EASYLIB

DD

Design Document

Kostandin Caushi 898749

Raffaele Bongo 900090

Date 18/02/2019

POLITECNICO
MILANO 1863

Table of Contents

List of Figures

1 Introduction

1.1 Purpose

The purpose of this document is to provide a full description of the design of the EasyLib and EasyLib - Librarian mobile applications, two native Android applications, providing insights into the goal of the system, the design of each component and how the project has been managed.

1.2 Scope

EasyLib is a mobile application that manages to make easier multiple management tasks that nowadays a librarians and libraries' users needs to deal with. It answers all the questions for which a library visitor usually stuck for an answer such as books description and availability, waiting queue for reserve a book, events taking place and news related to the library, and make really easy the book search and reservation. The application allow the user to interface with all the libraries that choose to use EasyLib as management system, using a single identification process to interact with all of them. EasyLib has been thought to cover also the Librarian necessity and help him into his daily work automatizing the book delivering and returning process and helping him to schedule his work. However, the librarian system that we are going to present in this document is a prototype with a fraction of the functionalities that have been designed and are left to a future upgraded of the app.

1.3 Definitions

- *System :* with the term system we will consider the client-side android applications and the server (not the DB).

1.4 Acronyms

- *API :* Application Programming Interface
- *DB :* Database

1.5 Abbreviations

- [Gn] : Goal n
- [Rn] : Requirement n

1.6 Document Structure

This paper is divided in six chapter.

The first chapter is composed by an introduction of the system, its application domain, its goals and a glossary containing the most common expression used in order to give to the reader a basic knowledge of the system and to make him understand better the subsequent parts.

In the second chapter we have provided a list of all the applications goals and the related requirements and we also listed their core functionalities with a synthetic description.

The third chapter is devoted to the exposure of the overall design architecture of the system, the identification and definition of all the software components that characterize our system end to end, the external services and libraries used to accomplish all the goals that we aimed to satisfy. We also shown a deployment proposal and non-functionals requirements that we need to respect.

The fourth chapter is exploited to show the most relevant decisions taken in the UI design and to show how the application looks to the final user using images taken during the real usage.

In the fifth chapter we showed some runtime view diagrams to explain how the most sensitive functionalities of the system are carried out and the interaction between the different hardware tiers and software components.

Finally, in the sixth chapter we have put in writing the implementation strategy that we followed to implement the system, how our team has been structured, the implementation process carried on and the test cases performed.

2 Overall Description

2.1 Goals & Requirements

2.1.1 EasyLib

[G_1] : Allow the user to register and login into the system (also throughout external services)

- [R_1] : The system has to check the credentials.
- [R_2] : A registered user must be able to log in the system.
- [R_3] : The System has to handle the connection and user requests.
- [R_4] : The system has to be able to save locally the user credentials to allow the automatic login.
- [R_5] : The system has to be able to communicate with the external services and receive the responses.
- [R_6] : The system has to be able to show to the user his profile information.

[G_2] : Allow the user to select his favourite library

- [R_1] : The system has to be able to check and retrieve libraries information from the DB.
- [R_2] : The system has to be able to recognize the libraries already set as favourite by the user.
- [R_3] : The system has to be able to set the preferred library selected by the user into the DB.
- [R_4] : The system has to be able to remove a library from the favourite libraries present in DB for the considered user.
- [R_5] : The system has to be able to show to the user his favourite libraries.

[G_3] : Allow the user to see all the libraries available into the system and their content

- [R_1] : The system has to be able to retrieve library's information and contents.
- [R_2] : The system has to be able to show the retrieved information and contents related to the libraries.

[G_4] : Allow the user to search a book in the available libraries

- [R_1] : The system has to be able to retrieve book's information from the libraries using title, author, genre and/or library name.
- [R_2] : The system has to be able to filter books using the book identifier, so it will show only one copy of the book even if it's present in more than one library.
- [R_3] : The system has to be able to show the book information to the user.

[G_5] : Allow user to retrieve book information through QRcode scan

- [R_1] : The android app needs to have access to the camera sensor.
- [R_2] : The android app has to be able to decode the QRcode attached to the book.
- [R_3] : The system has to be able to retrieve book information using the identifier decoded.

[G_6] : Allow the user to receive notifications from the server

- [R_1] : The Application Server has to be able to send notifications to the android app.
- [R_2] : The Android app as to be able to receive and handle incoming notifications.
- [R_3] : The Android app has to be able to show the notifications to the user.

[G_7] : Allow the user to reserve books

- [R_1] : The System has to retrieve the information of the book status from the DB.
- [R_2] : The System has to be able to change the status of the book to reserved / not reserved.
- [R_3] : The System has to show to the user all the books that he reserved.

[G_8] : Allow the user to get in line for a book

- [R_1] : The System has to retrieve the information of the book status from the DB.
- [R_2] : The System has to be able to change the status of the book to "on waiting list" / "not in waiting list".

- [R_3] : The System has to show to the user all the books for which he is in line.
- [R_4] : The System has to be able to manage the waiting list flowing when some reservation related to the considered book is removed.

[G_9] : Allow the user to see the books that he read

- [R_1] : The System has to be able to retrieve the books read from the user.
- [R_2] : The Android application has to be able to show the books read by the user.

[G_{10}] : Allow the user to rate a read book

- [R_1] : The System has to be able to retrieve the books' information.
- [R_2] : The System has to be able to recognize if the book has been read by the user.
- [R_3] : The System has to be able to insert the user rating into the DB.
- [R_4] : The DB has to be able to update the average rating after each rate insertion.

[G_{11}] : Allow the user to reserve a seat for an event

- [R_1] : The System has to be able to retrieve the events from the DB.
- [R_2] : The System has to check the available seats.
- [R_3] : The System has to allow the user to insert his participation to the event in the DB.
- [R_4] : The System has to allow the user to remove his participation to the event in the DB.
- [R_5] : The Android application has to show to the user his status related to the participation to an event.

[G_{12}] : Allow the user to see all the events that he joined

- [R_1] : The System has to be able to retrieve all the events joined by the considered user.
- [R_2] : The Android application has to be able to show the events joined by the user.

[G_{13}] : Allow the user to edit his profile information

- [R_1] : The Android application has to be able to show to the user his profile information.
- [R_2] : The System has to be able to retrieve the user's profile information from the DB.
- [R_3] : The System has to be able to update user's profile information into the DB.

2.1.2 EasyLib - Librarian

[G_1] : Allow the librarian to login into the system

- [R_1] : The system has to check the credentials.
- [R_2] : Registered user must be able to log in the system.
- [R_3] : The system has to handle the connection and user requests.
- [R_4] : The system has to be able to show to the user his relative library information.

[G_2] : Allow librarian to retrieve book information through QRcode scan

- [R_1] : The android app needs to have access to the camera sensor.
- [R_2] : The android app has to be able to decode the QRcode attached to the book.
- [R_3] : The system has to be able to retrieve the book information using the identifier decoded.

[G_3] : Allow the librarian to confirm/remove reservations for a specific book

- [R_1] : The system has to be able to retrieve the reservation list for a book.
- [R_2] : The system has to be able to get the librarian input and change the reservation status of a book.

[G_3] : Allow the librarian to communicate to the system when a book is returned

- [R_1] : The system has to be able to retrieve book information and check its status.
- [R_2] : The system has to be able to trigger the insert in the reservation list of the requests in waiting list and send them a notification.

2.1.3 EasyLib - Librarian (Future Work)

[FW_1] : Allow the librarian to register into the system.

[FW_2] : The system has to be able to save locally the librarian credentials to allow the automatic login.

[FW_3] : Allow the librarian to search for a book in his library.

[FW_4] : Allow the librarian to edit library information.

[FW_5] : Allow the librarian to change library contents (add/remove news, events and books).

2.2 Product Functions

2.2.1 EasyLib

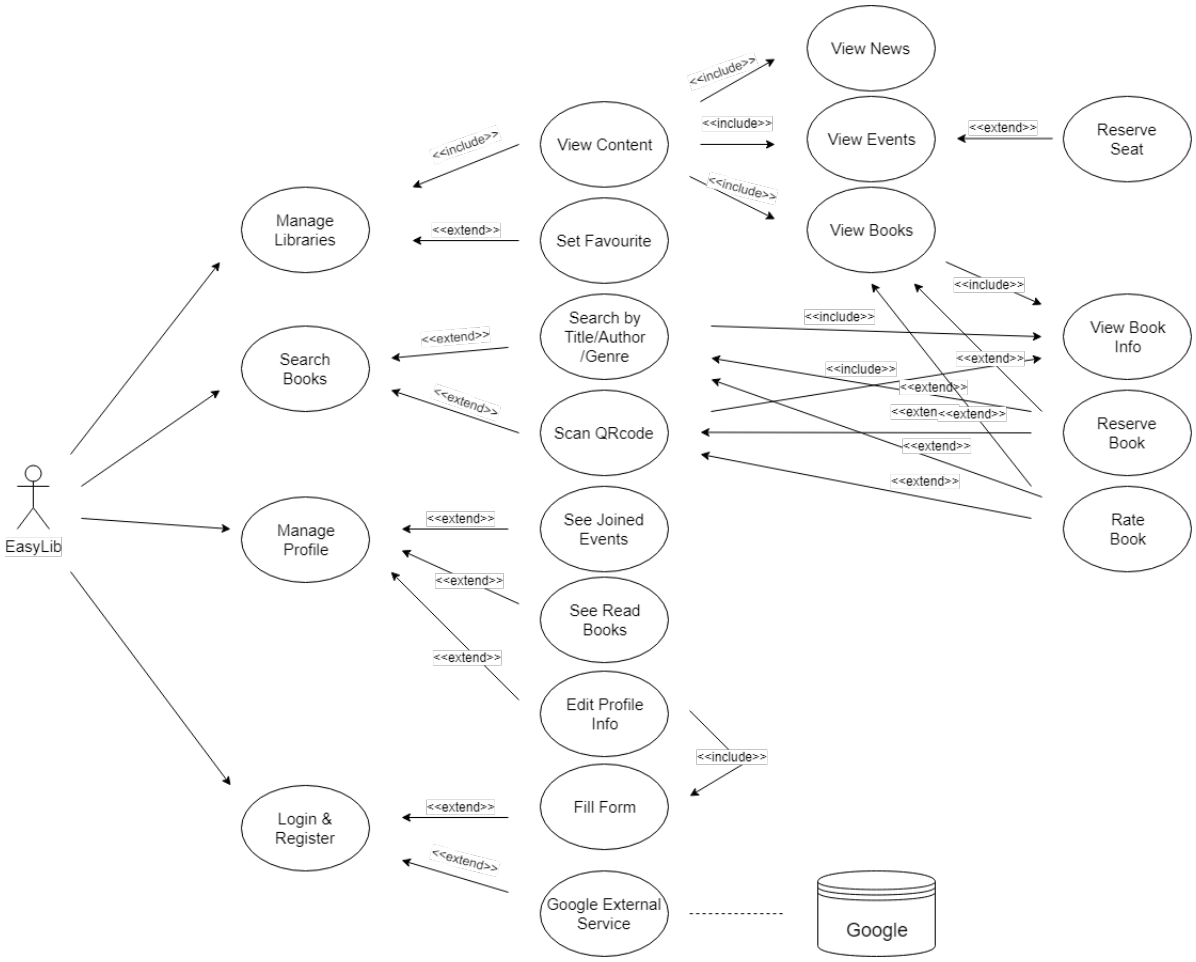

Figure 1: EasyLib - Use Case

1. *Advanced book search:* the user can search a book in the libraries using any combination of parameters between title, author's name and category and, if he needs, he can also restrict his search in only one of the available libraries. If the user is in the library, he can scan a QR code on the book to retrieve all its information.

2. *Consulting book information:* the user can consult all the available book's information such as title, author's name, the number of copies of the book available in each library, etc.

3. *Reserve or get in line for a book:* the user can reserve a copy of the book if available or get in line for it just tapping a bottom.

4. *Visualize libraries' news and events:* the user can remain constantly up-to-date with the new events organized and news posted by the available libraries checking out their dedicated pages.

5. *Book a seat to an event:* : the user can book a seat in the events organized by the libraries to avoid the risk of missing the meeting with his favourite authors.

6. *Save favourite libraries:* the user can select his favourite libraries between that ones present on the system in order to have a rapid access to their books, events and news.

7. *Set ratings:* the user can rate the books that he has read to spread his opinion to other readers.

8. *Keep track of read books:* the user can always access to the list of the book that he read directly and intuitively from his profile.

2.2.2 EasyLib - Librarian

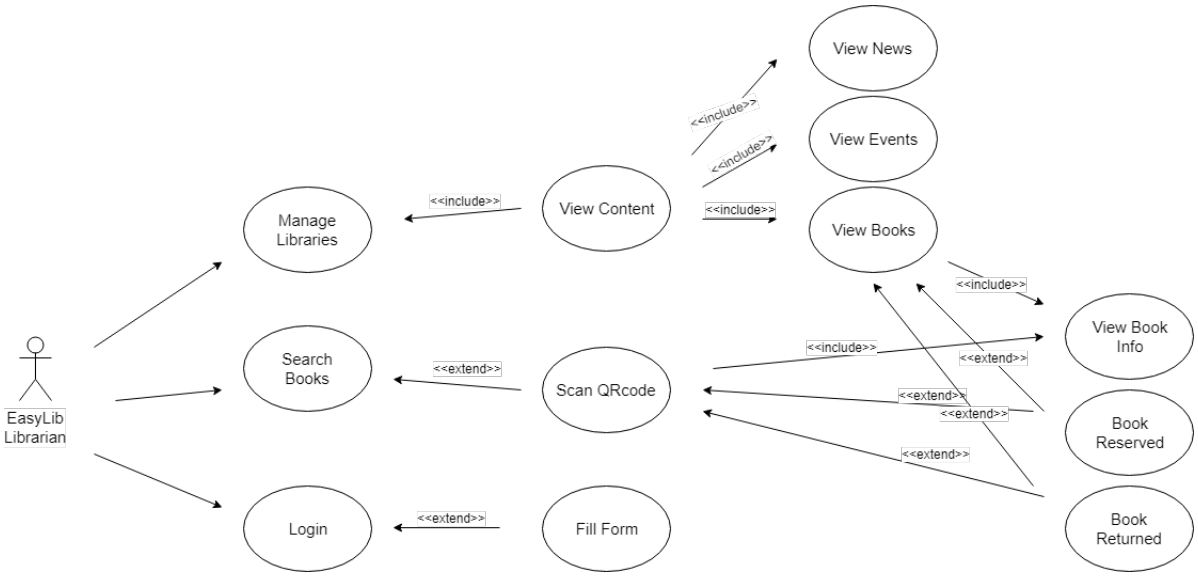

Figure 2: EasyLib - Librarian - Use Case

1. *Visualize his library information:* the librarian can visualize all the information inserted in his library such as books available, events, news, etc.
2. *Identify the user that reserved a book:* the librarian can scan the QRcode of a book and retrieve all the reservation for that book and the related user id. In this way the user can show to the librarian his personal Id authenticating himself and allowing the book delivering. A paired process is followed when the user returns a book.
3. *Record that a user has returned a book:* with the same approach of the former feature, the librarian can update into the Database a book status to returned.

2.3 Domanin Properties

In order to have the aimed behaviour of the app, few domain constraints must be satisfied.

- The user needs to have a mobile phone or a tablet that support Android OS.
- The Android OS version installed on the device must be KitKat (API level 19) or higher.
- An access to an internet connection is always needed.
- The users must be able to easily reach the physical library location to use the functionalities that need an interaction with the librarians, such as take and return a reserved book.

3 Architectural Design

3.1 Overview

In this section we describe the architectural arrangement that we designed for EasyLib's client and server side.

The proposed architecture is composed by three tiers:

- *Presentation Tier:* it's represented by the tablet and the Mobile Android App. They constitute the view part of our system.
- *Business Tier:* it's represented by the Socket Server, which responds to the user's requests, and the Application Server, which contains all the Business Logic. The application server communicates with the DBMS of the MySQL Database contained in the Database Tier.
- *Database Tier:* it's represented by the DB Server, that contains and manages persistent data in an efficient way.

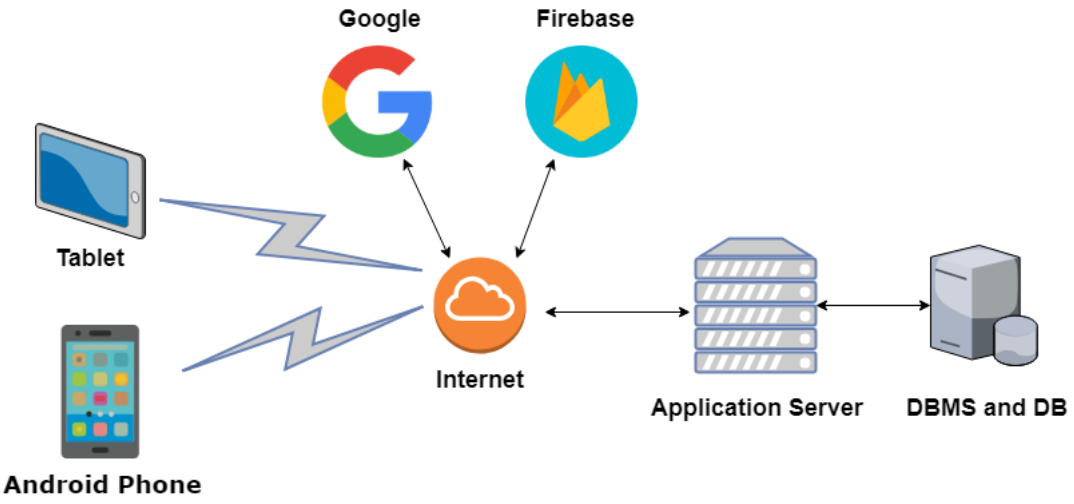

Figure 3: High level system architecture

3.2 High Level Component View

The system is divided in three main layers: Presentation Layer, Business Layer and Data Layer. The presentation layer is represented by both mobile and tablet application.

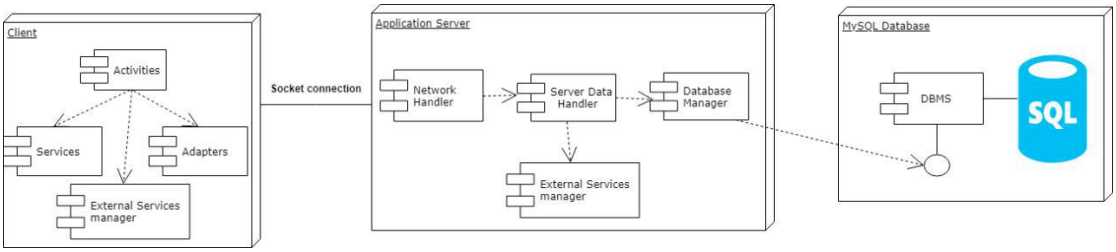

Figure 4: Component View

3.2.1 Client-side Application

In this section we are going to illustrate the general architecture of our client application. EasyLib can be fairly considered in the middle between a thin and a fat client. The data are requested client side and correctly retrieved with the work performed by the application server and the DBMS. Nonetheless, on the client side the data are checked, filtered and formatted in order to give the most pleasant screen view.
In order to accomplish efficiently the communication task, the client exploits two different android services:

- *Connection service:* it manages the communication with the server. It opens a socket connection and keep it alive in order to send and receive messages in any moment and in an asynchronous fashion.
- *Check connection service:* It constantly checks if there is internet connection available and, if is not the case, makes the user aware of this forbidding any dangerous actions that could lead to a misbehaviour of the app.

We implemented a system to decode the messages received from the server. The messages have to be passed through two different classes before reaching the application context and be manipulated to modify the view. The service thread is in charge of this massage handling. First of all the message received is used as key in a functional HashMap to trigger the correspondent method that receives the expected object from the input socket channel, wrap it in a bundle and pass it to the next class. Here another functional map allows to call the correct method and send the object along with its related message to current application context where a Broadcast receiver component is listening for incoming inputs.

3.2.2 Application Server

As said in the previous paragraphs, EasyLib is an application that allows the user to interface with several libraries. It does so by centralizing all the libraries data inside a server that can manage multiple clients with different access privileges, that is libraries' users and librarians, simultaneously. The clients establish a socket connection with the server from their devices and, previous authentication, it manages many kinds of stateless requests.
Our application server is divided in three parts: socket connection layer, request decoding layer and database manager.

- *Socket connection layer:* this layer is always listening for new connection requests. When a new client claims to log in or register to EasyLib, the connection layer grab its request creating a new socket connection. After that the TCP connection has been established, all the communications pass through this one and the messages delivered to the subsequent component.
- *Request decoding layer:* this layer implemented in the Java class called ServerDataHandler, receives the request grabbed by the connection layer and decodes it in order to trigger the functions to produce the requested answer on the DatabaseManager class and take back their output. The decoding of the requests has been made using a functional Hash Map that has as key a string and as value a function reference that is triggered when the correspondent string is received from the client.
- *Database manager:* this final layer is the core of the system business logic. It receives the directive from the previous layer, produces SQL statement and send them to the DBMS to retrieve the requested data. Then, it manipulates and format the data before sending them back to the client through the Socket connection layer.

3.2.3 External Services and libraries

EasyLib exploits some external services to accomplish its main tasks.

- *Google sign in:* we use google sign in API to allow the user to identify himself in our system with his google credentials.
- *Firebase Cloud Messaging:* we use Firebase Cloud Messaging system to manage the notification that our server needs to send to the user.
- *Google Books API:* we use google books API to fill the books schema of the libraries' database.

Furthemore, one of the most interesting feature of our application is to exploit the camera in order to decode a QRCode and look for the book related to it. To perform this operation, we exploit the potential of *Zxing* library.

3.3 Data Layer

The data layer is composed by a multi-table MySQL database that allow to persistently store all the data that the application need to provide its services. In the current moment it is composed by three schemas: the proprietary_db, library_1 and library_2. The EasyLib system is designed to be scalable and include any number of libraries that want to join our project. The only constraint that they have to follow is to homologate their database to our scheme, in order to provide the access to their information to the application.

3.3.1 Database structure

The Database is composed by two main schemas: proprietary_db and library_x. The former contains the information about the users and librarians' identity, the libraries' info needed to distinguish them, the book read by all the users and their favourite libraries.
The latter contains all the information strictly related with a single library. The x stands for an integer and represents the id related with the library. It contains information about the books, the news, the events and their participants, the books reserved and the queue for them and the ratings related to the book read in that library.
Some of the tables in these schemas contains triggers used to maintain the consistence of the data and to build more meaningful information to show to the final user.

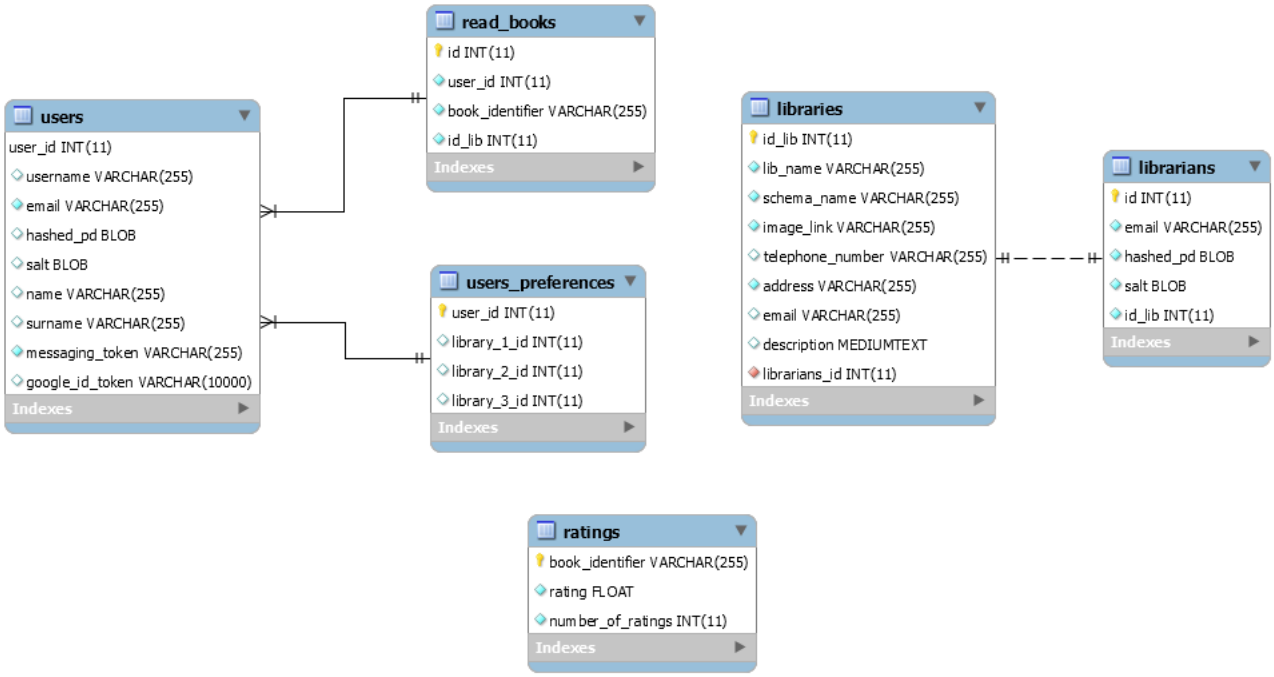

Figure 5: Proprietary database

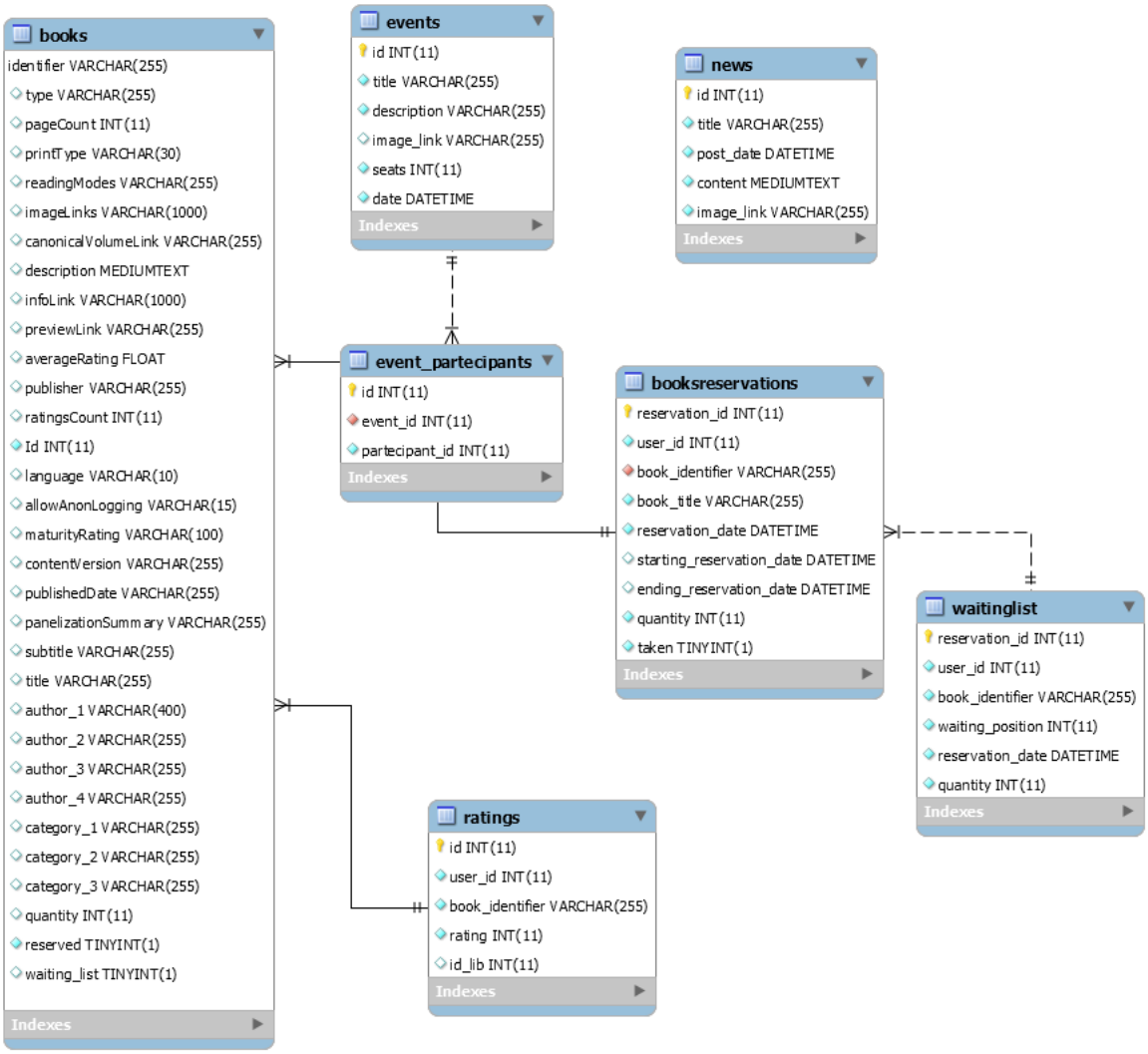

Figure 6: Library database

In order to maintain the data consistency after that users perform state changing operations, we have implemented 9 triggers distributed between the various schemas and tables. We list all of them below along with the description of their behaviour.

3.3.2 Triggers - Library schema

Trigger name	Table name	When
On_insert_rating	ratings	after insert

It computes the updated average rating of a book including the inserted rating and store it in the table ratings of the proprietary_db if it does not exist or update a currently existing record.

Trigger name	Table name	When
update_count_on _delivering	booksreservations	after delete

It updates the count of available copies for a book in books table in library_x schema after that one of them has been delivered. If the current available number of copy result to be zero after the action, the Boolean reserved is set to true.

Trigger name	Table name	When
waiting_list_flowing	booksreservations	after delete

It updates the position of the users in line of the book with the identifier equal to that one of the record deleted in the booksreservations table and remove the first waiting user for the book since it has to be inserted in the bookreservations table.

Trigger name	Table name	When
update_on_reservation	booksreservations	after insert

It updates the count of available copies for a book in books table in library_x schema after that one of them has been returned to the librarian of library x.

Trigger name	Table name	When
update_on_reservation	booksreservations	after insert

It updates the count of available copies for a book in books table in library_x schema after that one of them has been returned to the librarian

of library x.

Trigger name	Table name	When
update_available_seats_ins	event_partecipants	after insert

It updates the count of available seats for an event after a registration for it has been performed.

Trigger name	Table name	When
update_available_seats_del	event_partecipants	after delete

It updates the count of available seats for an event after that a registration for it has been removed.

Trigger name	Table name	When
on_insert_waiting_person	waitinglist	before insert

It sets the position in the waiting list of the new user inserted in the table for a book.

Trigger name	Table name	When
update_waitinglist_position	waitinglist	after delete

It updates the position of the users in line for a book after that a user in line has deleted is presence in the queue.

3.3.3 Triggers - Proprietary schema

Trigger name	Table name	When
on_insert_rating_propDB	ratings	after insert

It updates the average rating for the considered book in all the libraries of the system.

3.3.4 Internal customized data structures

In order to make the communication easier and the data transmission between the clients and the application server manageable, we have created few serializable data structures that are sent over the internet. They are 13 java classes designed with the purpose of containing all the information that server and clients need to exchange such as: books information, reservation details, events schedule, etc.

3.4 Deployment View

Here we provide a deployment strategy for our system.

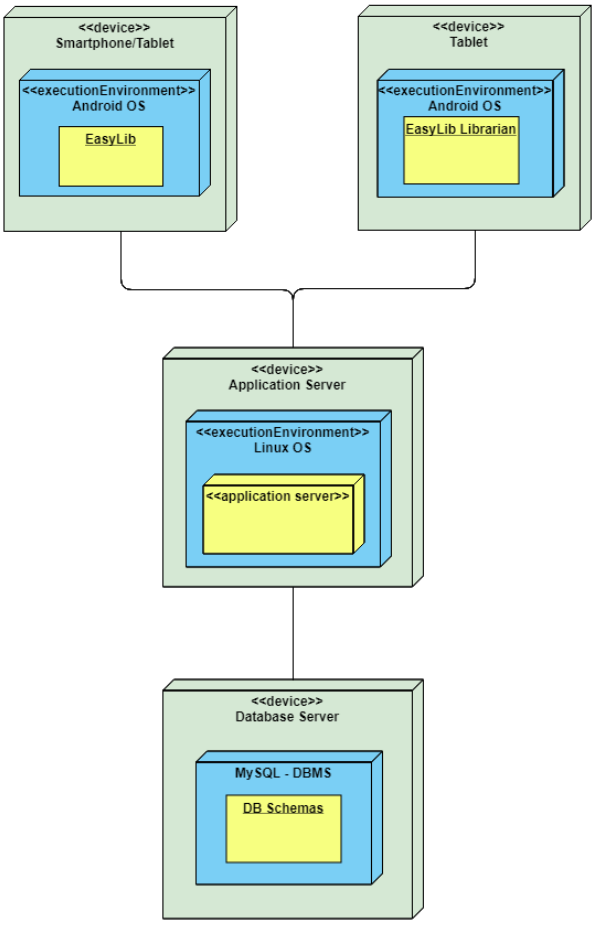

Figure 7: Deployment view EasyLib

3.5 Non-Functional Requirements

Here we state some non-functional requirements that our system must respect during the execution.

- *Performance:* The system must be able to respond to a consistent number of user requests simultaneously. All the data requested by the users and the needed computations must respectively be retrieved and performed as soon as possible and as efficiently as possible.
- *Reliability:* The system must guarantee a 24/7 service. The information must be stored in proprietary Database in order to be always available to the user when required.
- *Scalability:* The system must ensure a high level of scalability, since new libraries can join the system in any moment enlarging EasyLib supply.
- *Security:* All users' passwords are encrypted using a hash function with salt. In this way the attackers cannot see the plain text password and steal sensitive information.
- *Accuracy:* The data retrieved by the application has to be accurate and up-to-date in a real-time fashion.

4 User Interface Design

The two Android applications are available for multiple screen sizes. In particular :

- *EasyLib :* is available for normal and xlarge sizes, so this means for smartphone around 5" and tablet of 10". Moreover it's available only on portrait orientation.
- *EasyLib - Librarian :* for now, it's only provided for tablet (so xlarge size), but it has portrait and landscape orientation, in order to improve the usability.

4.1 Distinct Features used for UI

We have created and used some customized components in order to improve the usability and look of our applications (*Better Look & Feel*).

Here are some of the main aspects :

- *Customized Adapters :* we have used different adapters for different recyclerViews in order to distinguish different element types and make our app less monotonous. There are some very simple ones and others more complex and challenging to make (like the one that has recyclerViews inside other recyclerViews).
- *Transparent Theme :* we have created a theme, associated to the single activities, that makes their background transparent instead of the "white" default one. This allows us to let the MainActivity, that after the Login, is the activity at the bottom of the stack, to fit all the screen size. While we make all the other activities, that comes after, look like pop-ups, with a black background with 80% of opacity, that looks like shadow.
- *SwipeBack Gesture :* for the "pop-up" activities, previously described, we added the "swipe right" gesture that actually destroys the activity.

4.2 App Functions & UI

4.2.1 Register & Login

When the app opens up the user can Login filling up the form with his Email or Password (Image 1) or he can Register opening the Register Activity and filling that form (Image 2). In other cases can happen that the user needs to reset his password or maybe he is already logged in. In the former case the app will show just a Loading (Image 3).

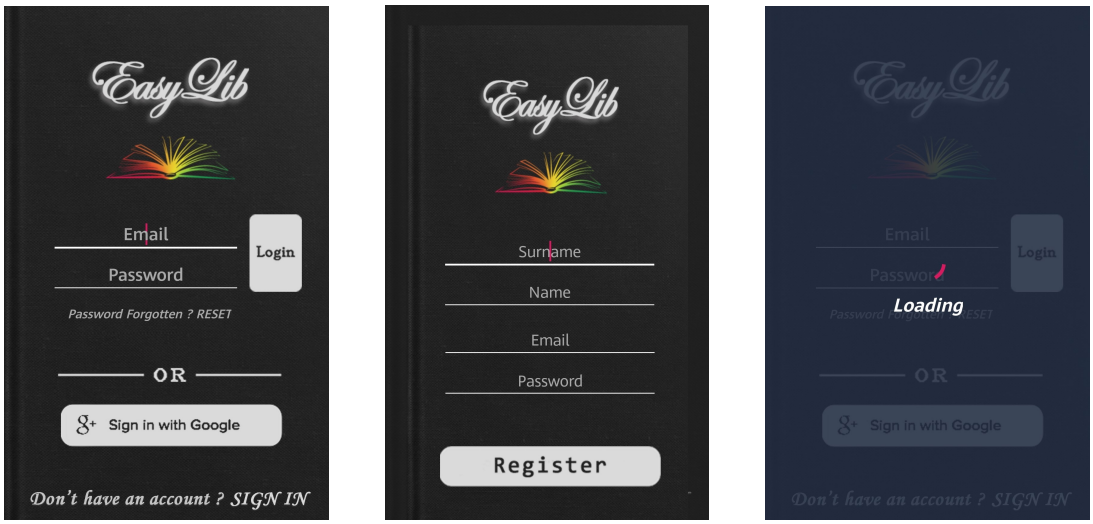

Figure 8: Login & Registration - UI

4.2.2 Libraries + Set Favourite

On MainActivity with the Home Fragment (Image 1) the user can see the Libraries that he has set as favourite and open them in order to see their content. Otherwise he can tap on "All Libraries" button and the list of the all available ones will appear (Image 2).

Once a library is selected the LibraryActivity will open up showing at the top the library information and its contents underneath (news, events, books). At the end a button will be shown in order to make the user able to set the library as Favourite / remove it from Favourites (Image 3).

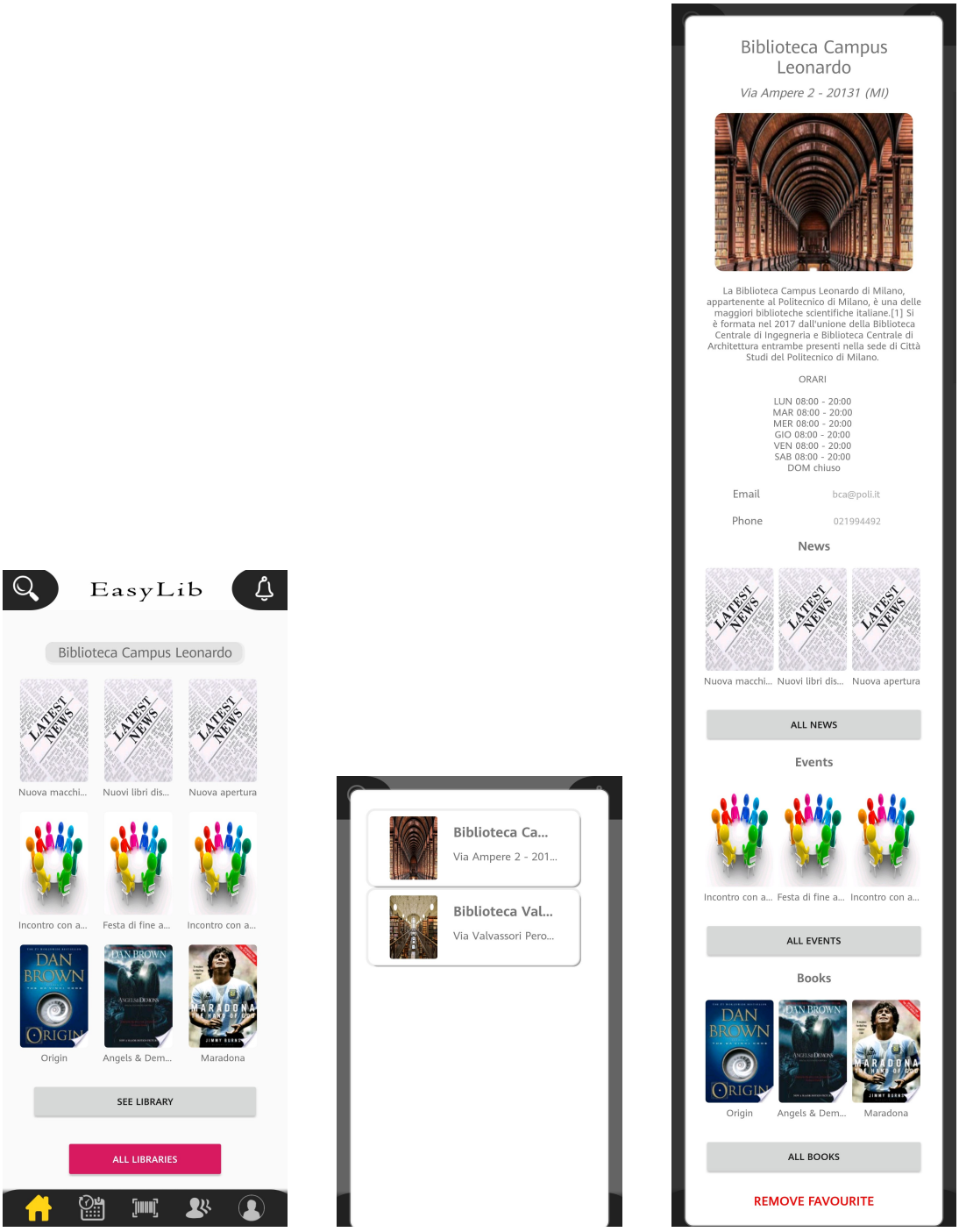

Figure 9: Libraries + Set Favourite - UI

4.2.3 Events + Reserve Seat

As previously seen, inside the LibraryActivity the events organized by the library are shown. The user can open them and see their info (Image 1). Moreover in case there are still available seats he can reserve one of them.

All the events joined by the user are shown on the Profile (Image 2).

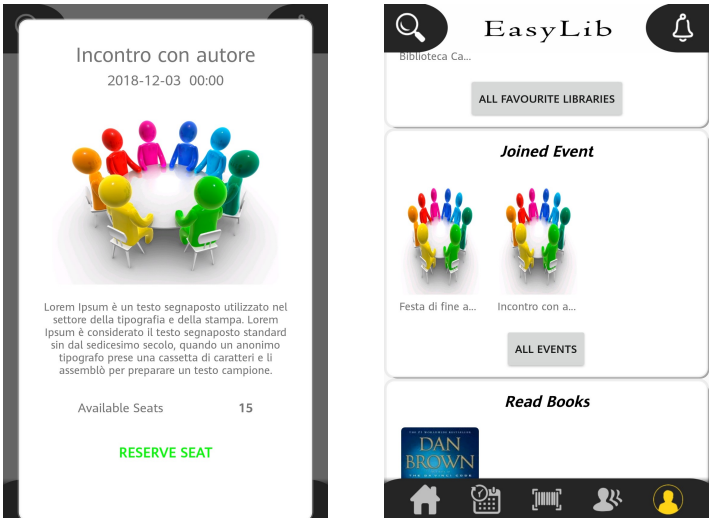

Figure 10: Events + Reserve Seat - UI

4.2.4 Search Books

There are 2 ways to search for a book :

- *By Title/Author/Genre :* the user can tap the "Search Icon" on top of the MainActivity (shown on the previous figures) and fill the Title field or activate the Advanced Search fields (Image 1).
- *QR code :* in the bottom NavBar on MainActivity there is a "Scan Icon" that when tapped opens up an activity that activates the camera and scans the qr-code (Image 2).

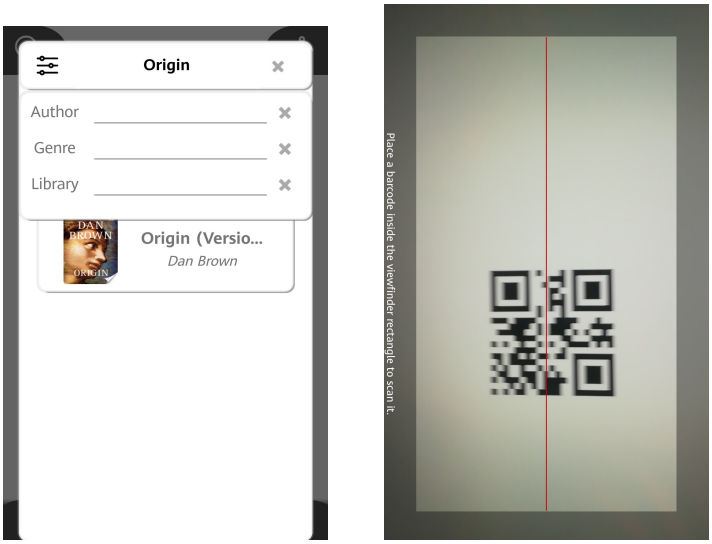

Figure 11: Search Books - UI

4.2.5 Books + Reservation

Once the user selects a book the "BookActivity" opens up and shows him its information and cover image (Image 1). At the end the current status of the book is displayed (if it's been already read or not) and also the list of all the libraries where the book available (Image 2).

If the user wants, he can reserve the book by tapping the "Reserve" button and update its status (Image 3).

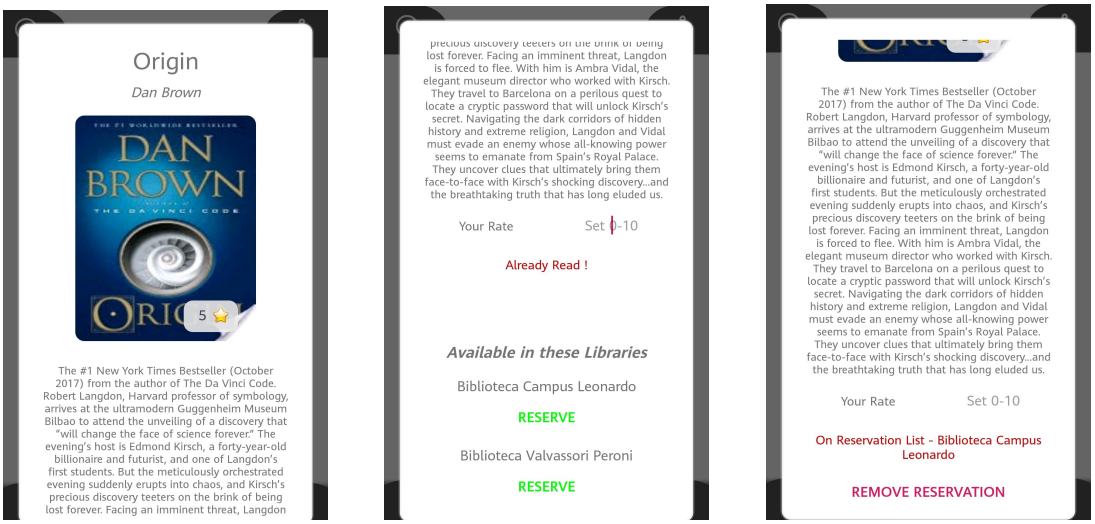

Figure 12: Books + Reservation - UI

The user can do that only in case there are available books, in a specific library, and see all the reservations in the "Calendar Fragment" on "MainActivity" (Image 4). Otherwise the user is added on the Waiting List. All the books for which the user is on Waiting List are shown on the "Queue Fragment" on "MainActivity" with the relative position on queue (Image 5).

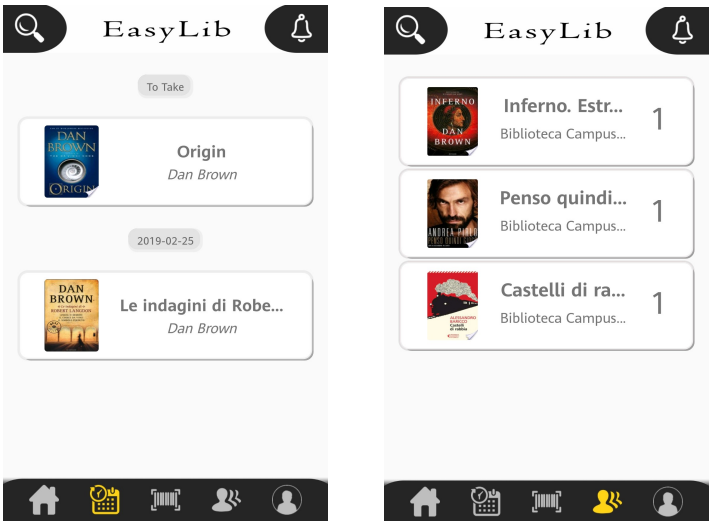

Figure 13: Calendar & Queue - UI

4.2.6 Profile

The user can see all his information on the "Profile Fragment" on "Main Activity", along with his USER ID (Image 1).

The fragment also displays all his favourite libraries, Joined Events and Read Books (Image 2).

In case of necessity he can modify it's credentials by tapping the "Edit Icon" on top of the "Profile Fragment" and by filling up the form shown on the "Edit Profile Activity" (Image 3).

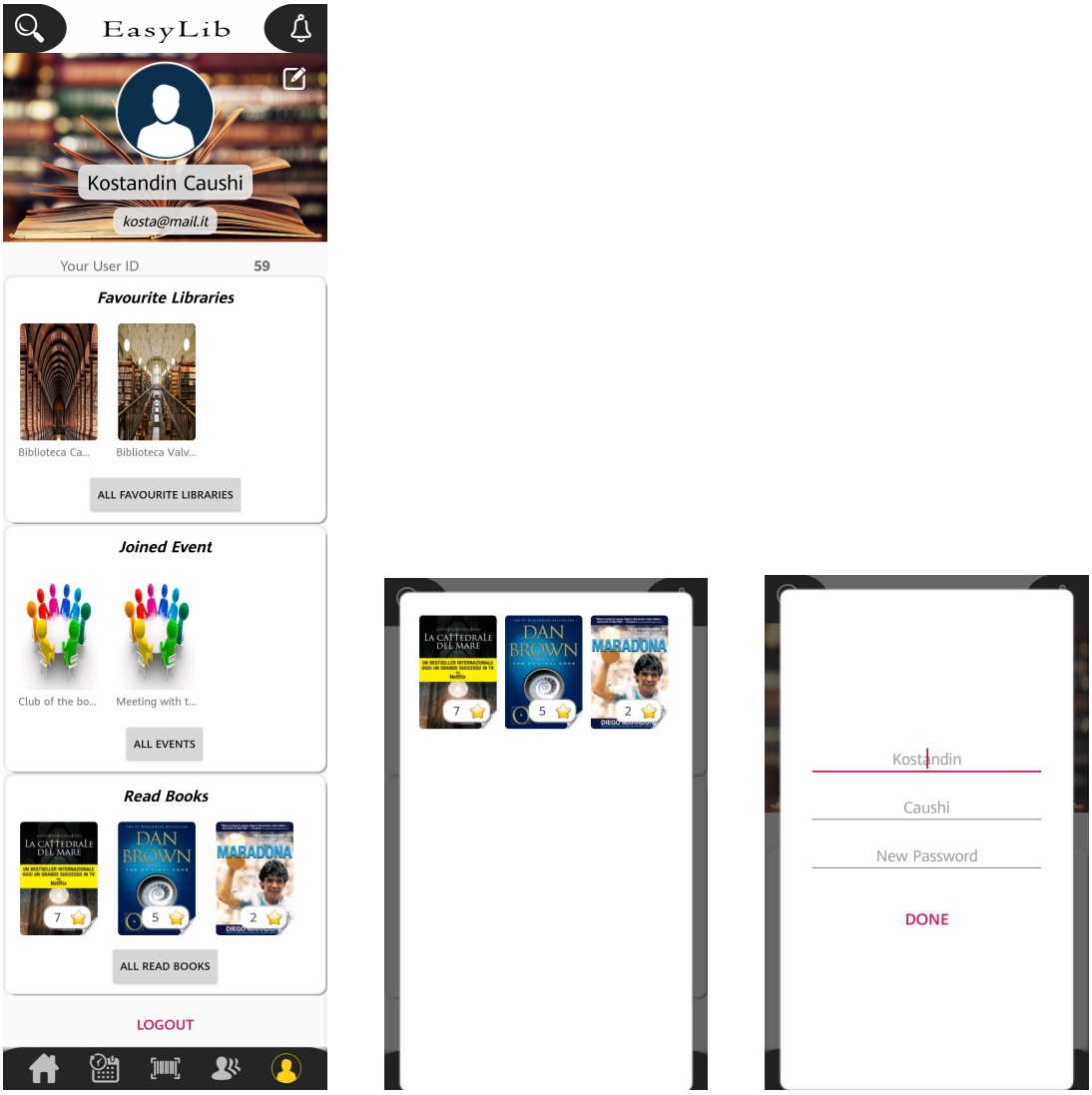

Figure 14: Profile - UI

4.2.7 Confirm Reservation / Book Returned (EasyLib - Librarian)

On the Librarian side, he can open the "Book Activity" scanning a qr-code and retrieving all the book's information and the list of the Reservations for it.

Based on the status of each reservation, different buttons are shown: if the book is already physically taken from the library then just the "Returned" button is shown; otherwise 2 buttons are shown: "Reserve", that confirms the user reservations and sets the status to "Taken", or "Remove" that removes the reservation.

The images underneath shows the portrait and landscape orientation of this activity on tablet.

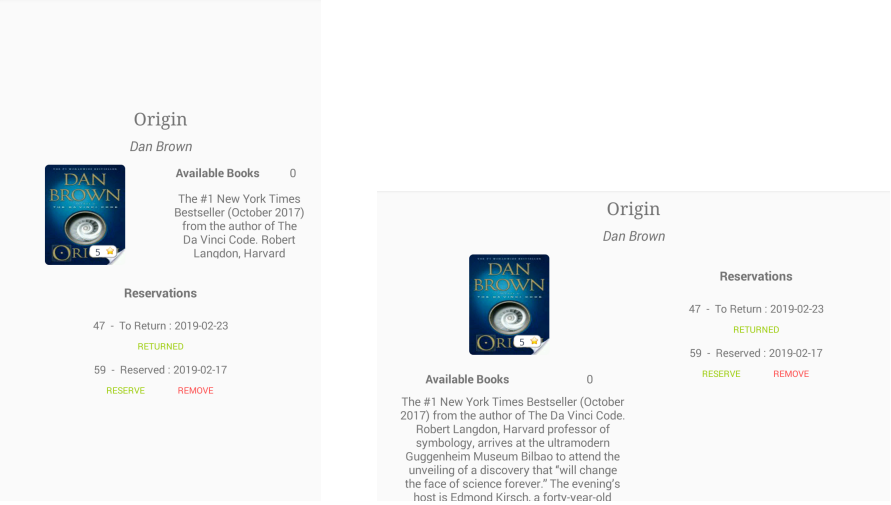

Figure 15: Confirm Reservation / Book Returned (EasyLib - Librarian) - UI

5 Runtime View

In this section we will show some Runtime Views in order to see how the components interact according to specific requests.

For the server-side we have taken into consideration the high-level Application Server only, without identifying its fine grained components. We also separated it from the DB, cause they rely on 2 different machines.

5.1 EasyLib

5.1.1 QR-Code Scan + Book Reservation

This Runtime View shows the different steps needed to retrieve book information through QR-code scan and reserve it.

We have to make an assumption : in the following diagram we reported only the most meaningful calls needed to achieve the goal.

So starting from the Main Activity the User has to tap the "QR scan" icon on the bottom NavBar and the EasyLib app will open the "QR Scan Activity". Thanks to the camera of the smartphone or tablet the QR-image is captured and an identifier is returned. This one is the book identifier and is sent to the server (and so the DB) to get back the information. The "Book Activity" is started and book information are set in the layout.

In order to let the user reserve the book, the app gets from the server (and so the DB) a list of libraries where is available. It is next passed to the BookActivityAdapter which creates the single items of the recyclerView placed under the book information.

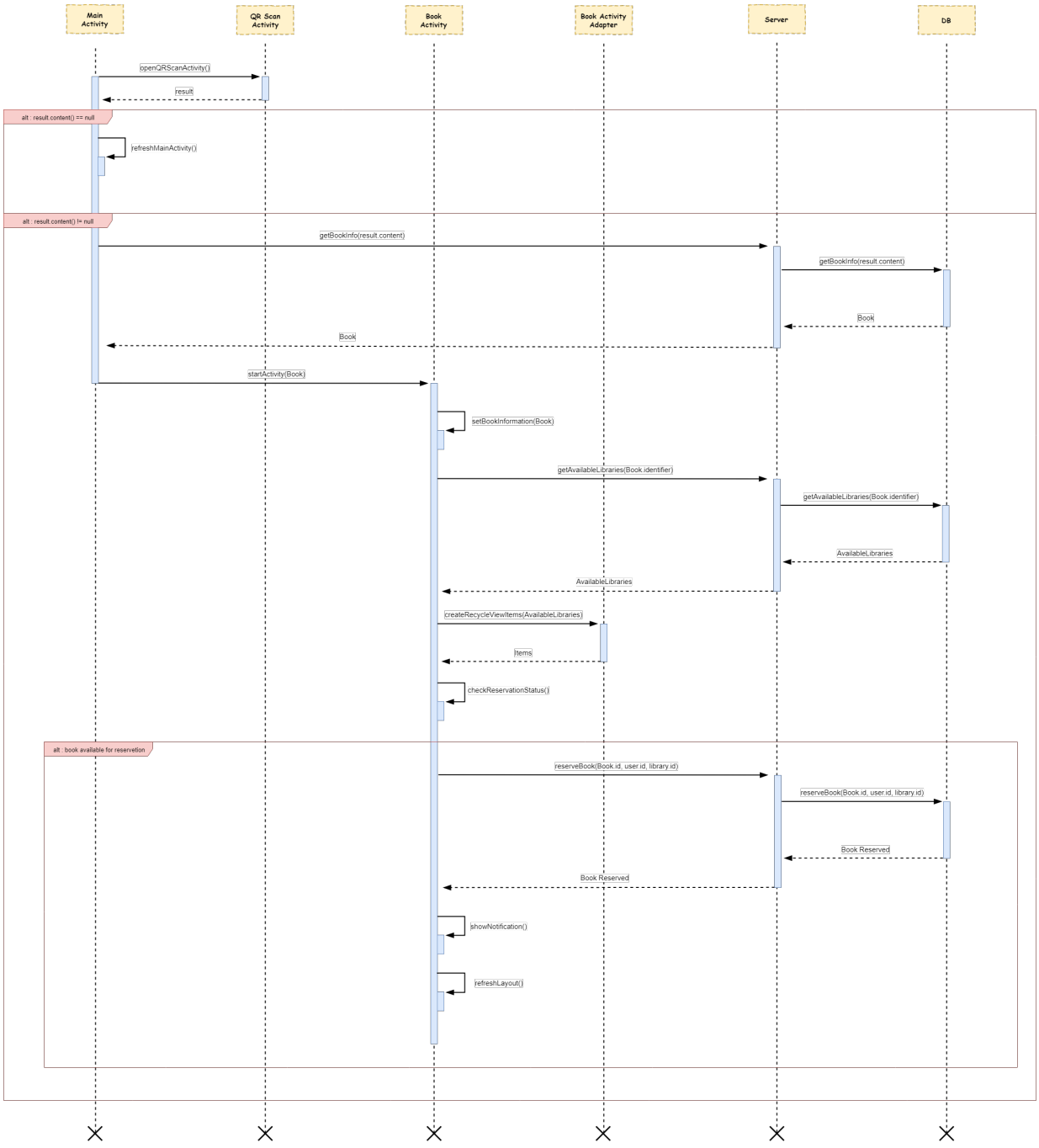

Figure 16: QR-Code Scan + Book Reservation - Runtime View

5.1.2 Rate Book

This Runtime View shows the steps performed to rate a book. We need to make some assumption :

- In the following diagram we reported only the most meaningful calls needed to achieve the goal.
- The book can be rated only if it was previously reserved through EasyLib app and returned by the user.

Starting from the Main Activity, the User has to tap the "Profile" icon on the bottom NavBar and the EasyLib app will ask the server for User information sending his identifier. Once data are back, "Main Activity" sets the "Profile Fragment" in the Main Frame.

Next the app asks the server for the read books and shows them in the "Read Books Activity" which contains a recyclerView. The user can select the wanted book and open it in the "Book Activity". When it starts the book information, passed with an Intent, are set in the layout and a check is excuted in order to see if it is already rated by the user or not. In case it is not, an EditText is showed and the user can fill it with a rate from 0 to 10.

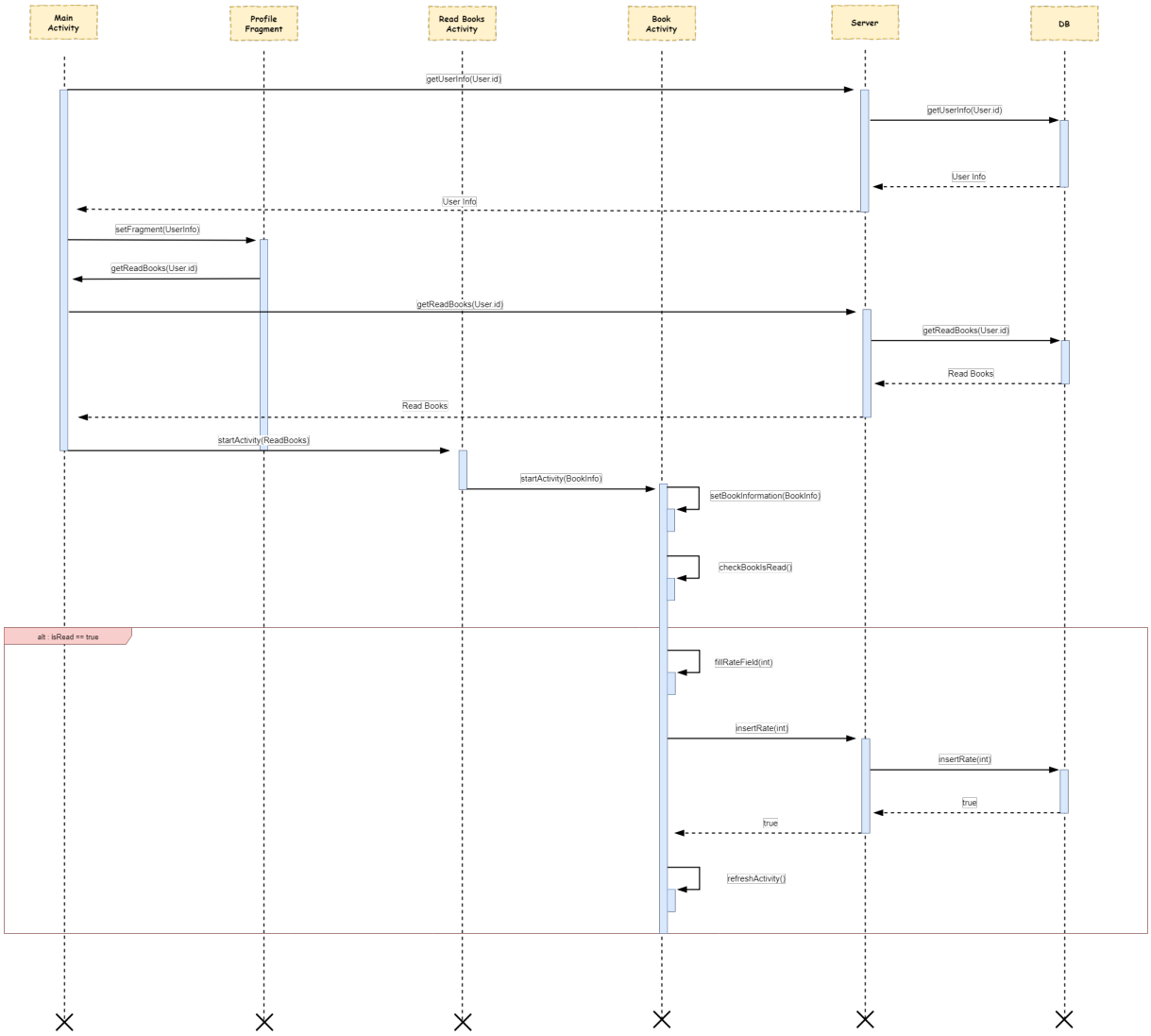

Figure 17: Rate Book - Runtime View

5.1.3 Event - Reserve Seat

This Runtime View shows the different steps needed to reserve a seat.

We need to make an assumption : in the following diagram we reported only the most meaningful calls needed to achieve the goal.

Starting from the Main Activity, the User can tap the "All Libraries" button and the EasyLib app will ask the server (and later to the DB) to get the list of all the libraries. When the return get back, a new activity is started where all the libraries is displayed by means of RecyclerView

The user can next select the Library that he wants and the information about it are asked to the server and next displayed in the Book Activity. In this activity also the "library contents" are displayed (news, events and books), so the user can select an event and automatically the EventActivity is opened and shown.

The app will check if there are available seats and if the user has already joined the event (in the diagram is called "checkEventStatus()"). If there are free seats and the event is not already joined the "Reserve Seat" button is shown and the User can reserve a seat.

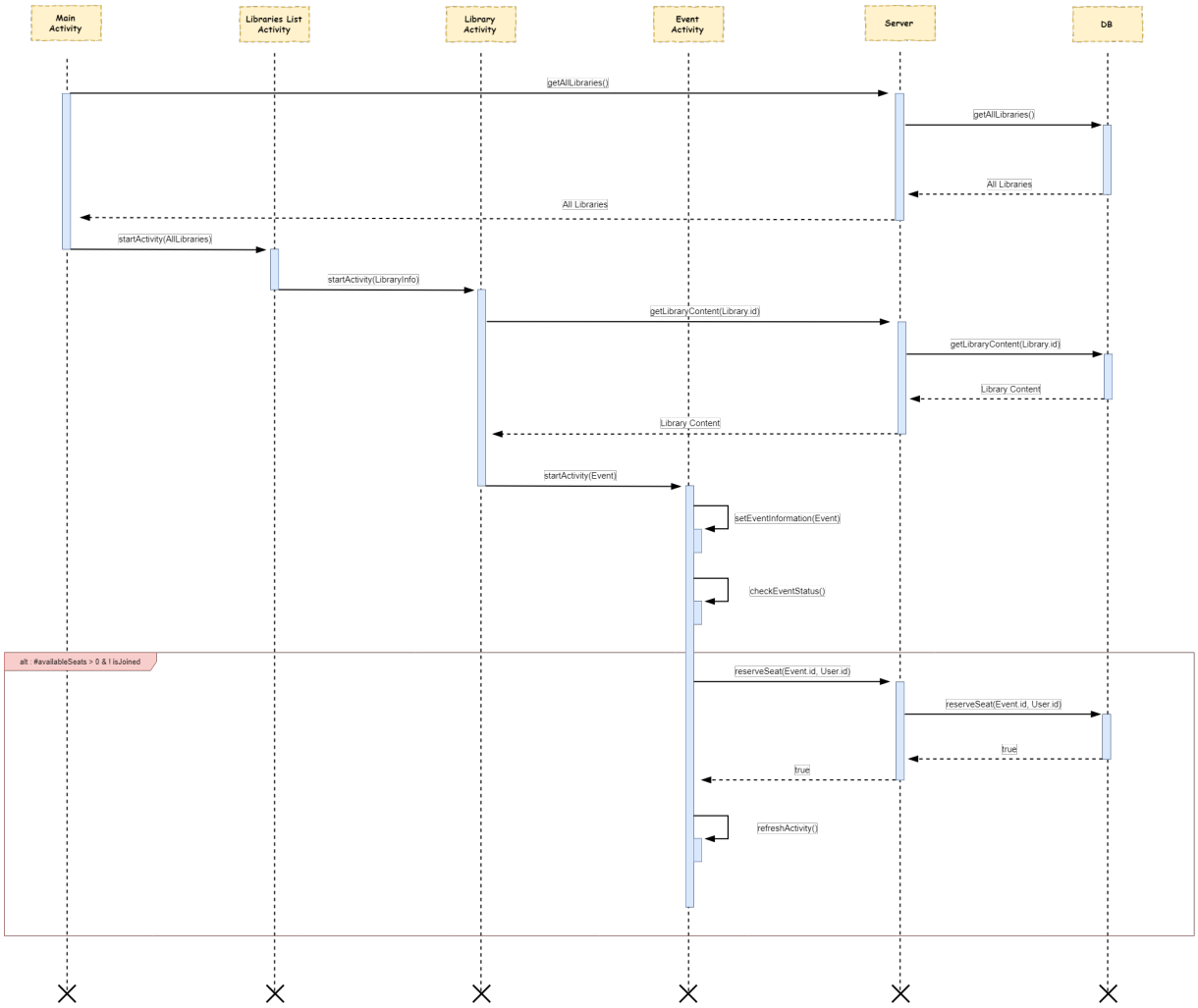

Figure 18: Event Reserve Seat - Runtime View

5.2 EasyLib - Librarian

5.2.1 QR-Code Scan + Book Returned

This Runtime View shows the different steps needed to find book information through QR-code scan and communicate to the server (and then the DB) that the book is returned.

We have to make an assumption : in the following diagram we reported only the most meaningful calls needed to achieve the goal.

Starting from the "Library Activity" the librarian has to tap the Floating Button with the QR-code icon that allows the app to open the "QR Scan Activity". As in the previous runtime, the qr-code is analyzed and the identifier returned is used to get from the server (and so the DB) the information of that specific book. The "Book Activity" is then opened and the data are set in the layout.

Then, the app has to retrieve all the reservations and pass the ArrayList to an Adapter that generates the single items of the recyclerView placed under the book layout.

At the end the librarian as to find the reservation that has the user id and tap the "Returned" button.

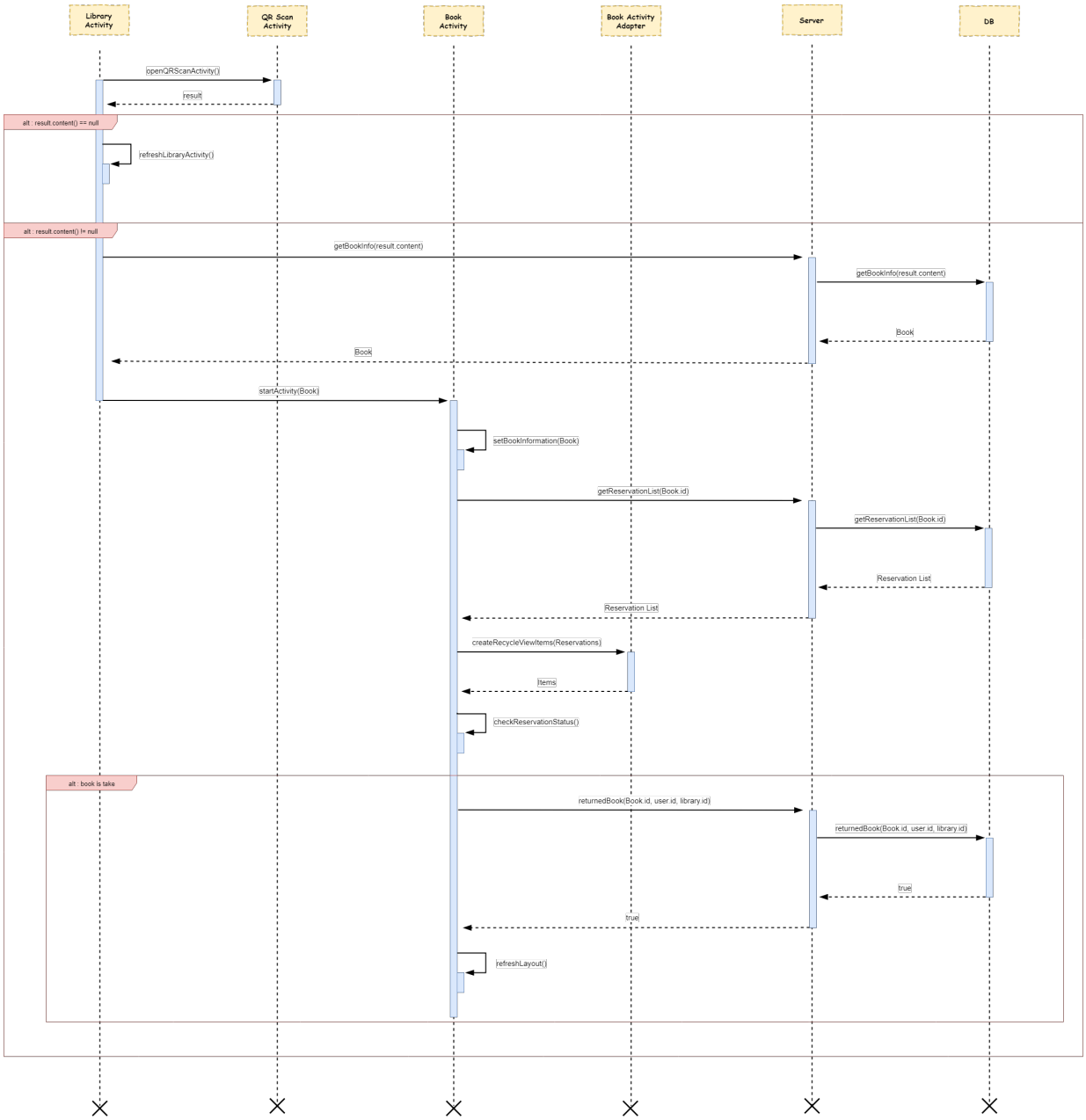

Figure 19: QR-Code Scan + Book Returned - Runtime View

6 Implementation, Integration & Test Plan

In this section we will show the plan we have projected for implementing EasyLib. We will describe the strategy for the processes chosen for approaching the project and the structure of our team, how it is divided and the tasks that each part has to accomplish. After we will state how each team's part has to interact with the others and the span of time of these interactions. Finally, we will supply a list of possible risks, the probability of their presence, their impact and a possible strategy to deal with them.

6.1 Strategy adopted

To give a quality assurance of the project and to be sure that the final product respected the stated goals, we have followed an Agile planning process. It consists in a first initiating phase in which we have stated an overall plan application that we have implemented. We have defined the EasyLib's requirements, discussed and stated the flow of each interaction and the specific technology to use. After this part, we followed the well know agility's cycle, composed by the subsequent phases: Analysis, design, implementation, testing & integration, mantainance and Planning again. Every cycle round has as input a specific process to accomplish, that is divided between the various team's parts. After the execution of all the tasks in a precise given timeframe, the work were checked by all the team's components. When the discussion on the work done had been terminated and an agreement had been reached, we started with another planning phase containing also the corrections agreed. This cycle has been repeated until the end of the project.

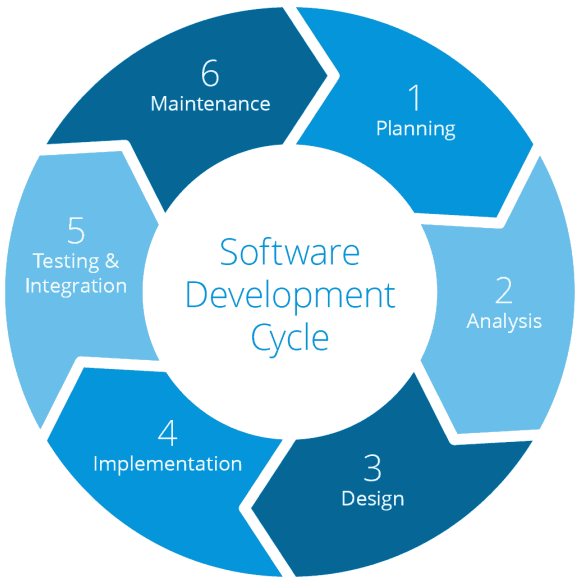

Figure 20: Agility cycle

6.2 Team Structure

Our team is composed by two Computer engineers with complementary competences and that have dealt with different tasks throughout the development process. The main tasks performed are that ones listed and commented right below.

- *Client Front-End development:* this role required the full understanding of the android components related to the visualization part, the high level knowledge of image-editing software, in order to produce the desidered layout and the management of the application server requests and the arrangement of the information contained in the responses to produce the whished view result.
- *Client Back-End and Application Server development:* this role required a deep knowledge of how to build asynchronous socket multi-client network infrastructure, a good understanding in multi-table MySQL databases management, including SQL statement and triggers implementation. Nonetheless, this team component has dealt with the Android services and their communication with the application server and the main thread of the app.

6.3 Implementation process

The work has been divided in three milestones. In these spans of time the team members were required to complete the tasks that have been established in the previous planning phase. Every week a meeting to discuss about the progresses were attended and new agreement finalized to proceed at the same speed were taken. If a team's part terminated his tasks before the milestone's day, he usually anticipated the work programmed for the subsequent one but remaining ready for eventually applying changes or corrections on the software already implemented. After each milestone, some days were employed for integrating front-end and back-end part to have always a working prototype.
At the end of the implementation process the team has performed a Unit testing and functional testing campaign, in order to assure that all the functionalities worked correctly. Also a User acceptance test have been carried on, involving a restrict group of people which gave back valuable feedbacks on their user experience that drove the developing process.

6.3.1 Test cases

As said before, in order to assure the correct behaviour of our application, we carried out tests of the main functionalities, trying to cover the wider surface of possible cases. We report below a list of the tests performed.

1.
   - *Goal:* Login using Email and Password
   - *Input:* The user taps on "Sign in" in Login Activity
   - *Outcome:* If the user inserts a valid email and password pair, after clicking on the "Sign in" button he is authenticated.
2.
   - *Goal:* Registration using Email and Password
   - *Input:* The user inserts surname, mail, password and tap on "Register" in Register Activity
   - *Outcome:* if the user inserts a valid email, password and username, after clicking on "Register" button a new account is register
3.
   - *Goal:* Consult book information
   - *Input:* The user taps on a book object in any part of the app in which a book icon is available
   - *Outcome:* The library page is opened and all its information such as name, address, telephone number and opening time are shown to the user
4.
   - *Goal:* Receive notification when the user pass from the first place of the queue to the list of the users that reserved the book
   - *Input:* The user that have borrowed the book returns it to the library and the librarian confirm the transaction through the EasyLib librarian app or delete its reservation
   - *Outcome:* The user receives a notification informing him that he can go to the library to take the book for which he was waiting for. The book become visible tapping the calendar icon on the navbar and is not more visible tapping the queue icon of the navbar. The notification is visible tapping the bell icon on the upper right part of the screen
5.
   - *Goal:* Search a book
   - *Input:* The user inserts title and/or author and/or category and/or library name and click on search icon
   - *Outcome:* If the book exists in one of the libraries present in the system, the books matching with the research are shown
6.
   - *Goal:* Reserve a copy of a book
   - *Input:* The user taps on on "reserve" button in the book page
   - *Outcome:* When the user taps on reserve, a new record in the database with the reservation's information is created the user can consult it tapping on the calendar icon on the navbar
7.
   - *Goal:* Get in line for a book
   - *Input:* The user taps on "add queue" button in a book page

- *Outcome:* When the user taps on "add queue", a new record in the database with the queuing information is created and the user can consult it tapping on the queue icon on the navbar

8.
- *Goal:* Register for an event
- *Input:* The user taps on "reserve a seat" button in the event page
- *Outcome:* When the user taps on "reserve a seat", a new record in the database with the seat reservation information is created and the user can consult it tapping on the profile icon on the navbar

9.
- *Goal:* Rate a read book
- *Input:* The user taps on a book in the list of the read books available in the profile and set a rating between 0 and 10
- *Outcome:* The rate is recorded in the Database, a new average rating is computed using also the new tuple and it is show to the user on the book image

10.
- *Goal:* Edit profile information
- *Input:* The user inserts new name and/or surname and/or password in the edit profile page and submit the change with a tap
- *Outcome:* The user's profile information is updated in the Database and the user can consult the change from the profile page

11.
- *Goal:* Set a library as favourite
- *Input:* The user taps the button "set as favourite" at the bottom of the library's page
- *Outcome:* The library is set as favourite for the user in the database and he can see it in the profile page or tapping on the home icon in the navbar in which he can also consult some info or the library which results to be the first that he prefers