cambridgeData/map.geojson",  
  "San Francisco restaurants": "https://onvaessayer.github.io/SanFrancisco/map.geojson",  
  "sample errors": "https://onvaessayer.github.io/gitshareErrors1/map.geojson"  
}
```

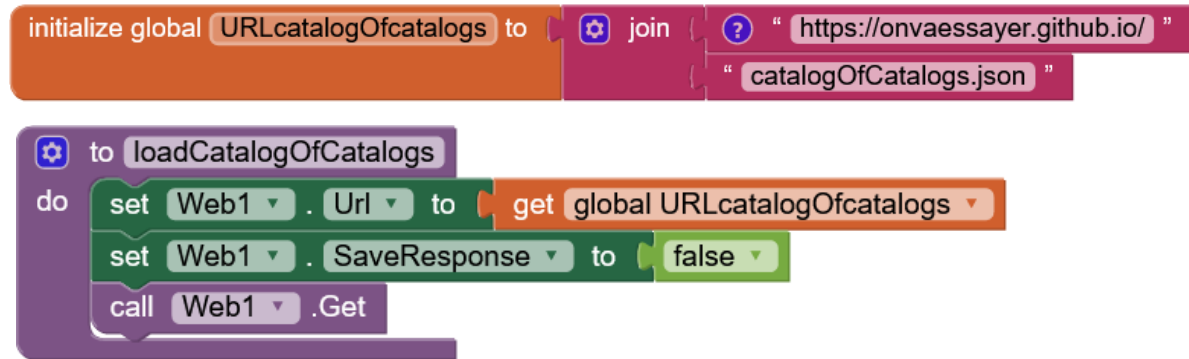
GITSHARE 3C : WEB catalog of catalogs

1. Create & upload catalog of catalogs (github, dropbox, ...)
2. Define URL and load web data

```
initialize global URLcatalogOfcatalogs to join [ "https://onvaessayer.github.io/" , "catalogOfCatalogs.json" ]
```

GITSHARE 3C : WEB catalog of catalogs

1. Create & upload catalog of catalogs (github, dropbox, ...)
2. Define URL and load web data



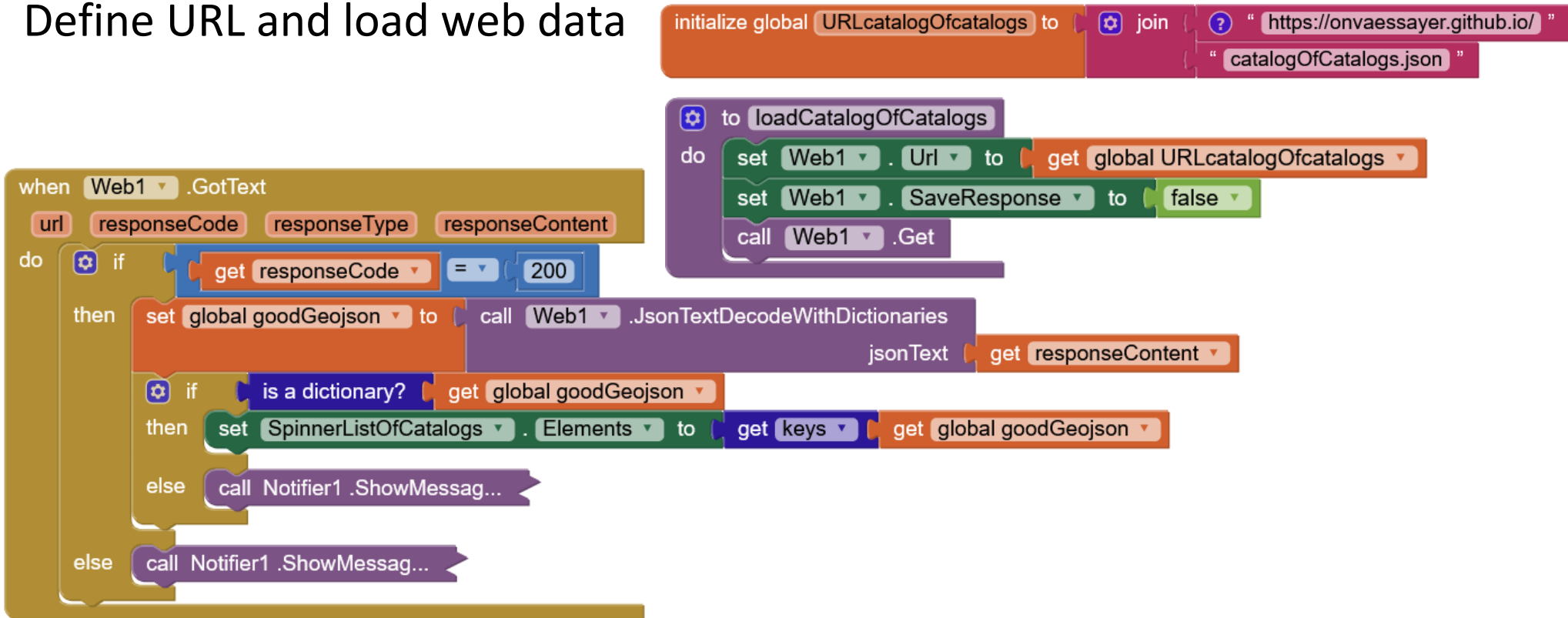
```
initialize global URLcatalogOfcatalogs to join ["https://onvaessayer.github.io/", "catalogOfCatalogs.json"]

to loadCatalogOfCatalogs
do
  set Web1 . Url to get global URLcatalogOfcatalogs
  set Web1 . SaveResponse to false
  call Web1 . Get
```

The image shows a Scratch script with two main parts. The first part is an 'initialize global' block that sets a global variable named 'URLcatalogOfcatalogs' to the value of a 'join' block. The 'join' block concatenates the string 'https://onvaessayer.github.io/' and the string 'catalogOfCatalogs.json'. The second part is a 'to loadCatalogOfCatalogs' function block. It contains a 'do' loop with three steps: 1) 'set Web1 . Url to get global URLcatalogOfcatalogs', 2) 'set Web1 . SaveResponse to false', and 3) 'call Web1 . Get'.

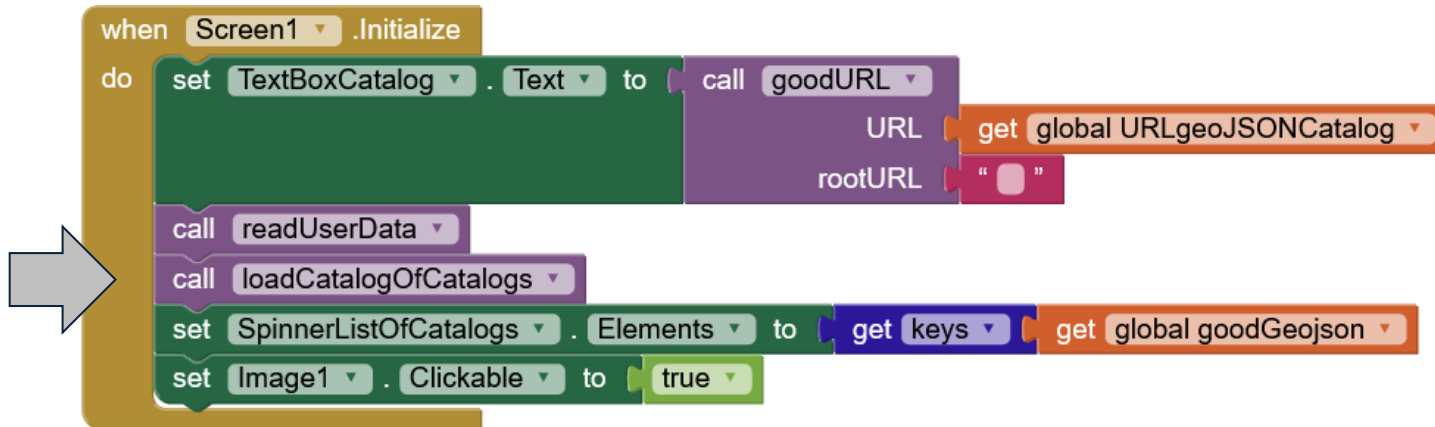
GITSHARE 3C : WEB catalog of catalogs

1. Create & upload catalog of catalogs (github, dropbox, ...)
2. Define URL and load web data



GITSHARE 3C : WEB catalog of catalogs

1. Create & upload catalog of catalogs (github, dropbox, ...)
2. Define URL & procedure to load web data
3. Call load catalogs on startup



The image shows a Scratch code block for the initialization of a screen. The code is as follows:

```
when Screen1 .Initialize
do
  set TextBoxCatalog . Text to call goodURL
  call readUserData
  call loadCatalogOfCatalogs
  set SpinnerListOfCatalogs . Elements to get keys
  set Image1 . Clickable to true
```

The code block is a "when Screen1 .Initialize" block. It contains a "do" block with the following steps:

- set TextBoxCatalog . Text to call goodURL (with URL and rootURL sub-blocks)
- call readUserData
- call loadCatalogOfCatalogs
- set SpinnerListOfCatalogs . Elements to get keys (with global goodGeojson sub-block)
- set Image1 . Clickable to true

A grey arrow points to the "call loadCatalogOfCatalogs" block.

A large red circle with a thin outline, centered on the page. Inside the circle, the text '3.6.4' is written in a bold, black, sans-serif font.

3.6.4

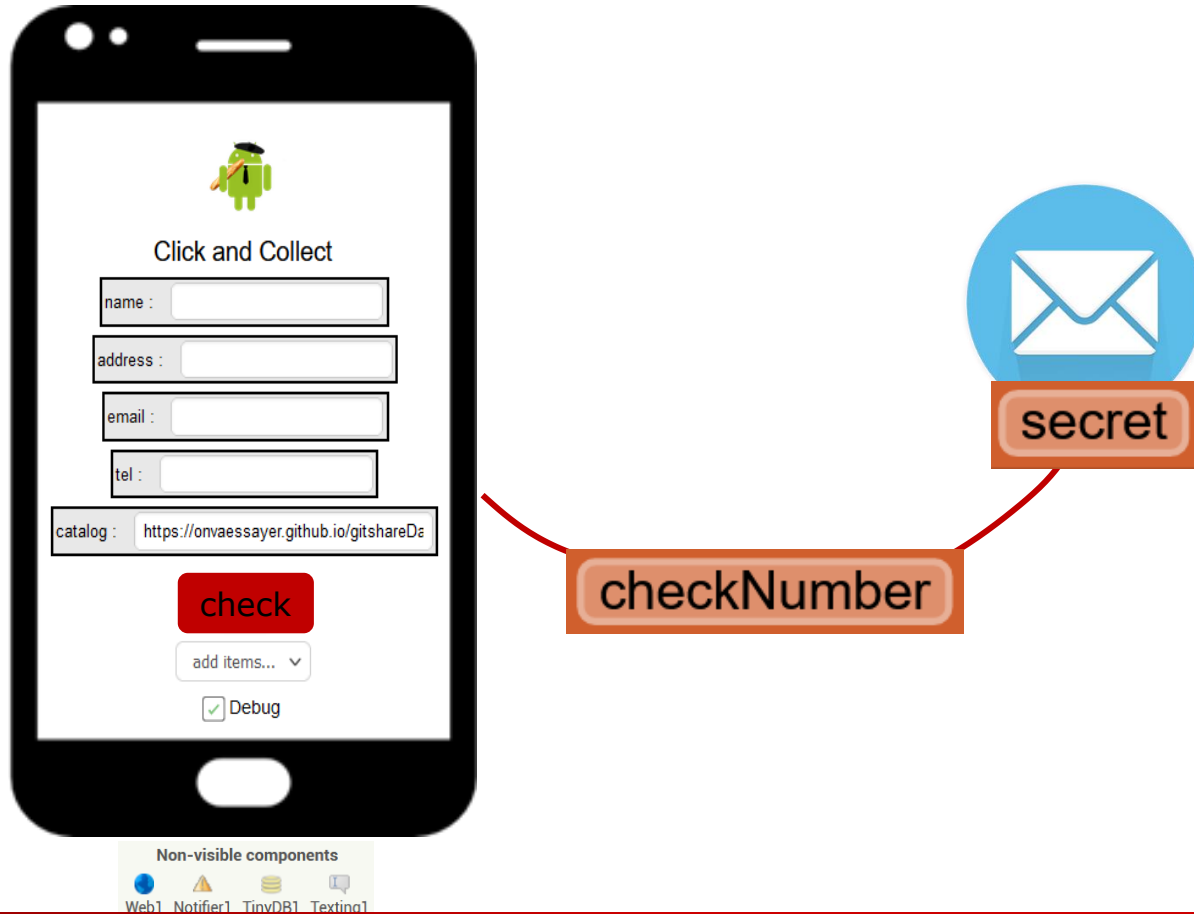
Telephone #
verification

GITSHARE 3C : TELEPHONE VERIFICATION

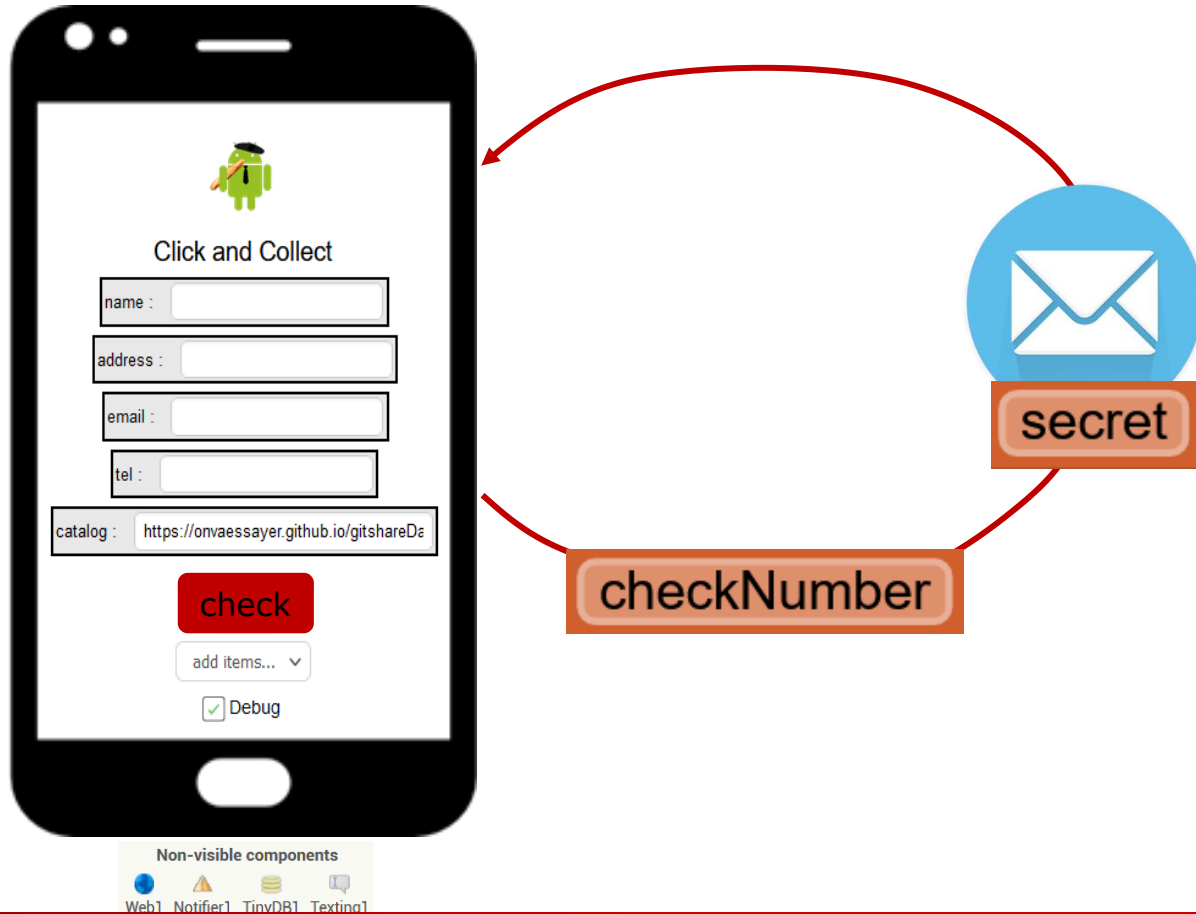


Non-visible components
Web1 Notifier1 TinyDB1 Texting1

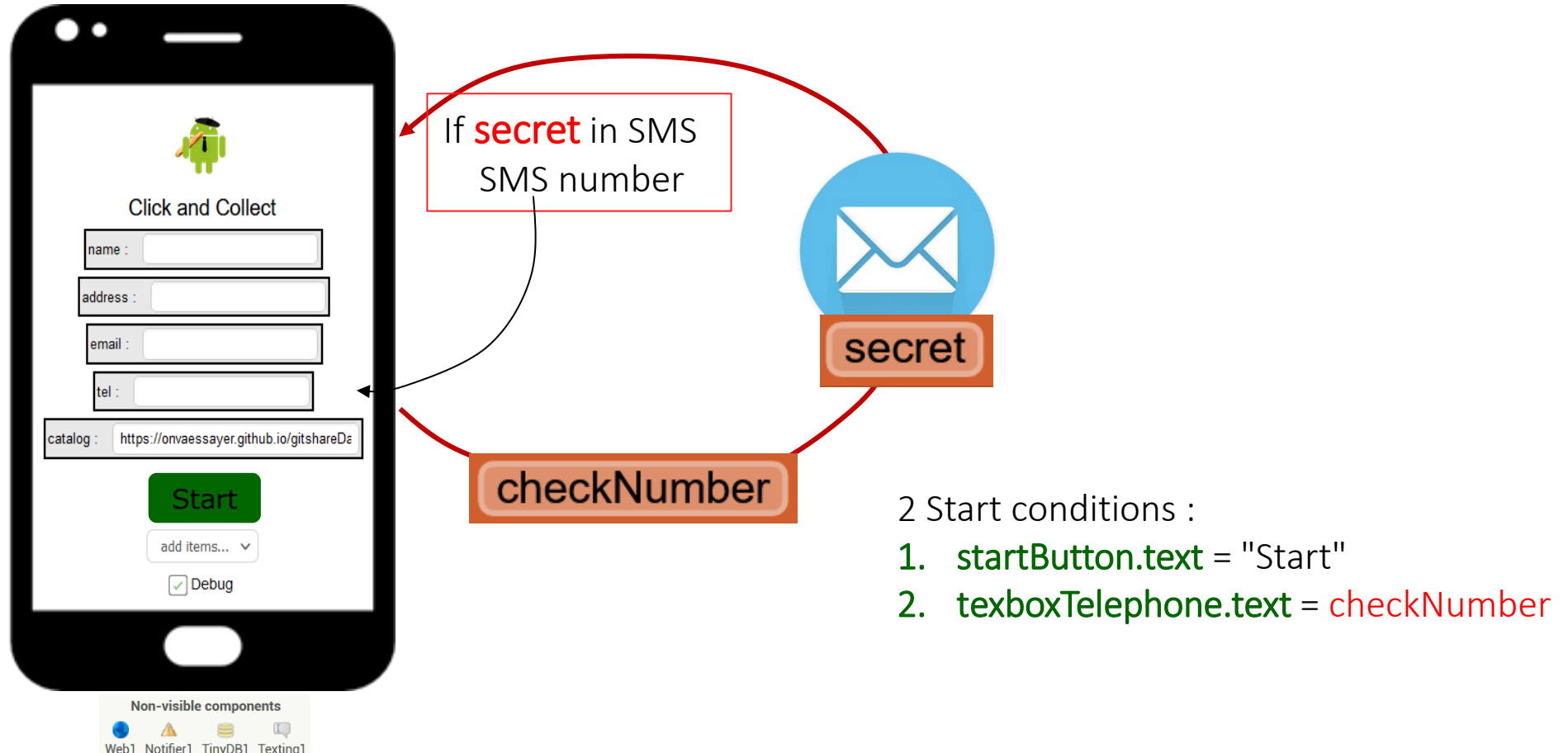
GITSHARE 3C : TELEPHONE VERIFICATION



GITSHARE 3C : TELEPHONE VERIFICATION

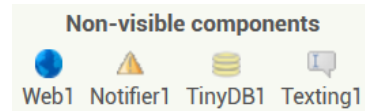


GITSHARE 3C : TELEPHONE VERIFICATION



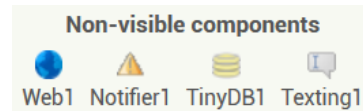
GITSHARE 3C : TELEPHONE VERIFICATION

1. Screen1 design : Add SMS/texting component



GITSHARE 3C : TELEPHONE VERIFICATION

1. Screen1 design : Add SMS/texting component

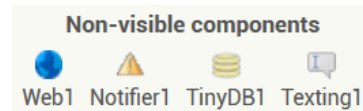


2. Init variables :



GITSHARE 3C : TELEPHONE VERIFICATION

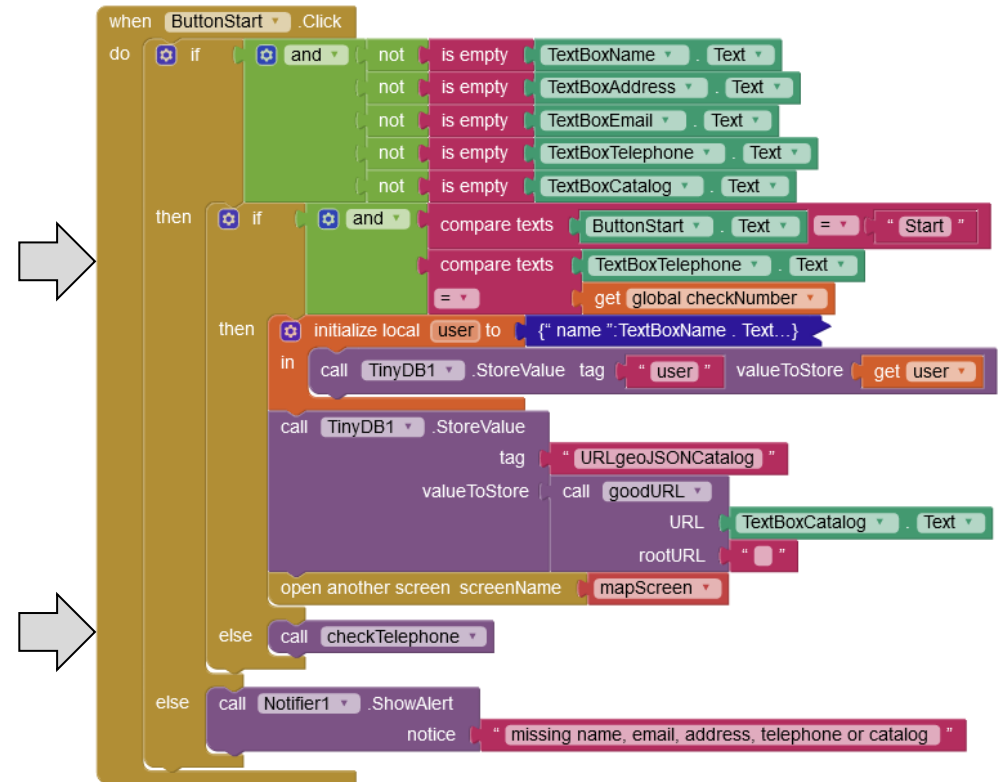
1. Screen1 design : Add SMS/texting component



2. Init variables :




3. Check conditions on buttonStart
 - checkTelephone if not met



GITSHARE 3C : TELEPHONE VERIFICATION

4. Checktelephone procedure


- set button text and color
- set checkNumber and secret
- activate reception & send SMS

```
to checkTelephone
do
  set ButtonStart . Text to "check telephone"
  set ButtonStart . BackgroundColor to 
  call Notifier1 . ShowAlert notice "téléphone SMS check"
  set Texting1 . PhoneNumber to TextBoxTelephone . Text
  set global secret to random integer from 1 to 9999
  set Texting1 . Message to get global secret
  set Texting1 . ReceivingEnabled to ReceivingState Foreground
  call Texting1 . SendMessageDirect
```

GITSHARE 3C : TELEPHONE VERIFICATION


4. Checktelephone procedure

- set button text and color
- set checkNumber and secret
- activate reception & send SMS

```
to checkTelephone
do
  set ButtonStart . Text to "check telephone"
  set ButtonStart . BackgroundColor to 
  call Notifier1 .ShowAlert notice "téléphone SMS check"
  set Texting1 . PhoneNumber to TextBoxTelephone . Text
  set global secret to random integer from 1 to 9999
  set Texting1 . Message to get global secret
  set Texting1 . ReceivingEnabled to ReceivingState Foreground
  call Texting1 .SendMessageDirect
```

5. Check / SMS reception

- get SMS number
- If sms message = secret
 textBoxNumber = SMS number
 checkNumber = textBoxNumber
 buttonStart.text= Start

```
when Texting1 .MessageReceived
  number messageText
do
  set Texting1 . ReceivingEnabled to ReceivingState Off
  if compare texts get messageText = get global secret
  then
    set TextBoxTelephone . Text to get number
    set global checkNumber to TextBoxTelephone . Text
    set ButtonStart . Text to "Start"
    set ButtonStart . BackgroundColor to 
  else
    call Notifier1 .ShowAlert notice "telephone check KO"
```

GITSHARE 3C : TELEPHONE VERIFICATION

4. Checktelephone procedure

- set button text and color
- set checkNumber and secret
- activate reception & send SMS

```
to checkTelephone
do
  set ButtonStart . Text to " check telephone "
  set ButtonStart . BackgroundColor to 
  call Notifier1 . showAlert notice " téléphone SMS check "
  set Texting1 . PhoneNumber to TextBoxTelephone . Text
  set global secret to random integer from 1 to 9999
  set Texting1 . Message to get global secret
  set Texting1 . ReceivingEnabled to ReceivingState Foreground
  call Texting1 . SendMessageDirect
```

5. Check / SMS reception

- get SMS number
- If sms message = secret
 textBoxNumber = SMS number
 checkNumber = textBoxNumber
 buttonStart.text= Start

```
when Texting1 . MessageReceived
  number messageText
do
  set Texting1 . ReceivingEnabled to ReceivingState Off
  if compare texts get messageText = get global secret
  then
    set TextBoxTelephone . Text to get number
    set global checkNumber to TextBoxTelephone . Text
    set ButtonStart . Text to " Start "
    set ButtonStart . BackgroundColor to 
  else
    call Notifier1 . showAlert notice " telephone check KO "
```

6. Set checkNumber in readUserData

```
set TextBoxTelephone . Text to get value for key " telephone " in dictionary get user or if not found " "
```

GITSHARE 3C : SCREEN1 - BONUS & EXTENSIONS

```

initialize global URLGeoJSONCatalog to join
  https://onvaessayer.github.io/
  gitshareData3/map.geojson

when Screen1.Initialize
do
  set TextBoxCatalog .Text to call goodURL .
  URL get global URLGeoJSONCatalog .
  rootURL .

  call readUserData .
  call loadCatalogOfCatalogs .

  set SpinnerListOfCatalogs .Elements to get keys .
  get global goodGeojson .
  set Image1 .Clickable to true

when ButtonStart .Click
do
  if
    and
      not is empty TextBoxName .Text .
      not is empty TextBoxAddress .Text .
      not is empty TextBoxEmail .Text .
      not is empty TextBoxTelephone .Text .
      not is empty TextBoxCatalog .Text .
  if
    and
      compare texts ButtonStart .Text = Start .
      compare texts TextBoxTelephone .Text =
        get global checkNumber .
  then
    initialize local user to make a dictionary
      key name .
      value TextBoxName .Text .
      key address .
      value TextBoxAddress .Text .
      key email .
      value TextBoxEmail .Text .
      key telephone .
      value TextBoxTelephone .Text .
      key catalog .
      value TextBoxCatalog .Text .
    in
      call TinyDB1 .StoreValue tag user .
      valueToStore get user .
    open another screen with start value screenName mapScreen .
      startValue call goodURL .
      URL TextBoxCatalog .Text .
      rootURL .
  else
    call checkTelephone .
  
```

Load Catalog of Catalogs
Image clickable

Telephone valid ?
checkNumber ?
Button start ?

Telephone INvalid ?
checkTelephone

```

initialize global goodGeojson to { *select data ** data... }

to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1 .GetValue
    tag user .
    valueIfTagNotThere create empty dictionary
  in
    set TextBoxName .Text to get value for key name .
      in dictionary get user . or if not found .
    set TextBoxAddress .Text to get value for key address .
      in dictionary get user . or if not found .
    set TextBoxEmail .Text to get value for key email .
      in dictionary get user . or if not found .
    set TextBoxTelephone .Text to get value for key telephone .
      in dictionary get user . or if not found .
    set global checkNumber to TextBoxTelephone .Text .
    set TextBoxCatalog .Text to get value for key catalog .
      in dictionary get user . or if not found .
      get global URLGeoJSONCatalog .
    set CheckBoxDebug .Enabled to call TinyDB1 .GetValue
      tag debug .
      valueIfTagNotThere true .

when SpinnerListOfCatalogs...
when CheckBoxDebug .Change...

when Image1 .Click
do
  if compare texts Image1 .P...
    initialize global flascode to join
      https://chart.googleapis.com/chart?
      cht=qr&chs=320x320&chl=
      https://onvaessayer.github.io/gitshare.apk .
    flascode

initialize global URLCatalogOfCatalogs to join
  https://onvaessayer.github.io/
  catalogOfCatalogs.json .
to loadCatalogOfCatalogs do...
when Web1 .GotText url ...

initialize global secret to 1
initialize global checkNumber to .
to checkTelephone do set B...
when Texting1 .MessageRece...
  
```

Telephone number

flashcode

Catalog of catalogs

Check telephone n°



3.6.5

Display items
extension

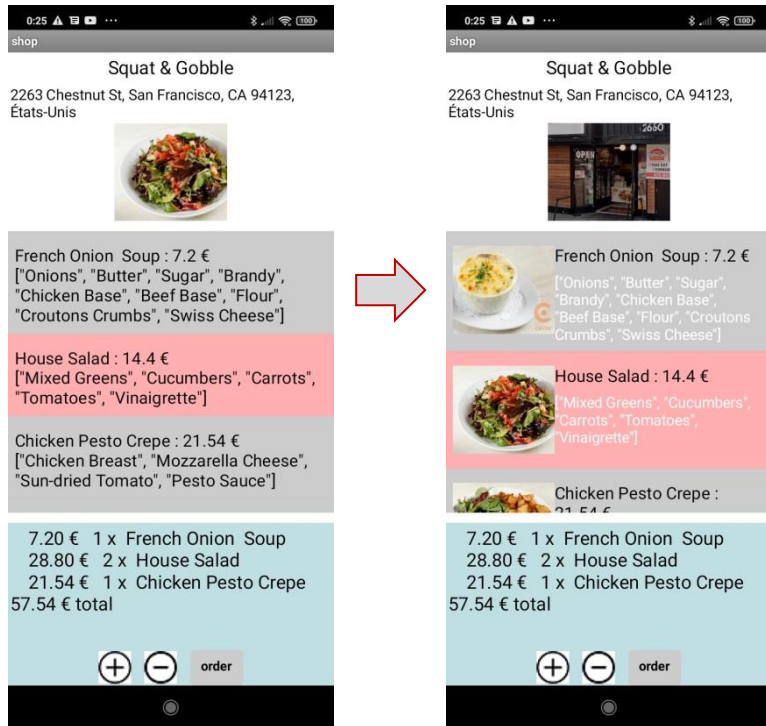
GITSHARE 3C : SHOP SCREEN – DISPLAY BONUS

- Display



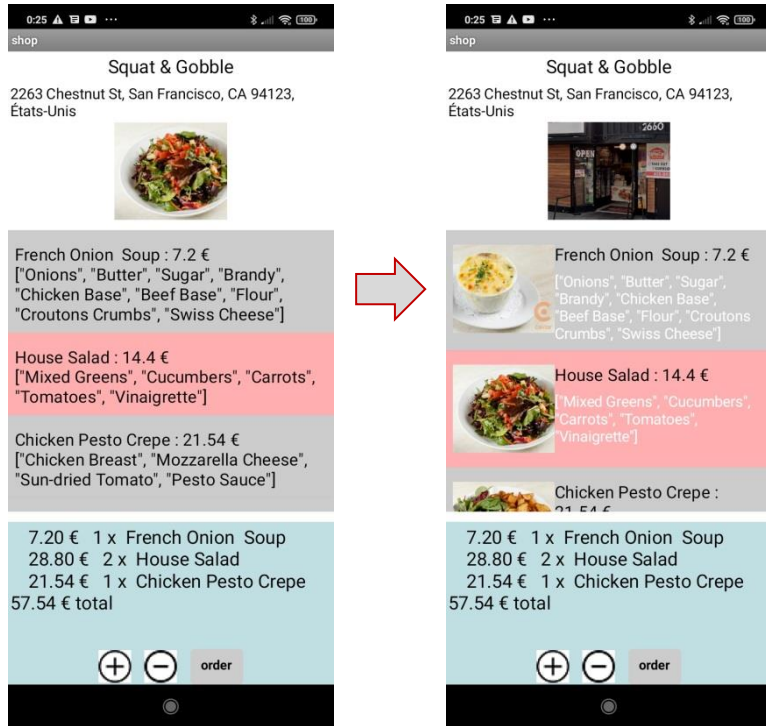
GITSHARE 3C : SHOP SCREEN – DISPLAY BONUS

- Display

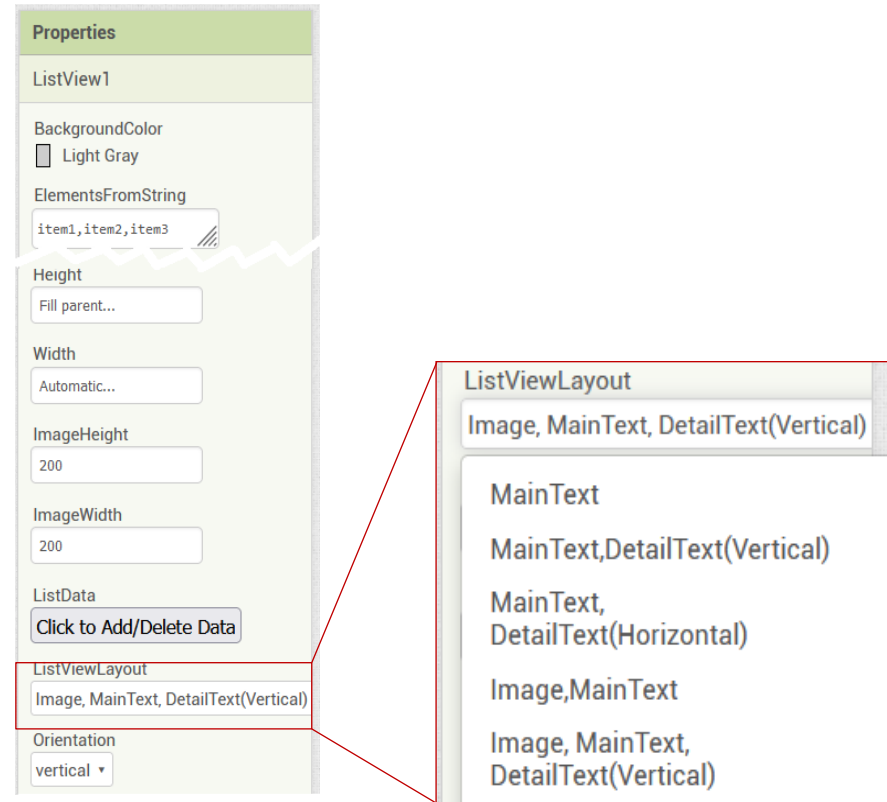


GITSHARE 3C : SHOP SCREEN – DISPLAY BONUS

- Display

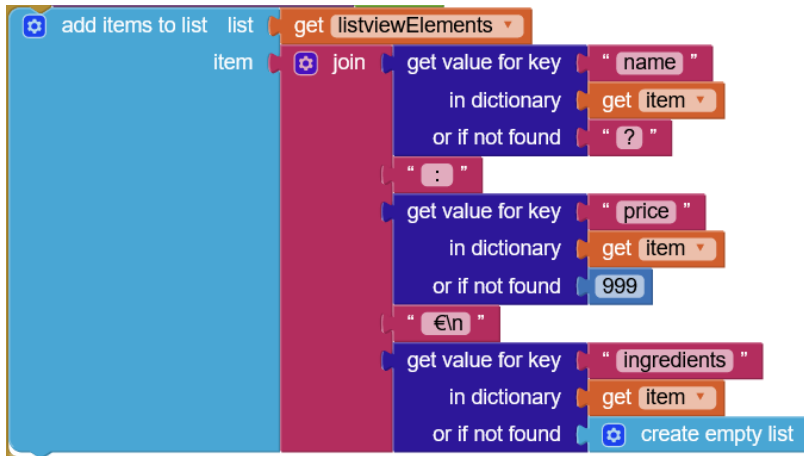


- Design



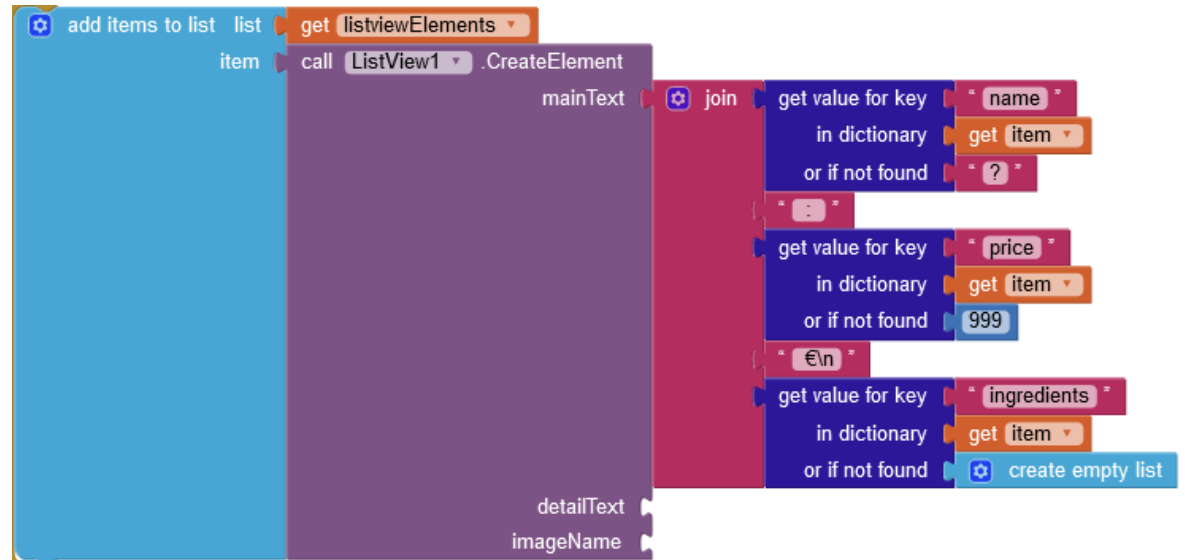
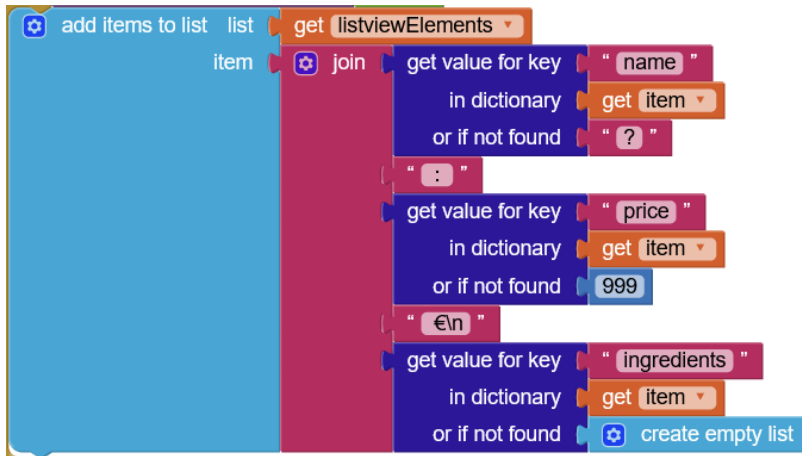
GITSHARE 3C : SHOP SCREEN – DISPLAY BONUS

- Blocks (procedure : listViewElements)



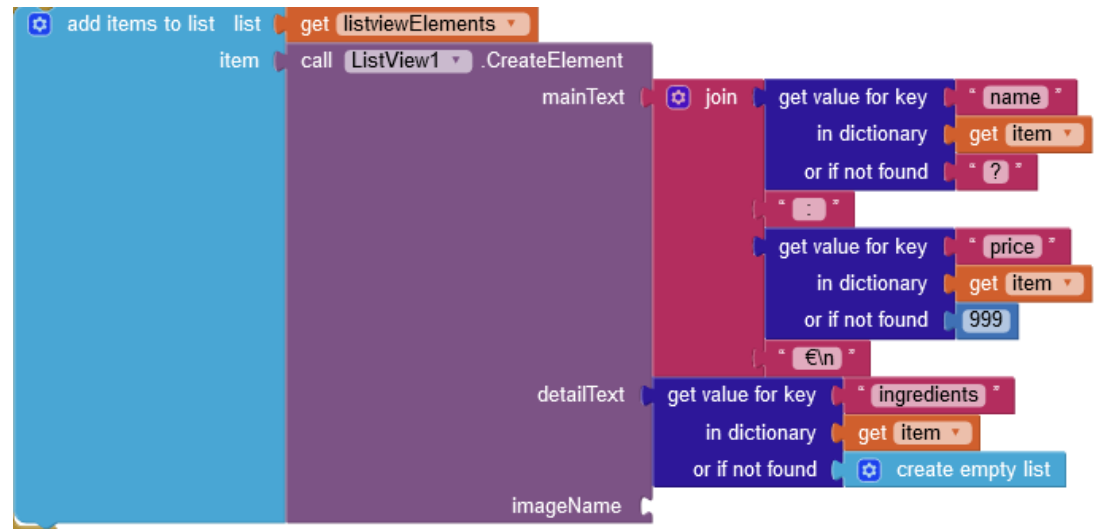
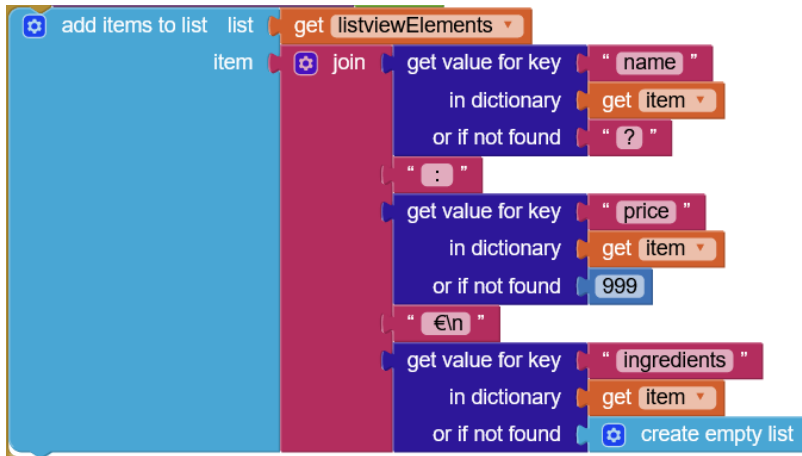
GITSHARE 3C : SHOP SCREEN – DISPLAY BONUS

- Blocks (procedure : listViewElements)



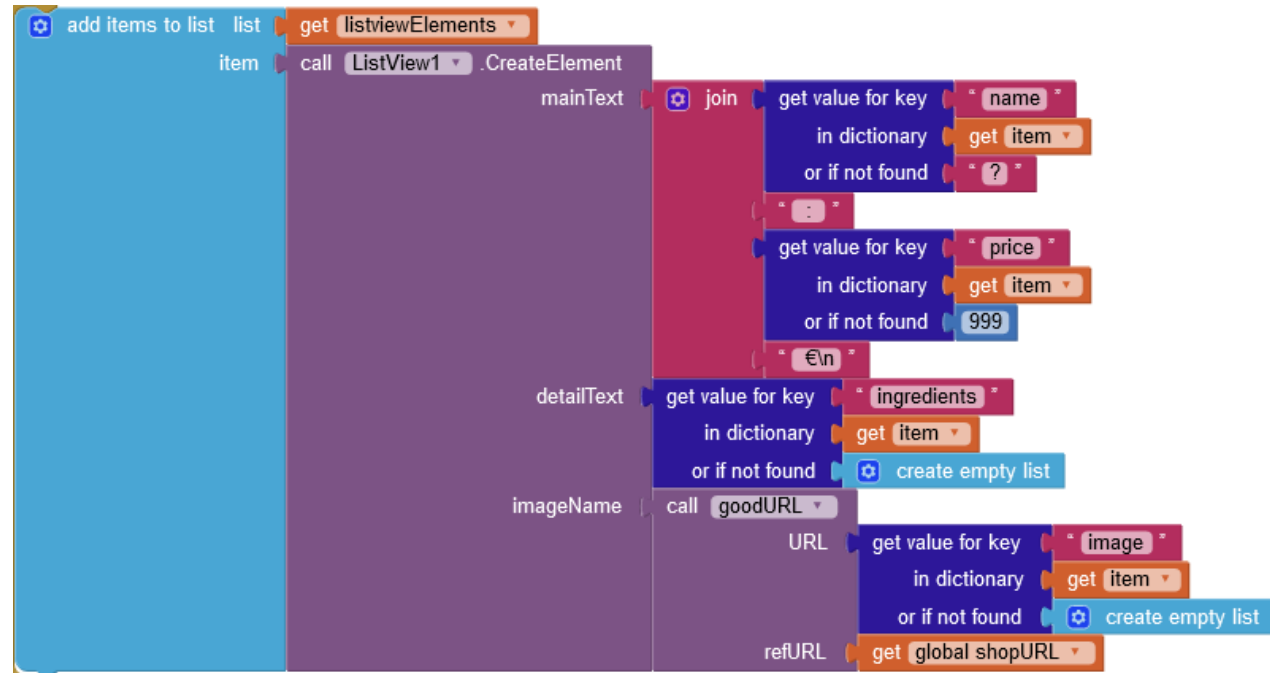
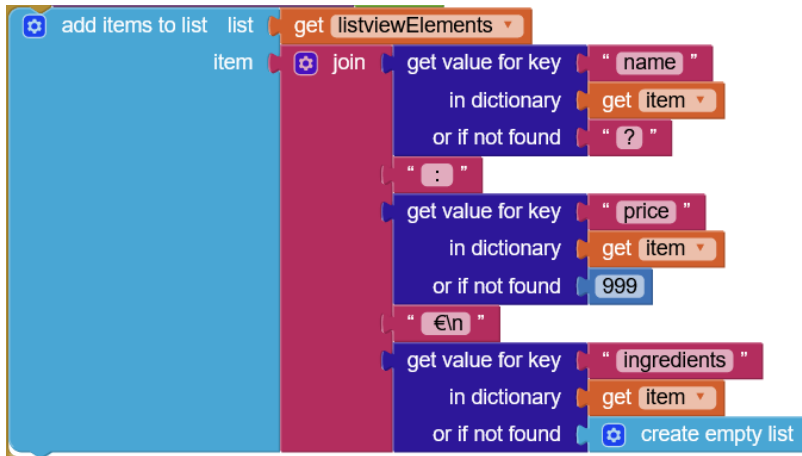
GITSHARE 3C : SHOP SCREEN – DISPLAY BONUS

- Blocks (procedure : listViewElements)



GITSHARE 3C : SHOP SCREEN – DISPLAY BONUS

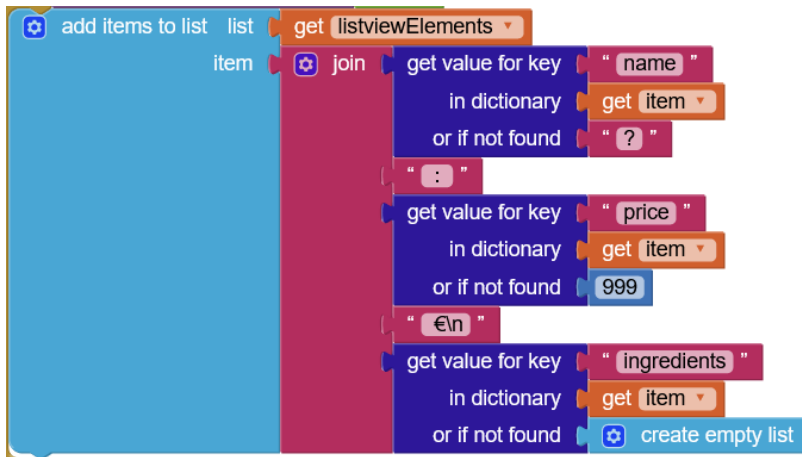
- Blocks (procedure : listViewElements)



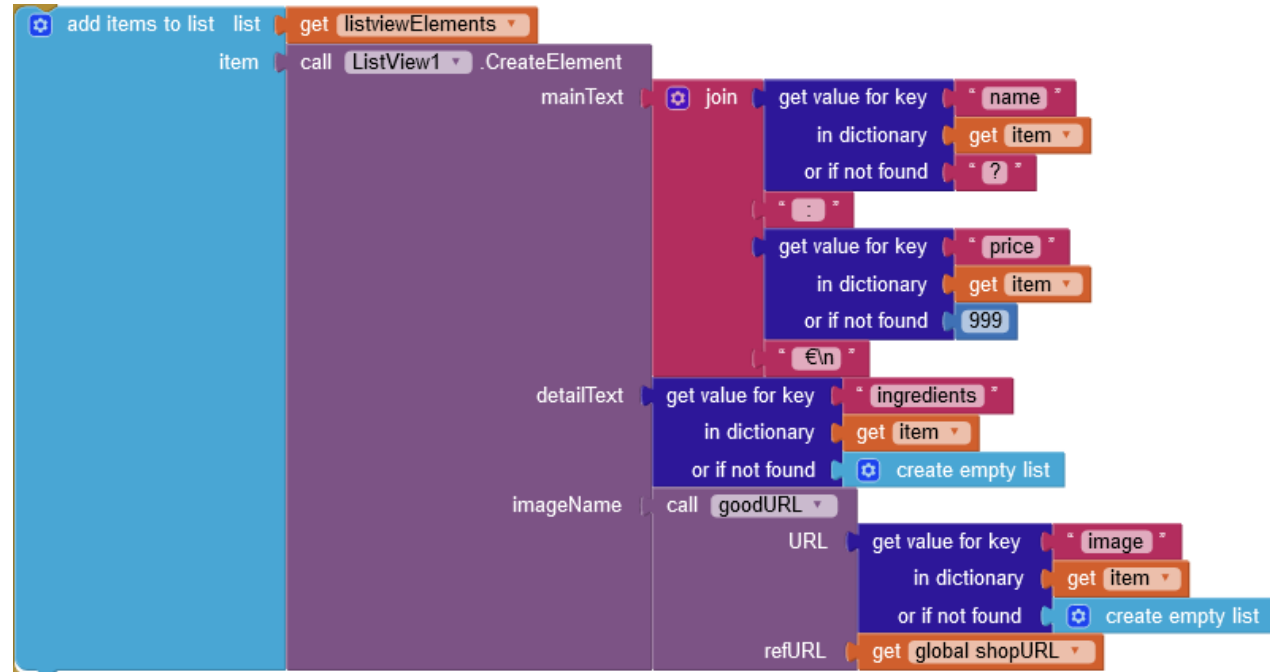
GITSHARE 3C : SHOP SCREEN – DISPLAY BONUS

- Blocks (procedure : listViewElements)

'display1' = true



'display1' = false



GITSHARE 3C : SHOP SCREEN – DISPLAY BONUS

```
to setListViewElements myItems
do
  initialize local listViewElements to create empty list
  in
    for each item in list get myItems
    do
      if is a dictionary? get item
      then
        if call TinyDB1 .GetValue tag "display1" valueIfTagNotThere true
        then
          add items to list list get listViewElements
          item
          join
            get value for key "name" in dictionary get item or if not found "?"
            " "
            get value for key "price" in dictionary get item or if not found 999
            " €\n"
            get value for key "ingredients" in dictionary get item or if not found create empty list
        else
          add items to list list get listViewElements
          item
          call ListView1 .CreateElement
          mainText
          join
            get value for key "name" in dictionary get item or if not found "?"
            " "
            get value for key "price" in dictionary get item or if not found 999
            " €\n"
          detailText
            get value for key "ingredients" in dictionary get item or if not found create empty list
          imageName
            call goodURL
            URL
            get value for key "image" in dictionary get item or if not found create empty list
            refURL
            get global shopURL
        else
          call Notifier1 .ShowMessageDialog message get item title "item is NOT a dictionary" buttonText "OK"
          break
      set ListView1 .Elements to get listViewElements
```



French Onion Soup : 7.2 €
["Onions", "Butter", "Sugar", "Brandy", "Chicken Base", "Beef Base", "Flour", "Croutons Crumbs", "Swiss Cheese"]

House Salad : 14.4 €
["Mixed Greens", "Cucumbers", "Carrots", "Tomatoes", "Vinaigrette"]



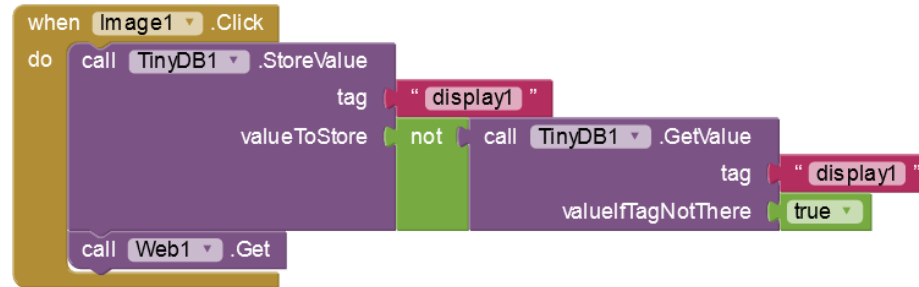
French Onion Soup : 7.2 €
["Onions", "Butter", "Sugar", "Brandy", "Chicken Base", "Beef Base", "Flour", "Croutons Crumbs", "Swiss Cheese"]

House Salad : 14.4 €
["Mixed Greens", "Cucumbers", "Carrots", "Tomatoes", "Vinaigrette"]

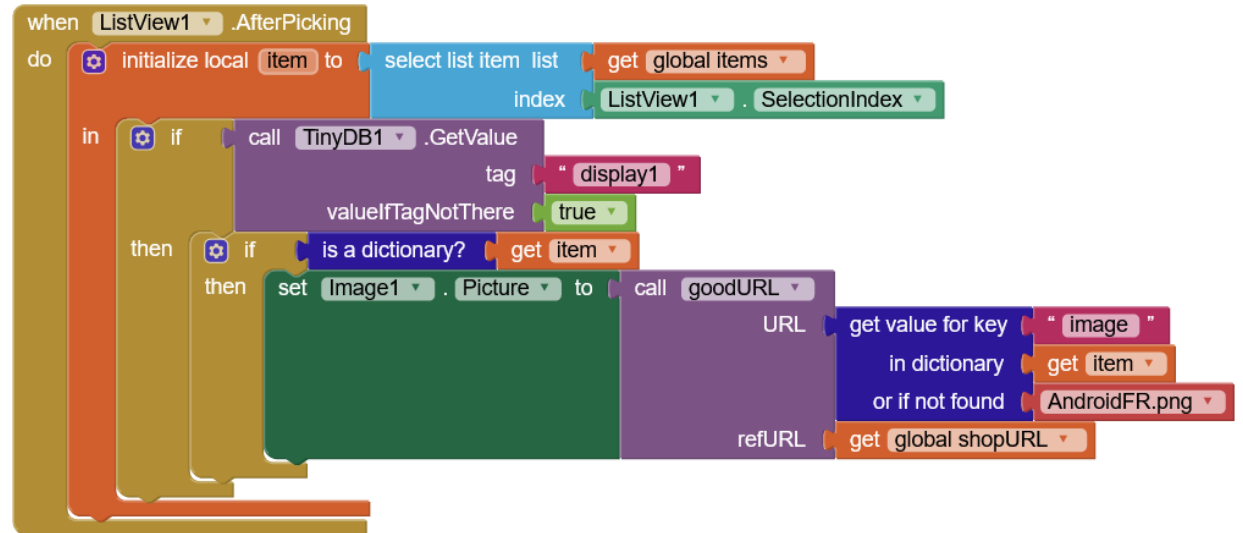
Chicken Pesto Crepe :
21.54 €

GITSHARE 3C : SHOP SCREEN – DISPLAY BONUS

- Image click
 - inverse display1
 - refresh : web.get



- Listview after picking
 - Replace top image only if display1 = false



Many images \Rightarrow delay

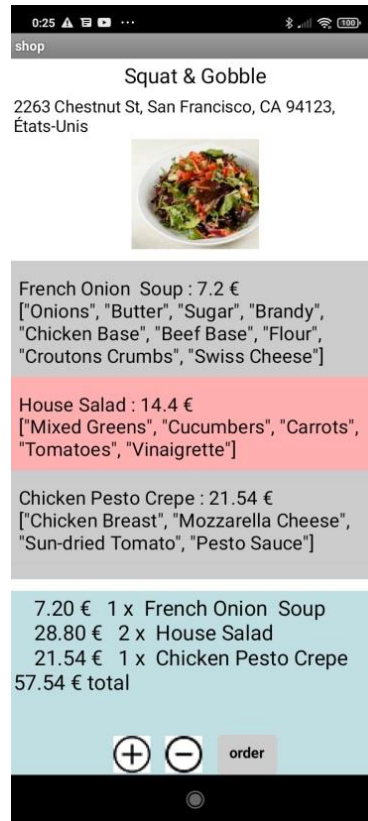
GITSHARE 3C : SHOP SCREEN – DISPLAY BONUS

- Image click
 - inverse display1
 - refresh : web.get
- Listview after picking
 - Replace top image only if display1 = false

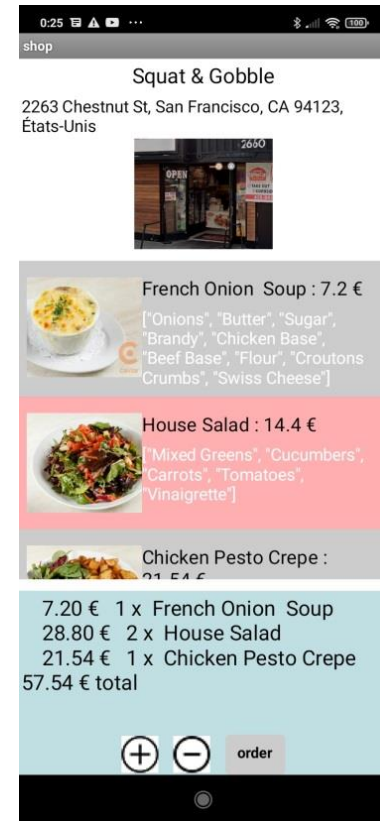


Many images ⇒ delay

'display1' = true



'display1' = false

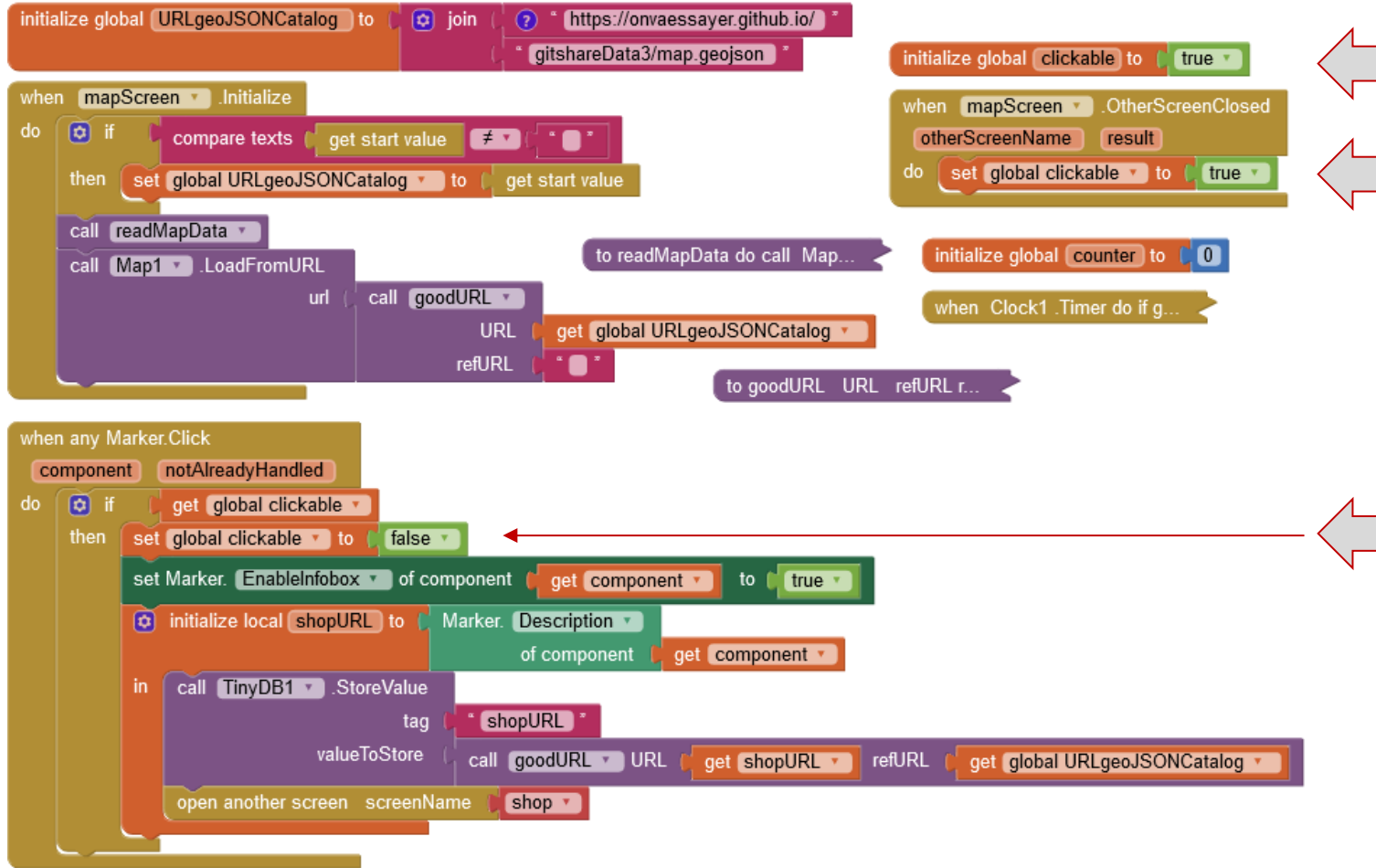


GITSHARE 3C : SHOP SCREEN – DISPLAY BONUS

The image displays a collection of code blocks for an Android application, organized into several functional sections:

- Initialization:** Blocks for initializing global variables like `shop`, `items`, and `shopURL` as empty dictionaries or lists.
- Web API Interaction:** A `when Web1 .GotText` block that checks the response code (200) and decodes the JSON response. It uses `get value for key` to extract item details like title, address, and image URLs.
- Image Handling:** A `when Image1 .Click` block that calls `goodURL` to get image data and `TinyDB1.StoreValue` to save it.
- UI Updates:** `when ListView1 .AfterPicking` and `when shop .ErrorOccured` blocks for handling user selections and errors.
- Ordering Logic:** `when ButtonAdd .Click` and `when ButtonSubtract .Click` blocks that call `addToSelectedItem` with quantities of 1 and -1 respectively.
- Order Confirmation:** A `when Notifier1 .AfterChoosing` block that compares the choice to "confirm". If confirmed, it calls `TinyDB1` to get user details and formats a date-time string.
- Dialog and Text Updates:** A `when ButtonOrder .Click` block that shows a `Notifier1 .ShowChooseDialog` with a message, title, and buttons. It also includes `replace all text` blocks for updating UI elements like `orderHeaderText` and `orderContentText`.

GITSHARE 3C : MAPSCREEN



GITSHARE 3C : SCREEN1

```
initialize global URLgeoJSONCatalog to join [
  "https://onvaessayer.github.io/"
  "gitshareData3/map_geojson"
]

when Screen1.Initialize
do
  set TextBoxCatalog . Text to call goodURL .
  URL get global URLgeoJSONCatalog
  rootURL .
  call readUserData
  call loadCatalogOfCatalogs
  set SpinnerListOfCatalogs . Elements to get keys get global goodGeojson
  set Image1 . Clickable to true

when ButtonStart . Click
do
  if and [
    not is empty TextBoxName . Text
    not is empty TextBoxAddress . Text
    not is empty TextBoxEmail . Text
    not is empty TextBoxTelephone . Text
    not is empty TextBoxCatalog . Text
  ]
  then
    if and [
      compare texts ButtonStart . Text = "Start"
      compare texts TextBoxTelephone . Text =
        get global checkNumber
    ]
    then
      initialize local user to make a dictionary
      key "name" value TextBoxName . Text
      key "address" value TextBoxAddress . Text
      key "email" value TextBoxEmail . Text
      key "telephone" value TextBoxTelephone . Text
      key "catalog" value TextBoxCatalog . Text
      in call TinyDB1 . StoreValue tag "user" valueToStore get user
      open another screen with start values screenName mapScreen
      startValue call goodURL .
      URL TextBoxCatalog . Text
      rootURL .
    else
      call checkTelephone
    else
      call Notifier1 . ShowAlert notice "missing name, email, address, telephone or catalog"
```

```
initialize global goodGeojson to { "select data" : "data" }

to goodURL URL rootURL
result do if starts at text get U...

to readUserData
do
  initialize local user to call TinyDB1 . GetValue
  tag "user"
  valueIfTagNotThere create empty dictionary
  in
    set TextBoxName . Text to get value for key "name" in dictionary get user or if not found .
    set TextBoxAddress . Text to get value for key "address" in dictionary get user or if not found .
    set TextBoxEmail . Text to get value for key "email" in dictionary get user or if not found .
    set TextBoxTelephone . Text to get value for key "telephone" in dictionary get user or if not found .
    set global checkNumber to TextBoxTelephone . Text
    set TextBoxCatalog . Text to get value for key "catalog"
    in dictionary get user
    or if not found get global URLgeoJSONCatalog
    set CheckBoxDebug . Enabled to call TinyDB1 . GetValue
    tag "debug"
    valueIfTagNotThere true

when SpinnerListOfCatalogs...
when CheckBoxDebug . Change...

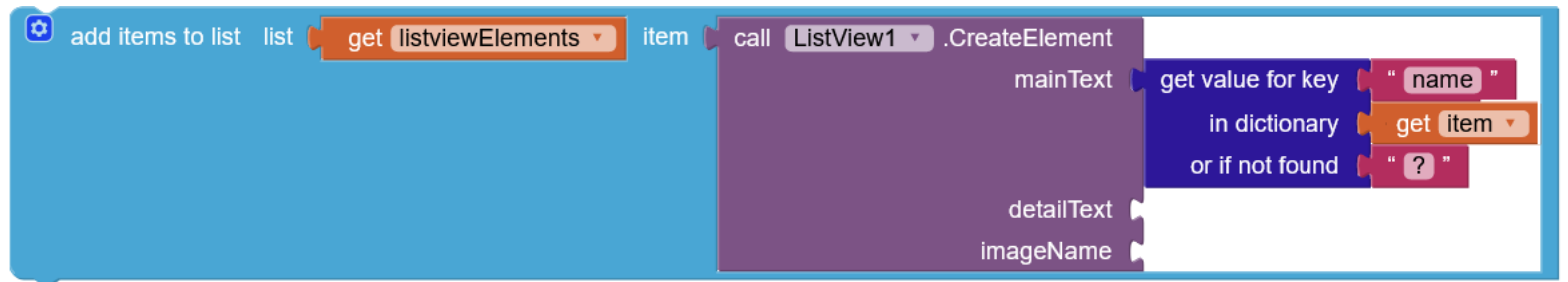
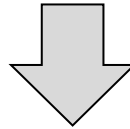
when Image1 . Click
do
  if compare texts Image1 . P...
  initialize global flascode to join [
    "https://chart.googleapis.com/chart?"
    "cht=qr&chs=320x320&chl="
    "https://onvaessayer.github.io/gitshare.apk"
  ]

initialize global URLcatalogOfCatalogs to join [
  "https://onvaessayer.github.io/"
  "catalogOfCatalogs.json"
]
to loadCatalogOfCatalogs do...
when Web1 . GotText url ...

initialize global secret to 1
to checkTelephone do set B...

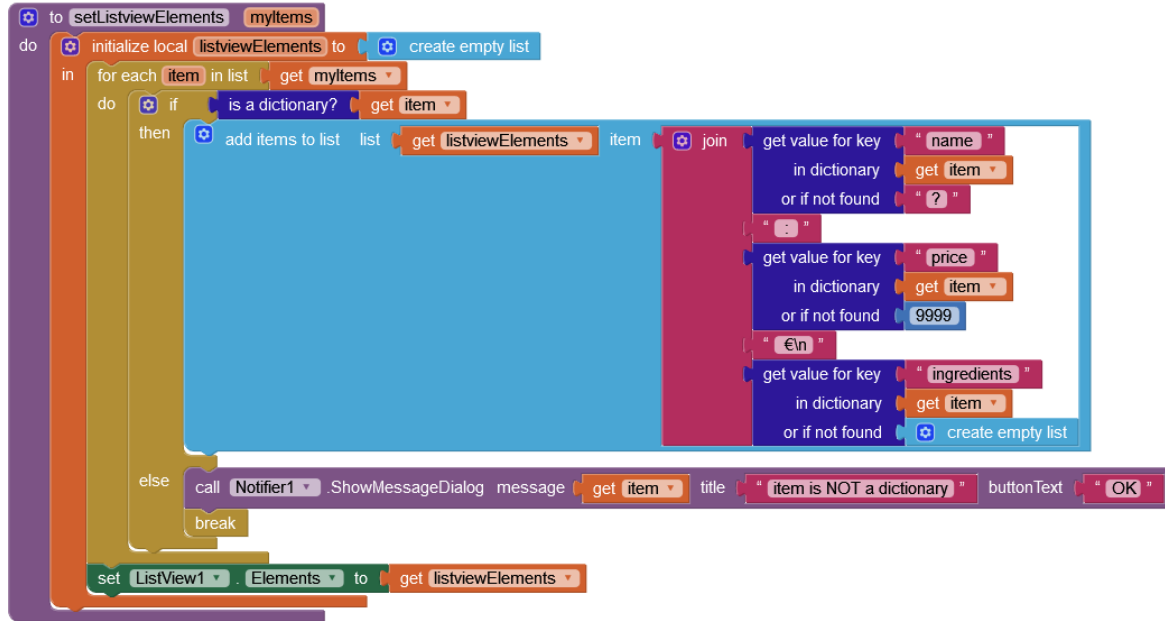
initialize global checkNumber to .
when Texting1 . MessageRece...
```


GITSHARE3d : LISTVIEW COMPONENT / DESIGN PROPERTIES

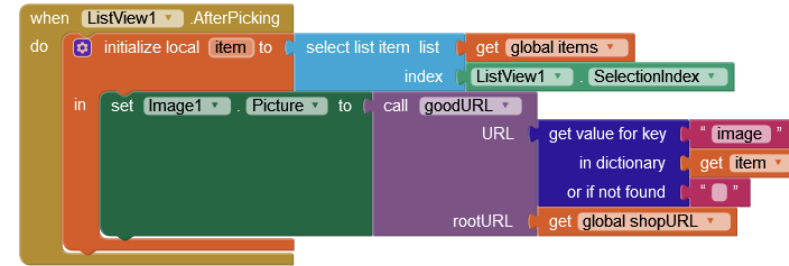


GITSHARE3B → 3d : LISTVIEW COMPONENT / BLOCKS (SHOP SCREEN)

Shop screen Version 3C

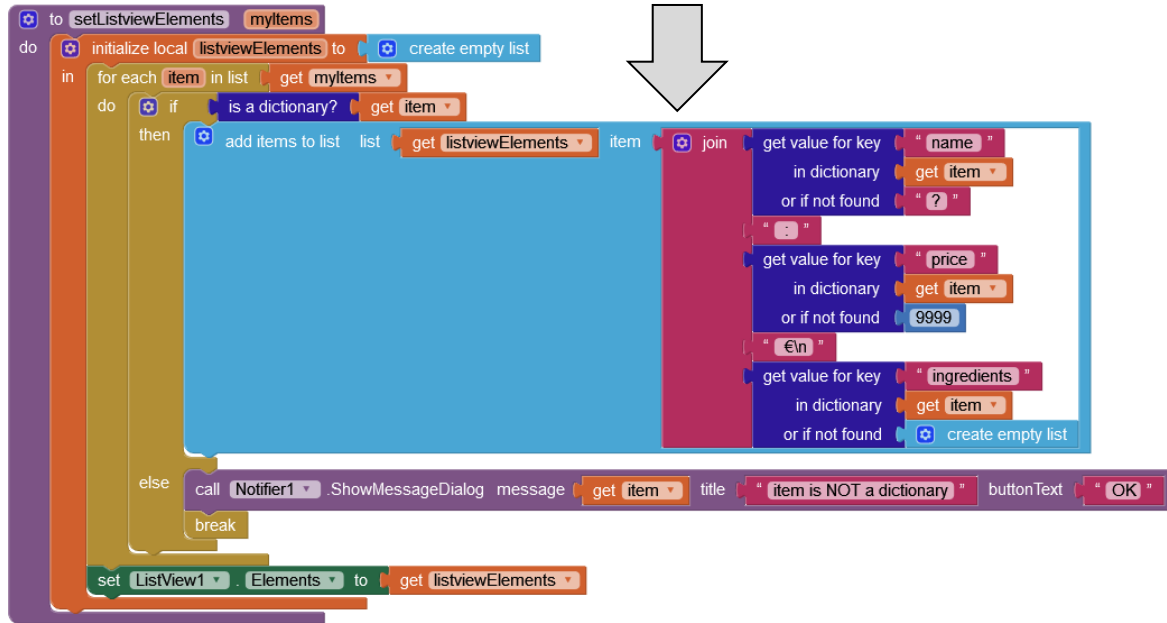


```
to setListViewElements myItems
do
  initialize local listViewElements to create empty list
  in for each item in list get myItems
  do
    if is a dictionary? get item
    then
      add items to list list get listViewElements item
      join
        get value for key "name"
        in dictionary get item
        or if not found "?"
        "."
        get value for key "price"
        in dictionary get item
        or if not found "9999"
        "€\n"
        get value for key "ingredients"
        in dictionary get item
        or if not found create empty list
    else
      call Notifier1.ShowDialog message get item title "item is NOT a dictionary" buttonText "OK"
      break
  set ListView1.Elements to get listViewElements
```



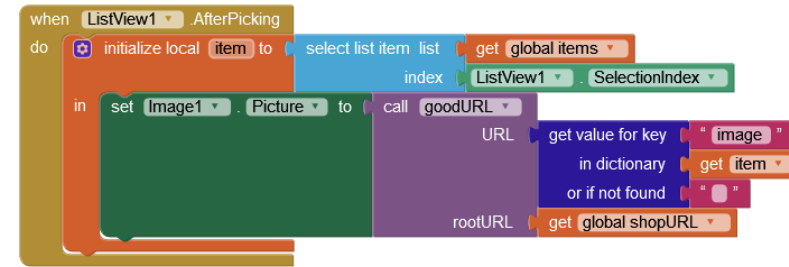
```
when ListView1.AfterPicking
do
  initialize local item to select list item list get global items
  index ListView1.SelectionIndex
  in set Image1.Picture to call goodURL
  URL
  get value for key "image"
  in dictionary get item
  or if not found ""
  rootURL get global shopURL
```


GITSHARE3B → 3d : LISTVIEW COMPONENT / BLOCKS (SHOP SCREEN)



```
to setListViewElements myItems
do
  initialize local listviewElements to create empty list
  in for each item in list get myItems
  do
    if is a dictionary? get item
    then
      add items to list list get listviewElements item
      join
      get value for key "name"
      in dictionary get item
      or if not found "?"
      "."
      get value for key "price"
      in dictionary get item
      or if not found 9999
      "€\n"
      get value for key "ingredients"
      in dictionary get item
      or if not found create empty list
    else
      call Notifier1 .ShowMessageDialog message get item title "item is NOT a dictionary" buttonText "OK"
      break
  set ListView1 . Elements to get listviewElements
```

The code block on the left is a Scratch script for a function named 'setListViewElements' that takes 'myItems' as an argument. It starts by initializing a local variable 'listviewElements' to an empty list. Then, it enters a loop 'in for each item in list' where it retrieves 'myItems'. Inside this loop, it checks 'if is a dictionary? get item'. If true, it performs several actions: 'add items to list list' with 'get listviewElements item', a 'join' block, and three 'get value for key' blocks for 'name', 'price', and 'ingredients'. Each 'get value for key' block has an 'in dictionary get item' block and an 'or if not found' block with default values: '?', '9999', and 'create empty list'. Additionally, there are string blocks for '.' and '€\n'. If the item is not a dictionary, it calls 'Notifier1 .ShowMessageDialog' with a message 'get item', title 'item is NOT a dictionary', and buttonText 'OK', then breaks the loop. Finally, it sets 'ListView1 . Elements' to 'get listviewElements'. A large grey arrow points from the 'add items to list' block to the right.



```
when ListView1 AfterPicking
do
  initialize local item to select list item list get global items
  index ListView1 . SelectionIndex
  in set Image1 . Picture to call goodURL
  URL
  get value for key "image"
  in dictionary get item
  or if not found ""
  rootURL get global shopURL
```

The code block on the right is a Scratch script for a 'when ListView1 AfterPicking' event. It starts with 'initialize local item to select list item list get global items' and 'index ListView1 . SelectionIndex'. Then, it enters a loop 'in set Image1 . Picture to call goodURL'. Inside this loop, it has three 'get value for key' blocks for 'image', 'image', and 'rootURL'. Each 'get value for key' block has an 'in dictionary get item' block and an 'or if not found' block with default values: '', '', and 'get global shopURL'.

GITSHARE3B → 3d : LISTVIEW COMPONENT / BLOCKS (SHOP SCREEN)

Listview create element

```
to setListViewElements myItems
do
  initialize local listviewElements to create empty list
  in for each item in list get myItems
  do
    if is a dictionary? get item
    then
      add items to list list get listviewElements item
      call ListView1 .CreateElement
      mainText
      detailText
      imageName
    else
      call Notifier1 .ShowMessageDialog message get item title "item is NOT a dictionary" buttonText "OK"
      break
  set ListView1 .Elements to get listviewElements

when ListView1 .AfterPicking
do
  initialize local item to select list item list get global items index ListView1 .SelectionIndex
  in set Image1 .Picture to call goodURL
  URL
  get value for key "image"
  in dictionary get item
  or if not found ""
  rootURL get global shopURL
```

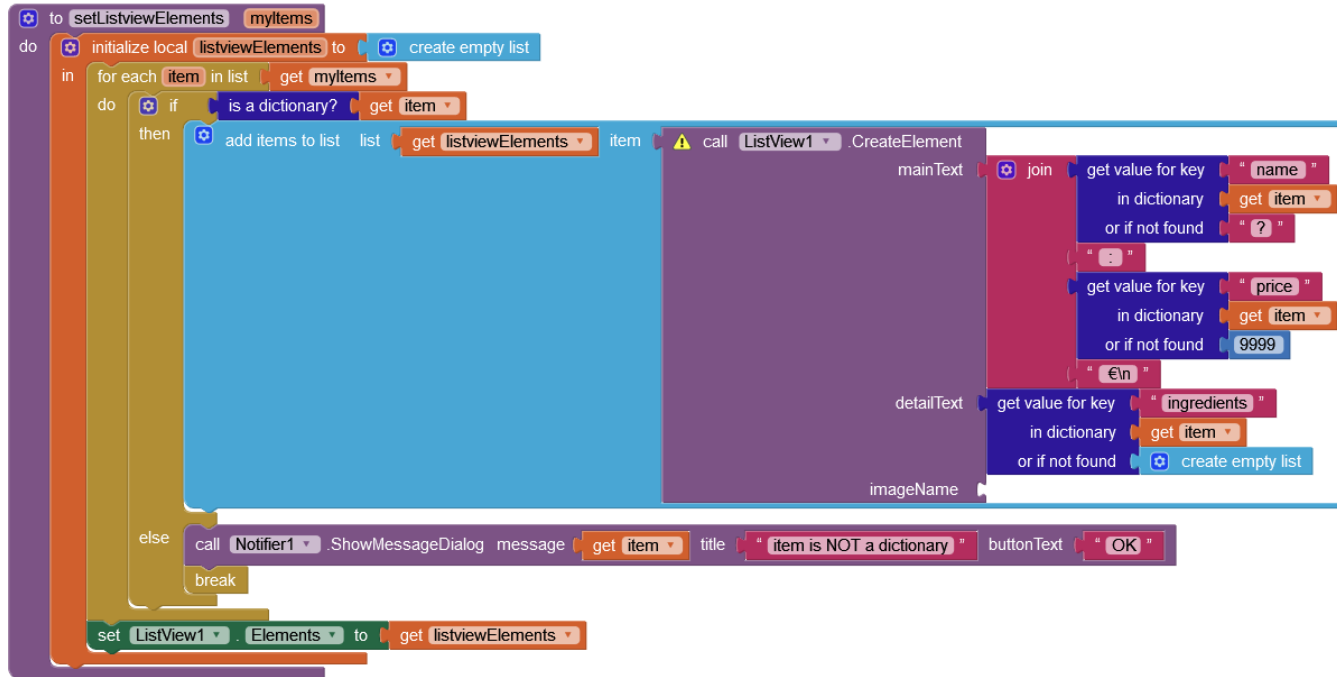
GITSHARE3B → 3d : LISTVIEW COMPONENT / BLOCKS (SHOP SCREEN)

```
to setListViewElements myItems
do
  initialize local listViewElements to create empty list
  in for each item in list get myItems
  do
    if is a dictionary? get item
    then
      add items to list list get listViewElements item
      call ListView1 .CreateElement
      mainText
      join
      get value for key "name"
      in dictionary get item
      or if not found "?"
      " "
      get value for key "price"
      in dictionary get item
      or if not found "9999"
      "€\n"
      get value for key "ingredients"
      in dictionary get item
      or if not found create empty list
      detailText
      imageName
    else
      call Notifier1 .ShowMessageDialog message get item title "item is NOT a dictionary" buttonText "OK"
      break
  set ListView1 .Elements to get listViewElements
```

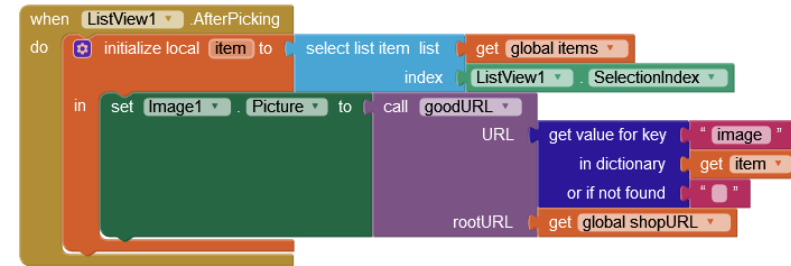
```
when ListView1 .AfterPicking
do
  initialize local item to select list item list get global items
  index ListView1 .SelectionIndex
  in set Image1 .Picture to call goodURL
  URL
  get value for key "image"
  in dictionary get item
  or if not found ""
  rootURL get global shopURL
```

Ingredients → detail

GITSHARE3B → 3d : LISTVIEW COMPONENT / BLOCKS (SHOP SCREEN)



```
to setListViewElements myItems
do
  initialize local listViewElements to create empty list
  in for each item in list get myItems
  do
    if is a dictionary? get item
    then
      add items to list list get listViewElements item
      call ListView1 .CreateElement
      mainText
      join
      get value for key "name"
      in dictionary get item
      or if not found "?"
      " "
      get value for key "price"
      in dictionary get item
      or if not found 9999
      " €\n"
      detailText
      get value for key "ingredients"
      in dictionary get item
      or if not found create empty list
      imageName
    else
      call Notifier1 .ShowMessageDialog message get item title "item is NOT a dictionary" buttonText "OK"
      break
  set ListView1 .Elements to get listViewElements
```



```
when ListView1 .AfterPicking
do
  initialize local item to select list item list get global items
  index ListView1 .SelectionIndex
  in set Image1 .Picture to call goodURL
  URL
  get value for key "image"
  in dictionary get item
  or if not found ""
  rootURL get global shopURL
```

GITSHARE3B → 3d : LISTVIEW COMPONENT / BLOCKS (SHOP SCREEN)

The code block 'to setListViewElements myItems' is a function that processes a list of items. It starts with 'do initialize local listviewElements to create empty list'. Then, it enters a 'for each item in list get myItems' loop. Inside the loop, it checks 'if is a dictionary? get item'. If true, it performs 'add items to list list get listviewElements item' and then calls 'call ListView1 .CreateElement'. The 'CreateElement' block has three fields: 'mainText' (a 'join' block with 'get value for key name in dictionary or if not found ?', 'get item', and '9999'), 'detailText' ('get value for key ingredients in dictionary or if not found create empty list'), and 'imageName' (empty). If the item is not a dictionary, it calls 'call Notifier1 .ShowMessageDialog message get item title item is NOT a dictionary buttonText OK' and then 'break'. Finally, it sets 'set ListView1 .Elements to get listviewElements'.

The code block 'when ListView1 .AfterPicking' is an event listener. It starts with 'do initialize local item to select list item list get global items index ListView1 .SelectionIndex'. Then, it enters a 'set Image1 .Picture to call goodURL' block. The 'goodURL' block has two fields: 'URL' ('get value for key image in dictionary or if not found') and 'rootURL' ('get global shopURL').

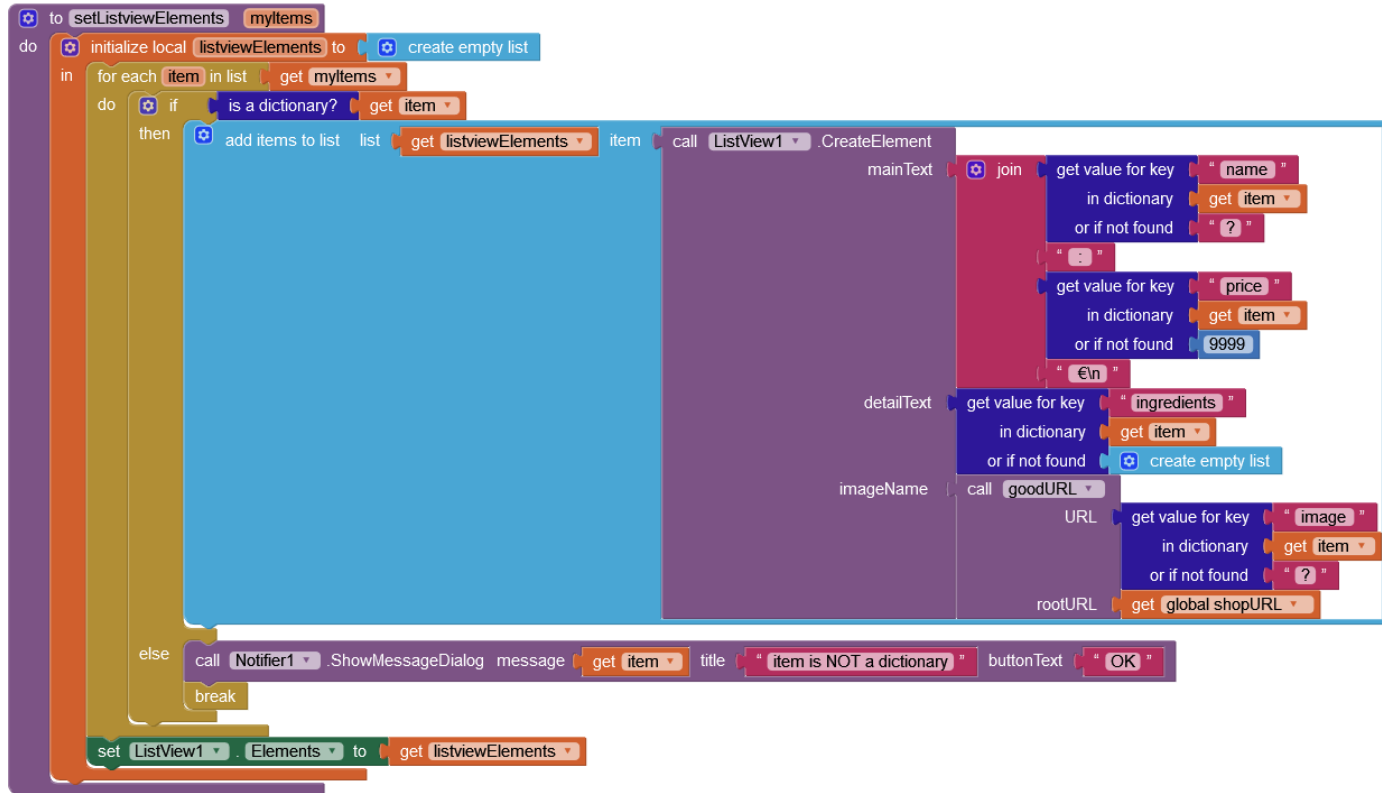
URL image

GITSHARE3B → 3d : LISTVIEW COMPONENT / BLOCKS (SHOP SCREEN)

The code block 'to setListViewElements myItems' is a 'do' loop. It starts with 'initialize local listviewElements to create empty list'. Then it enters an 'in for each (item) in list get myItems' loop. Inside, there is an 'if is a dictionary? get item' block. The 'then' branch contains a 'do' loop: 'add items to list list get listviewElements item' followed by a 'call ListView1 .CreateElement' block. This call block has three arguments: 'mainText', 'detailText', and 'imageName'. 'mainText' is a 'join' block with three parts: a 'get value for key "name" in dictionary get item or if not found "?"' block, a '" - "' block, and a 'get value for key "price" in dictionary get item or if not found 9999' block. 'detailText' is a 'get value for key "ingredients" in dictionary get item or if not found create empty list' block. 'imageName' is a 'call goodURL' block with 'URL' and 'rootURL' arguments. 'URL' is a 'get value for key "image" in dictionary get item or if not found "?"' block, and 'rootURL' is a 'get global shopURL' block. The 'else' branch of the 'if' block contains a 'call Notifier1 .ShowMessageDialog message get item title "item is NOT a dictionary" buttonText "OK"' block and a 'break' block. Finally, the 'do' loop ends with 'set ListView1 .Elements to get listviewElements'.

The code block 'when ListView1 .AfterPicking' is a 'do' loop. It starts with 'initialize local item to select list item list get global items index ListView1 .SelectionIndex'. Then it enters an 'in set Image1 .Picture to call goodURL' block. This block has 'URL' and 'rootURL' arguments. 'URL' is a 'get value for key "image" in dictionary get item or if not found "?"' block, and 'rootURL' is a 'get global shopURL' block.

GITSHARE3B → 3d : LISTVIEW COMPONENT / BLOCKS (SHOP SCREEN)



GITSHARE3B → 3d : LISTVIEW COMPONENT / BLOCKS (SHOP SCREEN)

Shop screen Version 3D

The image displays a Scratch script for a list view component. The script is organized as follows:

- Initialization:** A 'do' block containing 'initialize local listviewElements to create empty list'.
- Loop:** An 'in for each item in list get myItems' loop.
- Conditional Logic:** An 'if is a dictionary? get item' block. If true, it proceeds to the 'then' block; otherwise, it goes to the 'else' block.
- Then Block:** Contains 'add items to list list get listviewElements item join get value for key "name"', followed by a call to 'ListView1 .CreateElement'. This call has three arguments:
 - mainText:** A 'join' block with 'get value for key "name" in dictionary get item or if not found "?"', a separator ' " : "', 'get value for key "price" in dictionary get item or if not found "9999"', and a separator ' €\n '.
 - detailText:** 'get value for key "ingredients" in dictionary get item or if not found create empty list'.
 - imageName:** A call to 'goodURL .URL' with 'get value for key "image" in dictionary get item or if not found "?"' and 'get global shopURL'.
- Else Block:** 'call Notifier1 .ShowMessageDialog message get item title "item is NOT a dictionary" buttonText "OK"', followed by 'break'.
- Final Step:** 'set ListView1 .Elements to get listviewElements'.

Annotations on the right side of the image:

- An arrow points from the text 'Blocks version3C (collapse + disabled)' to a 'when ListView1 .AfterPicking do initialize local item to se...' block.
- Another arrow points from the text 'Blocks version3d' to the 'call ListView1 .CreateElement' block.

JSON & geoJSON links:

cours : <http://onvaessayer.org/appinventor?app=json>

videos : <http://onvaessayer.org/appinventor?video=json>

<http://onvaessayer.org/appinventor?video=geojson>

application resources : <http://onvaessayer.org/appinventor?res=gitshare>

App Inventor repository : <http://onvaessayer.org/appinventor/apps/gitshare/>

App Inventor starter:

cours : <http://onvaessayer.org/appinventor?app=getstarted>

video : <http://onvaessayer.org/appinventor?video=getstarted>

Dropbox video : <http://onvaessayer.org/appinventor?video=dropbox>

tinyDB : <http://onvaessayer.org/appinventor?video=tinyDB>

Version 1 URL : geoJSONcatalog

<https://onvaessayer.github.io/gitshareData1/map.geojson>

initialize global URLgeoJSONCatalog to `“s://onvaessayer.github.io/gitshareData1/map.geojson”`

Geo-Loc : <http://onvaessayer.org/appinventor?video=geoloc1>

Email : pierre.huguet50@gmail.com

Cours : <http://onvaessayer.org/appinventor?app=gitshare>

Vidéo : <http://onvaessayer.org/appinventor?video=gitshare>

Playlist : <http://onvaessayer.org/appinventor?playlist=gitshare>

Applications de base : <http://onvaessayer.org/appinventor?app=baseapps>

CLICK & COLLECT

Build your own mobile app

