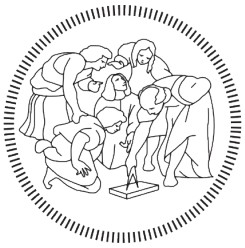
DESIGN AND IMPLEMENTATION OF MOBILE APPLICATIONS



POLITECNICO
MILANO 1863

NonSoloLibri

MATTEO COLOMBO - ANDREA TROIANIELLO

September 5, 2019

Abstract

Never loose a book again.
Organize your book collection so that you always know where they are.

Contents

List of Figures

Chapter 1

Introduction

1.1 Purpose

The purpose of this document is to describe all the aspects of the design and implementation of NonSoloLibri, a cross platform application whose aim is to help users to manage their book collections.

1.2 Scope

NonSoloLibri is a small social network application designed for book enthusiast that allows them to organize and review their collection.
Users will be able to create libraries and to organize books in them.

Beside the storing functionality, NonSoloLibri has a social part that allows them to connect with other people with a friendship system. Friends will be able to see the each other collections and wish-lists, so that they can surprise each other with presents.

A market place is available, where every verified user will be able to sell or buy books.

1.2.1 Goals

1. Allow the users to organize books in libraries.
2. Allow the users to manage libraries and books.
3. Allow the users to view their friends wishlist.
4. Allow the users to sell and buy books in a market place.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

Cross-platform: An application that runs on multiple operative systems, for example on both Android and iOS.

Document Oriented: Non relational database that uses JSON like documents to save data.

Flutter: A cross-platform development SDK written in Dart, to develop for iOS, Android and Web.

Library: A shelf characterized by a name, a photo and a list of books.

Market Place: A section of the application where users can sell and buy books.

Mockito: Testing tool used to mock classes in Flutter.

Verified user: A user whose identity has been verified by the system and is allowed to sell books in the market place.

Widget: A Flutter component that can represent anything, from a single layout item, to a whole page.

Wish-list: A list of books that a user would like to acquire in the future.

1.3.2 Acronyms

API: Application Programming Interface.

IEEE: Institute of Electrical and Electronic Engineers.

ISBN: International Standard Book Number.

JSON: JavaScript Object Notation.

SDK: Software Development Kit.

SQL: Structured Query Language.

UML: Unified Modeling Language.

1.3.3 Abbreviations

G.x: The design goal number X.

R.y: The design requirement number Y.

D.z: The domain assumption number Z.

1.4 Reference Documents

- Android Developers website.
- Course slides.
- Dart Language documentation.
- Firebase documentation.
- Flutter documentation.
- Material Design website.

1.5 Document Structure

This document is divided in multiple chapters:

Chapter 1: In this chapter it is explained the aim of this document and the project in general.

Chapter 2: This chapter cointains the description of the application and the list of its requirements, goals and the assumptions that were made during the development.

Chapter 3: This chapter contains the in-details description of the application architecture; all the components and their integration is explained.

Chapter 4: In this chapter all the interfaces used in the application are explained. This includes those related to the final users, but also those used during the implementation.

Chapter 5: In this chapter it is explained how the application was implemented and tested.

Chapter 2

Overall Description

2.1 Product Perspective

NonSoloLibri will be developed from scratch and will be a mobile application that runs on both iOS and Android.

Data will be stored online so that they can be retrieved from every device and to provide a sort of back-up option. Thus, users will be required to have a smartphone or tablet and a Google account to sign in.

In the application the users will be able to make use of their device hardware to capture pictures of books and libraries or to scan the book barcodes.

Verifying users will be mandatory in order to let them access to the market place, to add a security layer and to discourage bad behaviours.

2.2 Actors

We identified three main actors for the application:

Visitor A person who has yet to login into the application and can only see the login page.

User A visitor who successfully logged in into the application and can access to all the main feature of the aplication.

Verified User An user who successfully verified himself and can access to the market place.

2.3 Use Cases Diagram

The main actors have different functionalities, according to the previous definitions.

The visitior can only perform the login, the user can also make other things, like to manage the library or review a book. Verified users can interact with the market functions.

All the functions are shown using the following use case diagram:

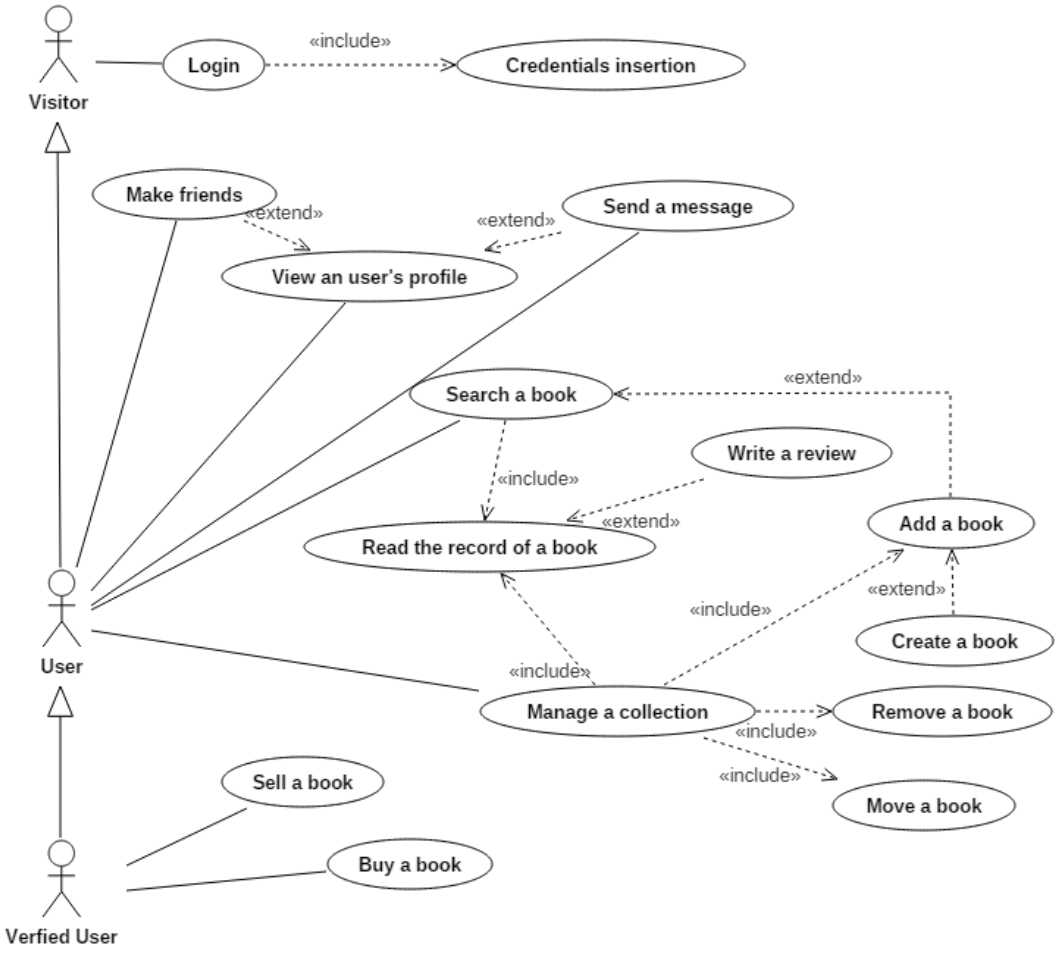


Figure 2.1: Use case diagram

2.4 Product Requirements

1. Visitors should be able to login with their Google account.
2. Users should be able to create libraries.
3. Users should be able to edit libraries.
4. Users should be able to customize libraries with photos or images.
5. Users should be able to delete libraries.
6. Users should be able to view libraries.
7. Users should be able to add books to libraries.
8. Users should be able to delete books from libraries.
9. Users should be able to move books from a library to another.
10. Users should be able to view books information.
11. Users should be able to create books if they are not present in the database.
12. Users should be able to suggest changes to books information.
13. Users should be able to search for books in the database.
14. Users should be able to send friendship requests.
15. Users should be able to manage friendships.
16. Users should be able to create a wish list.
17. Users should be able to view their friends wish lists.
18. Users should be able to verify themself.
19. Verified users should be able to access to the market place.
20. Verified users should be able to buy books in the market place.
21. Verified users should be able to sell books in the market place.

2.5 Constraints

2.5.1 Hardware and Software Constraints

1. To use the application, users are required to have a smartphone or a tablet. The device can have either Android or iOS
    - For Android the minimum supported versione is Android 5.0 Lollipop.
    - For iOS the minimium supported version is iOS 10.
2. The application requires a device configured and able to connect to the Internet.
3. The application requires a device with a photocamera.

2.5.2 Safety and Regulation Constraints

1. The system must guarantee data is protected and accessible only by its owner.
2. The application must ask for user permissions to acquire, store and elaborate images from their devices.
3. The system must complain with local laws.

2.6 Domain assumptions

1. Administration tasks such as verification of the users and data checks on books inserted by users are made through an independent web-application whose design is not part of this document.
2. Users should behave coherently and in respect of good manners.
3. Users should type in correct information when adding a new book.
4. A library is composed by: a name, a picture or an image and can be favourite or not.
5. A library can contain only one copy of a book.
6. Multiple books with the same ISBN code can be located in different libraries.

7. A book is characterized by: title, authors, publisher, release date, length, format, price and a ISBN code.

8. The application will work as an intermediary between users for book trading, but transactions and shipment will happen outside of the application context.

2.7 Software System Attributes

2.7.1 Reliability and Availability

The application should be tested so that reliability is not a problem and user do not experience any kind of error or interruption.

The application should have high availability and users should be able to rely on this application. The system should have a 99.8% of uptime.

The application should make use of a caching system so that in case of system or network failure users will still be able to access to data stored offline.

2.7.2 Security

All the communications between clients and the server will be encrypted and protected, thanks to the Firebase protocols.

All the information will be stored on the Firebase database and security will have high priority. All sensbile data will be encrypted, and as login and authentication will be performed through Google accounts, passwords and other user credentials will not be a concern.

2.7.3 Maintainability

During the design of the application, maintainability was taken in consideration and the application should have good maintainability and low code complexity.

The code should be well documented following the Dart Language documentation instructions, so that other developers can easily understand it.

The source code management should be done through a Version Control System.

2.7.4 Portability

The application should run on any Android and iOS smartphone or tablet that satisfies the software and hardware requirements.

2.7.5 Internationalization

The application should be internationalized and support multiple languages.

It should adapt to the system language or, if the language is not supported, it should use English as default.

Chapter 3

Architectural Design

3.1 Overview

This chapter focuses on the architectural structure, components will be described explaining how they interact with each other. The system will be illustrated both physically and logically.

The main high level components of the system are the following:

Application devices: User devices where the application is installed. This application implements most of the logic of the system.

Cloud server: The server side of the system is responsible for the data storage and sync.

This layer uses a NoSQL database, which stores data in flexible, JSON-like, documents. The main keys are flexibility, expressive querying, realtime updates, offline support and scalability.

Web Application: A pre-existing application used by administrators that allows to check and edit the stored data.

3.2 Component View

3.2.1 High-Level Component View

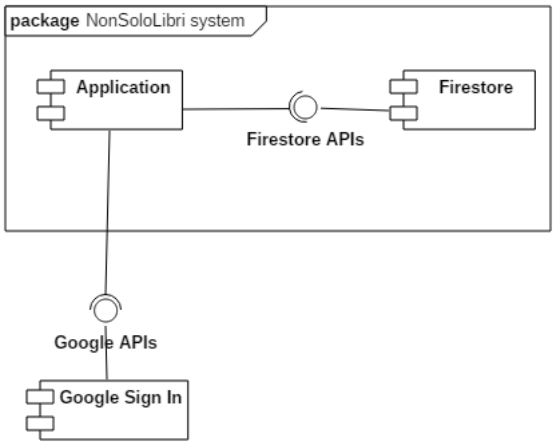


Figure 3.1: High level component diagram

This component diagram displays the high-level views of the system, focusing on application devices.

The components developed are:

Application: It is the core of the system, it manages all the information provided by the other services and performs the majority of the functions. It provides the client access to the entire system.

Firestore: This component has account management and backup roles. It receives the data from the application and provides them when necessary.

Moreover, the application is integrated with **Google Sign In**, which provides the authentication functions of the system, using directly the credential of Google.

3.2.2 Application

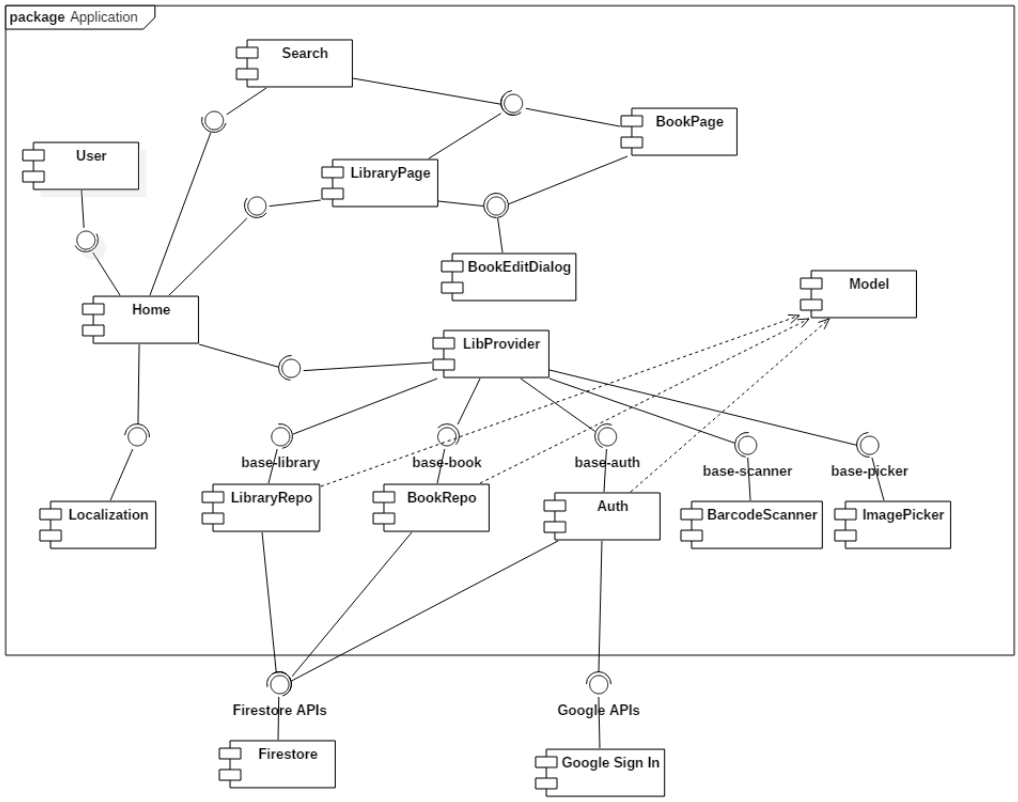


Figure 3.2: Application component diagram

Model: It represents how data are structured in the application and ready to be stored by Firestore.

Home: Is is the entry point of the application. Creates the Localization and LibProvider components. Handles login parts and the choice of the library.

Localization: It sets up the language of the application based on the language of the OS.

LibProvider: Provides LibraryRepo, BookRepo, Auth, BarcodeScanner and ImagePicker.

LibraryRepo: This components is used to communicate with Firebase and to manage the library collection.

BookRepo: It communicates with Firebase to save an retrieve books information.

Auth: This components is used to handle the authentication of the user.

BarcodeScanner: It uses the camera to read the ISBN from the barcode of a book.

ImagePicker: It uses the library that allows to take a photo with camera or to pick an image from the gallery of the device.

LibraryPage: It is a set of widgets that shows the books that belong to a given library.

BookPage: It handles the information of the book, its reviews and the associated insertions made by other users.

Search: Shows a list of the books, which match with a user's input.

User: It is the group of pages and widget that handles the profile management functionalities, including friendships and conversations between users.

3.3 Relationship between the model and the database

During the design of the application a database first approach was used. The database was the core during the design and the model was derived from it respecting the existing relationships between the entities.

The following is the ER diagram used to implement the database:

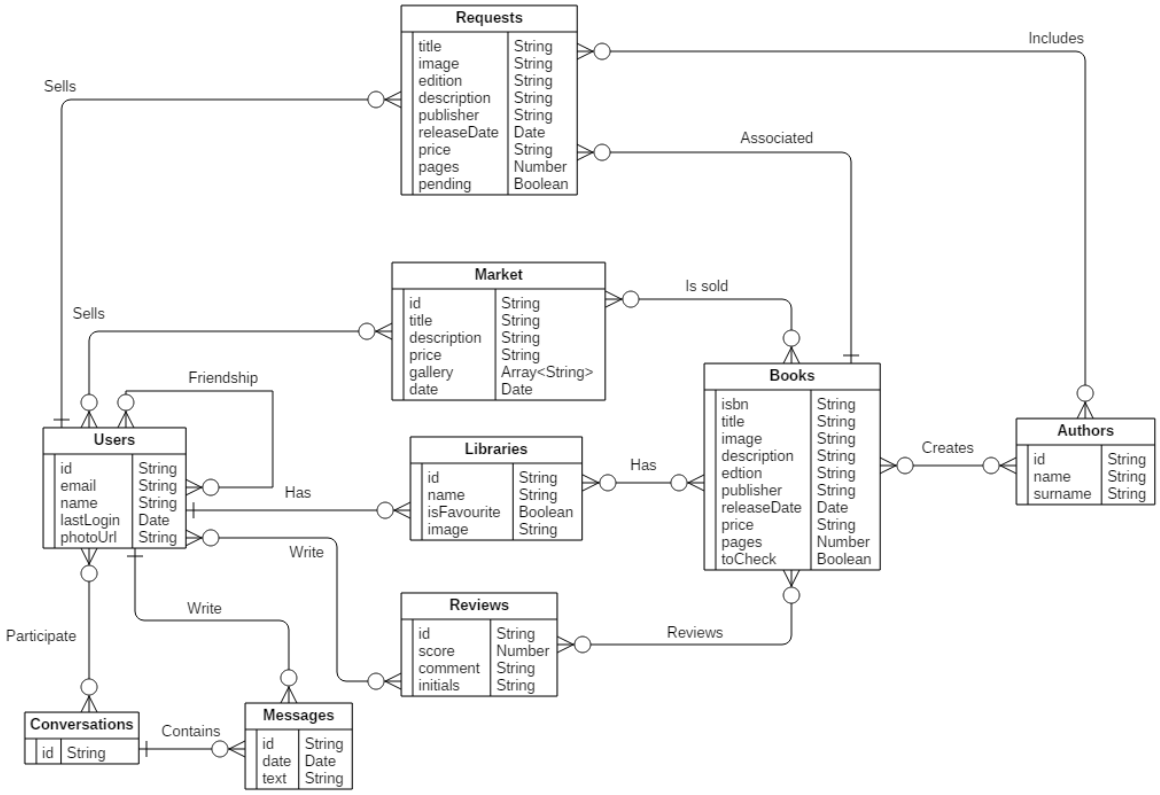


Figure 3.3: ER diagram

Books: Collects all information of the books contained in the database. Each book is characterized by a ISBN, which is its primary key, and other information including a title, a description and a image.

Users: The users use the login function of the application. The id is generated automatically, the name and email are saved, but the password no.

Libraries: The libraries created by each user.

Reviews: Includes a score and a comment written by a user associated to a book.

Authors: Contains the main information of the creator of one or more books.

Requests: Contains all the suggestions made by a user to edit the book information. The idetifier is the union of the ISBN of the book and the user's id.

Market: Collects the sales offers of books made by each user.

Conversations: The discussion between two or more users.

Messages: The texts exchanged during a conversation.

During the implementation, we opted for the introduction of data duplication. This allows to yield the main characteristics of the document-oriented database.

3.3.1 Runtime View

This section provides examples of how the main functionalities are implemented and their runtime behavior.

Login

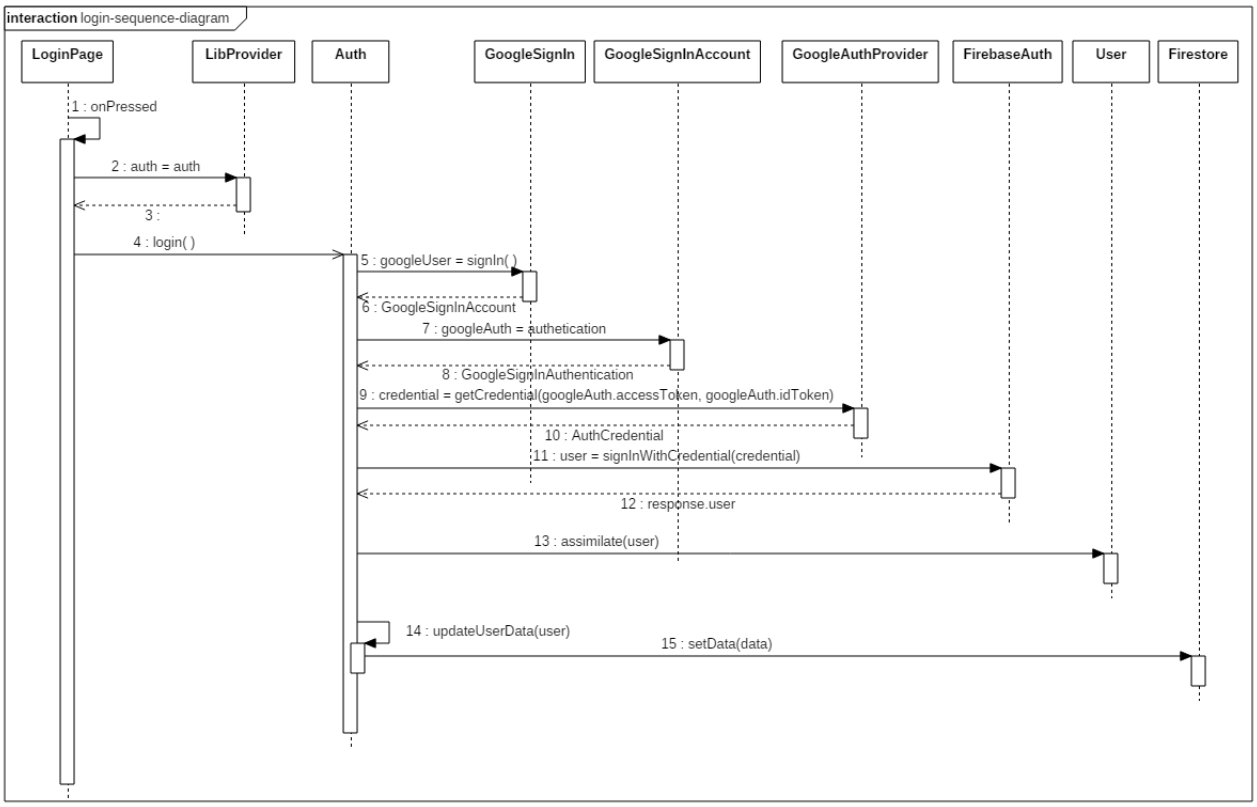


Figure 3.4: Login sequence diagram

The user taps the button on Login page, it retrieves the auth object from LibProvider. Auth handles the Google SignIn objects during the authentication phase and uses them to retrieve the user information (signInWithCredential.user).

Then, puts these information in a User object of the model using assimilate method and uses its updateUserDate method to save the user on Firestore.

Add new book

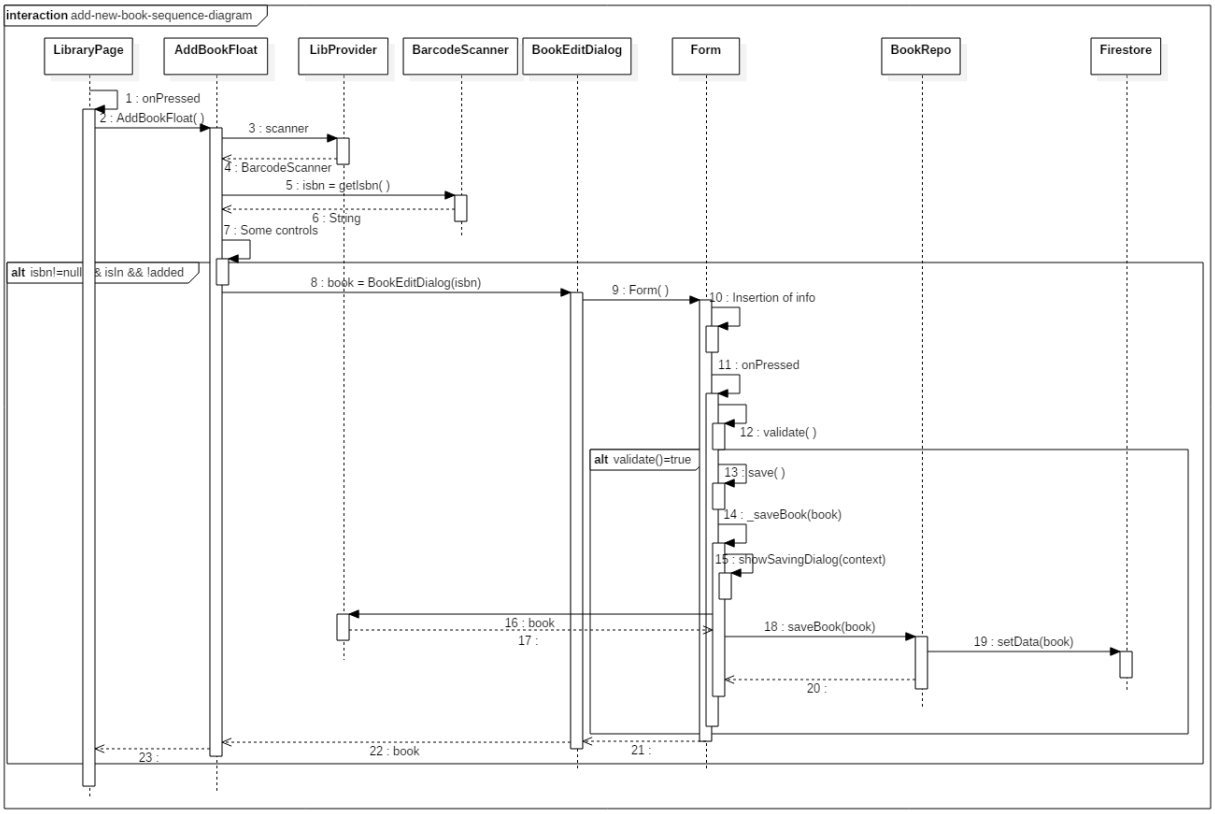


Figure 3.5: Add new book sequence diagram

After the button of Library page is pressed, AddBookFloat is created and when tapped, it opens the camera by using the BarcodeScanner. The camera reads the ISBN, written in barcode form, and returns it to AddBookFloat.

AddBookFloat controls if the string returned is a valid ISBN, by using a regular expression, and if the book already exists. If it exists and is not already present in the library, adds it. If the book does not exists, the BookEditDialog is opened. This dialog contains a form and allows to type in the information of the book. When the insertion is finished, the button "done" is tapped and the form is validated and the information are saved Firestore using BookRepo.

Chapter 4

Interface Design

4.1 UX Diagram

This diagram describes in details all the pages of the application, how the application can be navigated including the screens, the input forms and the possible errors.

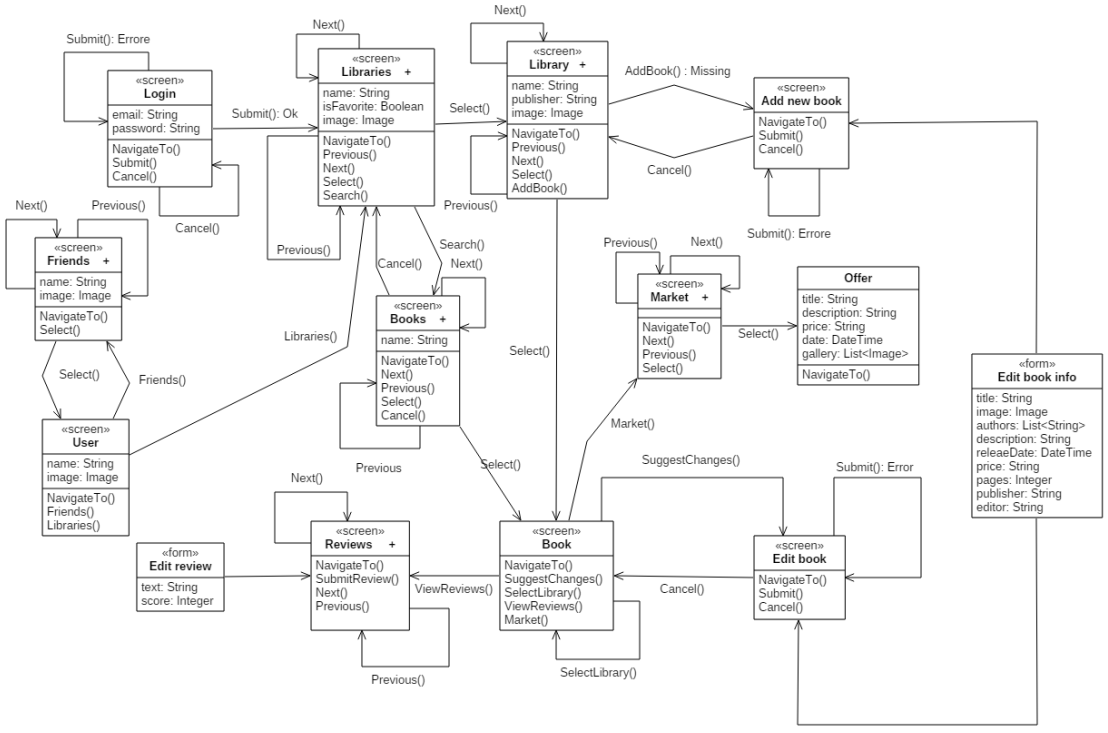


Figure 4.1: UX diagram

4.2 User Interfaces Design

The following images are the most important pages of the application.

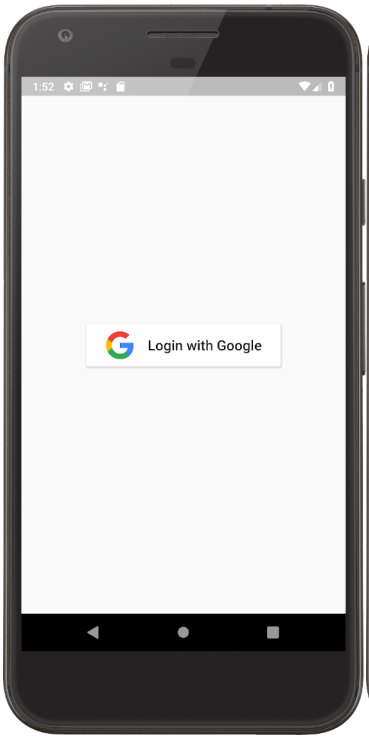


Figure 4.2: Login

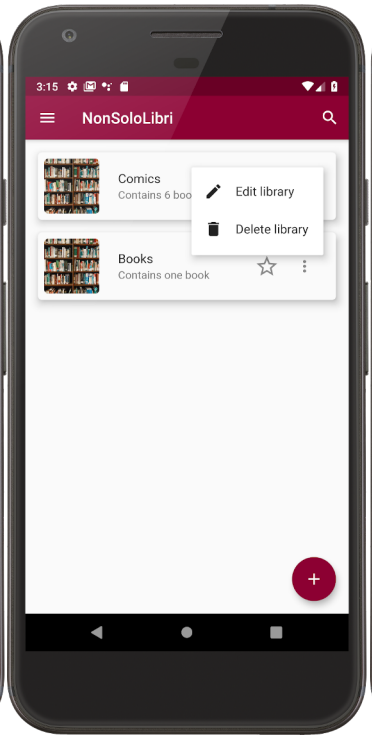


Figure 4.3: Libraries

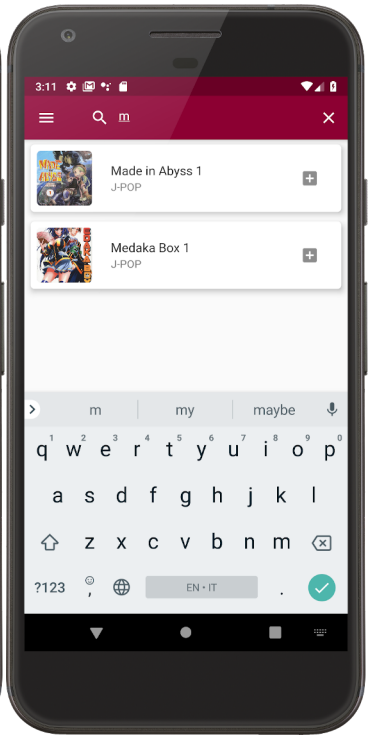
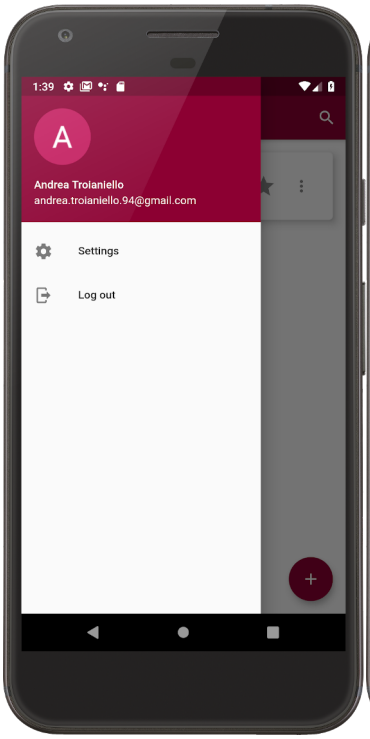
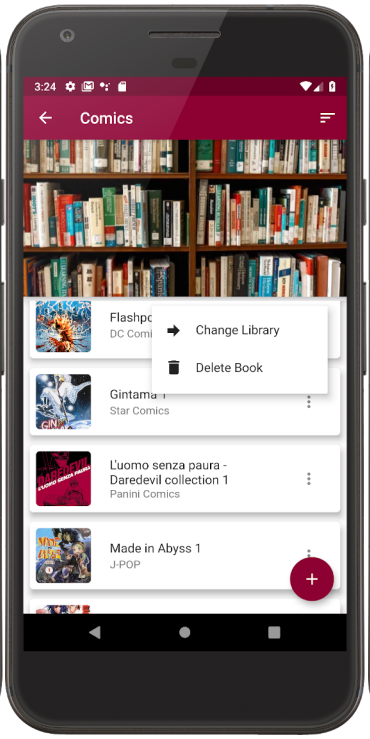


Figure 4.4: Search

The entry point of the application is login page (Figure 4.2), where the user inserts his Google credentials.

After, the page with the list of libraries is displayed. Each library can be edited or deleted.In this page there are a search icon, that allows to search through all books contained in the database (Figure 4.4) typing the desidered title, and a floating button, used to create of a new library.





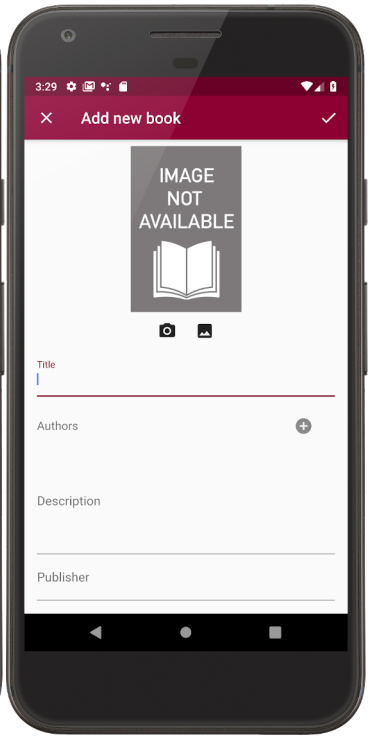


Figure 4.5: Menu Figure 4.6: Library Figure 4.7: New book

From the libraries page, can be opened a menu (Figure 4.5), which allows to reach the application settings, account functionalities and to log out.

After opening a library, all of books contained are displayed (Figure 4.6)and each of them can be moved to a another library or deleted.

The floating buttton opens the camera that reads the ISBN of the book and automatically add it. If it doesn't exists, opens a dialog, that allows to insert the infomration of a new book.

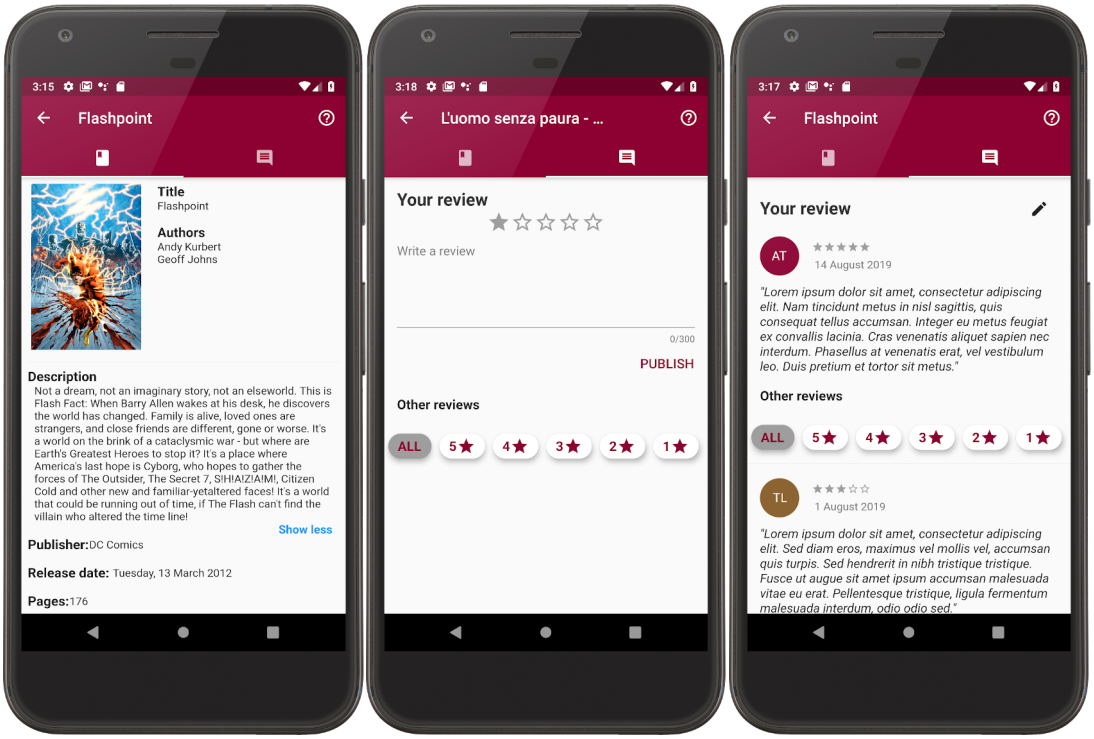


Figure 4.8: Book info Figure 4.9: No review Figure 4.10: Reviews

The book page (Figure 4.8) uses a tab bar to display all information releated to the book.

The first page contains the specific information, like the title, image of the cover, list of author and description.

The second one (Figures 4.9 and 4.10) shows all reviews. This section is divided in two parts. The fist part shows the review wrote by the user, if no exists allows to write it, otherwise displays it and, using a button, allows to edit it. The second part shows a scrollable list of reviews wrote by other users and can be filtred based on the scores.

The third page allows to search the best offer associated to the book and to make new one containg a descrption and a gallery of photos.

4.3 Hardware Interfaces

This project doesn't require any hardware interface.

4.4 Software Interfaces

- Database Management System (DBMS):
  - Name: Cloud Firestore
  - Source: `https://firebase.google.com/products/firestore/`
- Storage:
  - Name: Cloud Storage
  - Source: `https://firebase.google.com/products/storage`
- Server:
  - Name: Cloud Functions
  - Source: `https://firebase.google.com/products/functions/`
- Operating systems:
  - Name: Android
  - Minimum version: 5.0 Lollipop
  - API level: 21
  - Source: `https://www.android.com/`

  - Name: iOS
  - Minimum version: 10
  - Source: `https://www.apple.com/ios`

4.5 API Interfaces

The application communicates with the database on Firebase using:

firebase_core (version 0.4.0+8), a Flutter plugin to use the Firebase Core API, which enables connecting to multiple Firebase apps.

firebase_analytics (version 4.0.2), allows to use the Google Analytics for Firebase API, a free app measurement solution that provides insight on app usage and user engagement.

firebase_auth (version 0.14.0), a plugin to use the Firebase Authentication API, that aims to make building secure authentication systems easy, while improving the sign-in and onboarding experience for end users. It provides an end-to-end identity solution, supporting email and password accounts, phone auth, and Google, Twitter, Facebook, and GitHub login, and more.

cloud_firestore (version 0.12.9), a library that allows to communicates with the NoSQL cloud database of Google (Firestore).

firebase_storage (version 3.0.5), a Flutter plugin to use the Cloud Storage API. Cloud Storage is used to upload images or other files on Firebase.

During the adding new book phase, the ISBN is read using the barcode, that allowed by **flutter_barcode_scanner** (version 0.1.5+1).

For the user's login, the system uses the Google Sign In API. This API provides the authentication of the user and retrives his main information (name and email). The used libraries are **google_sign_in** (version 4.0.6) and **firebase_auth**.

The **plugins_intl** (version 0.15.8) and **intl_translation** (version 0.17.3) are used to generate and manage the language of the application, according to the language of the OS.

Chapter 5

Implementation and Test Planning

5.1 Implementation

For the client, the application has been implemented using Flutter, an SDK that simplifies cross-platform development and allows to write Dart code that can be compiled into native languages without loss of performances. Thanks to flutter we were able to write in only one language but to support two, different, operative systems.

For the backend it was decided to use Firebase to simplify the design and to take benefit of the infrastructure provided by Google, that offers both reliability and security.

Before the development, the application was divided in various components - called Widgets, in Flutter - that could be developed independently and a bottom up approach was choosen. Furthemore, the design of the application was thought with flexibility in mind, and the core of the application is completely independent from external libraries such those to access to hardware of the device, or those that implement the database.

At the moment of writing this document, all the features related to organizing a user library were implemented, while those related to the social network aspect of the application and to the market-place are yet to be implemented.

5.2 Testing

The application has been tested using the native tools provided by both Firebase and Flutter.

For firebase, the testing tools provided by the Firebase console were used. In this way we were able to test users permissions and API calls, that were later used in the application.

For Flutter the library *flutter_ test*, which is provided by the Flutter SDK, was used, together with *mockito*, an external library that simplifies mocking of dependences.

Flutter_ test was used to generate unit, widget and integration tests, that were used to test single functionalities and to test together the various widget that compose the pages.

Mockito was used to create mocks of external libraries that can not be tested within the application such as the photo camera, the barcode scanner and the Firebase implementation. With this approach we were able to test most of the implementation.

In addition, extensive tests were made by real users, by testing the application in daily conditions with both physical devices and emulators.

Bibliography

[1] Pantieri, Lorenzo e Tommaso Gordini (2017), *L'arte di scrivere con LaTeX*, `http://www.lorenzopantieri.net/LaTeX_files/ArteLaTeX.pdf`.