University of Namur

INFOB318: personnal project

Ticked-it programmer guide

Project workers :

Gonzague Yernaux
Adrien Suys
Stéphane leblanc

Teacher:
Vincent Englebert

Assistant:
Maouaheb Belarbi

Contents

1 Introduction

This document is the programmer guide of the Ticked-it application. It's an Android application so we assume you have the basic of Android development. When i started to work on Ticked-it project, the app was already launched and pretty large. Then, I'm not going to describe all the content because i'm not able to and i didn't work on.

2 What is Ticked-it

Ticked-it is an Android application made for events managers. With Ticked-it, any manager is able to manage the entrance for his events. After a login activity, the user can access to all his events and see statistics about, sell some tickets for the event, scan the barcode of any ticket to manage the entrance and look at the audience of the event.

3 Developpers

Adrien Suys and Stephane Leblanc worked on the development of Ticked-it.

4 Development environment

This application has been developed on Android Studio with the Java language. Android Studio is the standard to develop Android applications and can be downloaded here : Android Studio

5 How to compile

To compile the project on Android Studio, you just have to click on the green arrow like in image below. Make sure you have an android smartphone connected to your computer or set up a virtual device.

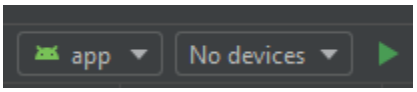

Figure 1: How to compile

6 How to generate APK

To make an executable version of your app, you have to generate an APK. To do that, go to "Build" and click on "generate signed APK"

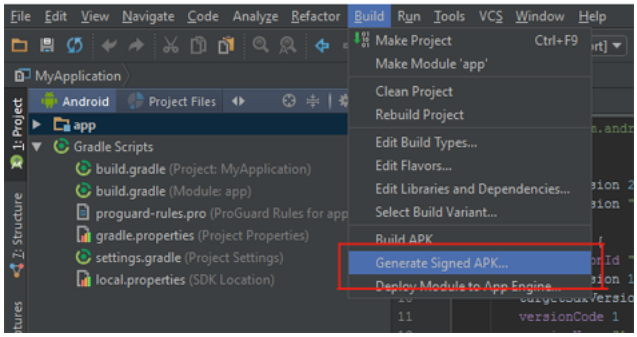

Figure 2: Generate APK

Then choose the APK option and click Next. If you don't have set up a key, click on "Create new" and fill in the different forms. Choose then this key to generate apk, click Next, choose release build and V2 (full APK signature) and click on finish. You executable version is now build in AndroidStudioProjects/MyApp/app/release.

7 MVP architecture

This project is based on the MVP architecture that is a derivative from the MVC. This architecture allows to cut the code into 3 parts : the model, the view and the presenter.

- Model: the data layer. Contains data represented by classes with the logic to manipule it. (Example in the project : Event class). Communicate with the database.
- View: represents the UI part of the app. Displays data and notify the presenter about user actions.
- Presenter: makes the link between Model and View. Gets data from Model to display in the View.

Basically, when the View has to display data, it asks to the Presenter. Presenter asks in turn to the Model, gets data and gives back to the View.
One of the main goals of MVP is to facilitate automated unit testing. That's why each Presenter and View are represented by an interface that forms a contract between the Presenter and the View. It helps to decouple the code and facilitate tests.

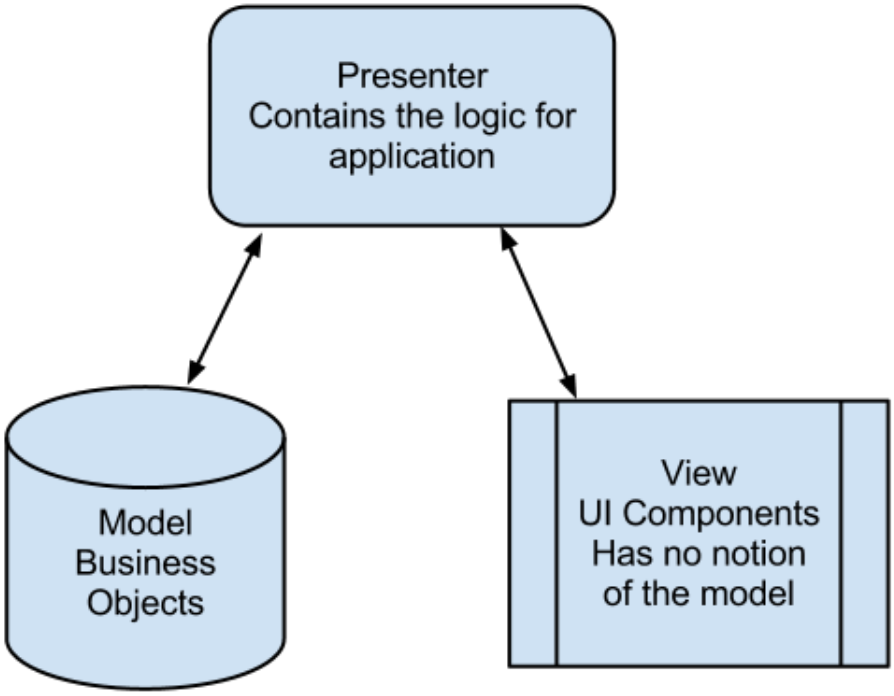

Figure 3: MVP Wikipedia

8 Architecture of packages and classes

8.1 Java part

In the Java folder, you can find the activities (the Views), apirest, fragments, model, presenter, security, utils and viewinterfaces packages.

8.1.1 Activities package

This package contains the different activities of the app. Each activity extends the Activity class that gathers the different methods common to each activity. Look at the figure 6 to see the class diagram of the Activity package.

8.1.2 Fragments package

This package contains the different fragments of the app. Except for the PreferenceFragment, all other fragments are a portion of the user interface of OneEventActivity. Check this figure below to see the relations between fragments and activities. To have more informations about fragments: Fragments
Note: activities like BarcodeManualActivity and CreateTicketActivity are reachable from fragments in the OneEventActivity.

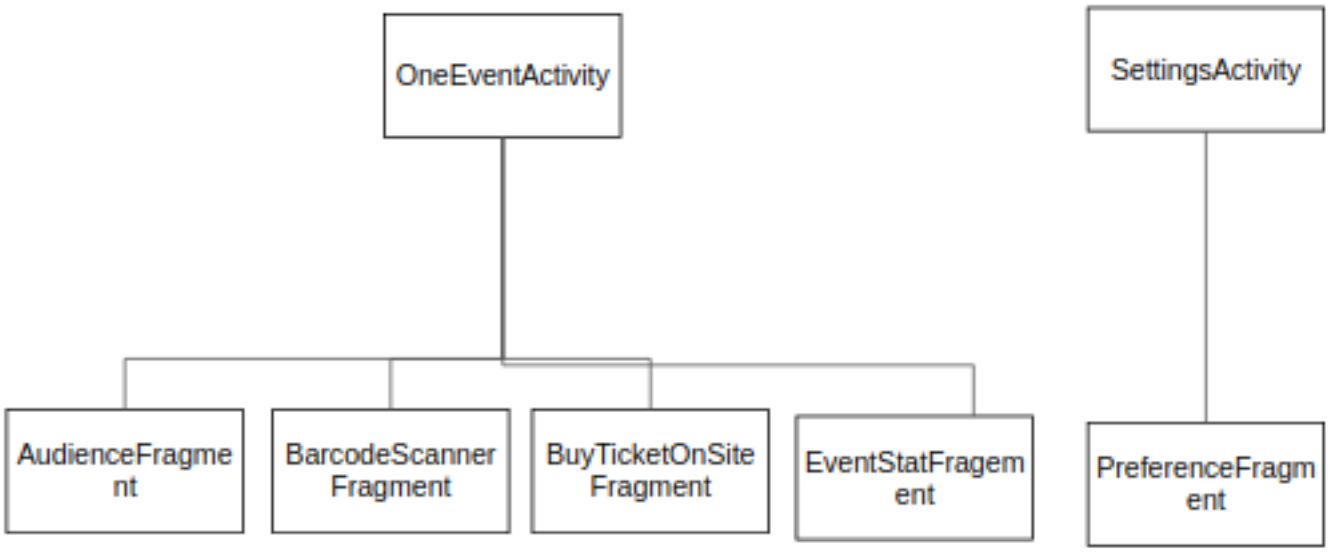

Figure 4: Relations between activities and fragments.

8.1.3 Models package

This package contains the different models of the app. Each model represents a type of data for the app with methods to manipulate it.

- The counterpart model is an object to represent the different types of tickets for a event. It is build with an id, a name and a price as attributes.
- The event model is an object to represent an event created by a user. It is build with an id, a name, the total number of tickets that can be sold for the event, the number of tickets that are already scanned, the number of tickets sold in pre-sale, the number of tickets sold on site at the gates and the date of the event.
- The ticket model is an object to represent a ticket created for an event. It is build the barcode value written in the ticket, the name of the buyer, the type of the ticket, the seat number (nothing if stand up) and an error message is the ticket is unvalid.
- The User model is an object to represent a user of the app. He must be in the database and organize an event to access to the app. It is build with an id, a username, a crypted password and an error message if the user has no access to the app.

8.1.4 Presenters package

This package contains the different presenters to make the link between Model and View. Then, it contains all the methods needed to retrieve data from Model and give it to the View. Each presenter extends the Presenter class, look at the figure 5 to see the class diagram.

8.1.5 Utils package

The LocalHelper class is a class to manage the settings of the app's locale. It's then used by the PreferenceFragment to change the language of the app.

8.2 XML part

This part that you cand find in the res package, contains all elements used to build the layouts of the app.

- Drawable package : contains all icons and pictures used in the app.
- Layout package : contains all xml files that build the layout of an activity or a fragment.
- Layout-land package : contains the xml file that build the login layout when the device is horizontal landed
- Menu package : menu.xml represents the options displayed when the user clicks on the three dots on the top right of the app. navigation_bottom_menu represents the different items displayed in the bottom bar of the OneEventActivity to load a new fragment.
- Mipmap packages : they each contain launcher icon of the app but with a different density to provide a better display. The device will automatically detect the density of the screen and display the icon in the package corresponding to the density detected.
- Value package : arrays.xml list the value proposed by the listPreference in the SettingsActivity to change the language. attrs.xml defines attributes used to apply the correct color to elements depending of the current theme. colors.xml lists different colors that can be used. dimens.xml fixes dimension to some elements in the app. strings.xml contains all the strings used in the app. Centralise it here allow to change the current language of the app. styles.xml defines some styles used in the app. For example, the light and dark themes or the theme of the dialog that pops up when user opens the app with no internet.

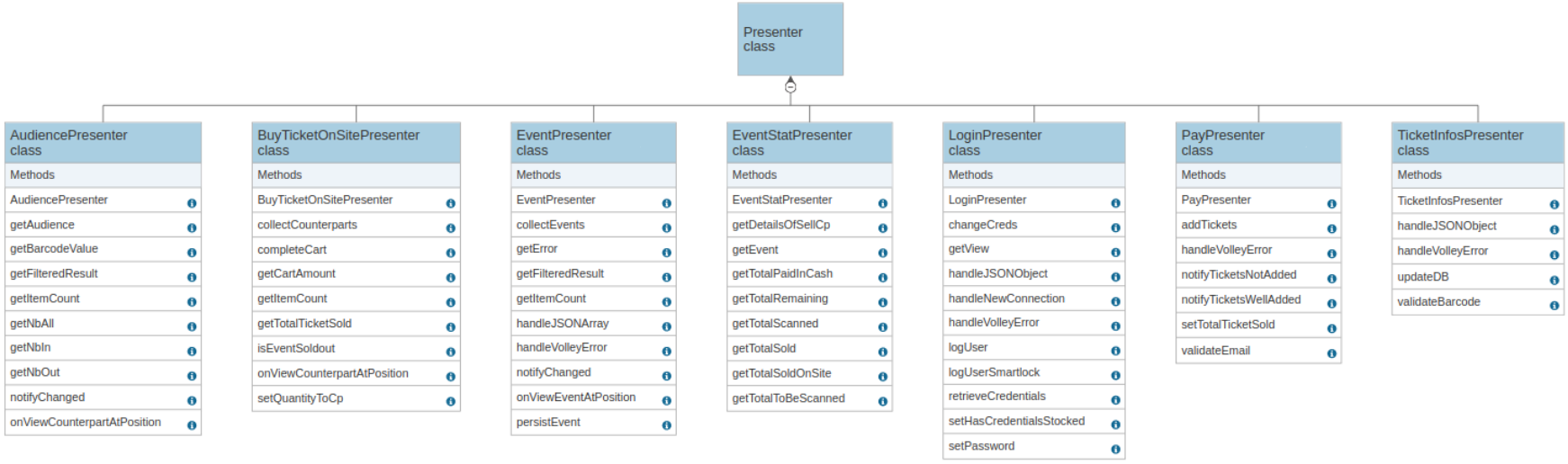

Figure 5: Presenters class diagram

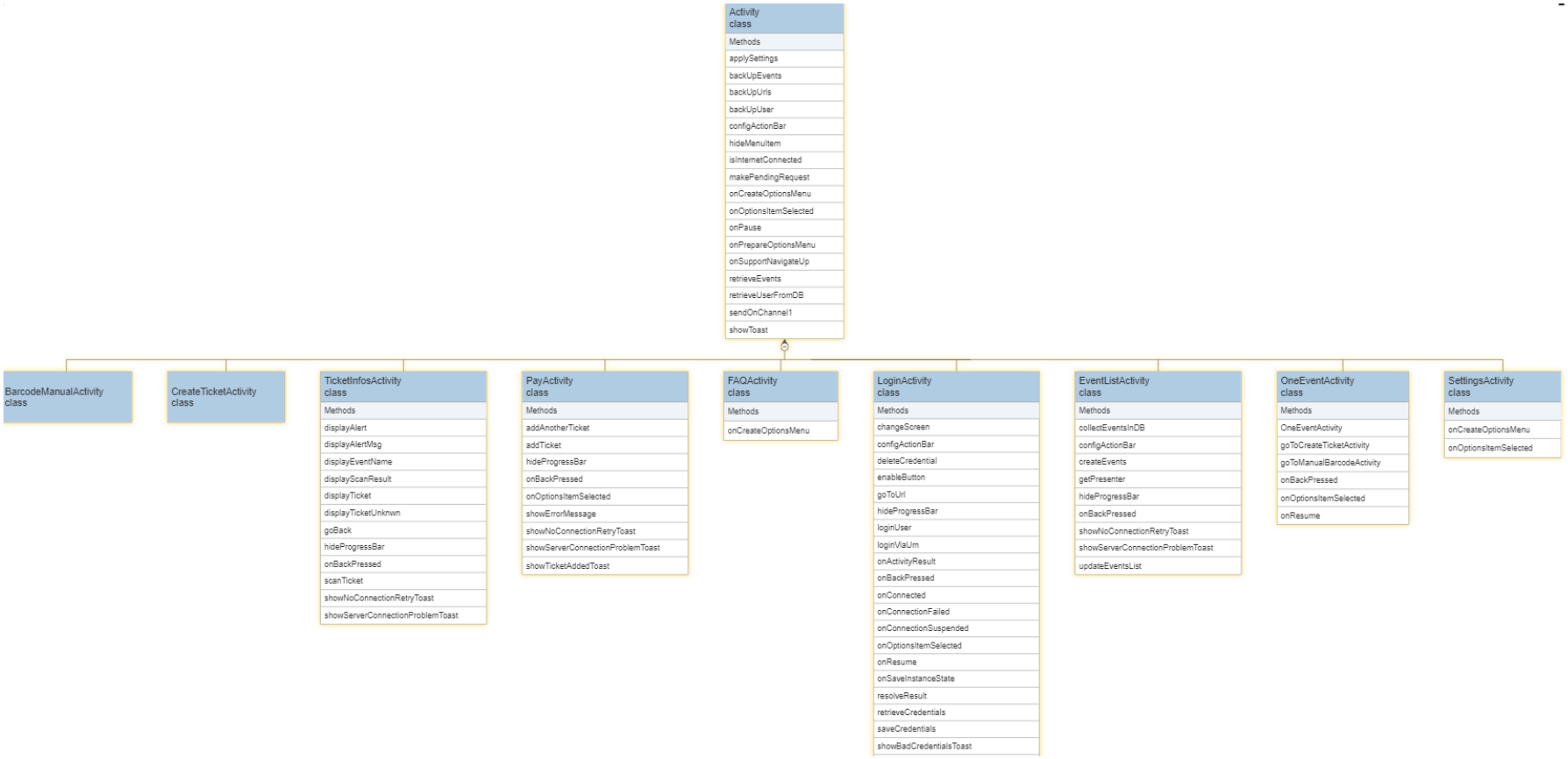

Figure 6: Activities class diagram