**Faculty: Business Information Technology**

**645-2 Flutter project – Technical guide**

Une image contenant clipart, Graphique, symbole, graphisme

Description générée automatiquement

Students: Milena Lonfat, Luca Del Buono, Elias Borrajo, Alexandre Martroye De Joly, Arnaud Michelet

Professors: Michael Schumacher & Alexandre Cotting

Submitted on 09.09.2023

**TABLE OF CONTENTS**

[INTRODUCTION 1](#_Toc145152677)

[1. Mockups 1](#_Toc145152678)

[2. Data Structure 4](#_Toc145152679)

[2.1. Filters 5](#_Toc145152680)

[2.2. Migrations 5](#_Toc145152681)

[2.3. MuseumObjects 6](#_Toc145152682)

[2.4. Museums 6](#_Toc145152683)

[2.5. Quiz 7](#_Toc145152684)

[2.6. QuizPlayers 7](#_Toc145152685)

[3. Frontend 7](#_Toc145152686)

[3.1. Development tools 7](#_Toc145152687)

[3.2. Architecture 8](#_Toc145152688)

[3.3. Libraries 10](#_Toc145152689)

[3.4. Virtual device configuration 10](#_Toc145152690)

[4. Configuration 11](#_Toc145152691)

[5. Deployment 12](#_Toc145152692)

[6. Admin credentials 12](#_Toc145152693)

[7. List of URLs 13](#_Toc145152694)

[8. Appendix 13](#_Toc145152695)

[8.1. Mockups 13](#_Toc145152696)

**LIST OF FIGURES**

[Figure 1: Application home page 1](#_Toc145152697)

[Figure 2: Details of the selected object 2](#_Toc145152698)

[Figure 3: Quiz on exhibition objects 2](#_Toc145152699)

[Figure 4: The result of the quiz can be sent by email 3](#_Toc145152700)

[Figure 5: Connection to the admin console 3](#_Toc145152701)

[Figure 6: Editing of an object 4](#_Toc145152702)

[Figure 7: Six main objects of the database 4](#_Toc145152703)

[Figure 8: “filters” object with his “typeName” and different possible “options” 5](#_Toc145152704)

[Figure 9: "migrations" object and his attributes 5](#_Toc145152705)

[Figure 10: "museumObject" and his attributes¨ 6](#_Toc145152706)

[Figure 11: "museum" and his attributes 6](#_Toc145152707)

[Figure 12: "quiz" and his attributes 7](#_Toc145152708)

[Figure 13: "quizPlayers" object and his attributes 7](#_Toc145152709)

[Figure 14: Global architecture of the project 8](#_Toc145152710)

[Figure 15: Modification in "local.properties" file 8](#_Toc145152711)

[Figure 16: "lib" and his sub-folders 9](#_Toc145152712)

[Figure 17: "admin\_forms" and his sub-folders 9](#_Toc145152713)

**LIST OF ACRONYMS:**

CRUD: Create, read, update, delete

INTRODUCTION

The objective of the project was to develop a mobile application for Valais museums. The aim of this application is to use pointers on a map to display the origin and exhibition location of historical objects. When clicking on the pointer, the user receives more information about the selected object. It is also possible to reduce the research spectrum by using filters that separate different categories of objects. After finding out about the various objects, users can take a quiz and send the results by e-mail.

The application also contains an admin interface in which administrators can carry out all necessary operations to maintain the application.

This technical report contains all the information needed to understand the solution that has been conceived in agreement with the client, to set it up in a new development environment and/or to deploy it on a server.

1. Mockups

To conceive this application, mockups have been made to easily communicate with the client and clarify his needs. In the following pictures, one can understand the basic interfaces that have been agreed. All the mockups are available in the appendix. It should be noted that these are only mockups, and that the final product may have changed in the meantime.

Launching the app, the user directly arrives on a map (Fig. 1).

Une image contenant texte, multimédia, carte

Description générée automatiquement

Figure 1: Application home page

Clicking on a pointer available on the map, a window pops up displaying the details of the selected object (Fig. 2).

Une image contenant texte, capture d’écran, carte

Description générée automatiquement

Figure 2: Details of the selected object

When clicking on the “Quiz” button of the home page, the user can start a quiz about the objects he found out just before (Fig. 3).

Une image contenant texte, capture d’écran, multimédia, affichage

Description générée automatiquement

Figure 3: Quiz on exhibition objects

Once the quiz is done, the user can send the results to his email address or come back to the home page to discover more (Fig. 4).

Une image contenant texte, capture d’écran, multimédia, affichage

Description générée automatiquement

Figure 4: The result of the quiz can be sent by email

From the administrator's point of view, a button on the home page allow to access to the admin console by entering login details (Fig. 5).

Une image contenant capture d’écran, texte, affichage, Rectangle

Description générée automatiquement

Figure 5: Connection to the admin console

From the admin console, the administrator can add, remove, or edit the different contents of the application (Fig. 6).

Une image contenant texte, capture d’écran, nombre, Parallèle

Description générée automatiquement

Figure 6: Editing of an object

1. Data Structure

To ensure that our mobile application is responsive, easy to deploy and maintain, we have chosen to use Firebase Realtime Database. This cloud-based solution offers the advantage of real-time synchronization with all connected clients, which makes it highly scalable and reliable. In addition, it can maintain responsiveness even if the app goes offline, which is especially important for mobile users. The CRUD of the database can be managed entirely from the admin console of the application or directly from the Firebase console.

Regarding the data structure itself the architecture is a JSON file divided in six main data objects: ***filters, migrations, museumObjects, museums, quiz*** and ***quizPlayers***.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 7: Six main objects of the database

* 1. Filters

The ***filters*** object contains all the filters that are used in the application to help the user to focus his research on a specific theme, location, or type of object. The ***filters*** object gathers three attributes: ***id, typeName*** which is the label of the filterand ***options*** which regroups all the possible choices for the filter***.***

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 8: “filters” object with his “typeName” and different possible “options”

* 1. Migrations

The ***migrations*** object contains the information about a migratory flow and where it must be displayed on the map. Six attributes define the ***migrations*** object: ***id, name, description, arrival, polygons, tags*** and ***images.*** Those are the attributes that are displayed on the information pop-up that appears when clicking on a zone of the map.

Une image contenant texte, capture d’écran, Police, document

Description générée automatiquement

Figure 9: "migrations" object and his attributes

* 1. MuseumObjects

The ***museumObjects*** object gathers all the objects that are exhibited by the Valais museums. Each ***museumObject*** is linked to a museum and a List of filters. Several attributes define the object: ***id, museumId, name, description, point, filters*** and ***images.***

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Figure 10: "museumObject" and his attributes¨

* 1. Museums

In the ***museums*** object, you find all the museums in which you find the different exhibited objects. Each museum is made of those attributes: ***id, name, address*** and ***website.***

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Figure 11: "museum" and his attributes

* 1. Quiz

In the ***quiz*** object, all the questions that are part of the quiz are stored. Each question has its own description, list of possible answers and the correct answer to the question. The attributes of the objects are therefore the following: ***id, questionText, answers*** and ***correctAnswer.***

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Figure 12: "quiz" and his attributes

* 1. QuizPlayers

Finally, the quizPlayers object contains all the necessary information about the quiz that have been filled by the users of the application. The time at which the quiz has been passed, the final score of the quiz, and the email to which the score has been sent are then stocked into the database. The attributes of the quizPlayers object are: ***id, dateTime, score*** and ***userEmail.***

Une image contenant texte, Police, capture d’écran, ligne

Description générée automatiquement

Figure 13: "quizPlayers" object and his attributes

1. Frontend
   1. Development tools

To carry out the project, several development tools were used:

* IDE: Andoid Studio Giraffe 2022.3.1 Patch 1
* Database hosting: Firebase RealTime Database
* DevOps platform: GitLab
* Development support: ChatGPT-3.5
* Deployment: Google Play
* Project Management: Microsoft Teams
  1. Architecture

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Figure 14: Global architecture of the project

The ***.dart\_tool*** folder contains files used by pub and various Dart tools. It must not be committed.

The ***android*** folder contains the files and folders required to run the application on an android system. These ones are autogenerated when creating the project. It is recommended to let them as they be. In our case, we’ve made a minor modification in le ***local.properties*** file changing the ***flutter.minSdkVersion*** to the version 33.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 15: Modification in "local.properties" file

The ***lib*** folder contains the code of the application:

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 16: "lib" and his sub-folders

* admin\_forms: contains all the forms used for the administration of the application. It is itself divided in sub-folders that group the components according to their usages. Each of these sub-folders gathers all the necessary classes to perform the CRUD of the components.

Une image contenant texte, Police, capture d’écran, conception

Description générée automatiquement

Figure 17: "admin\_forms" and his sub-folders

* dataModels: contains all the data models of the application.
* firebase: contains all the interactions between the application and the database like the authentication, the CRUD, the options and the settings.
* map: contains all the classes to display the map and all the other components as filters, pointers, areas that are available on the map page.
* quiz: contains the classes of the different pages of the quiz.

We also have some documents in the root which are:

* .flutter-plugins: contains all the flutter plugins used in the project.
* .flutter-plugins-dependencies: contains the dependencies.
* .gitignore: what shouldn’t be pushed in an online repository.
* .metadata: tracks the properties of the Flutter project. It is used to assess capabilities and perform upgrades.
* Aaalysis\_options.yaml: configures the analyser that analyses Dart code to check for errors, warnings and lints.
* flutter\_exponomade.iml: stores information about development module and saves the paths, dependencies and other settings.
* pubspec.loc: snapshot of all dependencies, their versions and their transitive dependencies at a specific time.
* pubspec.yaml: defines the dependencies of the project with their versions.
* README.md: explains how to correctly set up and run the project.
  1. Libraries

Below, a list of all the libraries and frameworks used to develop this application (source: pubspec.yaml file):

* ***Flutter (3.13.2):*** Core package
* ***flutter\_map\_tappable\_polyline (^5.0.0***): a flutter\_map plugin that adds Polyline class with onTap event.
* ***latlong2 (^0.9.0):*** library for common latitude and longitude calculation.
* ***flutter\_map (^5.0.0):*** mapping package for flutter
* ***firebase\_core (^2.15.1):*** plugin for Firebase that enable to connect to multiple Firebase apps.
* ***cloud\_firestore (^4.9.0):*** plugin for Cloud Firestore noSQL database.
* ***firebase\_auth (^4.8.0):*** plugin enabling Android and iOS authentication.
* ***firebase\_database (^10.2.5):*** plugin for Firebase Database, a cloud-hosted NoSQL database with realtime data syncing across Android and iOS clients, and offline access.
* ***Intl (^0.18.1):*** provides internationalization and localization facilities.
* ***firebase\_storage (^11.2.6):*** plugin for Firebase Cloud Storage, a powerful, simple, and cost-effective object storage service for Android and iOS.
* ***google\_fonts (^5.1.0)****:* package to use fonts from fonts.google.com.
* ***carousel\_slider (^4.0.0):*** a carousel slider widget supporting scroll and custom child widget.
* ***image\_picker (^1.0.4):*** plugin for selecting images from the image library.
* ***cupertino\_icons (^1.0.2):*** asset repo containing a set of icons.
* ***flutter\_lints (^2.0.0):*** package containing a set of lints for Flutter apps, packages, and plugins to encourage good coding practices.
  1. Virtual device configuration

The system image of the virtual device used in this project is the following:

* Release name: Tiramisu
* API level 33
* Android: 13.0
* System image: x86\_64

1. Configuration

To run your project properly, you have some installations to perform:

1. Install Flutter on your laptop.
2. Install Android Studio.
3. Install Flutter and Dart plugins on your Android Studio.
4. Clone the project from Git.
5. Create an Android emulator on Android Studio.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

1. Launch the emulator.
2. Launch the project from the terminal with the command ***flutter run***
3. Deployment

The application is deployed on Google Play. This is done through the Google Play Console, by providing an App bundle in order to create a release.

Lauch the command “flutter build appbundle” to generate the App bundle of the release.

To sign your release App bundle, you need to create a local file name “key.properties” located in the “/android” directory. The content of this file should look like this:

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

1. Admin credentials

To access to the admin console, you need to authenticate as an admin user. Here are admin credentials:

Email: [admin@exponomade.com](mailto:admin@exponomade.com)

Password: ExpoNomade2023

1. List of URLs

Application:

* Deployed application: <https://play.google.com/store/apps/details?id=ch.hevs.expo_nomade&hl=fr&pli=1>

Repository and database:

* Git : <https://gitlab.com/elias.borrajo/flutter_exponomade>
* Firebase Realtime Database: <https://exponomade-6452-default-rtdb.europe-west1.firebasedatabase.app/>

Libraries:

* Flutter 3.13.2: <https://flutter.dev/>
* Package repository for Dart and Flutter apps: <https://pub.dev/>

1. Appendix
   1. Mockups

Une image contenant texte, capture d’écran, multimédia, affichage

Description générée automatiquement

Une image contenant texte, carte, capture d’écran, multimédia

Description générée automatiquement

Une image contenant texte, carte

Description générée automatiquement

Une image contenant texte, capture d’écran, carte

Description générée automatiquement

Une image contenant texte, capture d’écran, carte

Description générée automatiquement

Une image contenant texte, capture d’écran, multimédia, ordinateur

Description générée automatiquement

Une image contenant texte, capture d’écran, multimédia, logiciel

Description générée automatiquement

Une image contenant texte, capture d’écran, multimédia, gadget

Description générée automatiquement

Une image contenant texte, capture d’écran, multimédia, affichage

Description générée automatiquement

Une image contenant capture d’écran, Rectangle, texte, ligne

Description générée automatiquement

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement

Une image contenant texte, capture d’écran, Rectangle, Parallèle

Description générée automatiquement

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement

Une image contenant texte, capture d’écran, nombre, ligne

Description générée automatiquement

Une image contenant texte, capture d’écran, affichage, nombre

Description générée automatiquement

Une image contenant texte, capture d’écran, affichage, nombre

Description générée automatiquement

Une image contenant texte, capture d’écran, diagramme, Rectangle

Description générée automatiquement