Politecnico di Milano

School of Industrial and Information Engineering

SOFTWARE ENGINEERING 2 PROJECT

Design Document (DD)

Customer Line-up

Version 1.0

Authors
Hamza Haddaoui
Giuseppe Piccirillo
Alessandro Restifo

Supervisor
Dr. Elisabetta di Nitto

January 9, 2021

Contents

List of Figures

1 Introduction

1.1 Purpose

This document aims to extend in further details what has already been specified in the RASD document, by giving more technical details about the CLup system. It's mainly intended for developers and testers who will find a functional description of the components of the System, their interactions, their interfaces and how they will be implemented. The document also provides an overview of the system architecture; moreover, chosen design patterns and interface mockups are presented. Finally, this document describes the implementation, integration and testing plan for the system.

1.2 Scope

The purpose of this product is to help preventing crowds into stores and supermarkets, by planning each customer's visit. Before the coronavirus pandemic outbreak, crowds were not a problem inside shops: customers could go to buy goods at any time, without having to worry. COVID-19 has undeniably changed the way most people live their day-to-day lives. Controlling distancing between people and ensuring observance of the hygienic guidelines are both essential parts of managing the impact of the disease.
CLup is born for this purpose: helping managers to regulate the admissions to stores, preventing people to queue up outside and therefore lowering the risk of infection. When a customer has the necessity to buy something, all she/he has to do is to select the preferred store from the application, and then get a ticket. This ticket is nothing more than the digital counterpart of the situation where people line up for a service, but with more advantages. Furthermore, the product will have an additional function: we want customer to have the opportunity to book a visit in another day or time, without having to queue up. Finally, CLup can also build up statistics, collecting some data for a better service: for example, the best forecast for new customers can be predicted by checking the average permanence of a customer.

1.3 Definitions, acronyms, abbreviations

1.3.1 Definitions

- User: a citizen registered to CLup service
- Guest: a citizen using CLup services without being registered
- Ticket: reservation of a place on a virtual queue
- Visit: reservation of a place on scheduled day and time
- Timestamp: time and day at which a certain operation is performed
- Basic services: CLup functionality of booking a ticket, accessible both on the app and at the shop without registration
- Advanced services: CLup functionalities accessible only on app, upon registration
- Time to leave: estimated time at which the user should leave to get at shop at the right time

1.3.2 Acronyms

- GPS: Global Positioning System
- QR: Quick Response, refers to QR Code
- API: Application Programming Interface
- OS: Operative System
- AWS: Amazon Web Services
- JSON: JavaScript Object Notation
- REST: REpresentational State Transfer
- HTTP: HyperText Transfer Protocol
- HTTPS: HyperText Transfer Protocol Secure
- CDN: Content Delivery Network
- HTML: HyperText Markup Language
- CSS: Cascading Style Sheets
- JS: JavaScript
- NoSQL: No Structured Query Language

1.3.3 Abbreviations

- Gn: n-th goal
- Dn: n-th domain assumption
- Rn: n-th requirement

1.4 Revision history

- Version 1.0: Initial release of the DD

1.5 Reference documents

- Document: "R&DD Assignment A.Y. 2020-2021.pdf", Requirement Engineering and Design Project: goal, schedule, and rules
- ISO/IEC/IEEE 29148 dated Dec 2011, Systems and software engineering — Life cycle processes — Requirements engineering

1.6 Document structure

1. **Introduction**
   The first section identifies the product and the scope of this document. Furthermore, it defines the terms used in this Specification Document and it describes the content and structure of the DD.

2. **Architectural Design**
   This section shows the main components of the system and their relationships; it also focuses on design choices and architectural styles, patterns and paradigms.

3. **User Interface Design**
   According to what the RASD specifies, this section defines the UX through some flow diagrams.

4. **Requirements Traceability**.
   This section shows how the requirements in the RASD are satisfied by the design choices of the DD.

5. **Implementation, Integration and Test plan**
   This section is used to explain the order in which the implementation and integration of components will be done. Moreover, a high-level description of how the application will be tested is presented.

6. **Effort spent**
   The sixth section reports the details of the time spent by each member of the group on the various parts of the DD.

7. **References**
   The seventh section reports the notes found in the DD for a better explanation.

2 Architectural Design

2.1 Overview

The paradigm under which the CLup application will be developed is FaaS (Function as a Service) and PaaS (Platform as a Service), in contrast with the typical architecture in which systems are developed (4-tier architecture). This approach was chosen, instead of the previous mentioned classic type of architecture, because the functionalities required are easy enough to be offered through a serverless architecture, and there are many advantages in this kind of architecture.

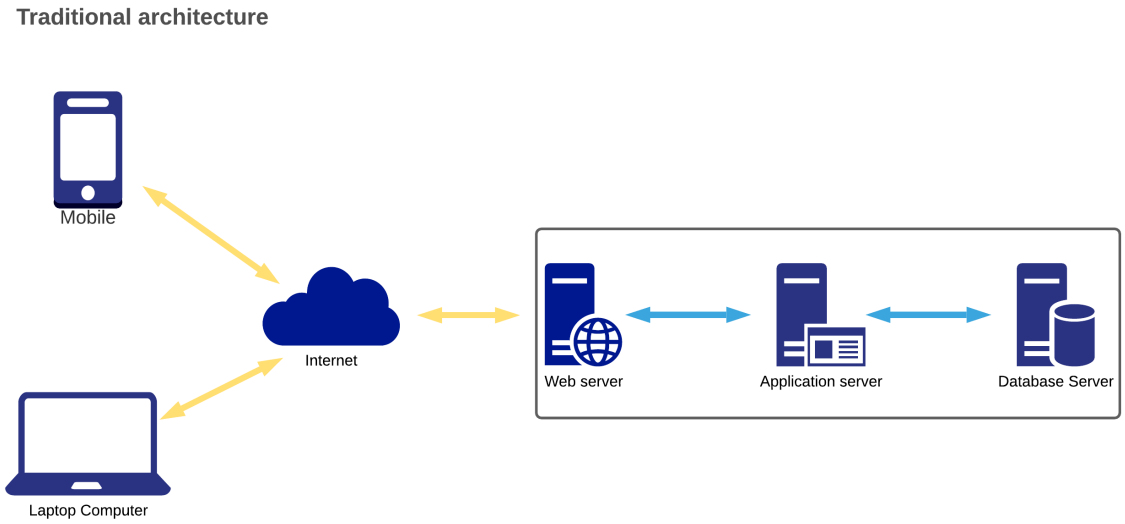

Figure 1: Representation of the traditional server architecture

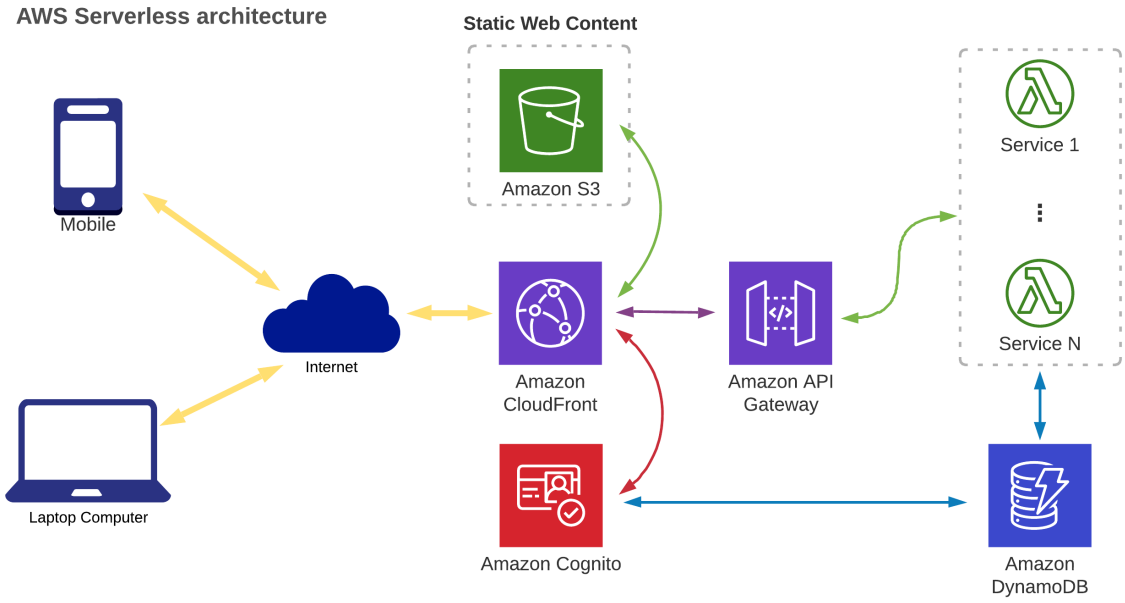

Figure 2: Representation of the serverless architecture

In this kind of system, the client interacts directly with the services offered by the application and the data stored in the database, without the need of any intermediate tier. This architecture allows a flexible scaling: the server, in fact, allocates as many resources as needed. Moreover, high availability is granted and server maintenance is not required. The provider chosen for the deployment of the application is AWS[1] . This provider offers a few services that are accessible over HTTP protocol, using REST architectural style and JSON messages protocol. The access point of the whole serverless architecture is the Amazon CloudFront[2] , which is a fast and reliable CDN that offers high speed transfer, field level encryption for sensible data and HTTPS support. Static content of the web application, including HTML, CSS and JS files will be stored into the Amazon Simple Storage (Amazon S3[3]). Amazon Cognito[4] is used for everything concerning the registration of the user, the authentication and the password reset. Microservice functionalities will be provided through the Amazon API Gateway[5] that, through Amazon CloudFront, grants access to the Lambda services[6] in which CLup functionalities will be embedded. Finally, data storing will be held by Amazon DynamoDB[7] which is a NoSQL database that offers a high integration with AWS platform.

2.2 Component View

The following diagrams will depict the internal structure of the CLup system, by means of components and interfaces that allow the communication between them. The architecture is made of three different parts: the CLup mobile application which will be used by citizens to create bookings, the CLup web application used by store managers and the CLup cloud that hosts all the previously introduced functionalities. An additional component, Google Directions API[8] , is presented for completeness, since it is used by the mobile application for the TTL functionality. For sake of simplicity, these three parts, will be presented at different levels of abstraction, starting from the highest level, shown below.

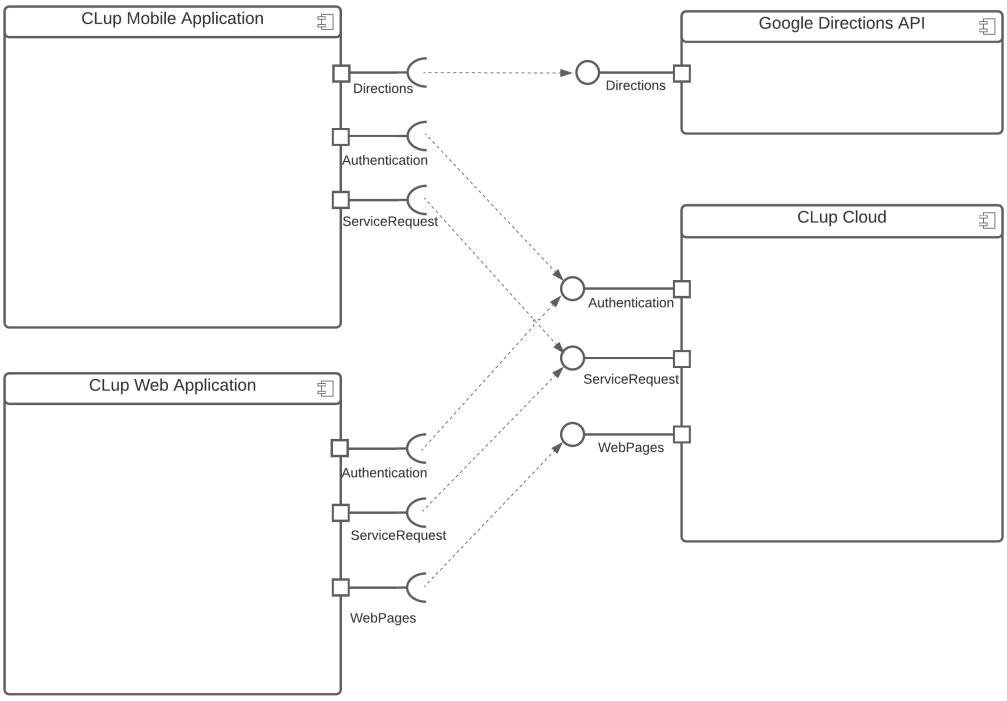

Figure 3: High level component diagram

2.2.1 Component diagram of the cloud

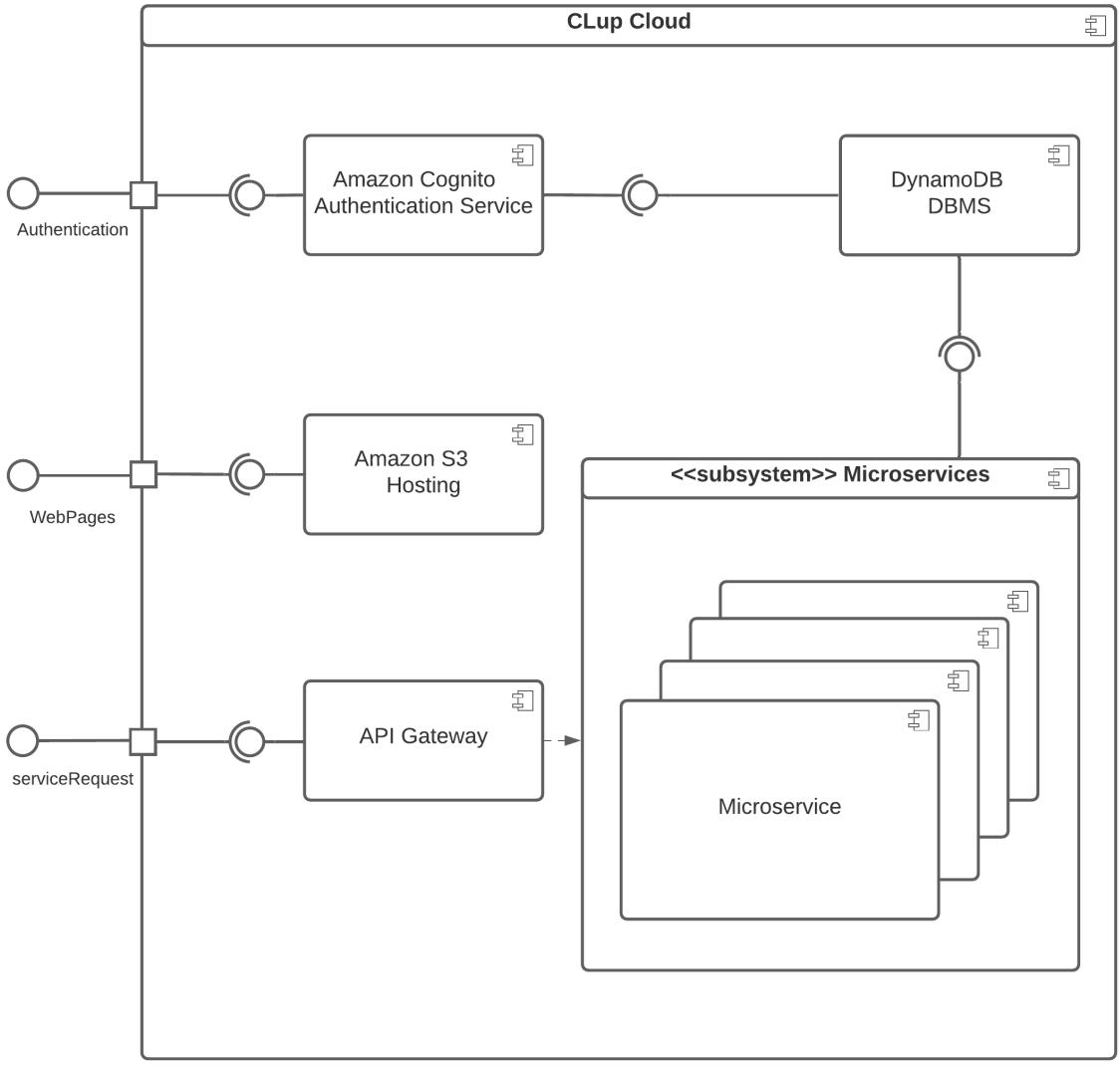

Figure 4: Component diagram of the cloud

The cloud architecture is based upon these components:

- Amazon Cognito, which manages the user authentication data, granting access to the system both to citizens and shop managers;
- Amazon S3, which is a data hosting space, that contains all the information needed to display web pages for the shop managers version of the application;
- The API Gateway handles all the REST APIs calls, accepting and processing them concurrently;
- DynamoDB DBMS;
- Microservices component, is a collection of microservices, that will be discussed in higher detail in the next page.

2.2.2 Component diagram of the microservices

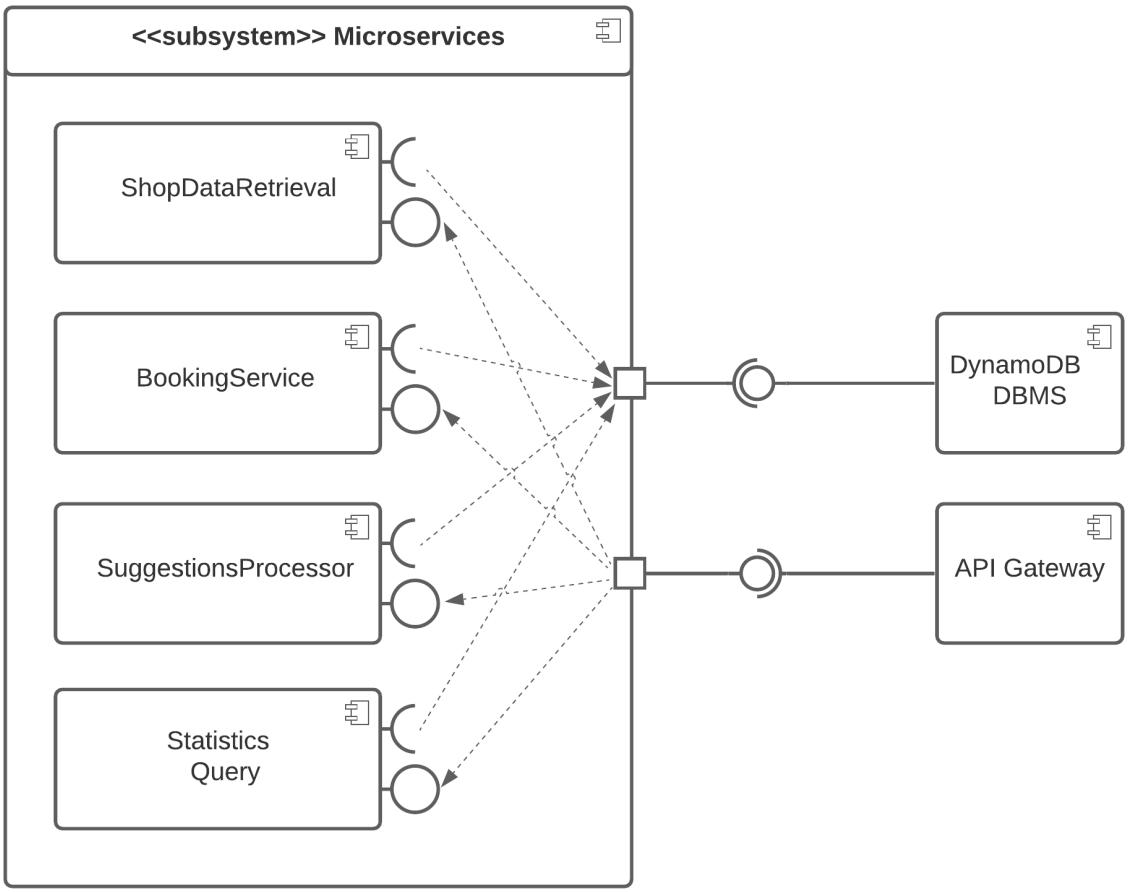

Figure 5: Component diagram of the cloud microservices

The main microservices that are mentioned in this component diagram are:

- The shop data retrieval microservice, which supports all the API calls that request information about the shops, their available slots, the number of people currently queueing up, similar shops nearby...;
- The booking microservice supports API calls for booking a ticket or a visit, in a specific store, including all the required data needed for the transaction;
- The suggestion processor is a microservice that can be called during the booking process by the mobile application, to check the presence of useful tips for the user;
- Statistics query microservice, is nothing else but a data mining API on the database, that can search for useful information to be presented to the shop managers showing insights and statistics about the customers.

2.2.3 Component diagram of the CLup mobile application

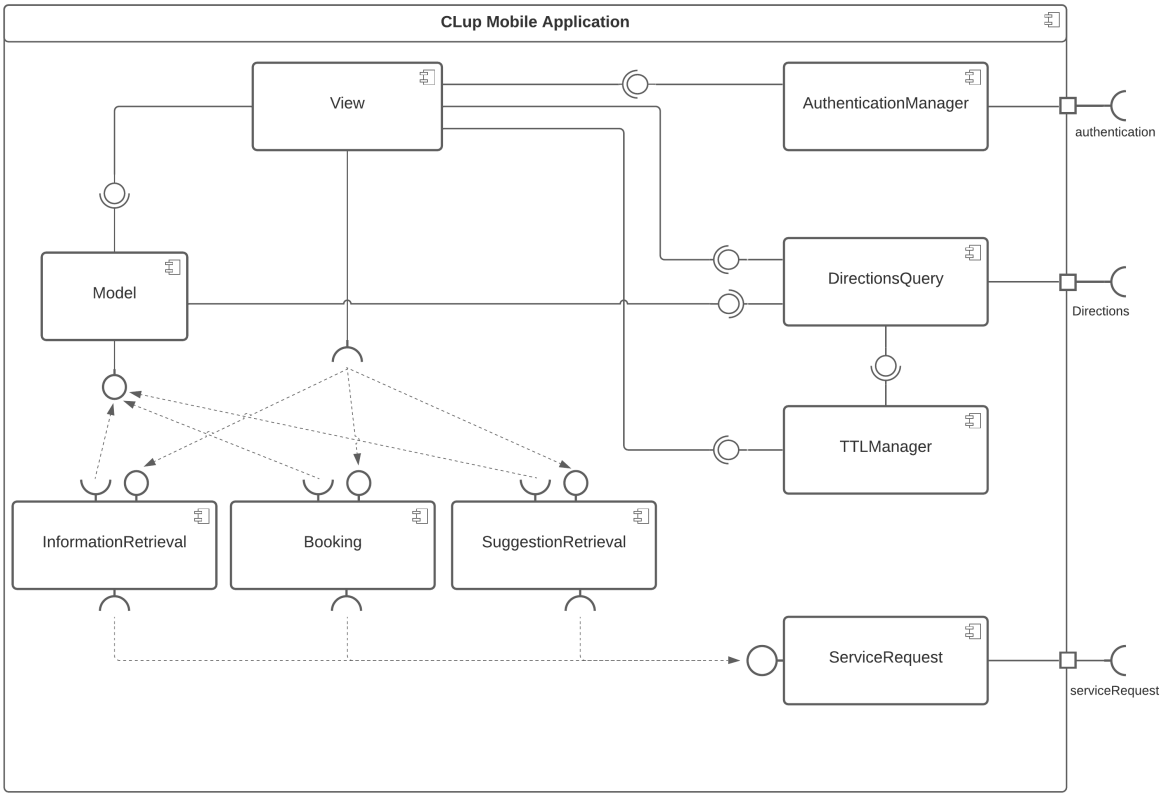

Figure 6: Component diagram of the CLup mobile application

The CLup mobile application is built using the MVC design pattern; this diagram puts a higher focus on the role of the controller, and its interaction with the other components of the architecture. The main controller components are:

- The AuthenticationManager module, whose role is to create a link with the cloud authentication service, providing this service to the user;
- The informationRetrieval module, which is liable of searching information about the desired shops, querying the cloud module;
- The booking module, which includes all the logic that allows the user to book a visit or a ticket;
- The suggestionRetrieval module, which is in charge of retrieving suggestions relative to the user, depending on the actions performed;
- The ServiceRequest module, which handles all the requests from the controller modules, and establishes the connection with the Amazon CloudFront through the Internet;
- The TTLManager module, which whenever a booking is pending, wakes up from time to time to check when it is time to leave for the shop;
- The DirectionsManager module, which handles the geo-localization process, computing the time needed to travel to the shop, making requests to the Google Directions API.

2.2.4 Component diagram of the CLup web application

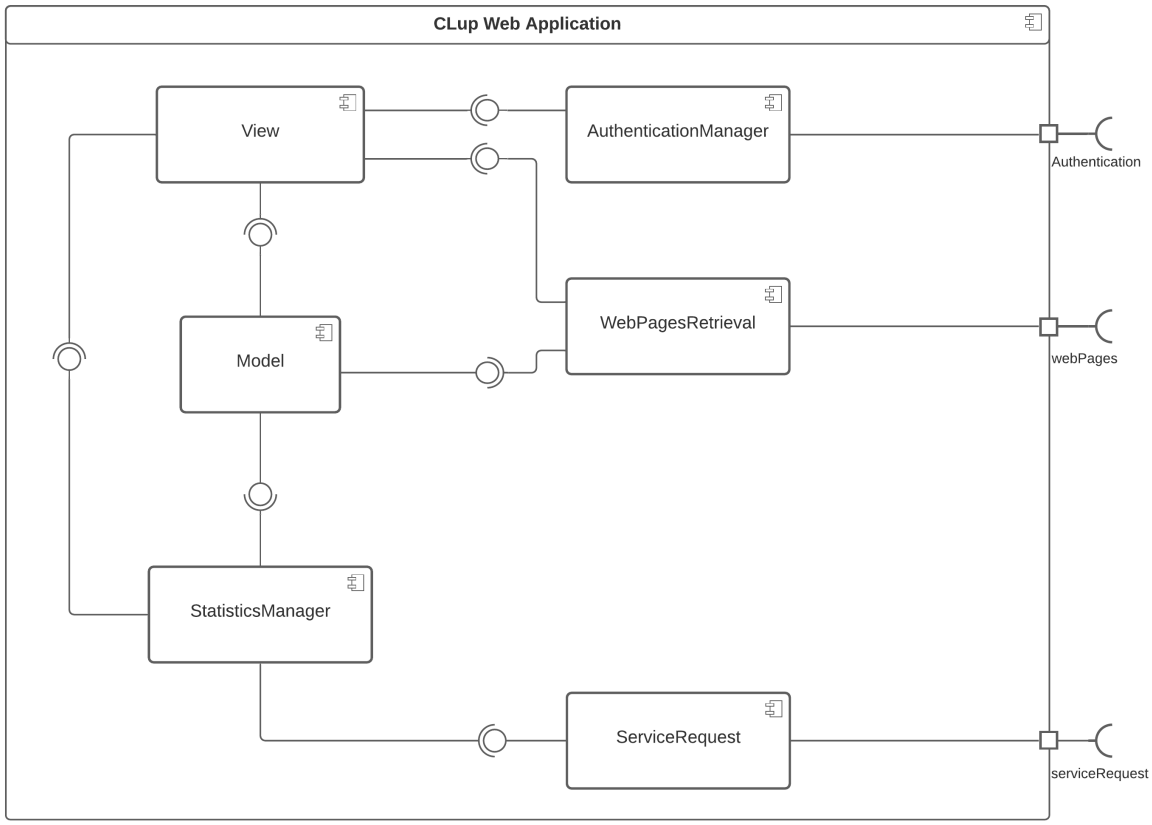

Figure 7: Component diagram of the CLup web application

The CLup web application is built using the MVC design pattern, like the mobile application. The main controller components are:

- The AuthenticationManager module, whose role is to create a link between the cloud authentication service, providing this service to the user;
- The WebPagesRetrieval module, which is liable of obtaining the web pages from the cloud service;
- The statisticsManager module, which is responsible of showing insighs and building graphs as well as visualize data provided by the cloud;
- The ServiceRequest module, which handles all the requests from the controller modules, and establishes the connection with Amazon CloudFront through the Internet.

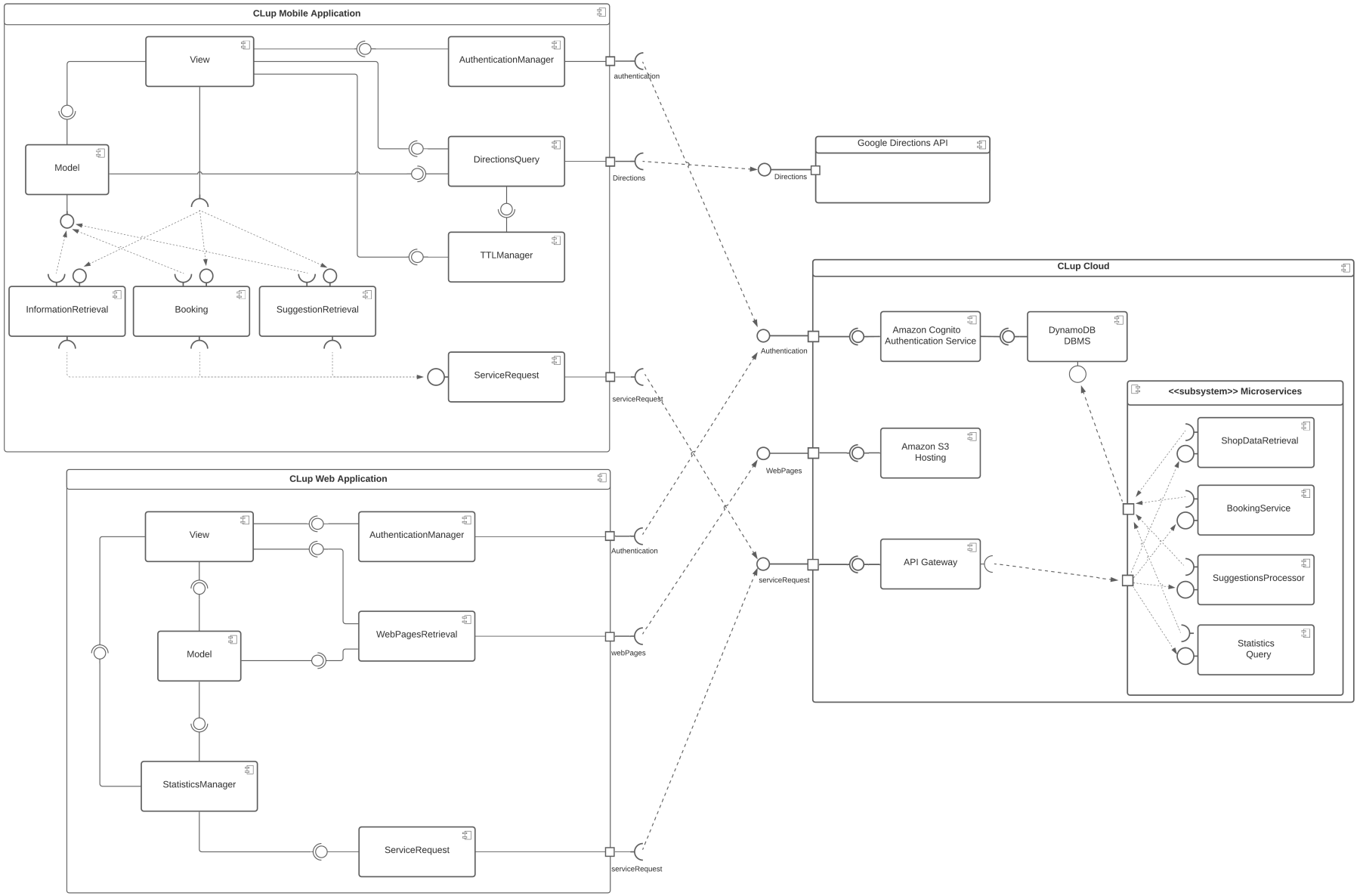

Figure 8: Full component diagram of the CLup system

2.3 Deployment View

The deployment view shows how the system will be composed when it will be deployed and therefore available to the user. As shown, there are three types of client devices that will use the application: mobile phones (of the users), tablets at disposal in stores (available for the user who do not have a mobile phone) and the computer with which the shop managers will access to the web app. On the left side of the graph, we can see the internal architecture of the cloud, which is composed of four nodes: the micro services node, that contains all the routines callable from remote devices, the authentication service, the DBMS and the Web hosting node.

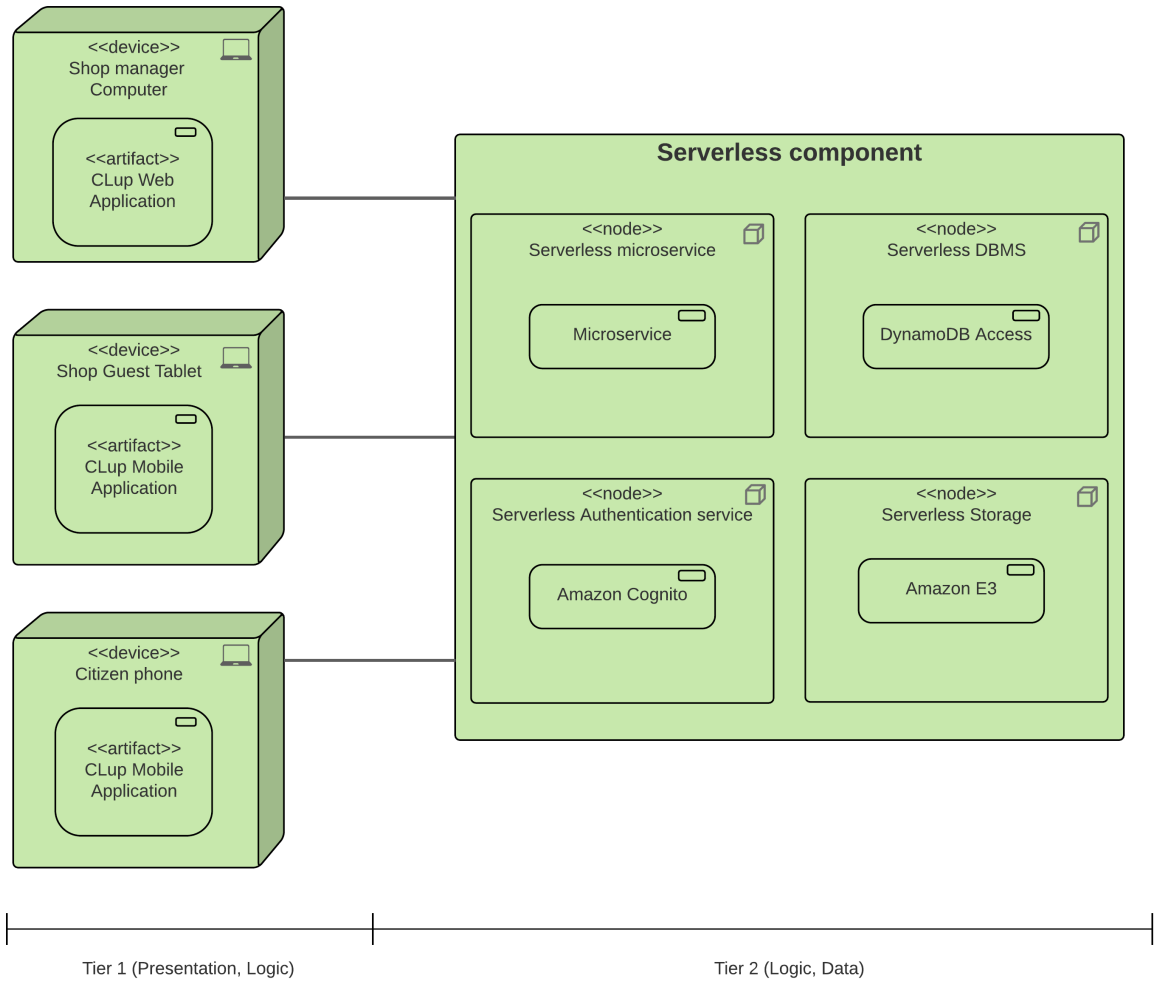

Figure 9: Deployment View

2.4 Runtime View

The following sequence diagrams represent some of the most important aspects of the system:

- **User login to CLup mobile application**: shows the interaction of the user with the CLup system, when a login is required. The main modules involved in this interaction are the authentication system, composed of the module on the client, and the service on the cloud. Another main component of this diagram is the DBMS in which the credentials are stored: if the keys match a tuple on the table, then the access is granted, otherwise the login is rejected.

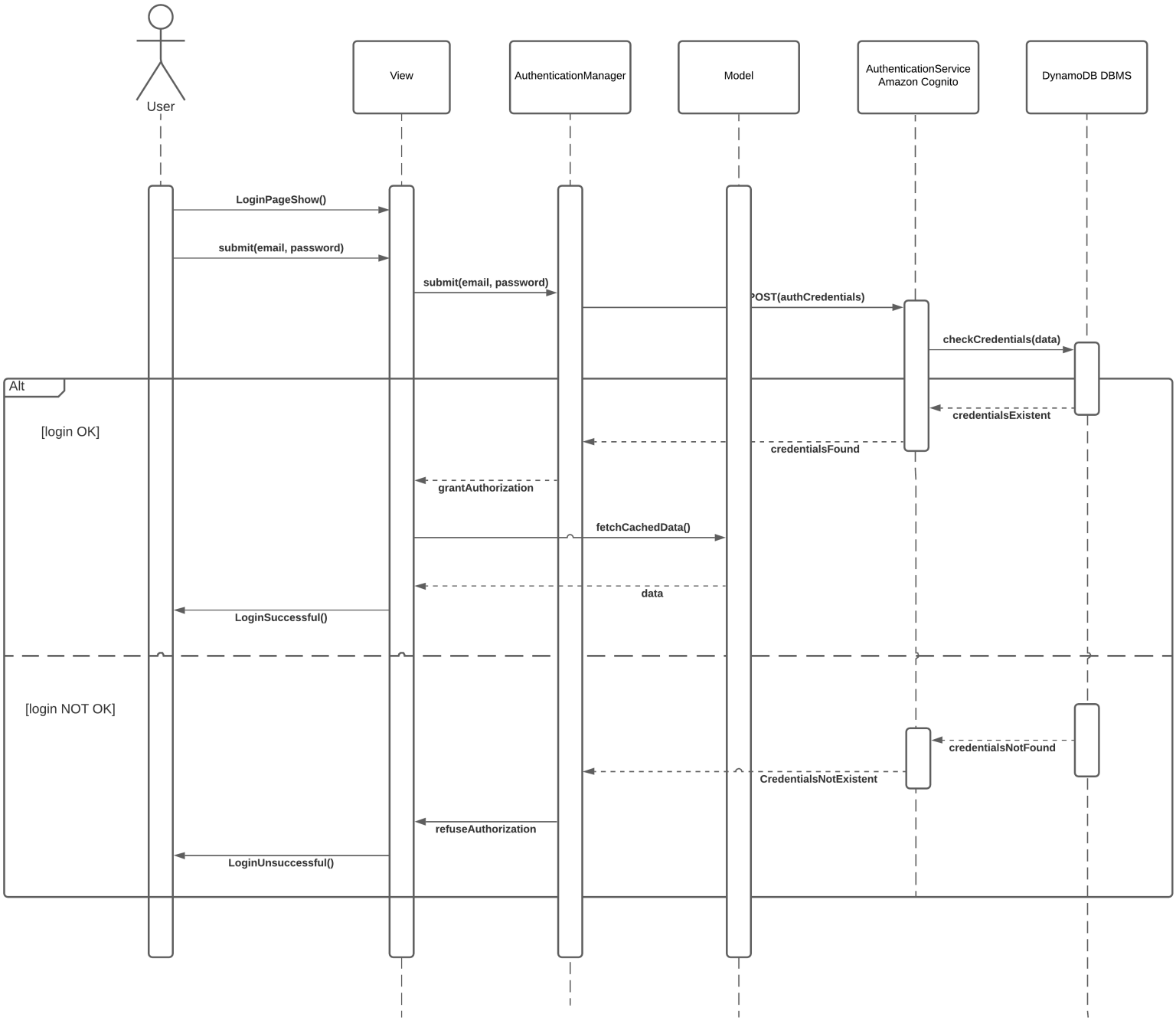

Figure 10: Sequence diagram of the user login

- **User booking (ticket and visits)**: depicts the situation in which the user wants to book a visit/ticket to a certain store: the first step is the login, which is shown in Fig.10. The second step is to search for a shop: the server will reply back with all the information regarding a certain store. The user will then fill-in the request by inserting all the required data (for the visit, date and time, as well as the grocery list if wanted) and will also receive suggestions based on her/his actions. Finally, the user will send the request to the server: if the request is successful then the QR code is sent to the user; otherwise, a fail message is sent to the user.

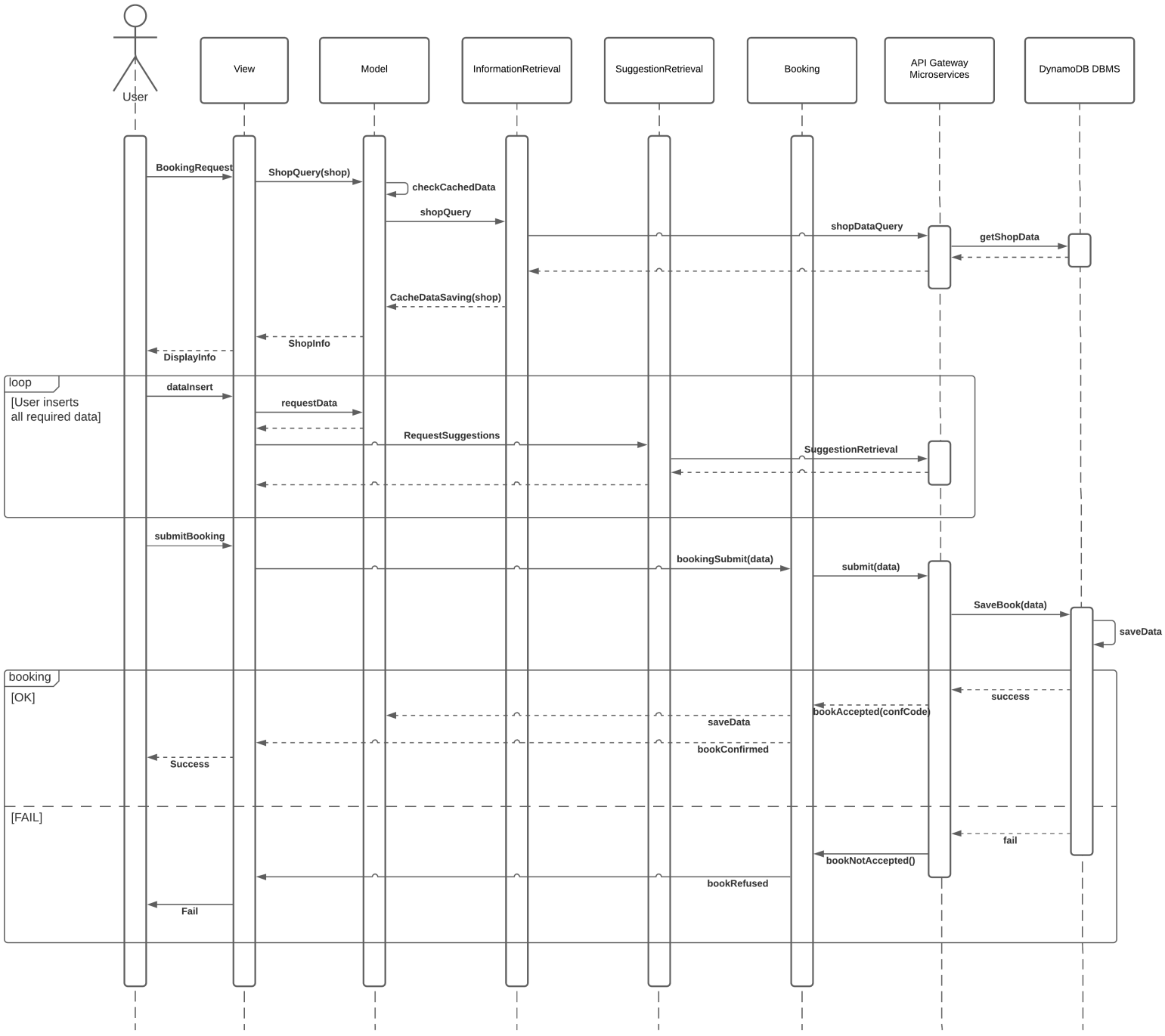

Figure 11: Sequence diagram of the booking process

- **Time to leave functionality**: shows the TTL functionality at work. At certain intervals, the application on the mobile phone wakes up and checks if there is an incoming booking. In case the booking is approaching, then a request to the Google Directions API is sent, by asking how long it takes to get from the user's position to the shop. In case the TTL condition is met, a notification is sent to the user, otherwise the wake up is rescheduled.

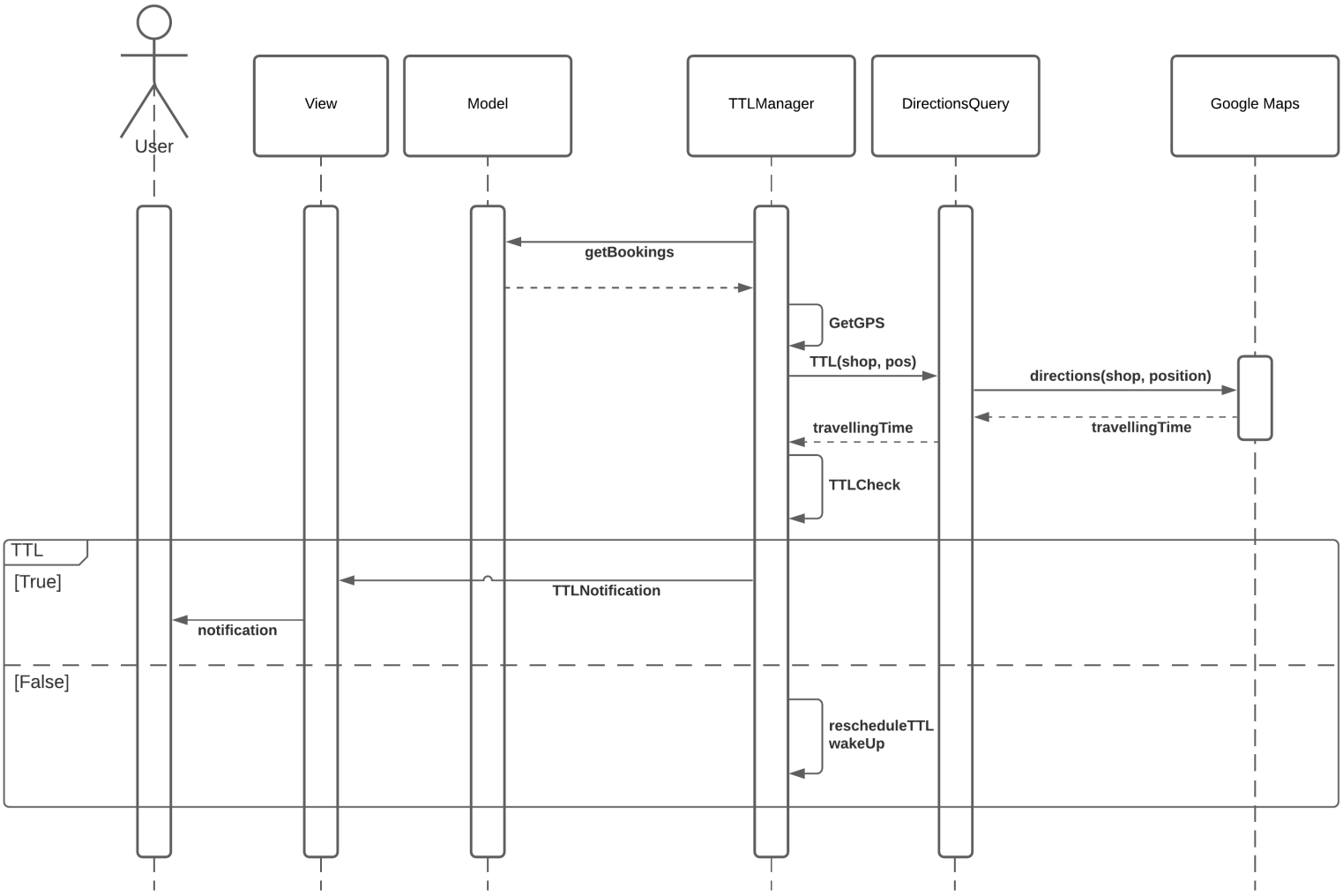

Figure 12: Sequence diagram of the TTL functionality

- **Shop manager data consulting**: The last sequence diagram shows the modules that are involved in the data consultation by the shop manager. When the shop manager logs in to the web interface, she/he can ask to consult the data on the platform by sending a request. All the available data are downloaded on local device to be mined in a faster way. In case additional data is required a new fetch on the cloud is made.

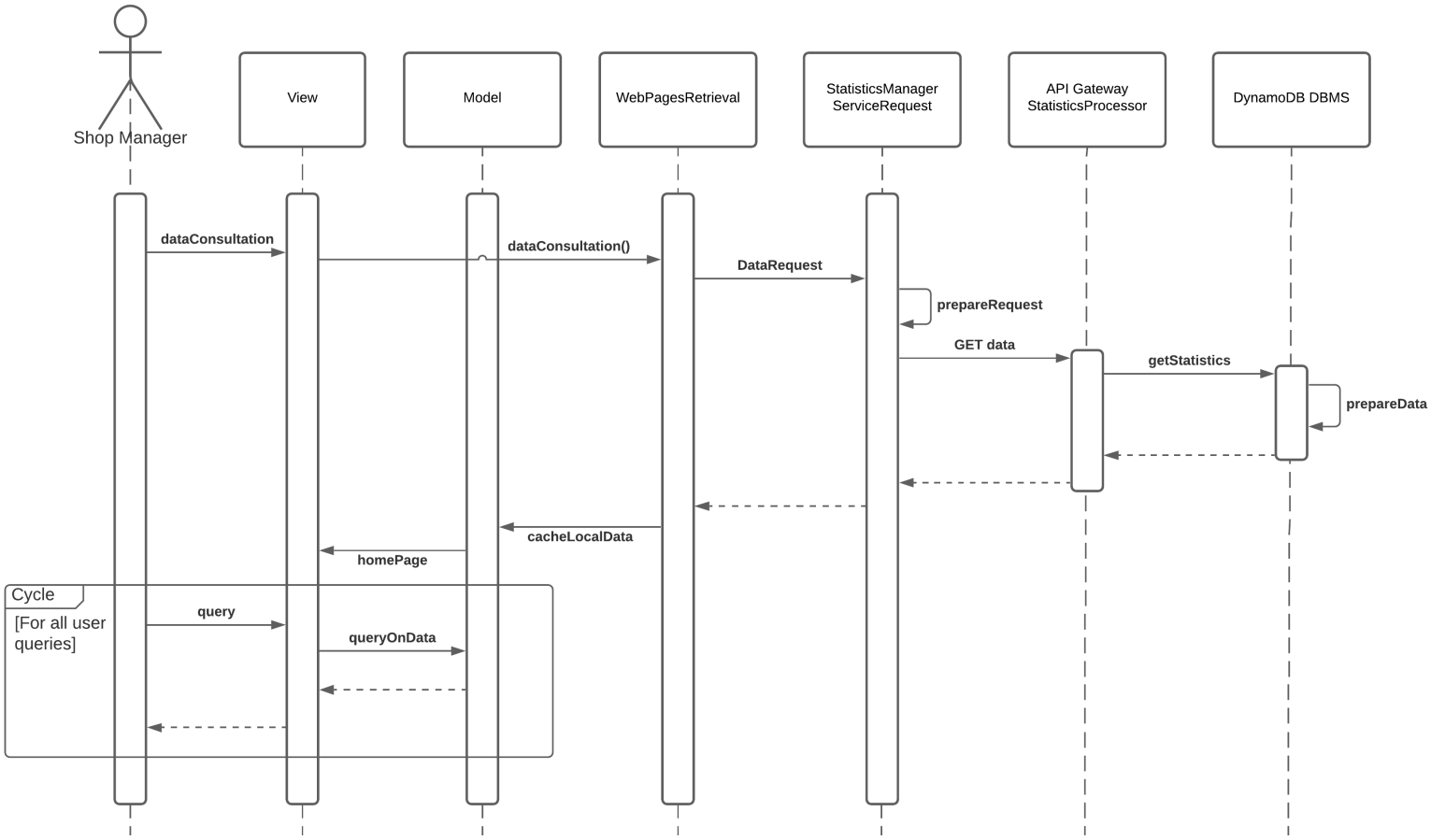

Figure 13: Sequence diagram of the shop manager usage

2.5 Component Interfaces

These diagrams show the components of the mobile application and the web application with the main accessible methods.

2.5.1 Component interfaces for the CLup mobile application

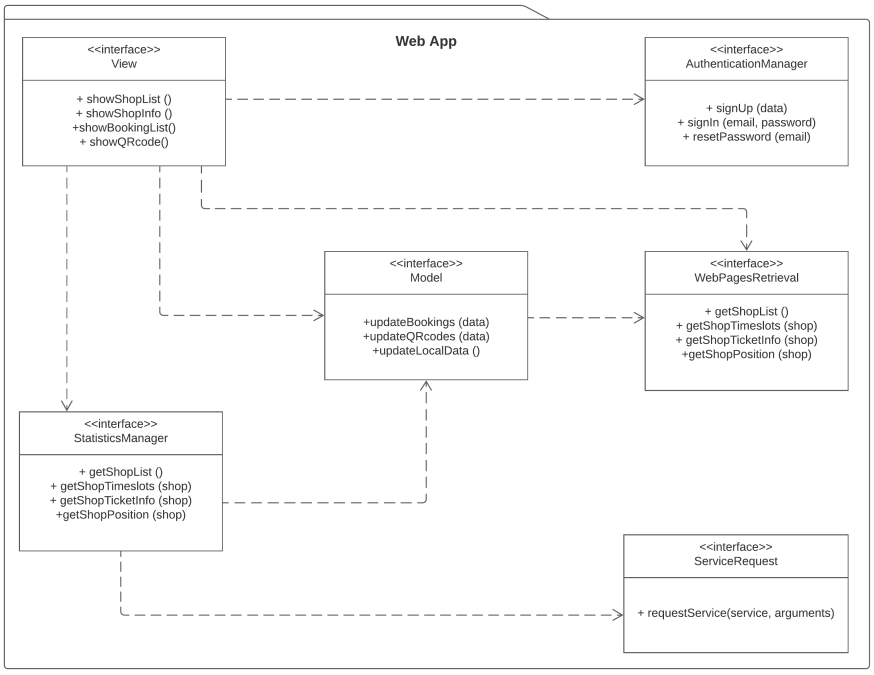

Figure 14: Sequence diagram of the shop manager usage

2.5.2 Component interfaces for the CLup mobile application

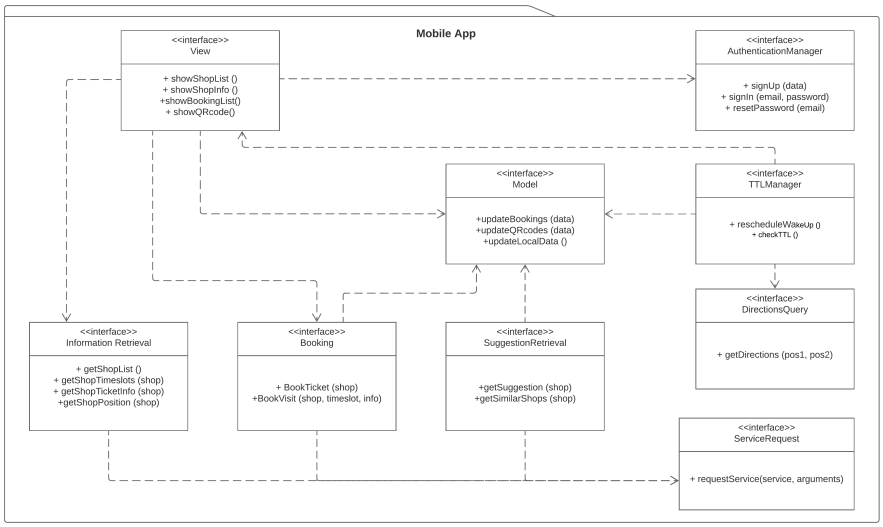

Figure 15: Component interface diagram of the web app

2.6 Selected architectural styles and patterns

Model View Controller
The MVC is a software design pattern very common in the development of user interfaces. It based on splitting the software in three main parts, connected through interfaces. The view, is in charge of showing visualizable content based on the model, which contains all the data needed for the application to run properly. Finally, the controller gets the input from the user and, depending on the type of input, modifies the model and/or the view.
Serverless computing
It is a cloud-computing execution model in which every part of the system is split into sub-functions that can run independently on the server. Serverless computing can simplify the process of deploying code into production, allowing the developers to focus on more important aspects of the project.

2.7 Other design decisions

Data will be stored in Amazon DynamoDB, which is a NoSql Database. This choice is motivated by the fact that this type of database is reliable and highly connected with the Amazon products that we are using in this solution. An entity-relationship diagram is presented to help the reader identify the key entities and the main relationships and attributes, though this will not represent the real DB structure, since the NoSql systems do not work using tables.

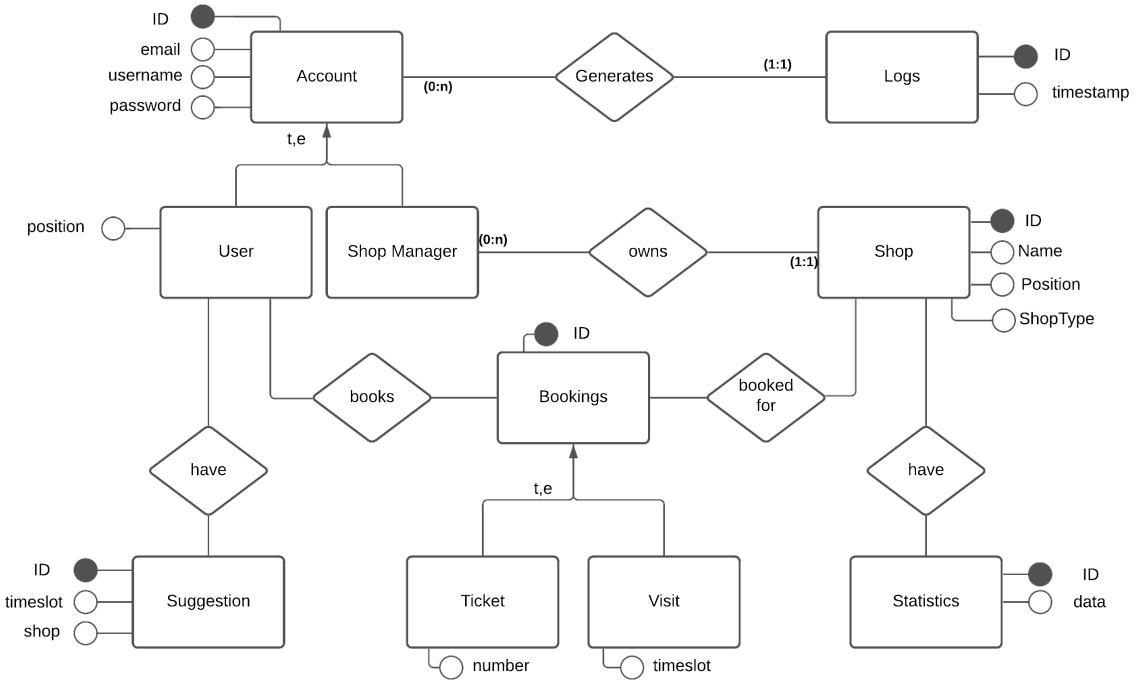

Figure 16: ER diagram of the system

The ER diagram is made of three main entities: the account (that can be of a user or of a shop manager), the bookings and the shop. The shop manager has an owner relationship with the shop, while "bookings" has two relationship, both with the "user" and with the "shop". Finally, additional tables are present: the "suggestions" relative to a certain user and the "statistics" for the shop.

3 User Interface Design

This section gives more details about the user interfaces of the mobile application through some general UX flowcharts about the main functions of the app. Furthermore, some interfaces of the web application for managers and the interface of the in-shop ticket hand out application for guests are shown.

3.1 Mobile App

3.1.1 Login/Signup UX flowchart

When the app is opened, the first interface displayed is **[e]** if the user isn't already logged-in, otherwise the home **[a]** is displayed. If the user does not have an account yet, she/he can signup clicking on "SIGNUP" in **[e]** and filling **[f]**.

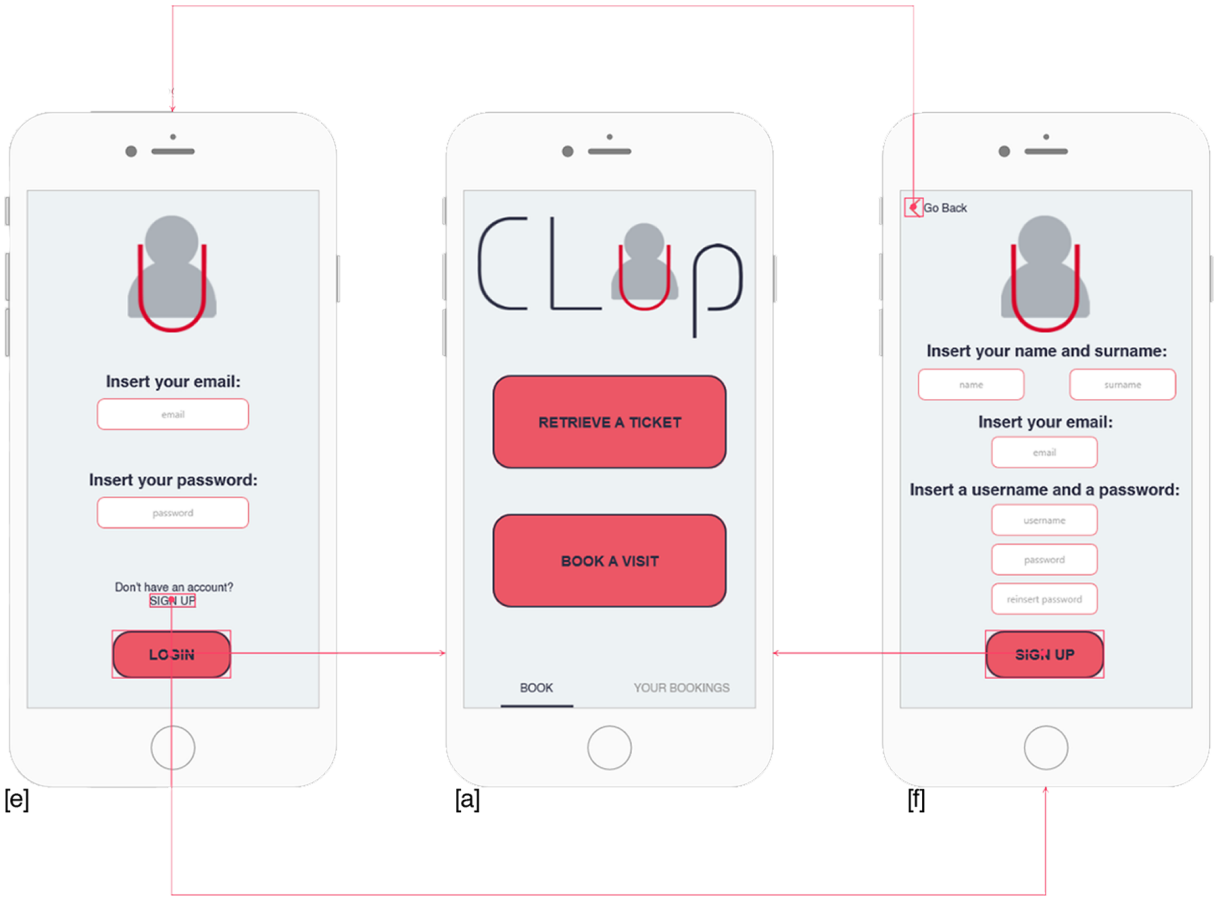

Figure 17: Login/Signup in mobile app UX flowchart

3.1.2 Home UX flowchart

As shown, from the home page **[a]** it is possible to retrieve a ticket and choose the store in **[b]**, or booking a visit and then choosing the store and a date in **[c]** (optionally it is possible to insert a list too).

Figure 18: Home in mobile app UX flowchart

Furthermore, from the home page it is possible to reach the bookings page **[d]**. Here it is possible to see the upcoming and past bookings and their details **[d.1]** by clicking on the QRcode icon. The user can delete a booking **[d.2]** clicking on the trash icon.

Figure 19: Options from "YOUR BOOKINGS" page

3.1.3 Taking Ticket UX flowchart

It is possible to take a ticket clicking "RETRIEVE A TICKET" in **[a]**, then choosing a store in **[b]** and finally taking the ticket in **[g]**.

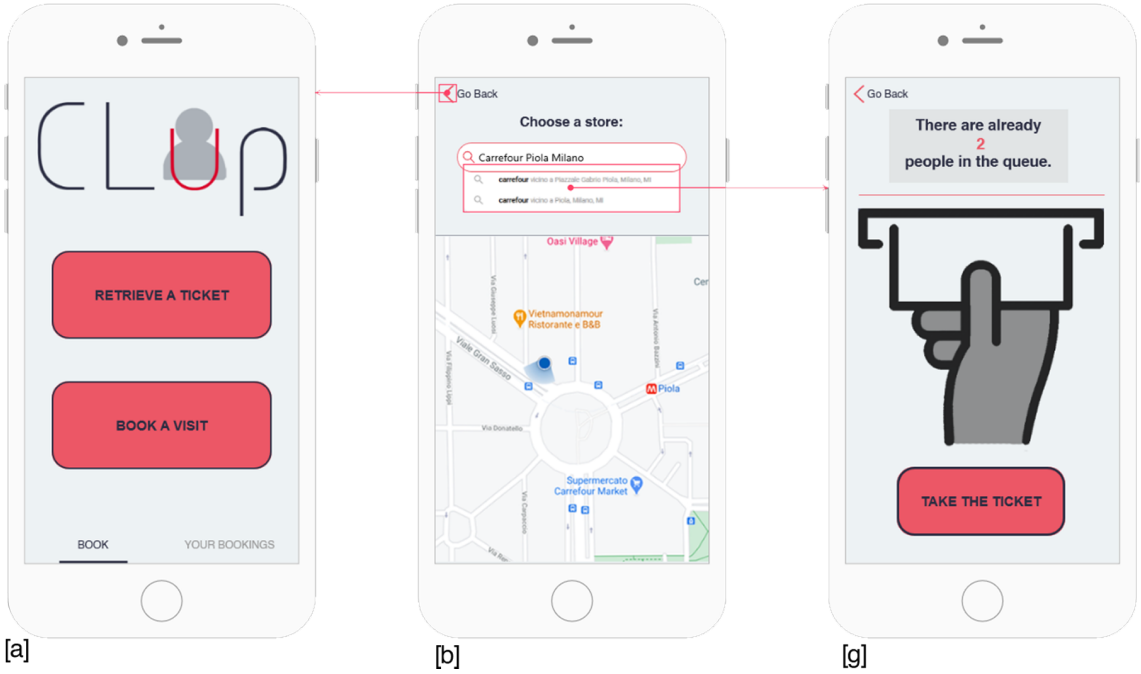

Figure 20: Taking ticket in mobile app UX flowchart

If the action can be performed and the ticket is correctly booked, a confirmation message is displayed **[g.1]**. The user can directly check the booking clicking on "My bookings" button or return to home page clicking "OK".
On the other hand, it is shown that there are no more available tickets **[g.2]**, so the app suggests to change the chosen store with a similar one of the same chain or another one but of the same category **[g.3]**.

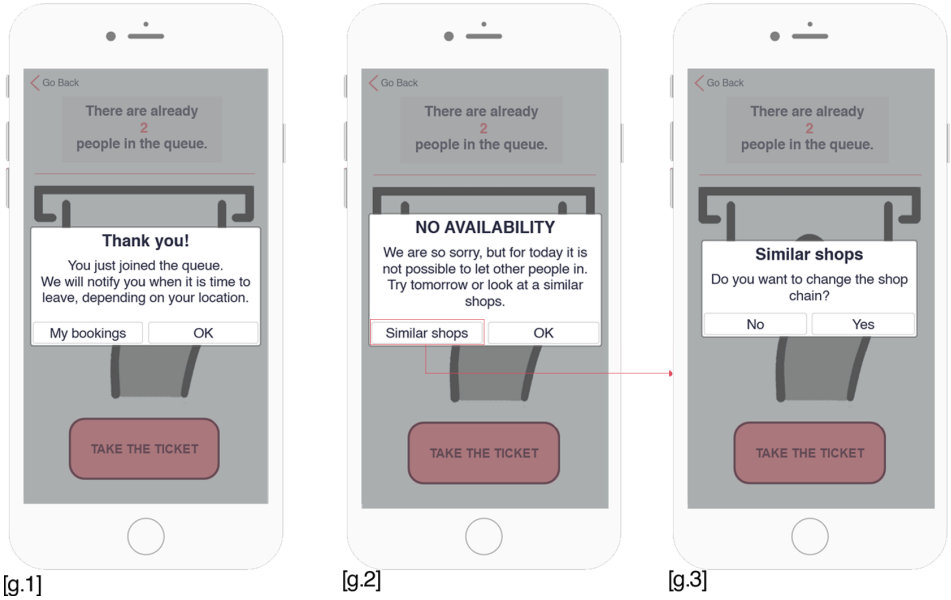

Figure 21: Taking ticket alerts

In picture**[h.1]** stores of the same chain are shown, while in **[h.2]** similar stores from others chains are shown.

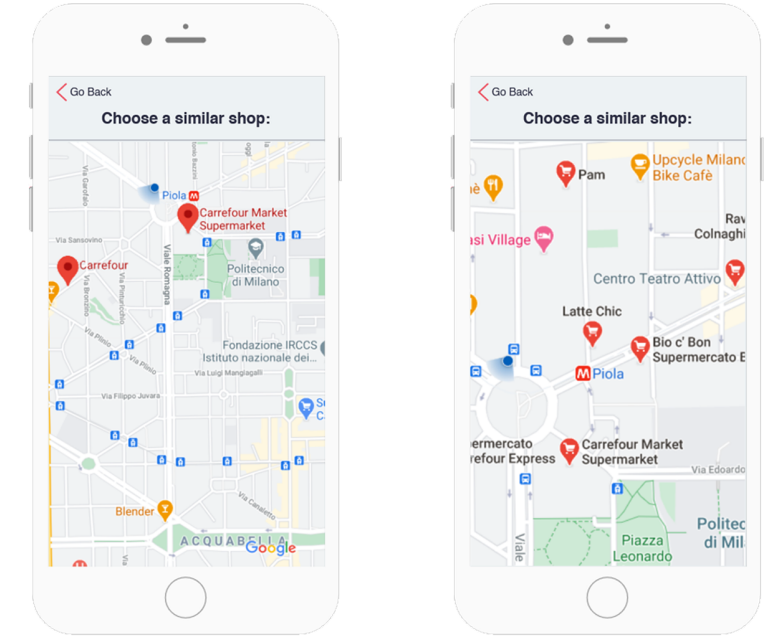

Figure 22: Change the chosen store

3.1.4 Booking Visit UX flowchart

It is possible to book a visit clicking "BOOK A VISIT" in **[a]**, choosing in **[c]** a store, a date and optionally a list (the user can insert products in **[i]**) and finally selecting a time-slot in **[l]**.

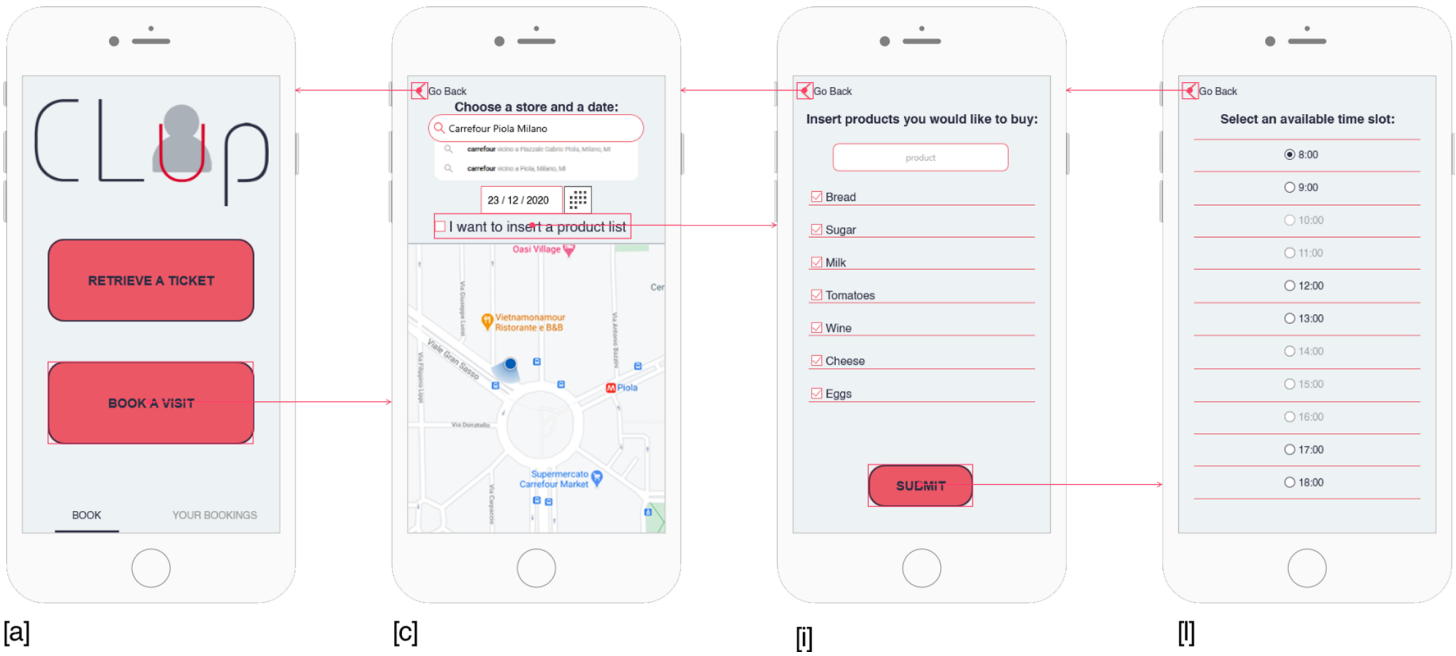

Figure 23: Booking a visit in mobile app UX flowchart

If there is enough data about the user, the app can suggest the expected duration of the visit **[c.1]**, however the user can choose to ignore this suggestion and set a personalized visit duration **[c.1.1]**.
If there are no available time slots in the chosen day, the app gives the possibility to change the store **[l.1]**, **[l.2]** in the same way for taking tickets **[h.1]**, **[h.2]**.

Figure 24: Booking visits alerts 1

Moreover, the app can suggest a specific time slot **[l.3]** in order to distribute bookings in an efficient way. The user can accept the suggestion **[l.3.1]** or ignore it **[l.3.2]**.

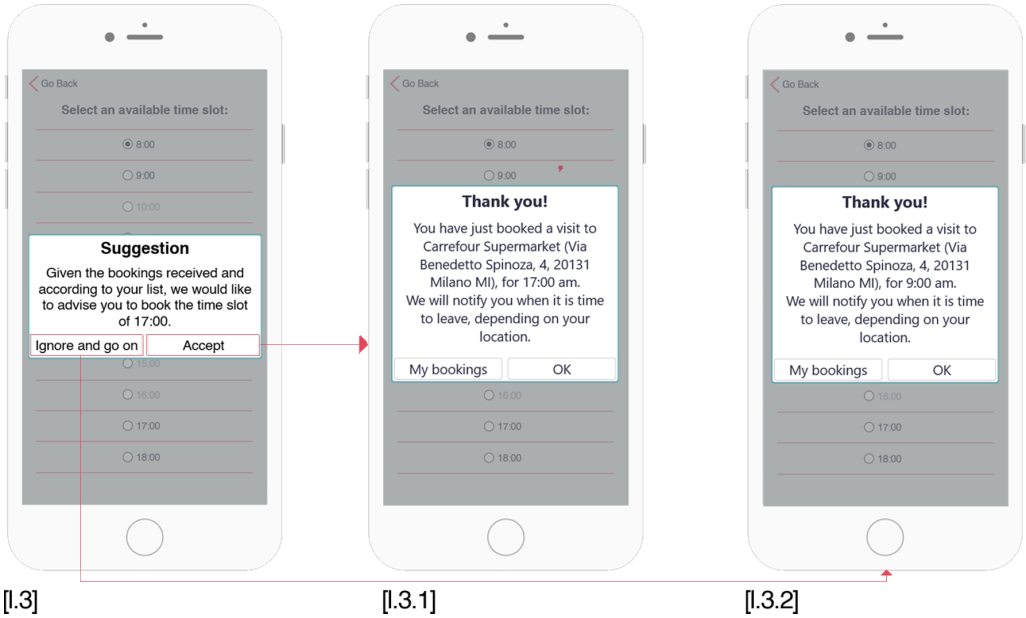

Figure 25: Booking visits alerts 2

3.2 Web App

In this section the main interfaces of the web application are shown.

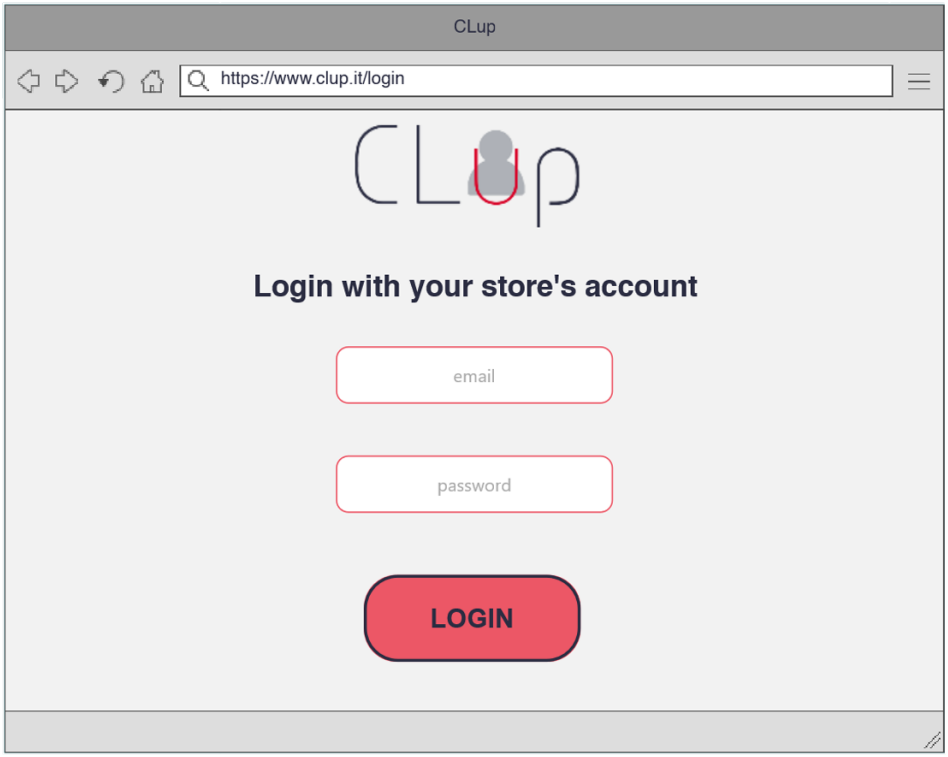

Figure 26: Login page of the web app

Figure 27: Home page of the web app

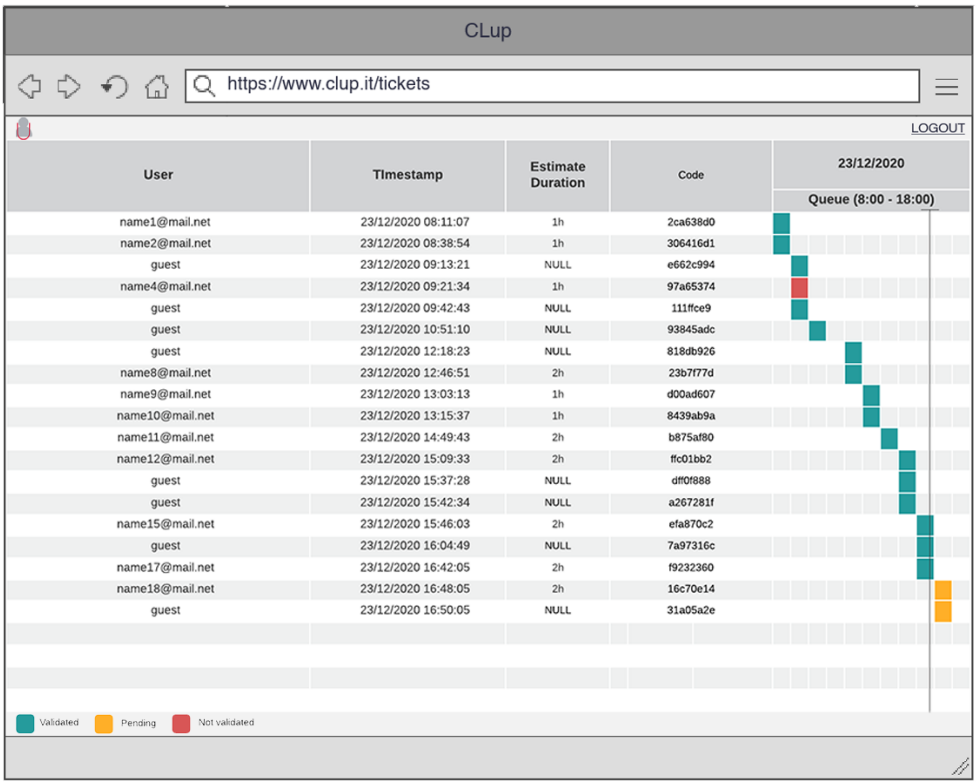

Figure 28: Tickets page of the web app

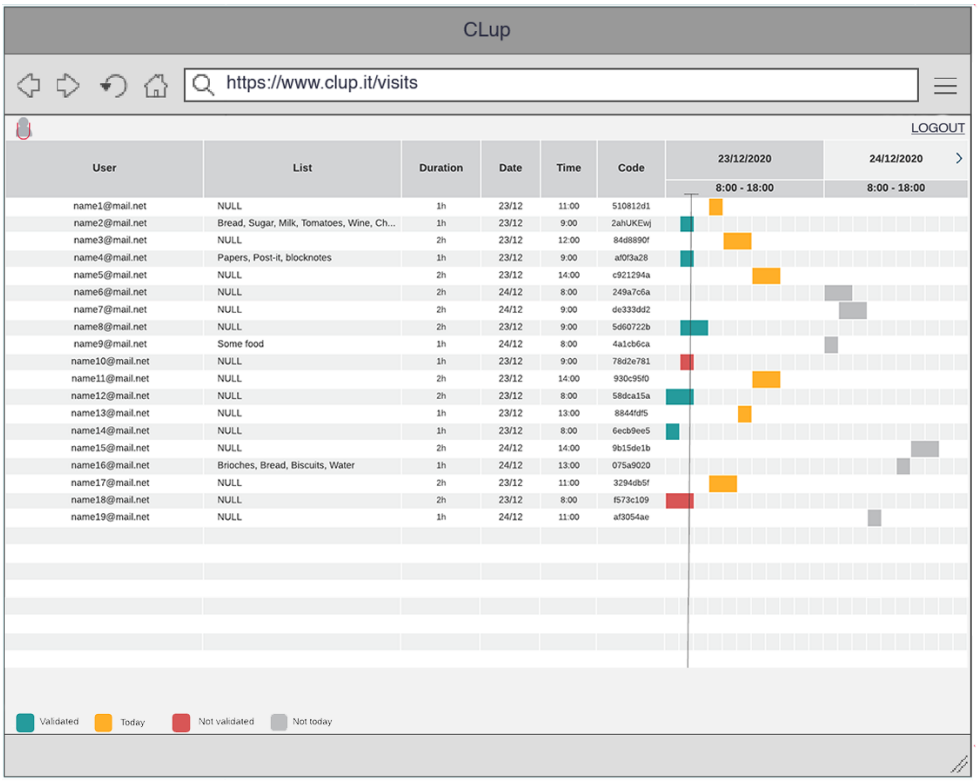

Figure 29: Visits page of the web app

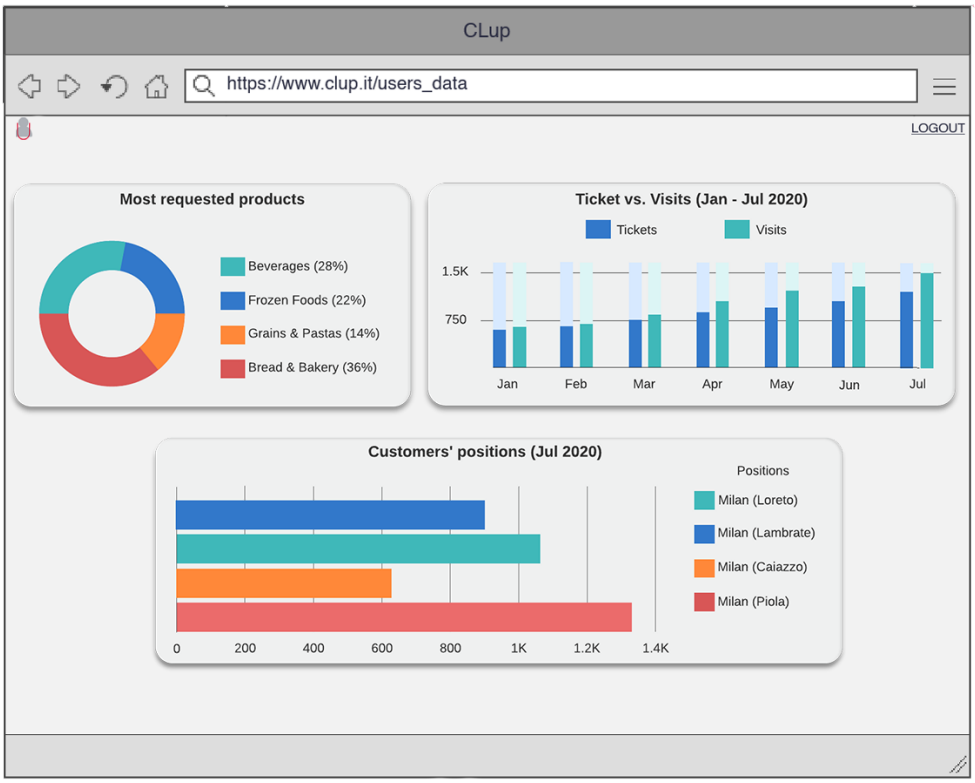

Figure 30: Users Data page of the web app

3.3 Guest

In this section the interface of the in-shop ticket hand out application is presented.

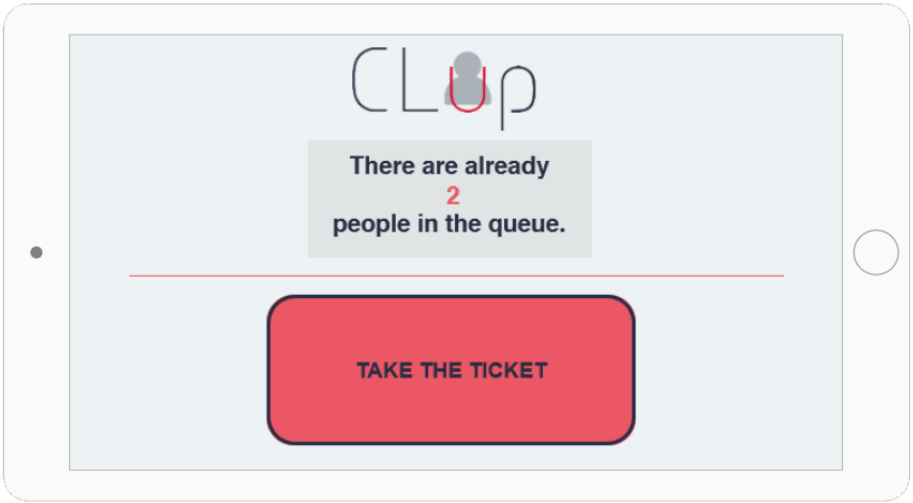

Figure 31: In-shop ticket hand out app interface

4 Requirements Traceability

[R1] A citizen not yet registered must be able to sign up and become a user

- WebPagesRetrieval (Web Client)
- Authentication Manager (App)
- Amazon S3 Hosting (Cloud)
- Amazon Cognito (Cloud)
- DynamoDB DBMS (Cloud)

[R2] The application must allow users to authenticate

- WebPagesRetrieval (Web Client)
- Authentication Manager (App)
- Amazon S3 Hosting (Cloud)
- Amazon Cognito (Cloud)
- DynamoDB DBMS (Cloud)

[R3] The application must allow the user to search for the desired store

- Information Retrieval (App)
- Service Request (App)
- API Gateway (Cloud)
- Suggestion Processor (Cloud)
- Shop Data Retrieval (Cloud)
- DynamoDB DBMS (Cloud)

[R4] The application must allow user to book after filling the required data

- Booking Service (App)
- Service Request (App)
- API Gateway (Cloud)
- Booking Service (Cloud)
- DynamoDB DBMS (Cloud)

[R5] The system must correctly save data regarding issued bookings

- Booking Service (Cloud)
- DynamoDB DBMS (Cloud)

[R6] The application must offer an interface to retrieve issued bookings

- Authentication Manager (Web Client)
- Web Pages Retrieval (Web Client)
- Statistics Manager (Web Client)
- Service Request (Web Client)
- Authentication Manager (App)
- Information Retrieval (App)
- Amazon S3 Hosting (Cloud)
- Amazon Cognito (Cloud)
- DynamoDB DBMS (Cloud)
- API Gateway (Cloud)
- Booking Service (Cloud)

[R7] The application must allow the user to select a date and a time for the visit

- Suggestion Retrieval (App)
- Service Request (App)
- API Gateway (Cloud)
- DynamoDB DBMS (Cloud)
- Statistics Query (Cloud)
- Suggestions Processor (Cloud)

[R8] When a booking is pending, the application must be able to wake up at a certain frequency to check when it is time to leave for the user

- Directions Query (App)
- TTL Manager (App)
- Google Directions API (Third party)

[R9] The application must send a notification when it is time to leave

- TTL Manager (App)

[R10] The application must print out the booking digest

- Booking Service (App)

[R11] The application must keep data about previous visits to shops

- DynamoDB DBMS (Cloud)

[R12] The application must popup suggestion during the booking process

- Suggestion Retrieval (App)
- Service Request (App)
- API Gateway (Cloud)
- Suggestions Processor (Cloud)
- Shop Data Retrieval (Cloud)
- DynamoDB DBMS (Cloud)

[R13] The generated QR code must contain all useful data for the transaction

- Information Retrieval (App)
- Service Request (App)
- API Gateway (Cloud)
- Booking Service (Cloud)
- DynamoDB DBMS (Cloud)

[R14] The system must be able to analyze data and show statistics in order to monitor the entrances

- Statistics Manager (Web Client)
- Service Request (Web Client)
- DynamoDB DBMS (Cloud)
- API Gateway (Cloud)
- Statistics Query (Cloud)
- Suggestion Processor (Cloud)

[R15] The system must allow the shop manager to see only data related to her/his store

- DynamoDB DBMS (Cloud)

[R16] The system must be able to interface with third party services

- Directions Query (App)
- TTL Manager (App)
- Google Directions API (Third Party)

[R17] The application must allow the user to book only if there is an available place

- Booking Service (App)
- Service Request (App)
- API Gateway (Cloud)
- Booking Service (Cloud)

5 Implementation, integration and testing

5.1 Description

In order to implement the system a bottom-up approach will be used. Bottom-up allows for early testing, which can begin as soon as the first module has been specified, and the different modules can be developed independently. Later, the components can be tested together.
CLup application is composed of three different high-level components:

- the Mobile App, the gateway for common users to access CLup services
- the Web Client Interface, the preferred way in which store managers can analyze data and statistics about the performance of their stores
- the Cloud computing platform, offered through a FAAS approach within AWS

These three major components can be broadly developed in parallel in order to allow a faster deployment of the application.
That being said, some lower-level components are still shared in the platform, making it necessary to determine the main features of the system from a lower-level point of view.
The main functions of the application, as stated in the RASD [Par. 2.2] are:

- Booking a ticket
- Booking a visit
- Monitoring the entrances and the statistics

5.2 Functions and components mapping

In order to undestand which core components have to be prioritized during the development stage, a mapping between the features and the components is required. Here the components are specified for each function.

5.2.1 Services specific for booking a ticket

On the application:

- Model and View of the mobile application
- Authentication Manager [shared development with the web client]
- Booking Service
- Service Request [shared development with the web client]

On the Cloud:

- DynamoDB DBMS
- Amazon Cognito
- API Gateway
- Booking Service

5.2.2 New services specific for booking a visit

- On the application:
- Information Retrieval
- Suggestion Retrieval
- Directions Query
- TTL Manager

On the Cloud:

- Suggestion Processor
- Shop Data Retrieval

5.2.3 New services specific for monitoring the entrances and the statistics

On the web client:

- Model and View
- Authentication Manager [shared development with the app]
- Statistics Manager
- Service Request [shared development with the app]
- Web Pages Retrieval

On the cloud:

- Statistics Query
- Amazon S3

The Google Directions APIs are available with no development needed, through HTTP requests.

5.3 Implementation plan

The implementation plan has to take into account the development of core features first.

5.3.1 Basic functionalities - App and Cloud

The primary part of the system is represented by the cloud infrastructure, since every other high-level component has to transfer information from and to it. The key component is the DBMS, «DynamoDB», which has to store every information in CLup infrastructure, and has to be developed first.
Another key component is the «Authentication Service», through Amazon Cognito, which has to be configured together with the «Authentication Manager» of the Web Client and the application.
In order to allow the basic functionalities of the application to work, other components are needed. The modules which constitute the «MVC» structure of the app have to be defined, along with the «Service Request» component, -developed both for the Web Client and for the app-. The «Booking» component has to be created as well.
As long as the Cloud counterparts to these services are concerned, the «API Gateway» is another key element that has to be early-stage defined. The Cloud component «Booking Service» has to be developed as well.
With the above mentioned components in place, the very basic functionalities of the app will work.

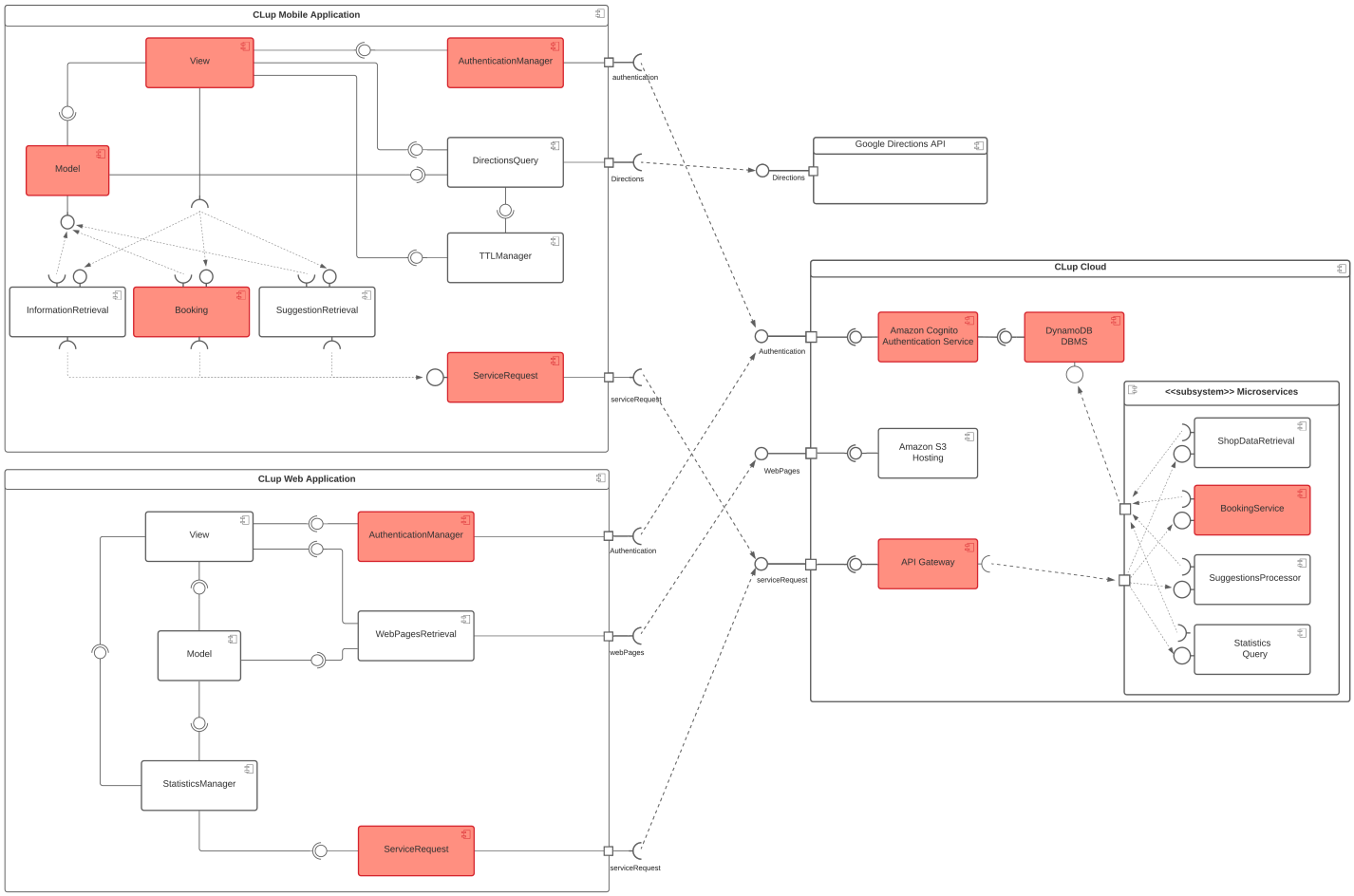

Figure 32: Components implemented for booking a ticket

5.3.2 Advanced functionalities - App and Cloud

The service can now be expanded with the functions for booking a visit.
In order to book a visit, it is necessary to retrieve many information from the Cloud, such as the available shop, timeslots, habits of the user as well the suggestions provided by the system. The «Information Retrieval» and the «Suggestion Retrieval» modules must be developed.
On the Cloud, the «Shop Data Retrieval» and the «Suggestion Processor» have to be developed in order to communicate with the respective components app-wise.
The application also needs to advise the user when it is time to leave in order to get to the appropriate shop in time. The «TTL Manager» and the «Directions Query» components are now strictly necessary, and have to be developed together, due to their high connection.

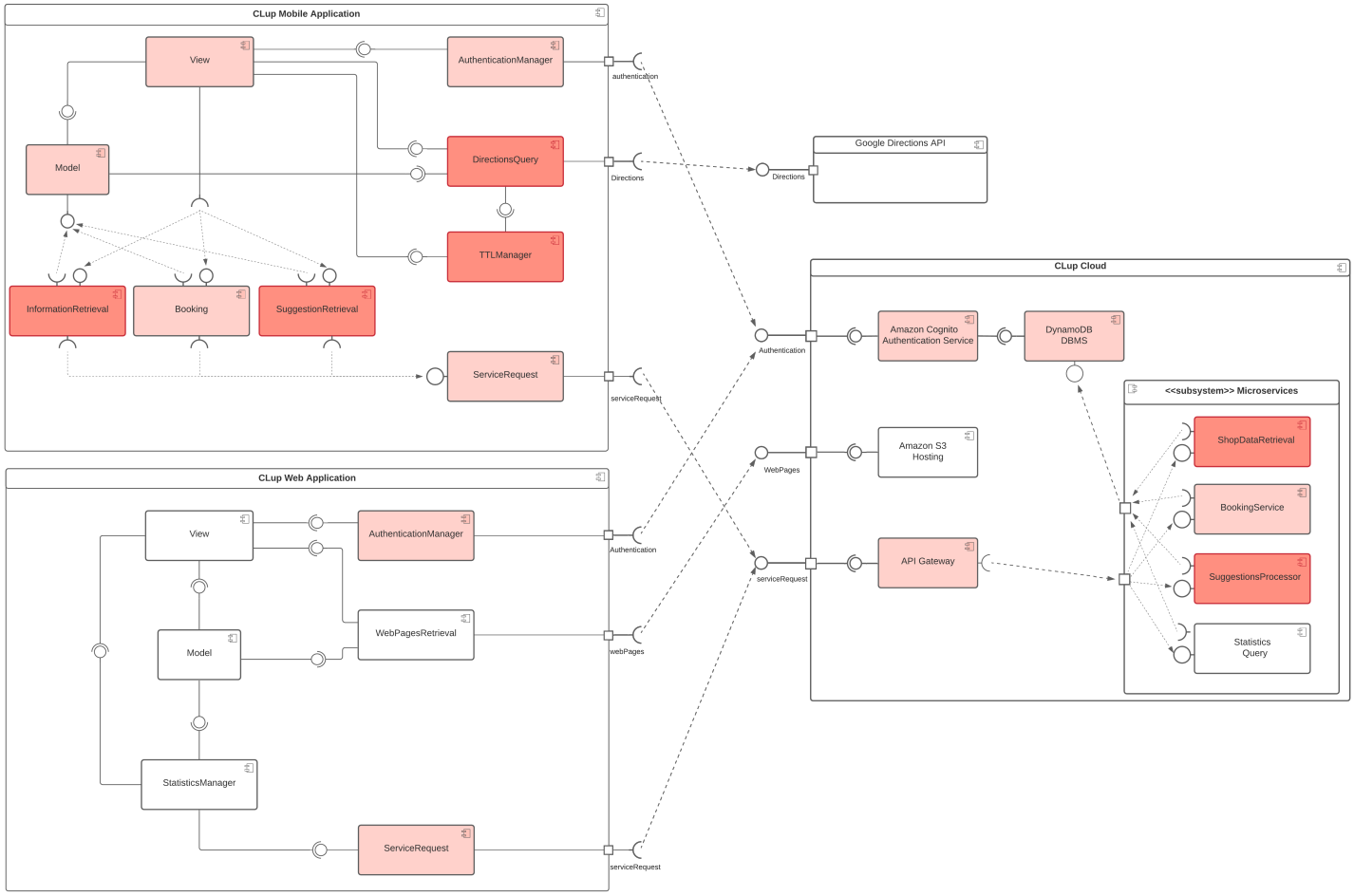

Figure 33: Components implemented for booking a visit

5.3.3 Advanced functionalities - Web Client and Cloud

CLup also offers the store managers the possibility to retrieve data and statistics about their own shop. To do that, the Web Client has to be completed, with an appropriate «MVC» structure, with the «Statistics Manager» and the «Web Pages Retrieval». «Authentication Manager» and «Service Requests» should already be defined in the early-stages, with the development of the basic functionalities.
On the Cloud, the «Statistics Query» and «Amazon S3 Hosting» components must be created.

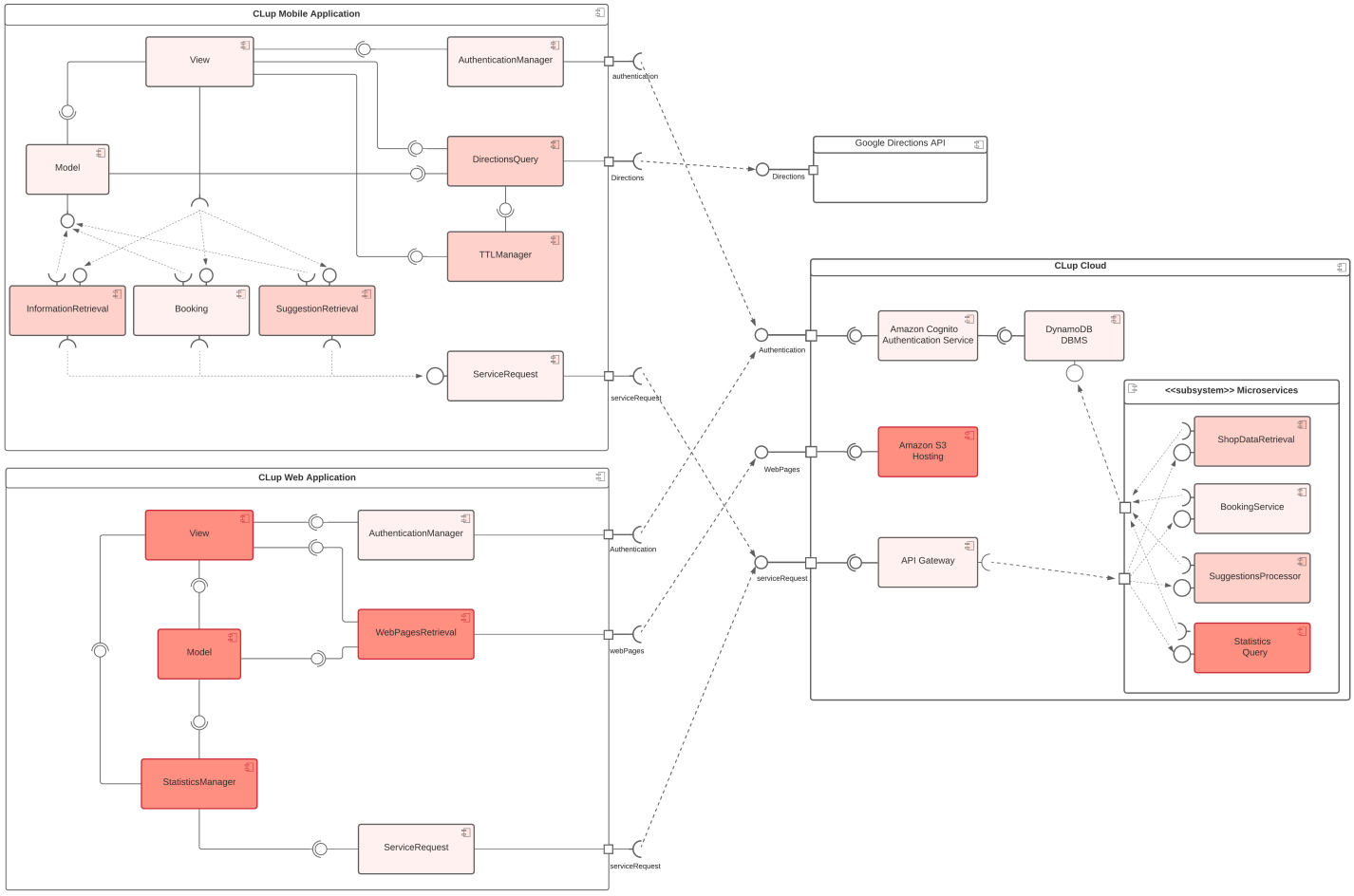

Figure 34: Components implemented for monitoring the entrances

5.4 Integration and Testing

The power of the function-based approach in defining the implementation plan relies on the ability to perform accurate integration testing, both with the backend and the frontend of the application already in place. Meanwhile the other components can be independently developed in parallel as well, since their creation affects the other functions behaviour only by a small extent.

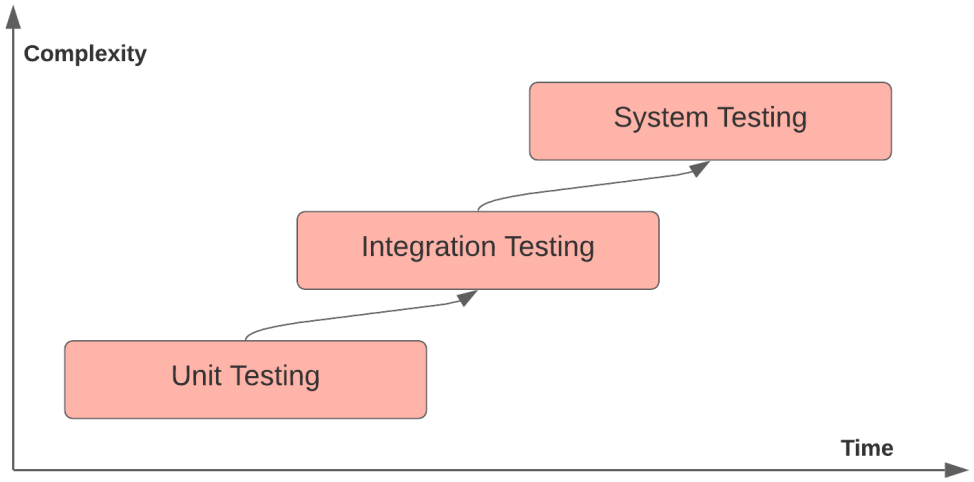

Figure 35: Unit, integration and system testing procedures compared

5.4.1 Unit Testing

Thanks to the bottom-up approach, every module can be extensivley tested and verified in isolation from the ground up. This is particularly useful because it allows to verify the integrity of each single module before connecting it to the others, thus having an adequate level of reliability before moving on to other testing methods.
Every single component needs to have its particular suite of tests, which has to always run correctly and to be built with the creation of the component itself during the implementation activity.

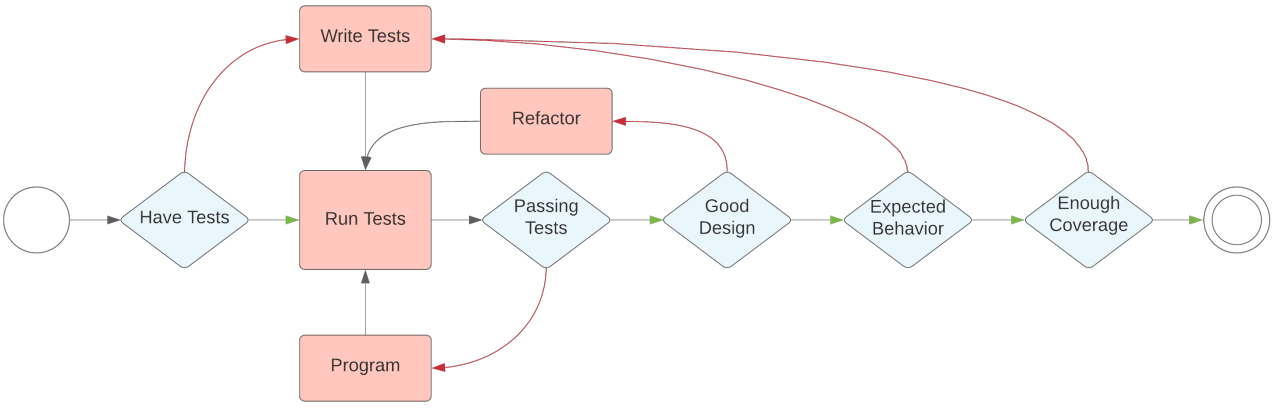

Figure 36: Unit testing flowchart

5.4.2 Integration Testing

The modules are gradually integrated into subsystems, carrying out appropriate checks on their correct integration, as soon as they have been tested with unit tests. The subsytems are the ones defined in section 5.2 and further expanded in section 5.3, they represent the main functionalities of the system, which can be tested on their own through the integration testing. This allows for a faster and easier testing.

5.4.3 System Testing

The complete system is validated against its functional (specification) and non-functional (performance, reliability...) requirements. This represents the last step in integration and testing. The system is then ready to be deployed, if it behaves properly.

6 Effort Spent

Hamza Haddaoui		
Task	**Time**	**Date**
First Meeting (work division)	3.5h	2020/12/12
Chapter 1	2.0h	2020/12/15
Second Meeting (work check)	3.0h	2020/12/16
Chapter 2	8.5h	2020/12/17
Third Meeting (merge)	3.0h	2020/12/18
Final Meeting	3.0h	2020/12/23
Refinements	3.0h	2020/12/26
Final refinement	3.0h	2020/12/27
Total	**29 h**	

Giuseppe Piccirillo		
Task	**Time**	**Date**
First Meeting (work division)	3.5h	2020/12/12
Mockups work	6.0h	2020/12/14
Second Meeting (work check)	3.0h	2020/12/16
Chapter 3	4.0h	2020/12/17
Third Meeting (merge)	3.0h	2020/12/18
Final Meeting	3.0h	2020/12/23
Refinements	3.0h	2020/12/26
Final refinement	3.0h	2020/12/27
Total	**28.5 h**	

Alessandro Restifo		
Task	**Time**	**Date**
First Meeting (work division)	3.5h	2020/12/12
Second Meeting (work check)	3.0h	2020/12/16
Chapter 4-5	6.0h	2020/12/17
Third Meeting (merge)	3.0h	2020/12/18
Chapter 4-5	2.5h	2020/12/21
Final Meeting	3.0h	2020/12/23
Refinements	3.0h	2020/12/26
Final refinement	3.0h	2020/12/27
LaTeX Refinement	1.5h	2020/12/29
Implementation and Testing Images	2.0h	2020/12/30
Total	**30.5 h**	

7 References

[1] - par 2.1: AWS - Amazon Web Services https://it.wikipedia.org/wiki/Amazon_Web_Services

[2] - par 2.1: Amazon CloudFront https://aws.amazon.com/cloudfront/

[3] - par 2.1: Amazon S3 https://aws.amazon.com/s3/

[4] - par 2.1: Amazon Cognito https://aws.amazon.com/cognito/

[5] - par 2.1: API Gateway https://aws.amazon.com/api-gateway/

[6] - par 2.1: Lambda Services https://aws.amazon.com/lambda/

[7] - par 2.1: Amazon DynamoDB https://aws.amazon.com/dynamodb/

[8] - par 2.2: Google Directions API Documentation at https://developers.google.com/maps/documentation/directions/overview?csw=1