Capstone Stage 1 - Maplenou

Komi Wolanyo KOUDO

March 11, 2019

Contents

GitHub Username: wolasoft

Maplenou

1 Description

In my mother tongue, **maplenou** means: **I want to buy something**.

The aim of my application is to allow people to offer items for sale. It's like french app leboncoin but targets the Togolese market.

With this application, people can consult the articles offered for sale, or can propose some.

2 Intended User

My application is intended for individuals and companies.

3 Features

My application will offer bellow functionalities:

- Display of the list of announcements
- Possibility to search announcements by defined criteria
- Display of the detail of an announcement
- Possibility to contact the marketer by Call, SMS or E-mail
- Possibility to offer an item for sale
- Possibility to mark an item as favorite
- Display of the list of announcements marked as favorite
- Possibility to register
- Possibility to login and logout
- When registered, possibility to display account details and related announcements
- Advertising displays

4 User interface mocks

4.1 Splash screen

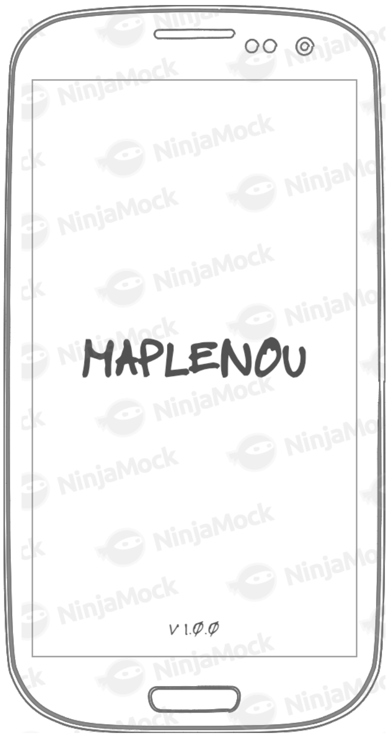


Figure 1: Splash screen

4.2 App default screen

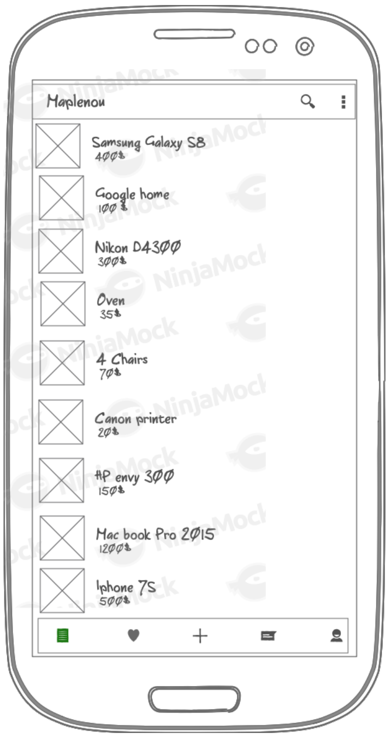


Figure 2: Announcement list - Phone screen

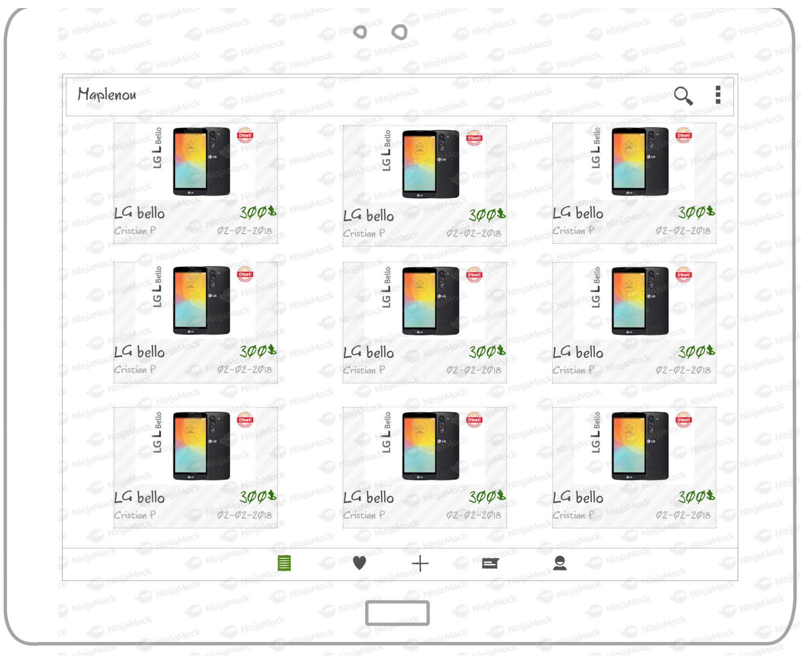
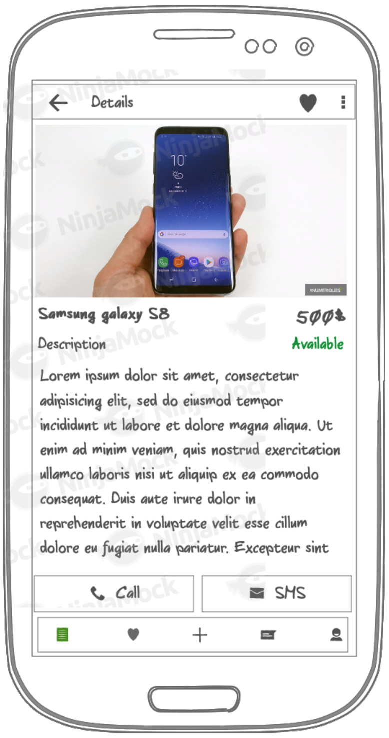


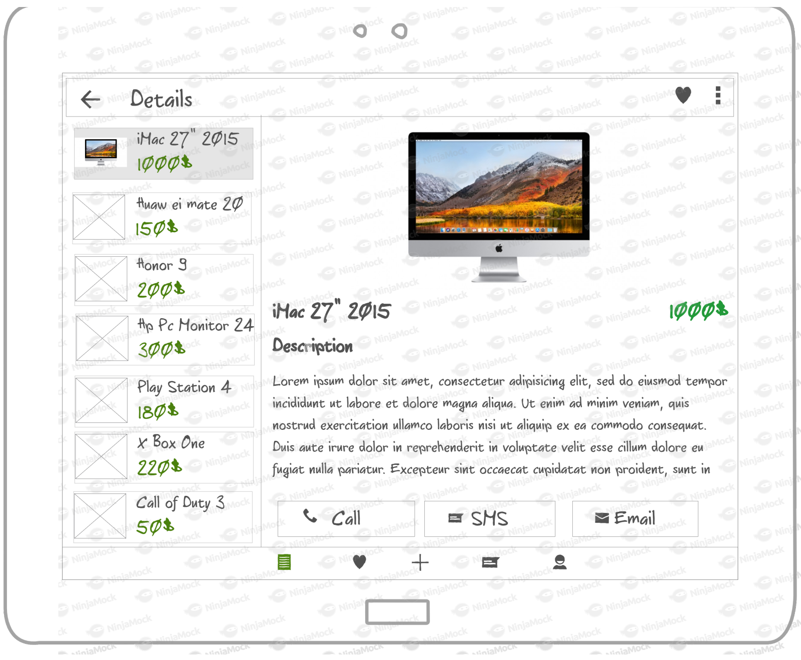
Figure 3: Announcement list - Tablet screen

4.3 Announcement details screen



Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint

Figure 4: Announcement details - Phone screen

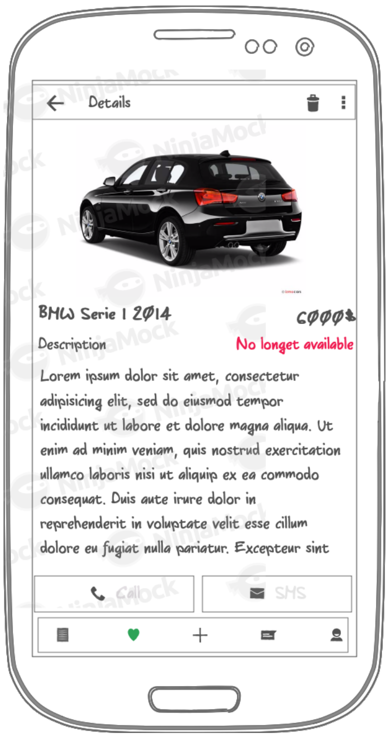


Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in

Figure 5: Announcement details - Tablet screen

4.4 Favorite Announcement details

Here announcement is deleted by marketer, so it is marked as **no longer available**



Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint

Figure 6: Favorite Announcement details - Phone screen

4.5 Registration screen

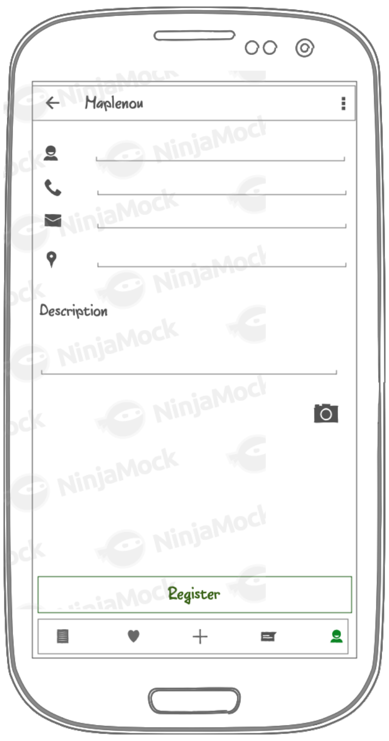


Figure 7: Registration - Phone screen

4.6 Login screen

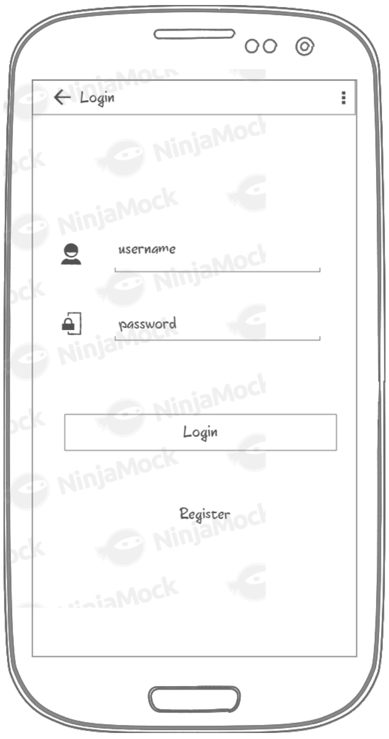


Figure 8: Login - Phone screen

4.7 Publish an anouncement

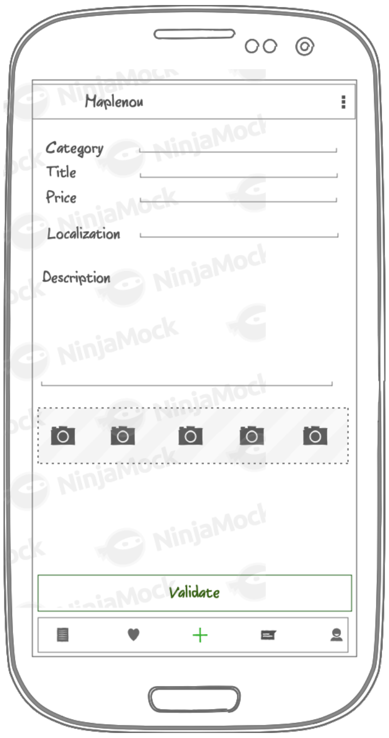
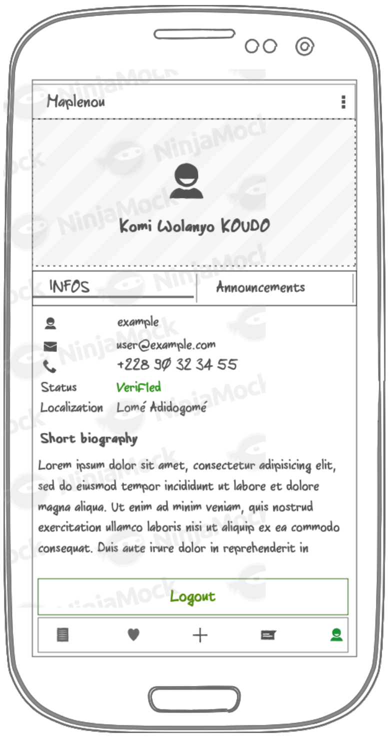


Figure 9: Publish an anouncement - Phone screen

4.8 User information screen



example
user@example.com
+228 9∅ 32 34 55

Status	Verified
Localization	Lomé Adidogomé

Short biography

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in

Figure 10: User information - Phone screen

4.9 User published announcement screen

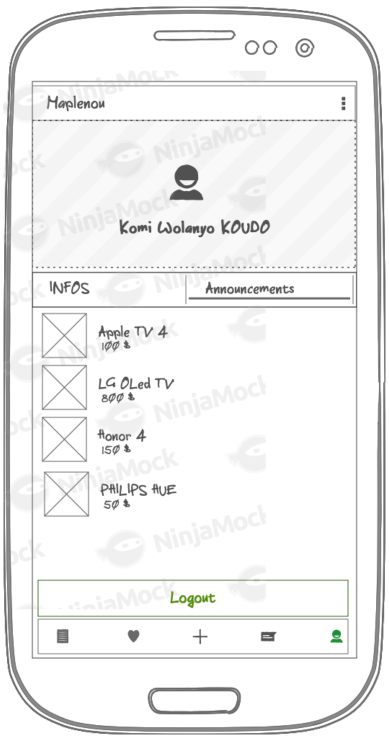


Figure 11: Announcement published by a user - Phone screen

4.10 *Bonus*: Message screen

I'm planning to add in app messaging feature. But it will depend on my backend architecture. If my backend is ready this app section will be visible, it will hidden. reason I marked it as bonus

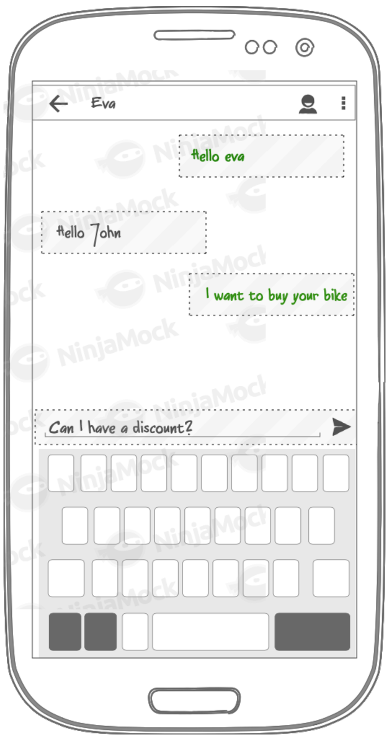


Figure 12: In app messaging - Phone screen

5 Key Considerations

5.1 Data persistence handling

The application will essentially communicate with an online custom restful API written in python flask. The available announcements will be fetched from this API. Announcements marked as favorite will be online linked to user account (if user is registered) and locally saved in a room database no matter if the user is registered or not.

5.2 Describe any corner cases in the UX

5.2.1 Network issue

If a network issue occurs during API call app will display a a screen to notify that to the user and invite him/her to touch screen to retry recontacting online API. Bellow is the screen i'm planning to display

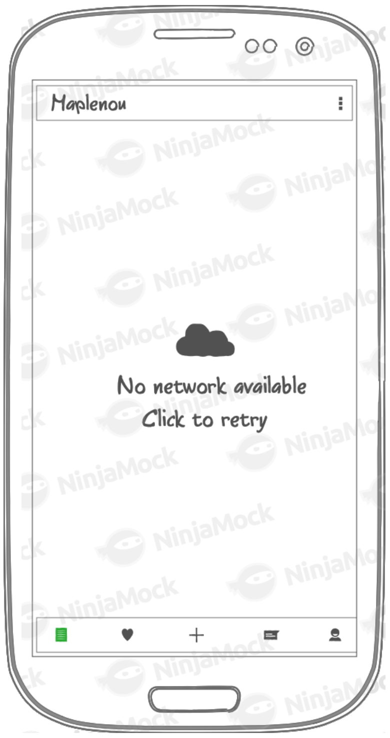


Figure 13: Network issue handling screen screen - Phone screen

5.2.2 Empty data issue

When no announcement is available, the application display bellow screen.

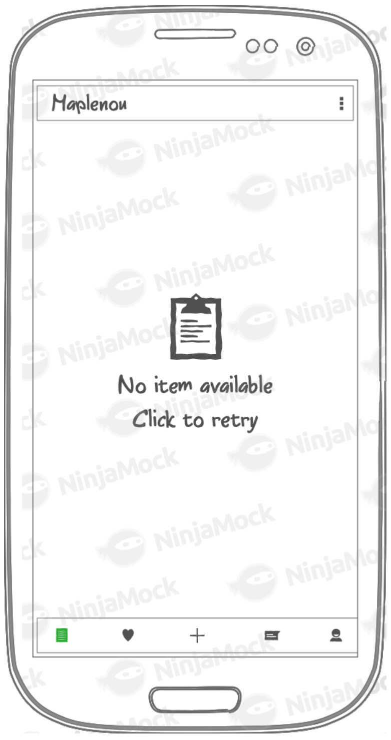


Figure 14: Empty Announcement list - Phone screen

5.2.3 Form validation

Data exchanged between the application and API will be validated by API and by application. In The Backend API, received data will be controlled before processed. In the mobile application, any form will be also validated before transferred via **http** to backend API. I do these validated to avoid user to send non conform data and to make the application work as it supposed to.

5.3 Libraries description

5.3.1 Retrofit

I will handle API http request with **Retrofit**.

I choose this library because it is an a type-safe http client for Android and Java. It is one of the most popular libraries in the Android world.

5.3.2 Gson

Gson is a Java library that can be used to convert Java Objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object.

I will use this library to parse object manipulated by my application.

It will be coupled with retrofit to parse data during API calls.

5.3.3 Room Persistence Library

The Room persistence library provides an abstraction layer over SQLite to allow for more robust database access while harnessing the full power of SQLite.

I will use this library to persist locally announcements marked as favorite by the user.

5.3.4 Dagger

Dagger is a replacement for these Factory classes. It allows you to focus on the interesting classes. Declare dependencies, specify how to satisfy them, and ship your app.

I will use this library to inject dependency in my application and make my app easy to test.

5.3.5 Junit

JUnit is a test framework which uses annotations to identify methods that specify a test. JUnit is an open source project hosted at Github.

I will use this library to unit test my application.

5.3.6 Espresso

The Espresso testing framework, provided by AndroidX Test, provides APIs for writing UI tests to simulate user interactions within a single target app.

Espresso tests can run on devices running Android 2.3.3 (API level 10) and higher.

I will use this library to test my application UIs.

5.3.7 Support libraries

> When developing apps that support multiple API versions, you may want a standard way to provide newer features on earlier versions of Android or gracefully fall back to equivalent functionality. Rather than building code to handle earlier versions of the platform, you can leverage these libraries to provide that compatibility layer. In addition, the Support Libraries provide additional convenience classes and features not available in the standard Framework API for easier development and support across more devices.

Because I want to provide same app behavior across a lot of android version, I will use this library to support my application features retro-compatibility.

5.3.8 Google AdSense library

Because I want to monetize my application, I will use this library.

Google AdSense is a free, simple way for application publishers of all sizes to earn money by displaying targeted Google ads on their application.

5.4 Google play service implementation

The only play service i will implement in my app is google AdSense. I'm planning to monetize my application with it. Everytime user start the application, an ad full-screen view will be display after the splash screen for a few second before launching application main activity.

6 Key steps / Required task

6.1 Task 1: Project setup

- Create online git repository
- Generate application basic skeleton with minimal gradle configuration
- Add application flavors
- Add application build types
- Add application base colors configuration
- Push project skeleton to remote git repository.

6.2 Task 2: Configure dagger

- Add dagger dependencies
- Add minimal dagger configuration

6.3 Task 3: Configure app data layout

- Create entities/models
- Add API base URL to module's gradle config
- Add Api connector
- Add retrofit and gson dependencies
- Configure retrofit services
- Add API repositories
- Add API dagger configuration
- Configure android room library
- Add room repository
- Add local data persistence dagger configuration
- Add application preference classe
- Add application preference dagger configuration

6.4 Task 4: Add main activity

- Add bottom navigation bar

6.5 Task 5: Add item list view

- Create item list fragment and logics
- Add Phone and tablet resources
- Configure list view
- Add API call
- Make view material

6.6 Task 6: Add item detail view

- Create item details fragment and logics
- Display data
- Add possibility to contact marketer by SMS/Email/Call
- Make view material

6.7 Task 7: Add search view

- Add item search fragment and logics
- Add Phone and tablet resources
- Display data
- Make view material

6.8 Task 8: Add favorite features

- Add possibility to mark announcement as favorite
- Add favorite list view fragment
- Add possibility to show favorite announcement details
- Make views material

6.9 Task 9: Add registering feature

- Add registering view and logics
- Make view material

6.10 Task 10: Add login features

- Add login view and logics
- Make view material

6.11 Task 11: Add logout features

- Add logout view and logics
- Make view material

6.12 Task 12: Add possibility to publish an announcement

- Add announcement publication view and logics
- Make view material

6.13 Task 13: Add user detail features

- Add user detail view and logics
- Show user information
- Show user published announcements
- Make view material

6.14 Task 14: Add Google AdSense

- Add AdSense dependencies
- Add Ad on application startup