



Politecnico di Milano

SOFTWARE ENGINEERING II PROJECT

---

# **DD**

## **DESIGN DOCUMENT**

---

*Professor:*  
Matteo Giovanni Rossi

*Authors:*  
Stefano Vanerio  
Agnese Straccia  
Cristiano Serafini

*Date:*  
10 January 2020

*Version:*  
2.0

# CONTENTS

## 1 Introduction

1.1 Purpose . . . . .	4
1.2 Context . . . . .	4
1.3 Scope . . . . .	4
1.4 Glossary . . . . .	5
1.4.1 Definitions . . . . .	5
1.4.2 Acronyms . . . . .	5
1.4.3 Abbreviations . . . . .	6
1.5 Document Structure . . . . .	6

## 2 Architectural Design

2.1 Overview. . . . .	8
2.1.1 General Context . . . . .	8
2.1.2 Composition Diagram . . . . .	10
2.2 Component View . . . . .	12
2.2.1 Server Component Diagram . . . . .	16
2.3 Deployment View . . . . .	23
2.3.1 Deployment Diagram . . . . .	25
2.4 Runtime View . . . . .	27
2.4.1 Customer . . . . .	27
2.4.2 Manager . . . . .	36
2.5 Component Interfaces . . . . .	39
2.6 Selected Architectural Styles and Patterns . . . . .	43
2.6.1 Client & Server . . . . .	43
2.6.2 Four - Tier Architecture . . . . .	44
2.6.3 RESTful Architecture . . . . .	46
2.7 Other Design Decisions . . . . .	47
2.7.1 Relational Database . . . . .	47

## 3 User Interface Design

3.1 UX Diagrams . . . . .	48
3.1.1 Customer Application FLOW Diagram . . . . .	49
3.1.2 Manager Application FlowDiagram . . . . .	50
3.2 Customer Application . . . . .	50
3.2.1 Login . . . . .	51
3.2.2 Registration . . . . .	52
3.2.3 Home Page . . . . .	53
3.2.4 Supermarket Selection (LineUp) . . . . .	54
3.2.5 Ticket Preview (LineUp) . . . . .	55
3.2.6 Ticket (LineUp) . . . . .	56
3.2.7 Ticket Machine Interface . . . . .	57
3.2.8 Supermarket Selection (BookAVisit) . . . . .	58
3.2.9 Time Slot Choice (BookAVisit) . . . . .	59
3.2.10 Fully Booked Notification (BookAVisit) . . . . .	60

3.2.11	Choice of Product Categories . . . . .	61
3.2.12	Shopping List . . . . .	62
3.2.13	Suggested Time Slots . . . . .	63
3.2.14	Alternative Stores . . . . .	64
3.2.15	Ticket (BookAVisit) . . . . .	65
3.2.16	Notification Management . . . . .	66
3.3	Manager Application . . . . .	67
3.3.1	Login Page . . . . .	67
3.3.2	Registration Page . . . . .	68
3.3.3	Home Page . . . . .	69
<b>4</b>	<b>Requirements Traceability</b>	<b>70</b>
<b>5</b>	<b>Implementation, Integration and Test Plan</b>	
5.1	Overview . . . . .	73
5.2	Implementation Plan . . . . .	74
5.3	Integration Plan . . . . .	77
5.4	Plan Definition . . . . .	78
5.5	Testing . . . . .	84
<b>6</b>	<b>Effort Spent</b>	
6.1	Teamwork . . . . .	85
6.2	Individual Work . . . . .	85
	<b>Appendices</b>	
A	Revision History . . . . .	87
B	Software And Tools Used . . . . .	87
<b>7</b>	<b>References . . . . .</b>	<b>87</b>

# 1 Introduction

## 1.1 Purpose

This document, “Design Document”, describes the architecture of the system illustrated in terms of functional and nonfunctional requirements in the Requirements Analysis and Specification Document previously written. It is important to underline that the “Design Document” relies entirely on it. We want to describe our software under different aspects but without contradictions and with a certain linearity with respect to the RASD, in order to satisfy the scope of the software itself. The architecture is represented by highlighting computational components and the interactions between them. In this first section there is a brief introduction in which the Context (Section 1.2) and the Scope (Section 1.3) of the software are indicated. In addition to that, it is also reported a Glossary (Section 1.4) for better readability and, at the end, the Document Structure (Section 1.5) that anticipates the entire structure of the document.

## 1.2 Context

The coronavirus emergency has put a strain on society on many levels, due to many countries imposing lockdowns that allow people to exit their homes only for essential needs and enforcing strict rules even when people are justified in going out such as keeping a distance of at least one meter between people. We have only focused on a small part of the real world affected by this situation: grocery stores, which are one of the essential needs for human beings. In fact in this situation people can avoid exiting their homes for shallow purposes but not for going shopping to the supermarkets. Supermarkets need to regulate the influx of people outside the building by avoiding the creation of long lines waiting to enter, but also to manage people inside by fixing a number of people allowed to stay at the same time. In this context CLup is an easy-to-use application that helps people to do shopping in a more safer way but also in a more technological way. It is very useful for providing a way to improve this situation by avoiding the spread of contagion.

## 1.3 Scope

The aim of the system is to ensure the distance between people and to develop an easy-to-use application that, on the one side, allows store managers to regulate the influx of people in the building and, on the other side, saves people from having to line up and stand outside of stores for hours on end. The software has to guarantee some useful functionalities on the application, the main ones are:

- “Lining up” feature: this functionality allows users to “line up” from their home and to retrieve an online ticket without going in front of the store and raising the possibility of creating crowds. In this way, people leave their home only when it is their turn and the system warns them about the expected time needed to reach the store.

- “Book a visit” feature: the application must give the opportunity to make a store visit reservation by inserting date, time and an expected duration of the visit and also by selecting a list of items they would like to buy. Moreover, the system stores reservation data in order to infer the duration of the visit for long-term customers. This would allow the application to plan visits in a better way, for example allowing more people in the store, if it knows that they are going to buy different things, hence they will occupy different spaces in the store when they visit.
- Suggest alternatives and options feature: the application must suggest to the customers alternatives slots in a day/time range for visiting the store, to balance out the number of people in the store, or different stores of the same chain if the preferred one is not available.

This system is useful also for store managers who can manage the limited number of people inside the building and monitor accesses to the stores.

The application, in order to be effective, needs to be very easy to use, in this way it can be used by a large range of people including people who do not have access to the required technology, indeed for this purpose there is also the possibility to hand out “tickets” on the spot, thus acting as proxies for the customers.

## 1.4 Glossary

### 1.4.1 Definitions

- QR code: is a type of Matrix Barcode, it consists in a group of black squares on a white background. It is possible to read it with a device like a camera. It contains some data codified in patterns that are present in horizontal and vertical lines.
- OTP: is a password that is valid only for one transaction or session in a computer system or digital device. It consents to be protected from some types of attacks and grants a higher level of security.
- DBMS: a database management system is a software system designed to allow efficient creation, manipulation and querying of data.

### 1.4.2 Acronyms

- API: Application Programming Interface
- GPS: Global Positioning System
- HTTPS: HyperText Transfer Protocol over Secure Socket Layer
- JVM: Java Virtual Machine
- OS: Operating System
- OTP: One Time Password

- QR Code: Quick Response Code
- RASD: Requirements Analysis and Specification Document
- REST: Representational State Transfer
- TCP/IP: Transmission Control Protocol/ Internet Protocol
- UX: User Experience Design

### 1.4.3 Abbreviations

- R: Requirement
- IIT: Implementation, Integration and Test

## 1.5 Document Structure

In this section, we want to briefly describe and anticipate the structure of the entire document, in order to clarify what each section is about. The structure is structured as follows:

- **Introduction:** in this section the Purpose of the document is highlighted followed by the Context and the Scope in order to give a first introduction of the system. There is also the Glossary that clarifies some acronyms, abbreviations and definitions of critical words.
- **Architectural Design:** this section gives a high-level description of the architecture of the software by pointing out the various components the system is composed of and how they interact with each other. Then it is followed by the Deployment Diagram which helps to understand the distribution of components among devices. The Runtime View specifies through sequence diagrams the dynamic behavior of the components and how they are connected together. Also the Component Interfaces are described. At the end of the section we have justified all the strategies and decisions applied to our system.
- **User Interface Design:** this section is dedicated to the mockups of CLup. It provides an overview on how the user interfaces of the system look like. All the interfaces already described in the RASD document are now illustrated in a more detailed way. It is described how the various screens are connected and what are the features that each interface offers.
- **Requirements Traceability:** this section explains how the requirements, defined in the RASD, are mapped to the design components that we have defined previously in the second section of this document.
- **Implementation, Integration and Test Plan:** this section identifies all the design decisions and plans before the start of the implementation. We will illustrate the decisions made regarding bottom-up or top-down strategies at the beginning of the section. Then we clarify the order in which we plan to implement the subcomponents

of the system and the order in which we plan to integrate such subcomponents. Also further specifications about testing are discussed.

- **Effort Spent:** in this section we will include information about the number of hours each group member has worked to each section and all the hours spent together.
- **References:** references used for the document.

## 2 Architectural Design

This is the core and the body of the entire document in which we will focus on each aspect of the CLup system. We are going to describe the architecture of our software. First, we will show all the external systems that CLup interacts with. This document will provide, unlike the RASD document, a precise specification of each component.

In the following part, we will provide a global description of all the different components needed to obtain all the functionalities of the system and how they interact, then their deployment and behavior at runtime. The section ends with the description of the architectural patterns that we have chosen and our design decisions.

### 2.1 Overview

#### 2.1.1 General Context

This section shows a high-level view of our system in order to understand what other systems are needed to provide all the functionalities of the system. CLup is an application that provides services to two types of users: **customers** and **managers**. The customers are the people who want a ticket to enter a supermarket thanks to the core functionalities (**Book A Visit** and **Line Up**) and are the most conspicuous part of the users. The managers, instead, are the users interested only in the management of the supermarket thanks to the Manager functionality.

For greater clarity, we will define the most important functions:

- **Access Functions:** services that allow both customers and managers to register and login to the system to be recognized.
- **Core Functions:** services provided to customers to line up online or book a visit to a specific supermarket on a chosen date and time.
- **Advanced Functions:** services that suggest to the customers alternative slots in a day/time range for visiting the store, to balance out the number of people in the store, or different stores of the same chain if the preferred one is not available. Also, it provides periodic notifications of free slots for customers who allow it explicitly.
- **Manager Functions:** services that manage the modification of the capacities of the stores, specifically defined for the managers of the supermarkets.

In the following picture, we are going to illustrate the **Context Viewpoint** (Fig. 1): a schema in which there are the systems that are needed to obtain the functionalities presented above. It is possible to notice that only the manager and the customer interact directly with the system. For all the other services, the system makes a request and the external service creates a response after the computing of a specific feature.



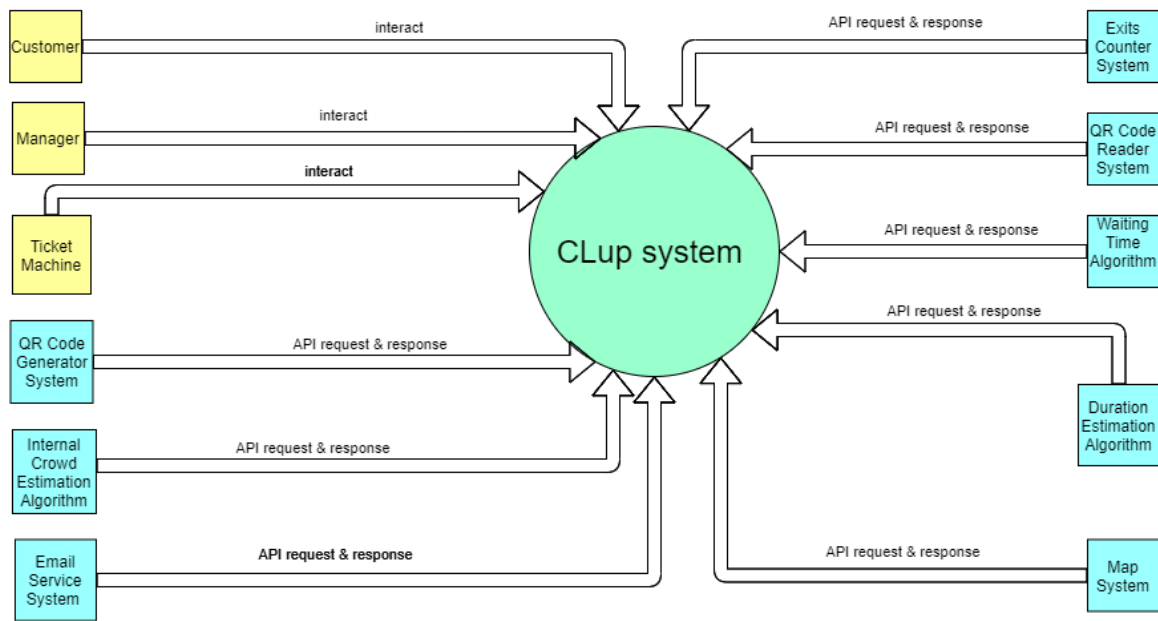


Figure 1: Context Viewpoint

The following table is useful to better understand the mapping of the various external services with the different functions of the system.

Functionality/Service	QR Code Generator System	QR Code Reader System	Exits Counter System	Email Service System	Map System	Internal Crowd Estimation Algorithm	Duration Estimation Algorithm	Waiting Time Algorithm
Access Functions				X				
Core Functions	X	X	X		X		X	X
Advanced Functionality						X		

Table 1: System Mapping Table

It is important to point out that all the systems, which name presents the word 'algorithm' (Waiting Time Algorithm, Internal Crowd Estimation Algorithm, Duration Estimation Algorithm), could be implemented internally. We have made this decision for many reasons that are better explained in Section 2.2.

### 2.1.2 Composition Diagram

After defining the external systems used by CLup, we are now able to highlight the modules used to complete the services. In the following picture, we are going to show the different modules considered crucial for the corresponding layer.

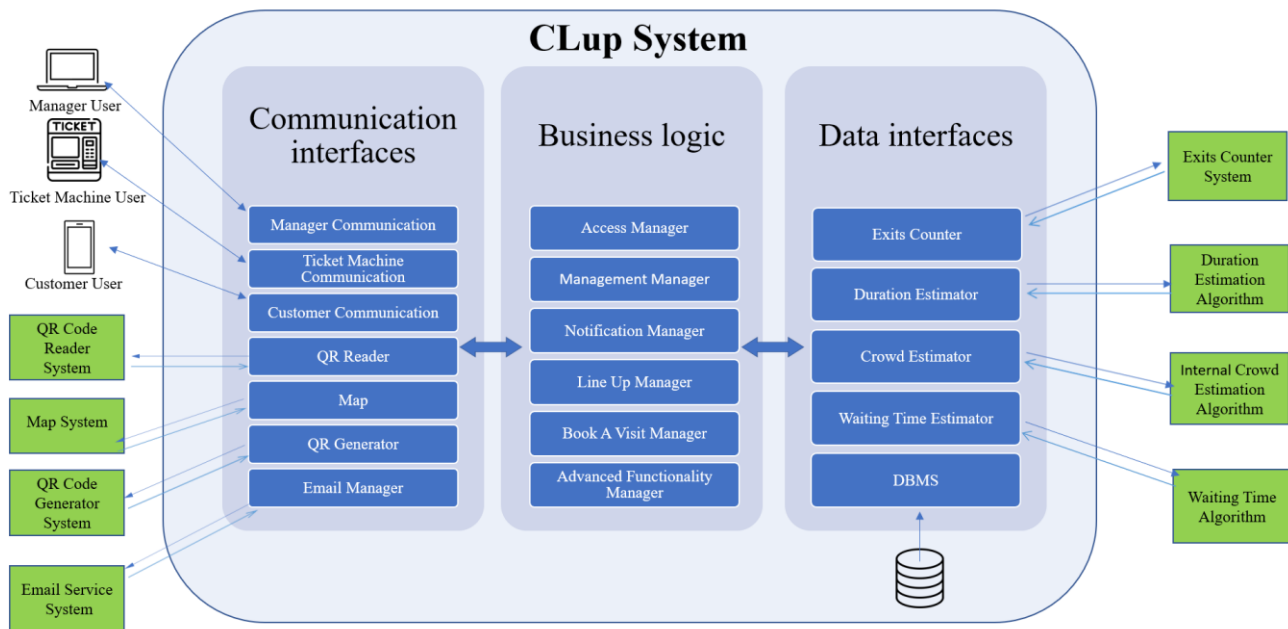


Figure 2: Composition Diagram

**Communication Interfaces:** these interfaces allow our system to communicate with external systems that provide the services that business logic needs. As we notice on the left-hand side of the diagram, we have:

- **Manager Client:** is the client used by a manager to handle CLup functionalities.
- **Ticket Machine Client:** is the client, placed outside a supermarket, used by a customer that does not have a device to benefit from CLup functions.
- **Customer Client:** is the client used by a customer to benefit from CLup functionalities.
- **QR Code Reader System:** is used to read the QR code generated during the “LineUp” or “Book a Visit” request formulation.
- **QR Code Generator System:** is used to generate a unique QR code during the “LineUp” or “Book a Visit” request formulation.

- **Map System:** is used to estimate the distance between customers and supermarkets and compute the time needed by the customer to reach his destination.
- **Email Service System:** it is used to send emails during the recovery of the account and the sending of the OTP code used by the manager in order to access.

**Business Logic:** in this layer, we have the modules that deal with the system functionalities:

- **Access Manager**
- **Notification Manager**
- **Line Up Manager**
- **Book A Visit Manager**
- **Management Manager**
- **Advanced Functionality Manager**

These modules are related to specific components and this connection is explained better in the next section. In the case of the “Advanced Functionality Manager” there are several different components that provide the functionalities needed.

**Data Interfaces:** the last interfaces provide data and are used in the different functionalities of the system. For this reason, the following interfaces can be found in this layer:

- **Exits Counter System:** is used to count how many customers left after they enter the supermarket.
- **Duration Estimation Algorithm:** is used to estimate the duration of the visit of a long-term customer based on his previous visits.
- **Internal Crowd Estimation Algorithm:** is used to estimate if it is possible to modify the maximum number of people inside the store without risks.
- **Waiting Time Algorithm:** is used to estimate how much time a user has to wait before his turn.

## 2.2 Component View

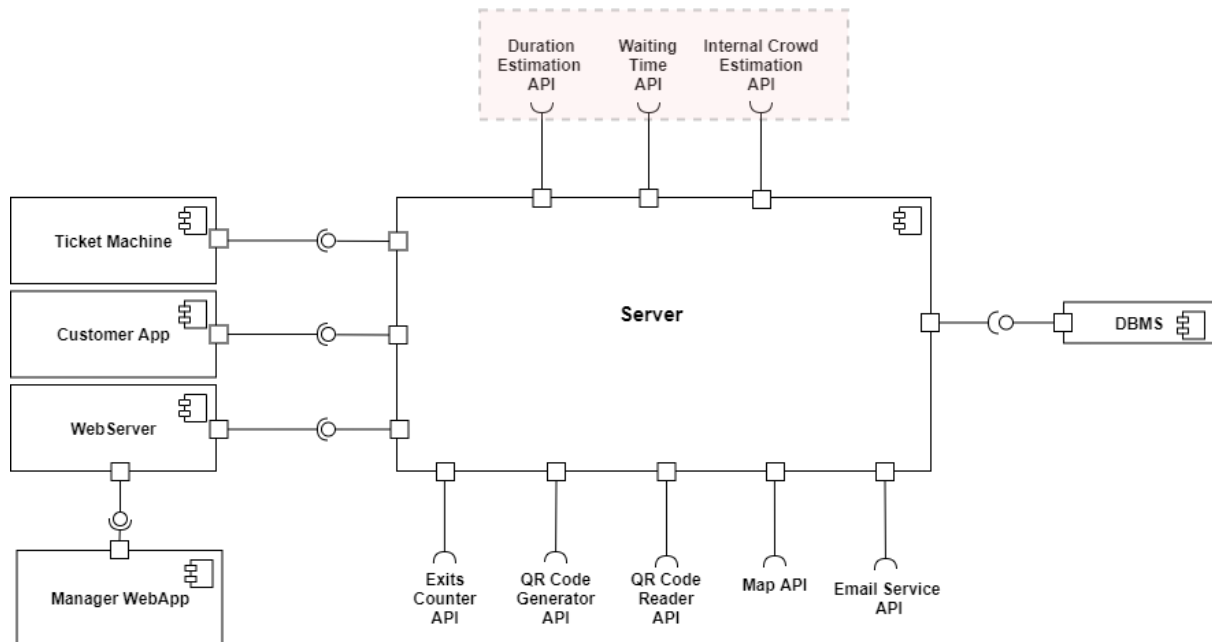


Figure 3: High-Level Component Diagram

Thanks to the introduction about the system, its interactions and its functionalities, we are now able to describe in a general view all the components that are used to perform the actions to reach the goal of the application.

Referring to Figure 2, that divides the system in three parts (Communication Interfaces, Business Logic and Data interfaces), it is possible to map each part to the different components of the system. Regarding the Communication Interfaces, as better explained in the previous section, it is possible to identify all the components and the external systems that provide services. The Business Logic is realized by the server and its internal components (as better explained in the section 2.2.1), and for the Data Interfaces the system works with the external system for the data management (DBMS).

Before the description of each component it is important to highlight one fundamental decision taken in this analysis. We have chosen to treat some services, highlighted in red in the picture above as external services. It is possible to implement all this services inside the system and, in this way, reduce the number of external interfaces needed, but we have chosen to use this specific approach for different reasons:

- CLup is an application dedicated to reservations and requests management, it is not a data mining application, so all the services related to the analysis of big amounts of data are not directly inserted inside the application, but they are provided by external services.

- In addition to that, the structure in which the data is stored in the CLup storage is most likely not optimized for searches of specific kinds of data. In the case of the Duration Estimation, for example, the function has to search for all the reservations of a specific customer and, using the storage of the system, this operation can take a while. Using external services it is possible to use a different kind of storage for the storing of a copy of the data (updated daily) resorted for a specific kind of search. In this way, it is possible to avoid bottle-neck situations inside the primary storage and speed-up the different kinds of search.
- For some of those services, like for example the Internal Crowd Estimation Service with the use of specific algorithms implemented for a certain functionality (like for example the Binary Space Partitioning) the efficiency grows in an important way in the case of a dedicated system.
- Thanks to the outsourcing of these services the developers can focus on the development of the application and in the optimization of all the other core services.

We are totally aware of the consequences of these decisions and in this document all the sections are written with this important assumption.

Now, we can proceed with the at high level description of the system (Fig. 3) and in the following section the fundamental components are better explained:

- **Server:** this component provides the business logic of the system. As better explained in the next section it is composed of various subcomponents that provide all the functionalities required by the goals. The interfaces provided by the system are the ones used by the users for the achieving of the functionalities granted by the system. One interface is dedicated to the “Customer App” component that represents the mobile software in the device of the customer. The second one is connected to a “Web Server” component that is in charge of the connection between the server and the Web Application used by the managers to regulate the influx of people inside the supermarket.
- **WebServer:** it is the necessary component that provides the necessary connection used by the browser where the web application runs and the functionalities of the system.
- **Manager WebApp:** it is the component representing the web browser used by the manager in order to use the services provided by the system for this specific kind of user.
- **Customer App:** it is the component representing the mobile application used by the customer in order to obtain the access to all the services granted to this specific class of users by the system.

- **Ticket Machine:** it is the component representing the client that runs on a physical ticket machine placed near the supermarket. It sends only LineUp requests with all the fields already compiled and, once the confirmation is received, prints a physical ticket.
- **Duration Estimation API:** it is the interface provided by the external system whose task is to compute, by analyzing a big amount of data, the duration of a visit for a long-term customer. This information is sometimes needed by the system during a Book A Visit reservation and, in that case, the external system after a request received by the system computes the operations and sends back to the system the result.
- **Waiting Time API:** it is the interface provided by an external system used for the computation of the waiting time in a specific situation. It has to take into account various information about the day of the week, the season, the weather and all the information related to the line (customer number, current number) to estimate a quite precise estimation of the waiting time for the customers. In this way the system can show this information to the customer during all the phases of the reservation for a specific service.
- **Internal Crowd Estimation API:** it is the interface provided by an external system used for the computation of the check of the internal crowd situation of specific departments of the store. It has to implement specific algorithms for the analysis of the internal space of the shops and, once consulted by the system for a specific request, has to respond by admitting or rejecting an over-capacity request according to the shopping list of the customer.
- **QR Code Generator API:** it is the interface provided by an external system in charge of the generation of the entry ticket. Every entry ticket contains all the information of a specific reservation and the external service, after the retrieval of those information, creates a unique ticket containing them. That ticket is used by the customer when he enters the supermarket
- **QR Code Reader API:** it is the interface of the external service used by the system to perform the reading of the tickets at the entrance of the store. This external service clearly needs a physical device with a camera to perform the action on spot (it can be an automatic machine or a smartphone of an employee for example). It reads the tickets of a fixed number of customers based on the capacity of the supermarket and stops when the maximum capacity is reached for the LineUp functionality.
- **Map API:** it is the interface provided by an external system that has to compute the distance between the customer that uses CLup and the supermarkets. This information is used during the selection of a specific supermarket during the creation of a LineUp or a BookAVisit Request to recommend the nearest supermarket.

In addition to that it is also used to compute the time needed by the customer to reach the supermarket and inform him in advance. We have decided in this document that the

customer moves to the supermarket walking. It is reasonable to think that in a pandemic situation it is not possible to move too far from home, due to the different restrictions imposed by the government. Obviously, it is possible to add also the choice of the transport mode and, in that case, the customer has to select it. In that case the developer has to take into account that this choice is stored with the user data but, for a better readability and a simplified model, this feature is not explained in detail in the document.

- **Exits Counter API:** it is the interface provided by the sensor (or some systems with the same goal) used for the count of the customer exits. In this way the system knows the correct number of customers inside the shop and can manage the requests in the right way without the risk of crowding. This information is sent to a specific component (better explained in the next section).
- **Email Service API:** it is the interface provided by an external email service. Thanks to this connection it is possible for the system to send emails to the customer in case of reset of the password or to the manager during the access with the OTP code.

## 2.2.1 Server Component Diagram

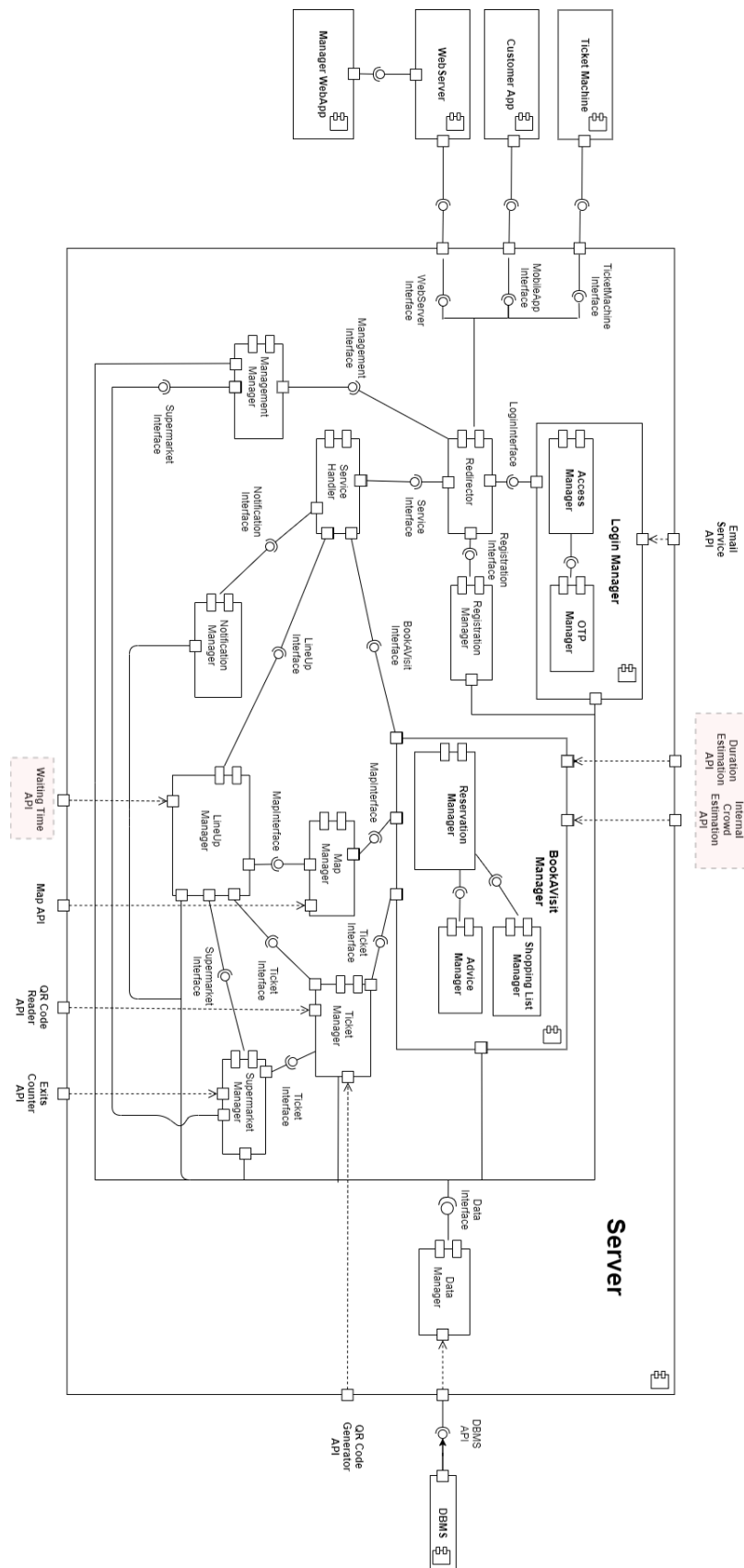


Figure 4: Server Component Diagram



It is possible to find the diagram with a better quality at this [link](#).

After some premises, it is possible to analyze in greater detail the internal components of the server used to achieve all the functionalities required by the goals.

First of all, it is possible to observe how the external connection to the various interfaces are the same of the previous picture (Fig. 3). The explanation provided by the previous section about the different external systems connected to the server remains the same and also the assumption of the “red boxes” components. In the graph, moreover, some subcomponents are encapsulated in other components. The explanation of these components will take place in the section below and it will better explain what is the function of each of them.

Now it is possible to explain in detail all the functionalities provided by the various components and how they interact:

**Redirector:** it is the first component reached by all kinds of requests. It redirects the different requests of the user to other components analyzing two characteristics:

- Sender: it distinguishes between customer requests and manager requests
- Type: there are different kind of requests that can reach the redirector, they are:
  - Login Requests: requests redirected to the Login Manager
  - Registration Requests: requests redirected to the Registration Manager
  - Service Requests:
    - Sent by a customer: redirected to the Service Handler
    - Sent by a manager: redirected to the Management Manager

In this way all the requests are forwarded to the correct component that can manage them and, in addition to that, requests that are not formatted in the right way (or with the wrong level of permission) are discarded.

**Registration Manager:** it is the component dedicated to the creation of the account both for customers and managers. Clearly, the component can distinguish the two different types of senders and has to provide different forms to the two different kinds of user.

This component manages the check of the correctness of all the fields of the registration form and has also a direct connection to the Data Manager component needed to check if usernames or emails are already stored.

**Login Manager:** it is the component dedicated to the management of the login requests, necessary to access all the other services of the application (with the exception of the registration). It is clearly connected to the Redirector and also with the DBMS interface through the Data Manager to check the correctness of the username and password inserted by the user. It also performs the credentials recovery in the case of forgotten password by interacting with the Email Service if the user has inserted his email during the registration.

It is important to notice the presence of two internal subcomponents dedicated to the functionalities of this component:

- **Access Manager:** it is the component dedicated to the management of the access, it manages completely the login requests of the customers and also, in part, the login requests of the managers. It performs the duplicate checks and communicates with the other components.
- **OTP Manager:** it is the component dedicated only to the management of the OTP code necessary for the login of the manager. It generates a Pseudo Random number thanks to a Pseudorandom Number Generator function and sends it to the manager through the email service. In this way the manager can finish the procedure of login by inserting that code. This step grants better security for this important access.

**Management Manager:** it is the component dedicated to the functionalities available to the manager. This component provides to the manager a way to modify the internal capacity of the store for the two kinds of requests. For the achievement of this functionality, it interacts with the Supermarket Manager component, explained later in the analysis, and can modify its internal data to achieve the desired purpose of the manager. Moreover, it lets the manager add/remove supermarkets in the managed ones using the identifier and, for this reason, the component is connected to the Data Manager for the storing of this information.

**Service Handler:** it is the component dedicated to redirect the requests made by the customers to the different services provided by the system.

We have chosen to divide this component and the Redirector for a better management of the requests and to avoid bottle-neck situations in which the entire system is unable to perform any kind of operation and, specially the management functions.

The possible services granted by the system are managed by three different components:

- Book A Visit Service: provided by the BookAVisit Manager
- LineUp Service: provided by the LineUp Manager
- Notification Service: provided by the Notification Manager

All the advanced features granted by the system are present in the internal components of the BookAVisit Manager, in particular the advice of different supermarkets and time slots and the possibility of fulfilling a shopping list.

**Notification Manager:** it is the component that deals with the generation of the periodic notifications and the sending of them to customers. It has to check if the customer has given his permission to this feature and then, using the information retrieved by the Database through the Data Manager component, creates a personalized notification for the customer containing advice related to a supermarket and a timeslot. Clearly, the customer has to switch on the “notification permission” also on his device.

**Map Manager:** it is the component that deals with all the requests that have to know some information by a system that computes paths in the city and retrieves specific positions. This component interacts with the Map API that must provide these services. In detail:

- It is used by the LineUp Manager and the BookAVisit Manager for the computation of the distance of the customer device and all the supermarkets inscribed to the CLup application.
- It is used for the advice of the nearest supermarket during the supermarket selection and for the filter of supermarkets by street and city.
- It is used to compute the time needed by the customer to reach the supermarket for the LineUp requests.

**LineUp Manager:** it is the component that must deal with all the LineUp requests sent by the customers or by the Ticket Machine. It has to guide the customer during the creation of the requests showing to him the available supermarkets and the nearest ones thanks to the Map Manager connection.

Once the selection of the store is done, the component has to use the Map Manager to ask to compute the time needed by the customer to reach the store.

In addition to that, it retrieves from the Supermarket Manager the number of the customer currently in the supermarket and a new number for the new customer that has made the request.

Moreover, the component has to ask the external system through the Waiting Time API in order to compute a reasonably accurate estimation of the time that the customer has to wait before his turn.

Once all these steps are completed the component sends back to the customer a preview of the reservation and, in the case of acceptance by the customer, the component asks the Ticket Manager for a ticket. At this point, it is finally possible to confirm the reservation and send back to the customer the final ticket.

During the lifetime of the ticket inside the system it is updated with the information about the passing of time and the updates of the lineup current ticket number (in the case of a mobile application ticket).

Clearly the component is also connected to the Data Manager for storing the information about the requests.

**BookAVisit Manager:** it is the component that deals with the requests related to the Book A Visit functionality sent by the customers. It is composed of different subcomponents in order to also achieve the advanced functionality of the generation of possible alternatives for the customer in the case of a request for a fully booked time slot.

The subcomponents are:

- **Reservation Manager:** it deals with all the primary operations of the management of requests (providing possible supermarkets, elaborating user choices, etc.). All these operations will be presented in order in the next paragraph.
- **Shopping List Manager:** it deals with the management of the shopping list that the customer can fulfil. It has to have access to specific kinds of data like an updated list of the products available in the store and reach this goal thanks to the connection with the Data Manager component.
- **Advice Manager:** it deals with the suggestion phase that takes place when a simple booking is not successful. In that case the component has to create alternatives for the customer retrieving at first available time slots for the supermarket and, if these slots are not what the customer wants, has to retrieve also a list of other supermarkets related to the first one. If the alternative is accepted, the process is concluded by the Reservation Manager.

The process of the creation of a booking request (as better explained in Fig. 10) has these fundamental steps:

1. At first, the customer chooses a supermarket also filtering the possibilities by City or Street (thanks to the connection of the BookAVisit component with the Map Manager that provides these kinds of information).
2. Once selected the supermarket the customer has to choose a day and a time slot and, optionally, can fulfil the shopping list.

3. If the customer does not fulfil the list and tries to generate a ticket the component asks the Ticket Manager to create a ticket if possible.
4. If the ticket is generated, the component sends back a confirmation and the process ends. If the choices of the customer are not available, the Advice Manager tries to propose different alternatives.

The component has a connection with the Data Manager for the storing of the tickets, for the retrieval of the capacity for the day chosen by the customer and, as explained before, for the retrieving of different dates and supermarkets for the suggestion of alternatives.

**Ticket Manager:** it is the component used to manage the communication between CLup and the external systems that guarantee the generation and the reading of the QR codes. It manages the creation of the ticket once the request is completed in this way:

- For every BookAVisit request: it asks the QR Code Generator system to create a ticket and then provides the ticket to the Book A Visit Manager.
- For every LineUp request: it asks the QR Code Generator system to create a ticket and provides it to the LineUp Manager.
- For every read request by the external QR Code Reader system: it checks if it is stored in the system and in that case accepts the ticket and updates the capacities, if it is wrong it refuses the ticket.

The Ticket Manager shares the information about the maximum capacity of the store, updated by the Supermarket Manager, with the QR Code Reader system and, in this way, the external system knows when to stop the reading of the tickets (in case of a full supermarket).

**Supermarket Manager:** it is the component that manages the characteristics of every supermarket (opening and closing hours, closing days, etc.). It is the component modified by the Management Manager with the requests of modification of the capacities made by the manager.

It manages these requests differentiating them in:

- Management of the LineUp capacity
- Management of the BookAVisit capacity

It is also connected to the LineUp Manager and, in fact, provides to it an interface used to retrieve the number of the customer currently in the store and the number assigned to the new customer. In addition to that, the component has also to compute the frequent inputs provided

by the Exits Counter System that counts the number of customers exiting the building. That information is fundamental to have updated data to provide to the other components.

Moreover, the number of customers currently in the store is sent to the Ticket Manager and, in this way, the QR Code Reader System can stop (or resume) the reading of the tickets.

**Data Manager:** it is the component that deals with the management of the data of the system. It has to communicate with the DBMS and retrieve all the data needed by the different components of the system for the different services.

A large number of components has to interact with the Data Manager, so it is important to take into account this characteristic of the system and develop it in the correct way. A possible solution for this problem is the replication of the component obviously with a correct management of the multiple requests from the system to the database.

## 2.3 Deployment View

After the previous explanations about the high-level architecture of the system in the Overview (Section 2.1), we went into more details with the Component View (Section 2.2) which individually specifies each component of the system and for which functionality it is needed. Now, in this section, we want to clarify about the distribution of these components among devices but, before the illustration of the Deployment Diagram (Fig. 6), we want to show the architectural style we have decided to apply to the system in order to better understand the diagram.

The architectural approach used is a **Client/Server** application due to all the services that the system needs to provide to the clients. In this way clients can make requests to the system and the system answers to them with the service (or services) requested.

Before going into details about the client-side we want to point out the difference between users and customers previously explained in the RASD document: customers are users that cannot be managers, whereas users are both customers and managers.

The client-side is implemented with three different types of system and all of these have a well-defined interface illustrated later on in the Section 3:

- **Mobile Application:** all the customers can access to the system through a Mobile Application
- **Web Application:** the Web Interface is for managers that can access and see different functionalities with respect to the customers
- **Ticket Machine:** the ticket machine is located at the entrance of supermarkets

Therefore, to make this type of architectural approach possible, we need a **Web Server** component. We have decided to use a dedicated hardware system optimized for managing all the interactions between the web application of the manager with the system. Hence, we applied a **Four-Tier Architecture** in order to clearly distinguish the clients from the server, but we describe in detail all of these decisions and architectural patterns used later in Section 2.6.

The following diagram is the same Component Diagram as in the Section 2.2 but we have highlighted the different components of the four-tier architecture with different colors in order to better distinguish Clients, Web Server, Application Server and DB Server.

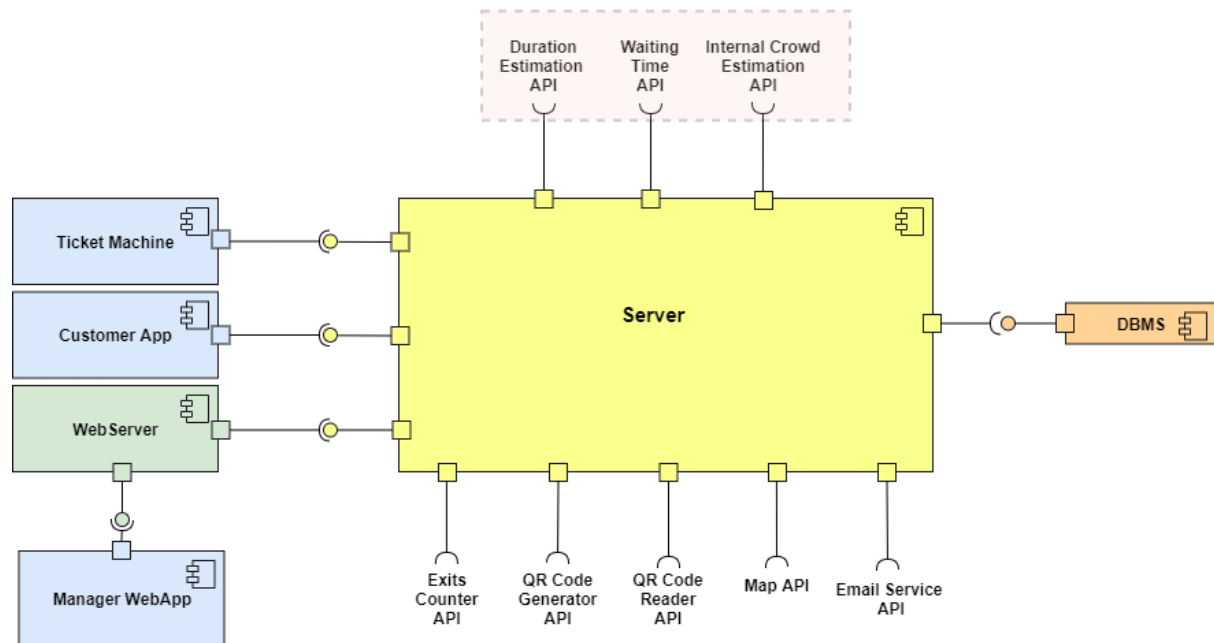


Figure 5: High-Level Component Diagram with coloured components



### 2.3.1 Deployment Diagram

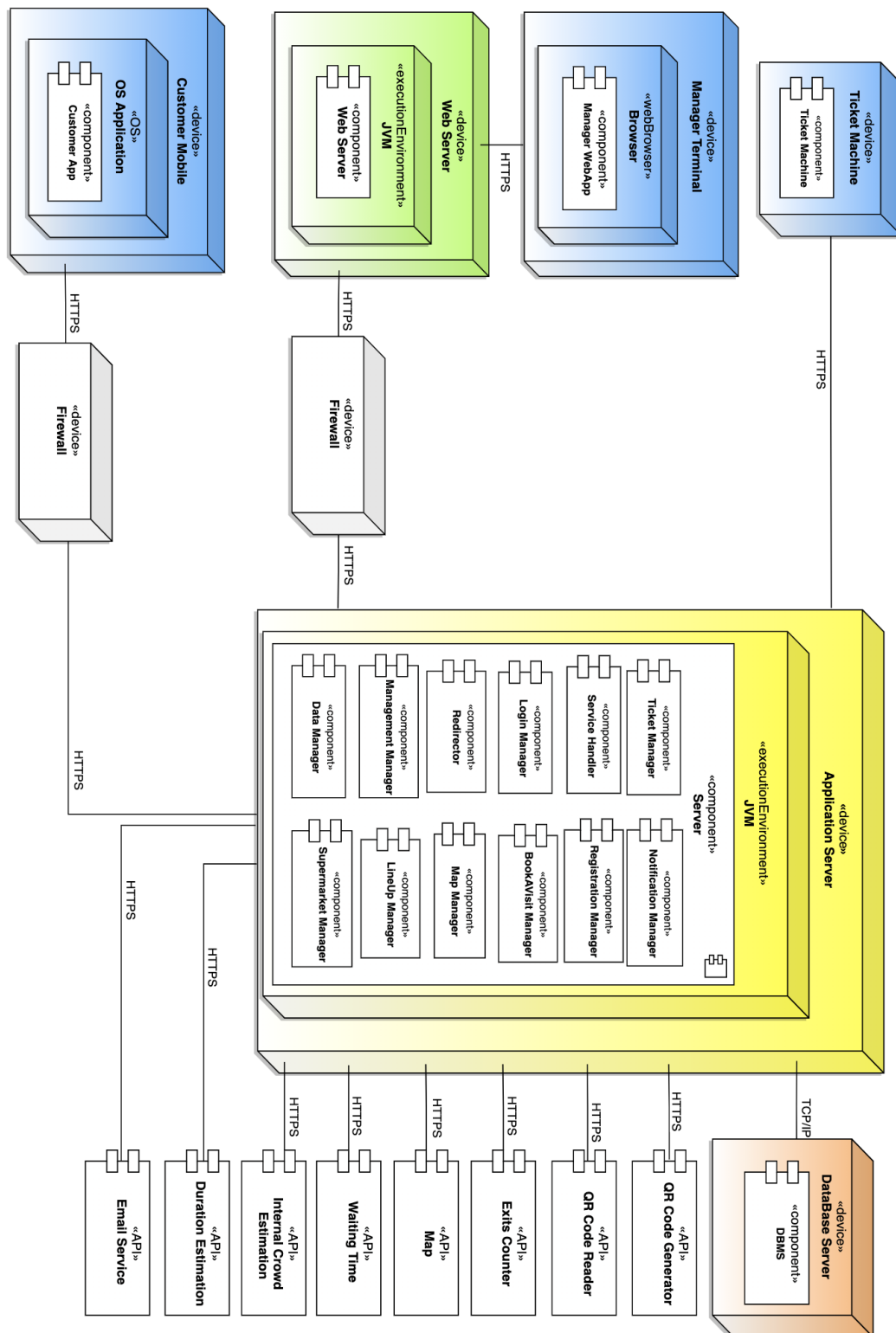


Figure 6: Deployment Diagram

In the above picture is presented the Deployment Diagram showing all the main devices in which client, server and DBMS are deployed. It is possible to notice that we have used the same colors as in the previous component diagram. In the right part of the diagram we have also added the external APIs provided to the software.

- **Manager Terminal:** this is the Manager client-side. It is used by the manager to registrate/login to CLup through a web browser and have access to all the functionalities.
- **Web Server:** this component is useful for the manager to communicate with the system. It is able to manage requests to transfer web pages to the Manager Terminal.
- **Customer Mobile:** this is the other client-side for all the customers. It is considered as a smartphone in which there is its own OS. CLup can be installed in all the mobile devices and hence it is compatible with all the main OSs.
- **Ticket Machine:** It is a physical machine located at the entrance of stores. Every time a customer goes to the store without a ticket, he has to physically retrieve it through this machine.
- **Application Server:** this is the application logic component. In this device there is all the business logic of CLup. It corresponds to the Server and all the server components previously described are included in it.
- **DataBase Server:** this is the device for the DBMS. It allows efficient creation, manipulation and querying of data.
- **Firewall:** firewalls are added in order to provide greater security and protect data from external attacks. They block and route packets between the internal and the external network depending on their header and their content.

The links between Manager Terminal and Web Server, Web Server and Application Server, Customer Mobile and Application Server, Ticket Machine and Application Server, Application Server and all the APIs are managed through a HTTPS protocol for secure communication. Whereas the link between Application Server and DataBase Server is managed through a TCP/IP protocol.

## 2.4 Runtime View

This section will present the most critical runtime views of the interactions between users and CLup. All the methods used in the following diagrams will be precisely described in the next section (2.5), where each of them will be mapped to the respective interface that realizes it. The Runtime View Diagrams of this section are presented first for the customers and then for the managers.

### 2.4.1 Customer

**Login:** In this sequence diagram (Fig. 7), the process of Customer login is shown. The Customer App asks the server to login.

The Redirector receives the request and forwards it to the Access Manager component. This component asks the Data Manager for customers' username and password from the database. If there are no customers with that username in the database, the client receives an error message. Otherwise, the check of the password is performed. If the password is correct, the customer will be logged and informed about the login success.

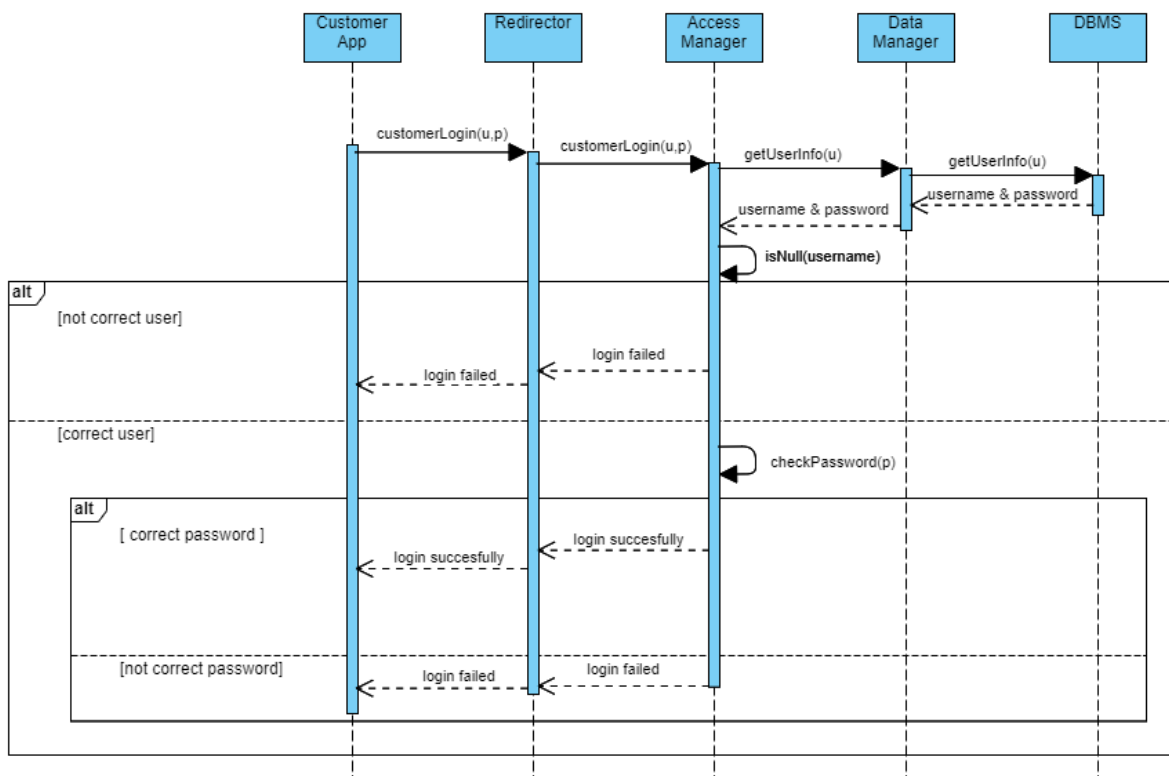


Figure 7: Customer Login Runtime View

**Registration:** In this sequence diagram (Fig. 8), the process of Customer registration is shown. The Customer App asks the server to register, passing username, password, confirm password, and an optional email. The Redirector receives the request and forwards it to the Registration Manager. This component asks the Data Manager if the received username is already present in the database. If the answer is negative, the Registration Manager checks that the two passwords entered are identical; differently, an error is given to the customer. If the two passwords match, the Registration Manager passes to the mail check. Otherwise, an error is given to the customer.

For what concerns the mail check, it is a bit complex: the customer can decide to register with or without email, with the only advantage that, in the first case, he will be allowed to recover the password in case he forgets it. If the customer has inserted the email, the Registration Manager asks the Data Manager if the received mail is already present. If the mail is not in the database, the customer is registered in the database and a confirmation message is sent; otherwise, an error message is sent to the customer.

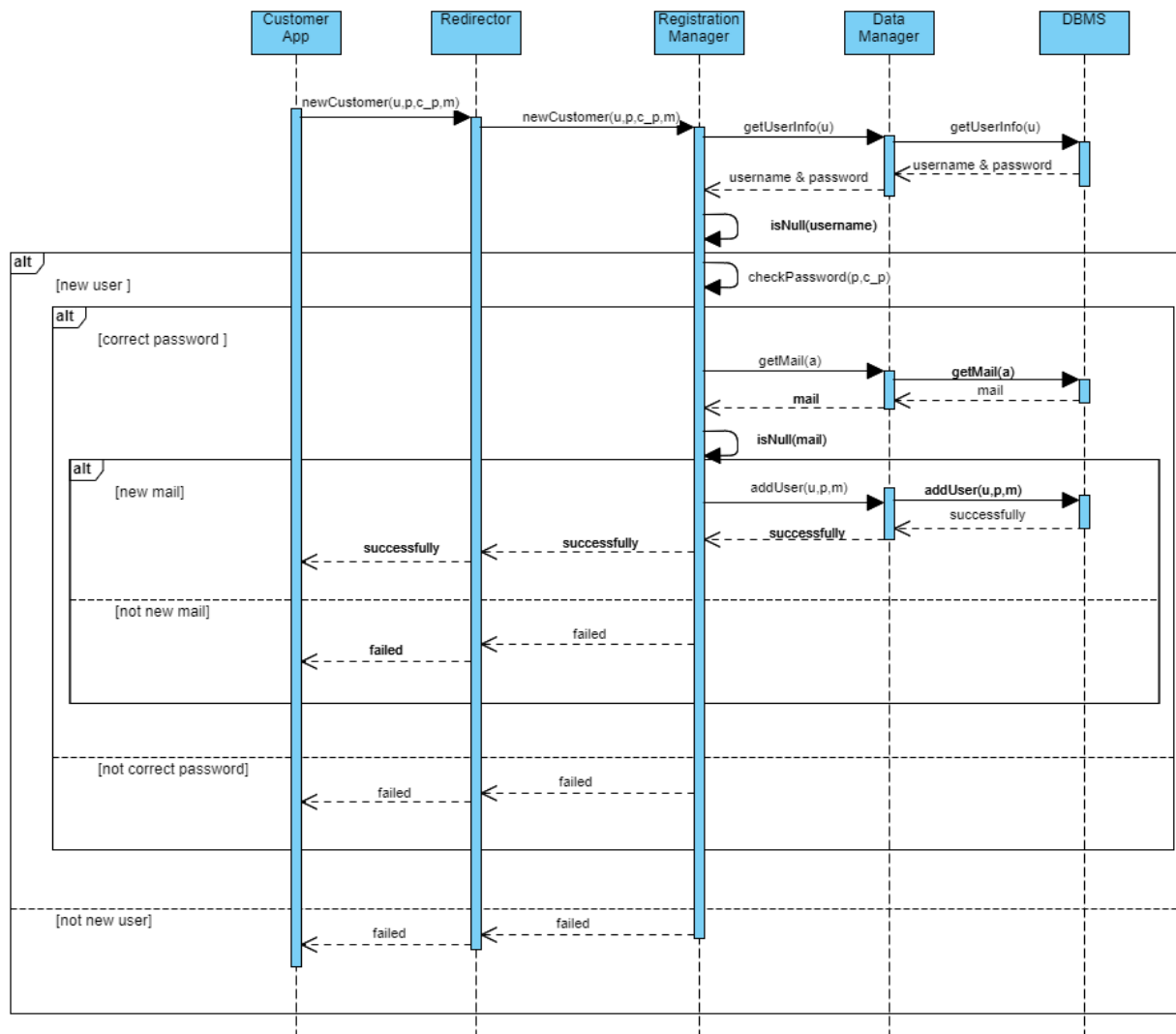


Figure 8: Customer Registration Runtime View

**Line Up Function ([link](#)):** In this sequence diagram (Fig. 9), the Line Up function, in the case of a customer using a mobile application is explained. The diagram has the fact that the customer is already logged in as an entry condition.

The Customer App asks the server for a lineup ticket. The Redirector receives the request and forwards it to the Service Handler, which sends it to the LineUp Manager. It checks if the customer is already in line asking it to the Data Manager, if it is the case the LineUp Manager cancels the request and sends an error message. Otherwise, the process that allows the customer to get in line starts.

At first, the LineUp Manager component uses the retrieved position of the customer to look for nearby supermarkets thanks to the Map Manager. At this point, the customer replies to the server with a message containing the chosen supermarket. The request is forwarded to the LineUp Manager through the same components of the previous customer's request. After receiving the request, the LineUp Manager sends another request to the Map Manager to estimate the correct time that the customer has to use to reach the supermarket. Then, it asks the Supermarket Manager for the customer number and the current number of people in the supermarket. Successively it asks the Waiting Time API to calculate the waiting time that the customer will have to wait before entering the supermarket.

Finally, the LineUp Manager asks the Ticket Manager to generate the ticket passing it the various information received from the other components (user, supermarket, estimated time, reaching time, line number). The Ticket Manager component asks a QR code to the QR Code Generator API. Once the ticket is received, the Ticket Manager sends it to the customer (obviously, the ticket before arriving at the customer must be forwarded to all components shown in the picture). It is important to note that a copy of the ticket is stored in the database for future computations.

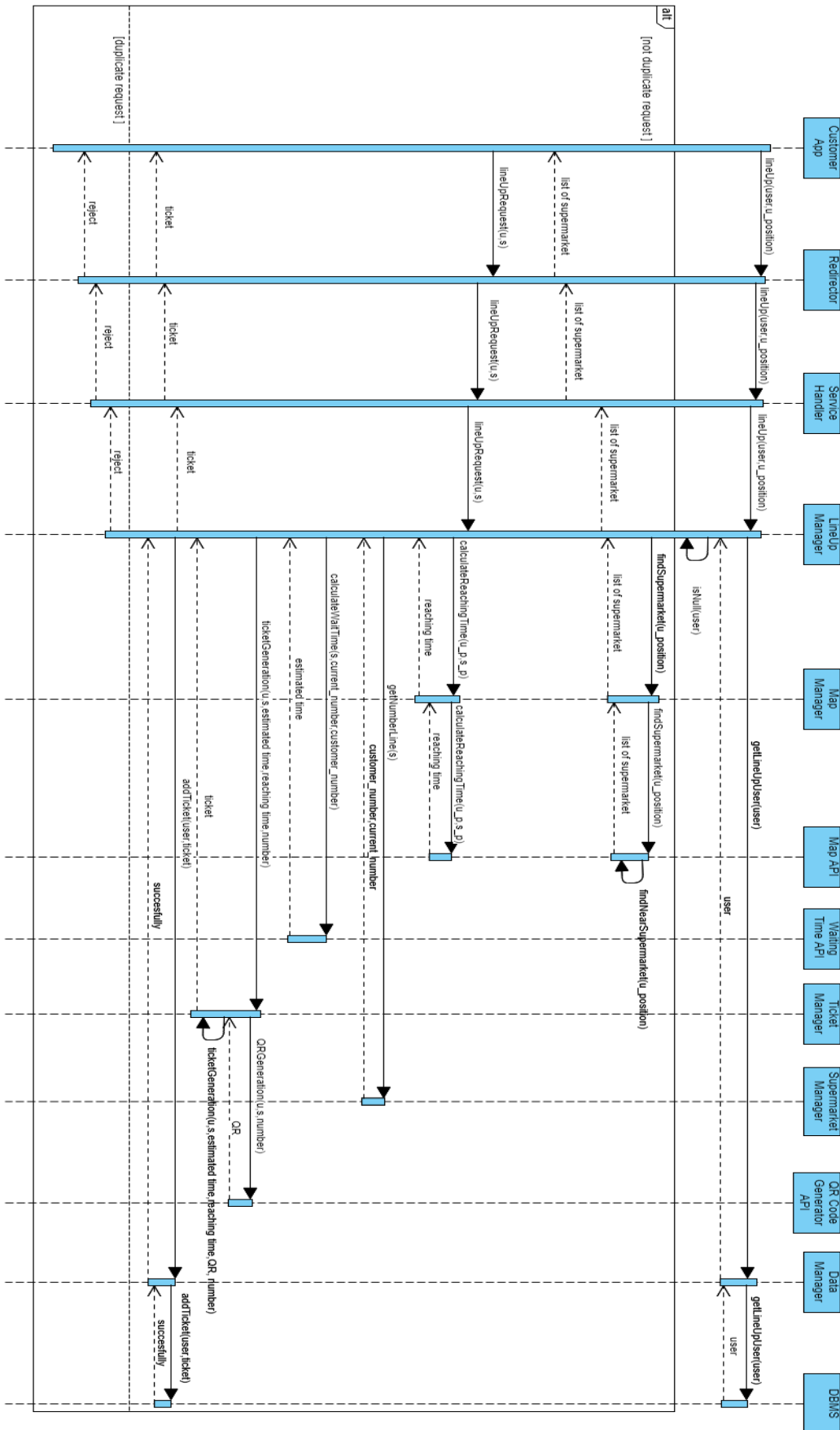


Figure 9: Line Up Function Runtime View

**Book A Visit Function** ([link](#)): In this sequence diagram (Fig. 10), the book a visit function is explained. The diagram has the fact that the customer is already logged in as an entry condition.

The Customer App asks the server for a reservation. The Redirector receives the request and forwards it to the Service Handler, which passes it to the Reservation Manager.

At first, the Reservation Manager component uses the retrieved position of the customer to look for nearby supermarkets thanks to the Map Manager. This list is sent to the client. The client chooses the preferred supermarket, and the server asks him for the reservations' date and time.

At this step, the customer sends a complete reservation request (the request contains the username, the preferred supermarket, the time and data, and optionally the expected duration of the visit). The request passes all the previous components up to the Reservation Manager. This component asks the Data Manager if the customer is already booked for that time slot. In this case, he is not allowed to book again. Otherwise, it checks if the customer has inserted the duration of the visit. In the affirmative case, it proceeds to verify that the slot is still free. To perform this job, it has to interact with the database.

If the selected slot is not full, the Reservation Manager asks the Ticket Manager to generate the ticket passing to it the various information received from the other components (slot which contains time, data and supermarket, duration, user). The Ticket Manager component asks a QR code to the QR Code Generator API. After receiving the code, the Ticket Manager sends the ticket to the customer (obviously, the ticket before arriving at the customer must be forwarded to all components shown in the picture).

Suppose the selected slot is no longer available: the customer is advised to add the shopping list or choose any alternative time slots or to change the supermarket with one of the suggested (features are shown in the diagrams below). It is important to note that a copy of the ticket is stored in the database for future computations, like the notification function.

Suppose the customer has not inserted the duration: the Reservation Manager asks the Duration Estimation API if he is a long-term customer. In that case, it sends an estimated duration and the process continues as in the previous case. Otherwise, a message is sent to the customer to inform him about the duration's lack.

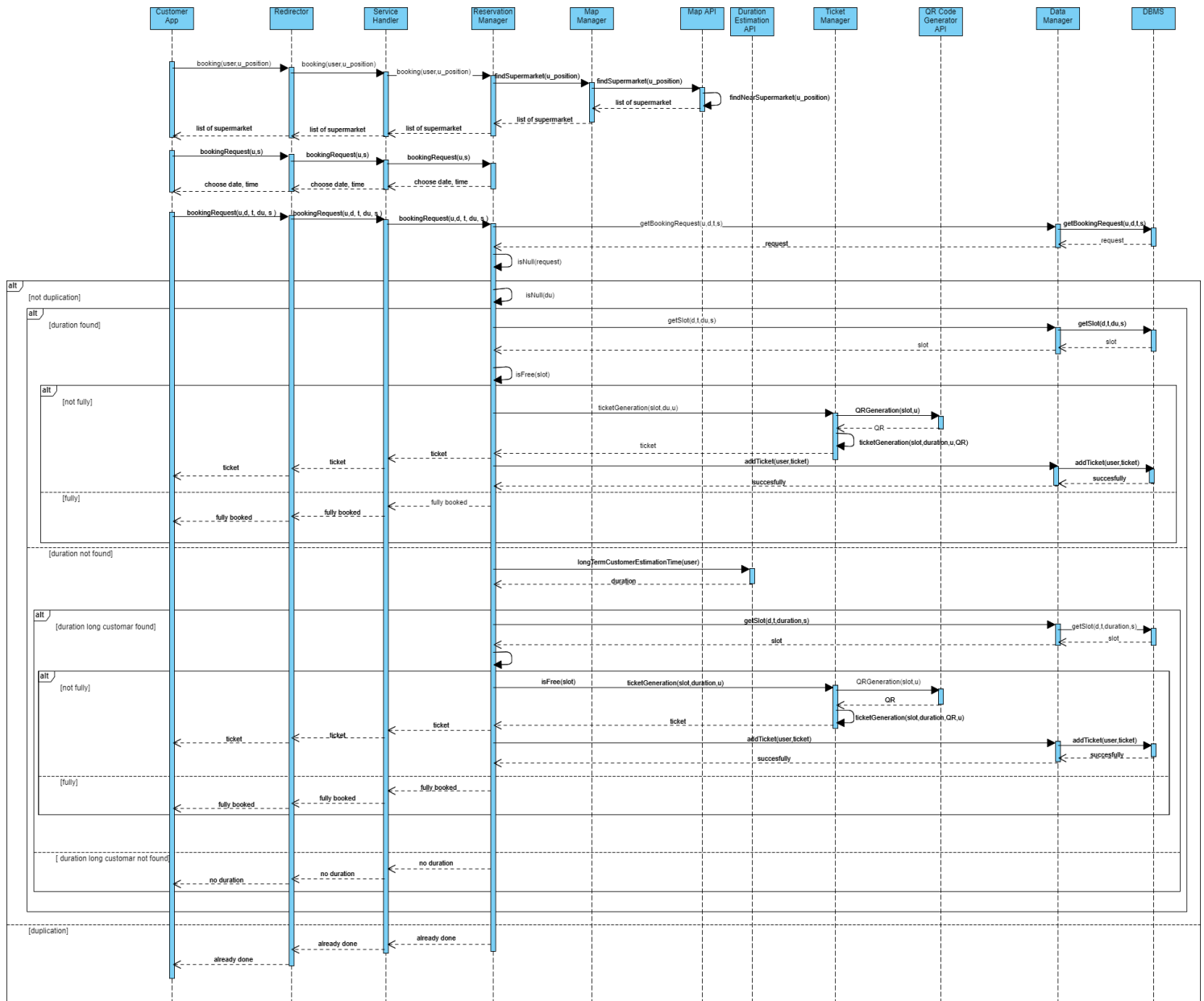


Figure 10: Book A Visit Function Runtime View



**Insertion of the Shopping List ([link](#)):** In this sequence diagram (Fig. 11), the shopping list insertion function is shown. The diagram has two entry conditions:

1. The customer is logged in.
2. The customer has just inserted the booking date and time.

The Customer App asks the server the possibility of adding a shopping list. The Redirector receives the request and forwards it to the Service Handler, which sends it to the Reservation Manager. It interacts with the Shopping List Manager component that asks the Data Manager to retrieve the various categories that the supermarket has and sends them to the client. If the customer does not want to insert an accurate shopping list, sends back the list of the chosen categories and the server responds with a message that informs him the list of categories is successfully stored. Otherwise, the server retrieves a list of the products belonging to the inserted categories and sends it back to the customer.

At this point, the customer sends the list of the chosen products to the Shopping List Manager component and this information is stored in the case of a possible use by the Internal Crowd Estimation System. Once done, a confirmation message is sent to the client.

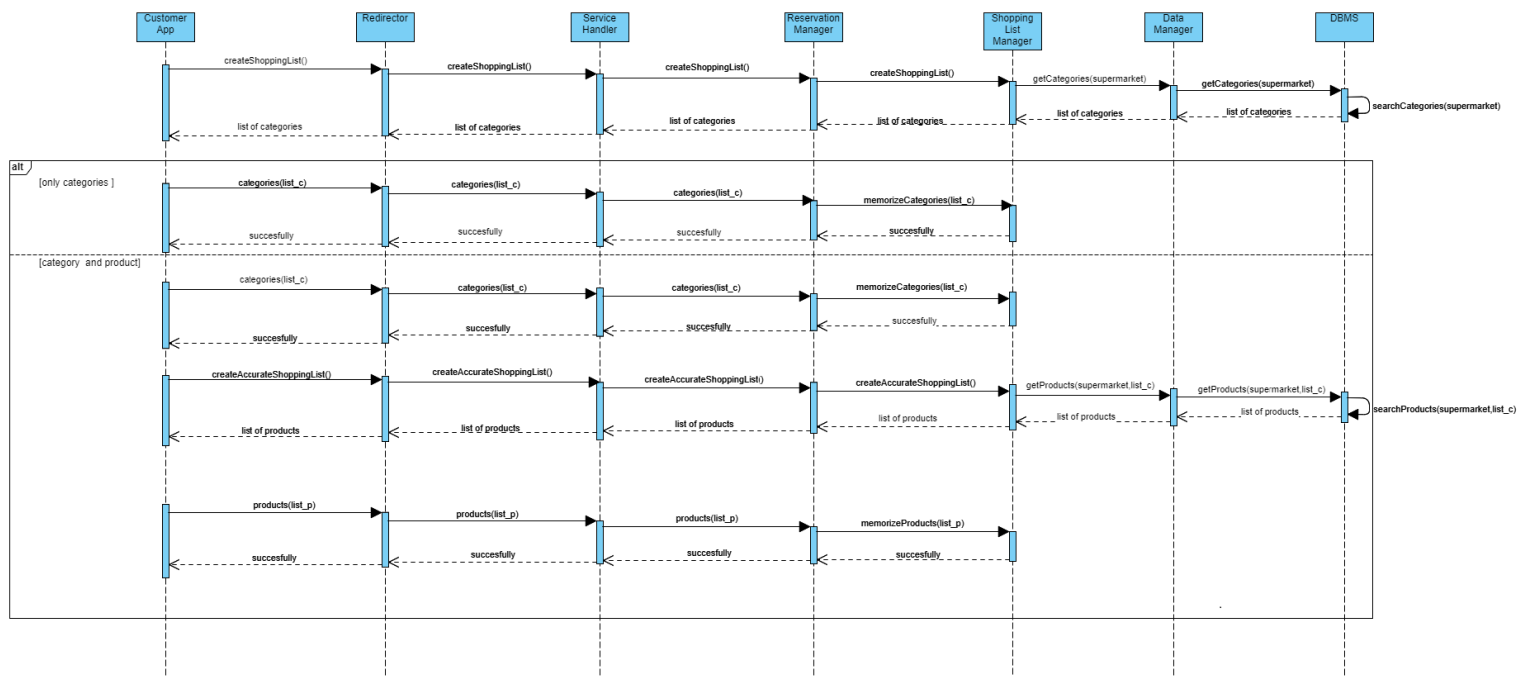


Figure 11: Insertion of the Shopping List Runtime View

**Alternatives Suggestion (link):** In this sequence diagram (Fig. 12), the alternative suggestion function is shown. The diagram has three entry conditions:

1. The customer is logged in.
2. The customer has sent a Book A Visit request without a shopping list or a shopping list, but the check crowding was unsuccessful.
3. The store is fully booked for the chosen time slot.

The Customer App asks the server for an alternative solution to get a reservation ticket. The Redirector receives the request and forwards it to the Service Handler, which sends it to the Reservation Manager. This last component interacts in two cases with the Advice Manager component.

The first one is to ask for an alternative slot. The Advice Manager component asks the Data Manager to find the free slots in the supermarket in which the customer had previously requested a booking. The second case asks for an alternative supermarket related to the first one. The request procedure is similar to the slots ones. The only difference is that the Data Manager has at its disposal the supermarket's position instead of the supermarket's name to return the supermarkets in the nearby. This last alternative is chosen by the client in the case in which the slots suggested are not compatible with the customer's needs.

If the suggestion fits the customer's needs, the Reservation Manager will proceed with the reservation function shown in the **Book A Visit sequence diagram**. If this is not the case the request is rejected.

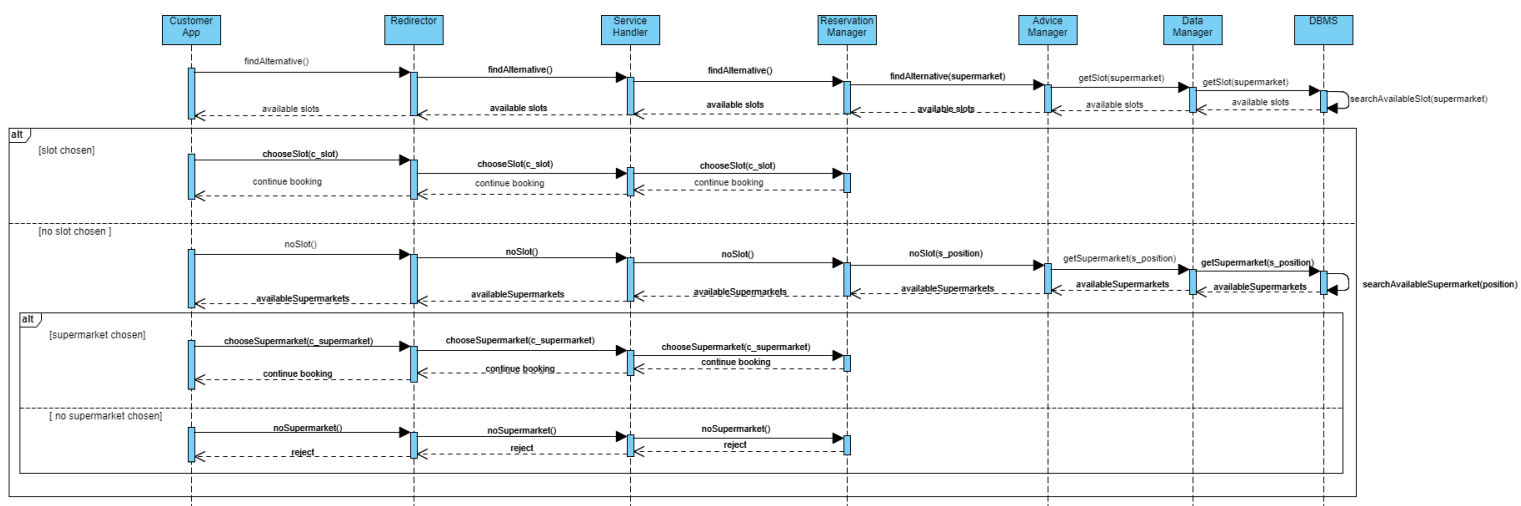


Figure 12: Alternatives Suggestion Runtime View

**Turn On Periodic Notifications Function:** In this sequence diagram (Fig. 13), the process of turning on periodic notifications is shown. The diagram has the fact that the customer is already logged in as an entry condition.

The Customer App asks the server to change the notification's permission. The Redirector receives the request and forwards it to the Service Handler, which sends it to the Notification Manager. This last component asks the Data Manager to update the permission of that customer. A message is sent to the customer to signal him that everything went well.

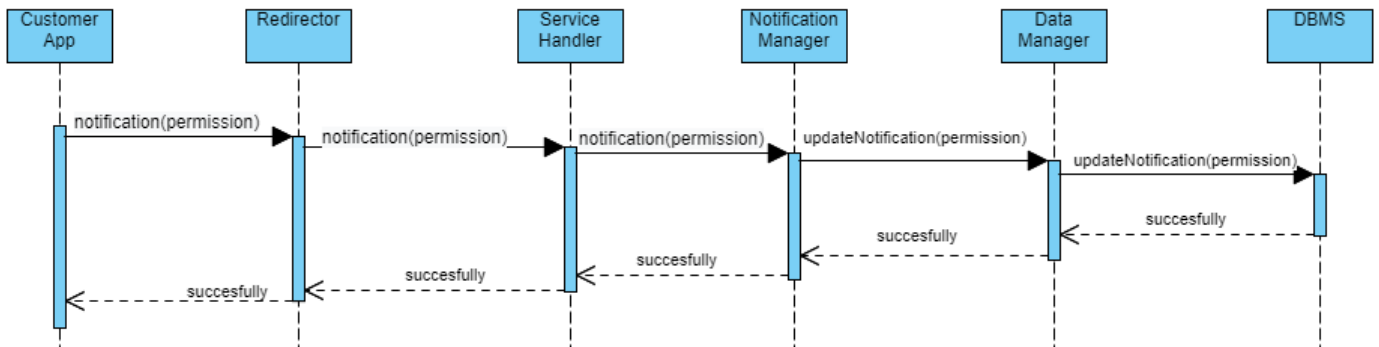


Figure 13: Turn On Periodic Notifications Function Runtime View

## 2.4.2 Manager

**Login ([link](#)):** In this sequence diagram (Fig. 14), the process of manager login is shown. The request, that contains only username and password, is sent by the manager using the Manager WebApp and then it passes through the Web Server that interacts with the Redirector. This last component receives the request and forwards it to the Access Manager component. It asks the Data Manager if the managers' username and password are in the database. If there is no manager with that username in the database, the manager receives an error message. Otherwise, the check of the password is performed. If the password is correct, the Access Manager sends an OTP request to the OTP Manager; it generates the OTP code and then asks the Data Manager to retrieve the manager's mail to send him the code thanks to the mail service.

If there is no manager with that username in the database, the manager receives an error message. Otherwise, the check of the password is performed. If the password is correct, the Access Manager sends an OTP request to the OTP Manager; it generates the OTP code and then asks the Data Manager to retrieve the manager's mail to send him the code thanks to the mail service.

To complete the login, the manager has to enter the OTP. This password will be forwarded through the other components until it reaches the OTP Manager to verify the correctness and allow the manager to login.

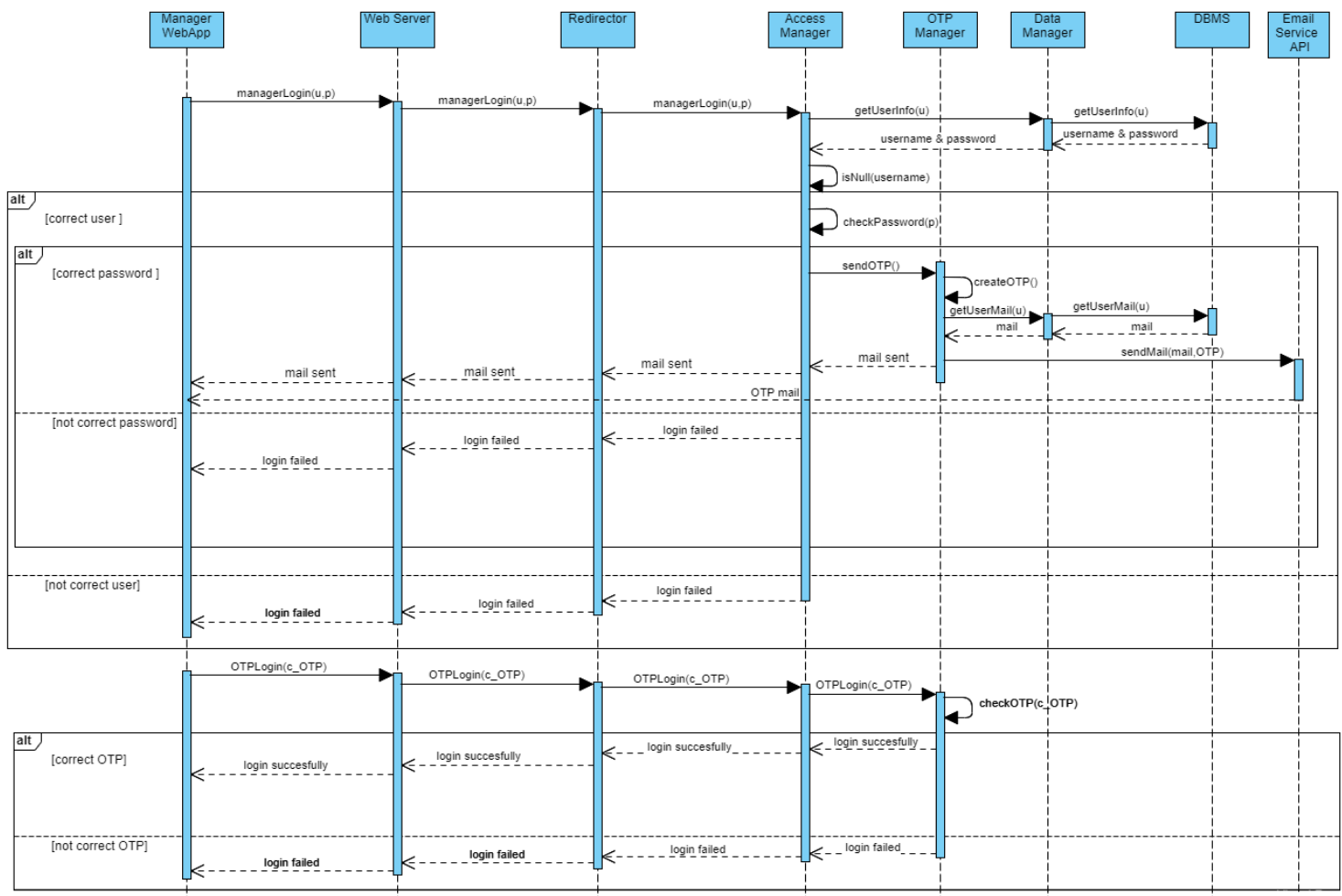


Figure 14: Manager Login Runtime View

**Registration ([link](#)):** In this sequence diagram (Fig. 14), the process of manager registration is shown.

The manager registration is like the customer one, but the request came from the Manager WebApp to be then passed to the Web Server that interacts with the Redirector.

There are two other differences:

1. The manager must insert an email.
2. There is another check regarding the shop ID to allow the manager to register for a specific supermarket.

It is possible to register also with multiple IDs, the diagram below represents the situation with only one ID, in that case all the IDs are checked by the system.

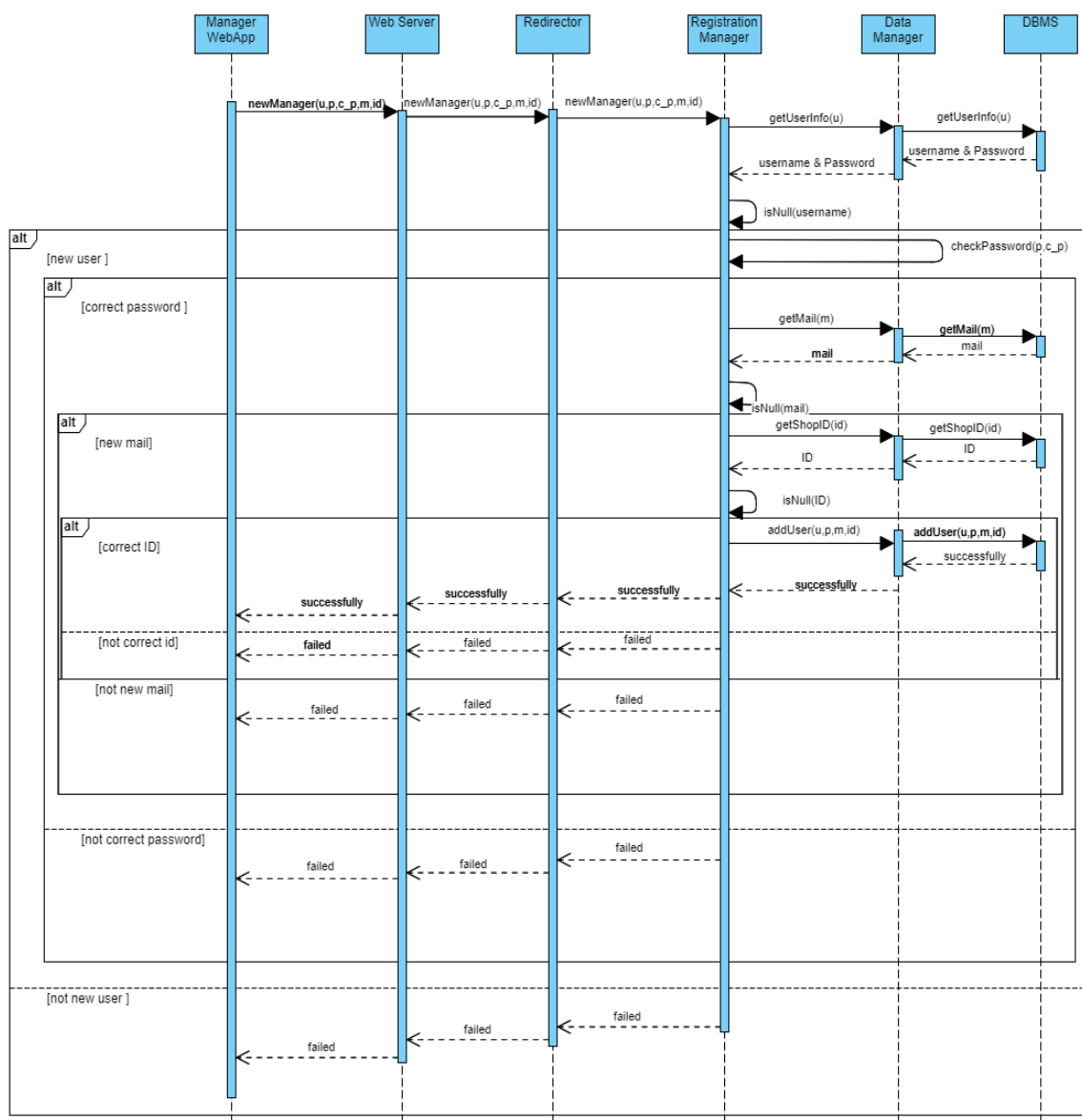


Figure 15: Manager Registration Runtime View

**Regulate The Influx Of People Function:** In this sequence diagram (Fig. 16), the process of changing the capacity of the supermarket is shown. The diagram has the fact that the manager is already logged in as an entry condition.

The request, which contains the new maximum number of people allowed to stay in the store and the type of service for which capacity change is required (book a visit or line up), came from the Manager WebApp to be then passed to the Web Server that interacts with the Redirector. The Redirector receives the request and forwards it to the Management Manager component, which passes it to the Supermarket Manager that asks the Ticket Manager to update the capacity.

At the end, the Supermarket Manager component asks the Ticket Manager to change the capacity in the QR Code Reader System. Once the capacity is updated, a message is sent to the manager to inform him that everything went well.

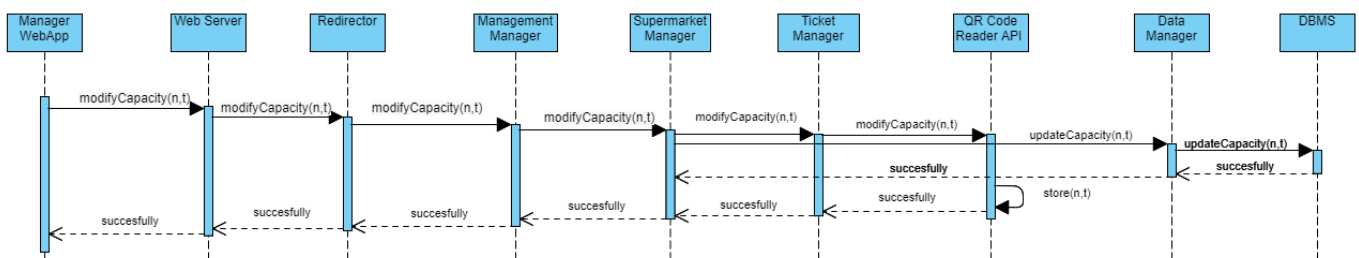


Figure 16: Regulate The Influx Of People Function Runtime View

## 2.5 Component Interfaces

The interfaces of the components of the application server and their methods are illustrated in Figure 17. For each component are illustrated the most critical methods to obtain the CLup functionalities shown in the previous section (Runtime View). For simplicity and better readability of the picture, the interfaces of the internal components of BookAVisit Manager and Login Manager are not shown.

It is also essential to point out that the methods, attributes, and interfaces listed in the figure are only a skeleton for the application's proper programming. In the implementation phase, it is possible to add others or modify the present ones to make them more efficient and easier to manage. At this ([link](#)) it is possible to find a picture with a better quality.

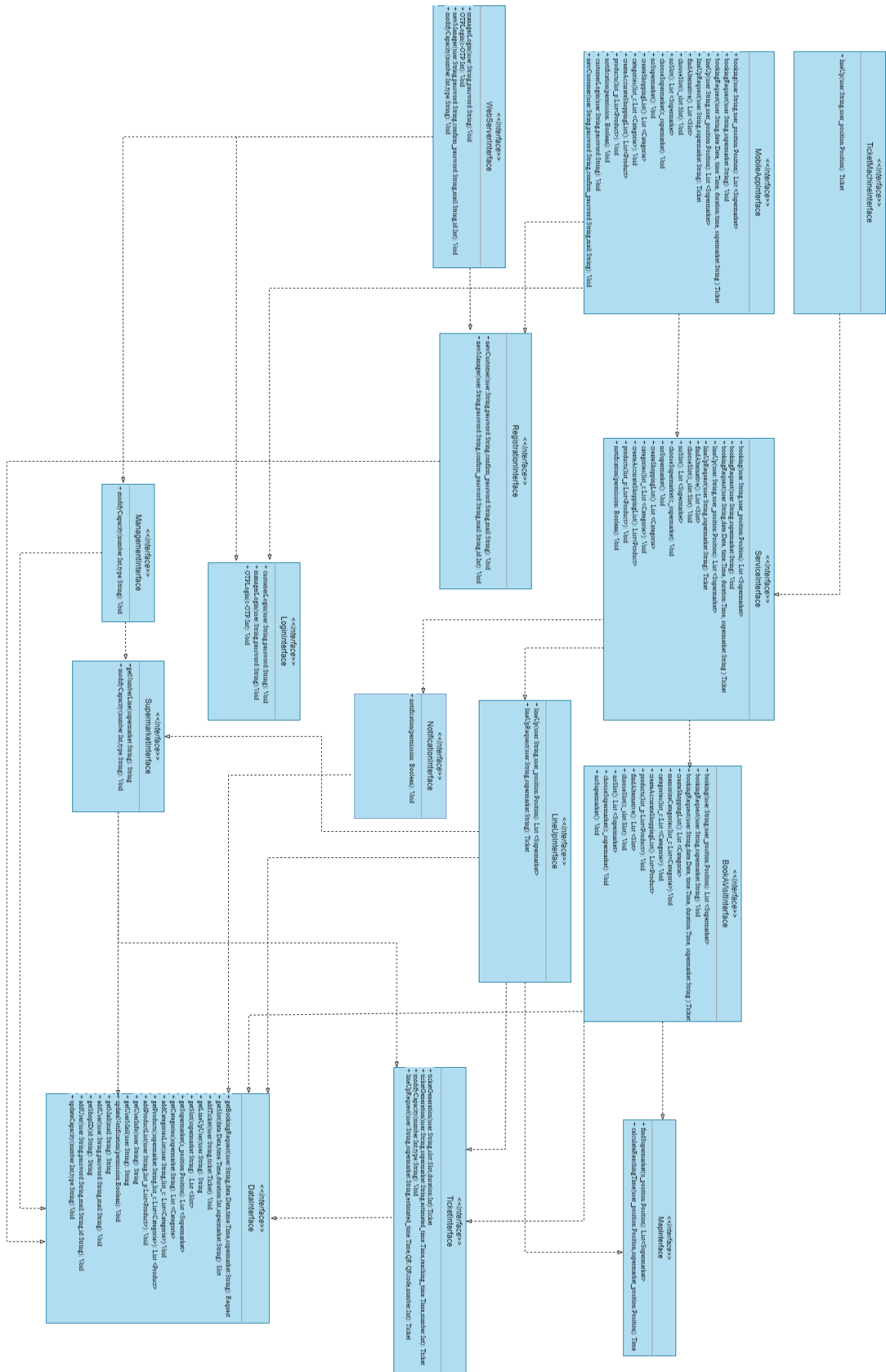


Figure 17: Component Interfaces Diagram



**MobileAppInterface:** It is the interface that allows the Customer App to connect with the Redirector. The only purpose of the methods of the interface is to forward requests to the right part of our system to handle them in the correct way.

**WebServerInterface:** It has the same functionality as the MobileAppInterface with a unique difference: it connects the Web Server to the Redirector. It is important to note that since the manager may be responsible for more than one supermarket, another "newManager" method could be added, replacing the integer id with a list of integers.

**TicketMachineInterface:** It is the interface that allows the Ticket Machine to connect with the Redirector. The only purpose of the interface methods is to forward requests to the right part of our system in order to handle them correctly. Note that for the ticket machine, the only functionality allowed is the LineUp request.

**ServiceInterface:** This interface is useful to better split the customer request. It connects the Redirector with the Service Handler. The methods can redirect the request to the component that provides the customer's service (service related to the LineUp, Book a Visit, or Notification function).

**ManagementInterface:** It is the interface whose method allows the manager to change some application parameters, for example, the LineUp and Book a Visit capacity. It connects the Redirector with the Management Manager.

**SupermarketInterface:** It is the interface whose methods allow changing some application parameters related to the supermarket. This interface also has a method `getNumberLine(Supermarket: String)` given as a parameter the supermarket requested by the client returns a string containing the number of people the customer has in line before him plus one and the current number (number of the customer just entered). As we can imagine, this interface connects both the Management Manager Component and the LineUp Management Component with the Supermarket Manager Component.

**MapManagerInterface:** The methods inside this interface are essential for the correct functioning of the system for the obtainment of the information related to the positions. This interface connects both the LineUp Manager Component and the BookAVisit Manager Component with the Map Manager Component.

**BookAVisitInterface:** It is the interface whose methods allow the reservation function. This interface connects the Service Handler with the BookAVisit Manager Component. It's important to point out that this interface could be the father of other interfaces, at least one for each BookAVisit Manager sub-component.

**LineUpInterface:** It is the interface whose methods allow the LineUp function. This interface connects the Service Handler with the LineUp Manager Component.

**NotificationInterface:** It is the interface whose methods allow the notification function. This interface connects the Service Handler with the Notification Manager Component.

**TicketInterface:** It is the interface whose methods allow ticket management. This interface connects the LineUp Manager Component, BookAVisit Manager Component, and the Supermarket Manager Component with the Ticket Manager Component.

**DataInterface:** This interface connects all the components that need to access the database with the Data Manager component. This component interacts directly with the database and queries/adds to the database what the previous components have asked. In this interface it is important to notice that the method `getUserInfo(user: String)` returns a string containing username and password.

**LoginInterface:** It is the interface whose methods allow the customer and the manager's login with different methods. This interface connects the Redirector with the Login Manager Component. As for the BookAVisitInterface, this interface could be a father for other two interfaces, one for the Access Manager Component and another for the OTP Manager Component.

**RegistrationInterface:** It is the interface whose methods allow the registration of both the customers and the managers in a different way. This interface connects the Redirector with the Registration Manager Component.

## 2.6 Selected Architectural Styles and Patterns

In this section we are going to explain and justify all the design decisions made to build our software architecture. As already mentioned in the previous section (Section 2.3), we have decided to apply a Client/Server application distributed on a four-tier architecture. We will also clarify how devices communicate with each other by explaining the different types of connection we decided to apply.

### 2.6.1 Client & Server

The architectural style adopted is a **Client/Server** architecture. This decision is mainly because CLup needs a central server that is active for receiving requests from all the clients connected, in this way it can manage different requests from clients that want to benefit from all functionalities that the system can provide. It can also manage multiple requests from different clients at the same time.

This type of architecture also facilitates the interactions between them. Client and Server have both a well-defined interface through which they communicate with each other across the Internet network: a client starts the communication channel and asks for a request, whereas the system accepts this new request and it takes it in charge. Moreover, we have three different client-sides: a Web Application for managers, a Mobile Application for customers and a Ticket Machine for customers without the application as we have already said and by applying a simple Client/Server architecture we can simply manage these three different types of interfaces.

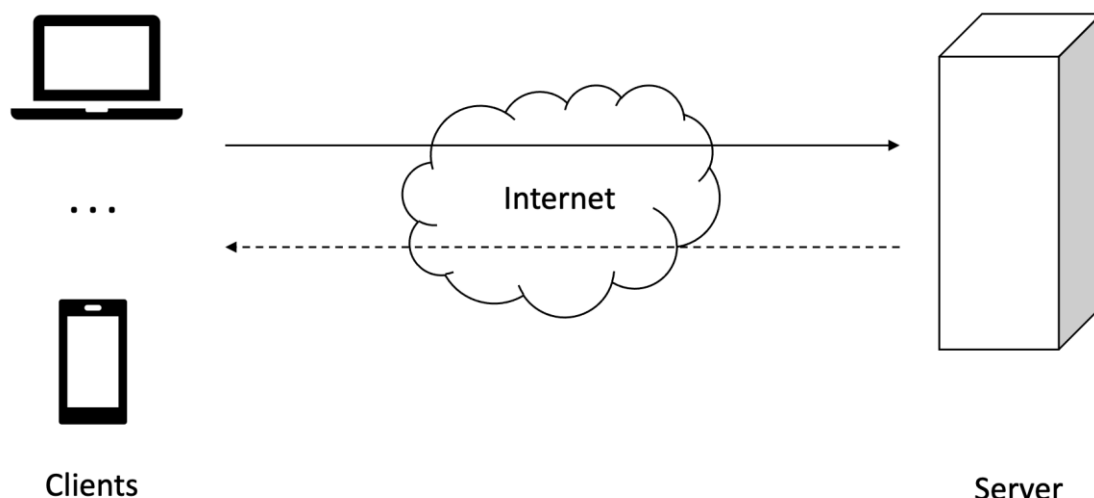


Figure 18: Client/Server application

## 2.6.2 Four - Tier Architecture

We have decided to apply a Client/Server application distributed among a **Four-Tier architecture** approach. The decision of four tiers arises from the need to have a Web Server. As we have already mentioned in Section 2.3 we need a design element capable of managing and transferring web pages to the client-side of the manager that interacts with a Web Browser in order to communicate with the system.

We have chosen a multilayered architecture in order to guarantee all the resulting properties: decoupling and scalability. Decoupling between application logic and data logic, but also between presentation logic and application logic. Scalability is intended in terms of ability of the infrastructure to meet growing demands from users with adequate updates.

We have adopted a **thin client** instead of a fat client (or thick client) because we prefer that the client need only to incorporate the presentation logic. Indeed, in our case the client does not require significant computing power as it is limited to interpreting and showing HTML pages for example in the case of a manager client. This type of solution, which has the advantage of working even with limited resources, requires a stable connection between the client and server side in order to respond to all user requests. In this situation, it is more convenient to centralize the application logic on an intermediate server. In this way it is sufficient to guarantee the robustness and efficiency of the intermediate server, which is shared between the different clients. This also allows the system to reduce the load on the DBMS server. The middle tier in fact maintains permanent connections with the DBMS. Furthermore, the number of open sessions between the intermediate tier and the DBMS is generally lower than the number of open connections between the clients and the intermediate tier. The server is sized so as to guarantee a predetermined level of performance for a certain number of simultaneously active users, a number which it is reasonable to assume lower than the total number of users who have access to the information system.

The following picture shows how layers are distributed among different subsystems and they are highlighted with the same colors adopted so far in order to distinguish Client, Server, Web Server and Data Server.

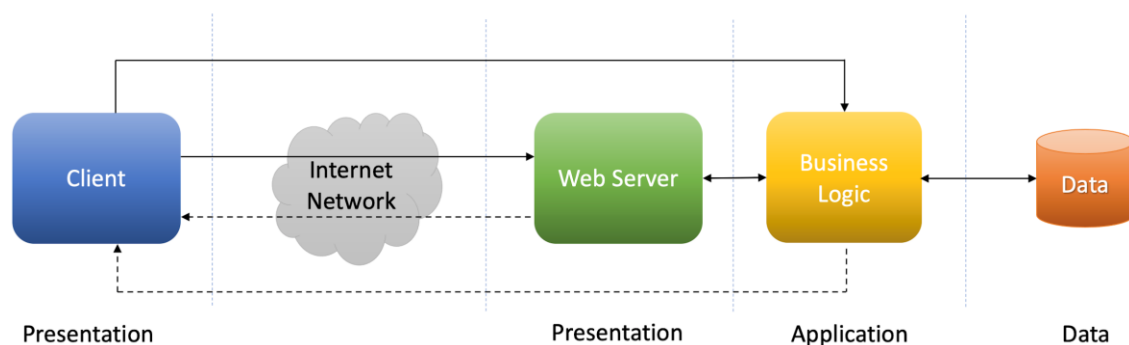


Figure 19: Four-Tier and multilayered architecture

The four-tiers we can see in the picture are:

- **Client:** the client-side is composed of three types of application that manages the interactions between clients and the system that allows the access to all of its functionalities.
  - Customer App
  - Manager WebApp
  - Ticket Machine
- **Web Server:** this architectural component allows the Manager WebApp to communicate with the system through the Web Interface. It is able to manage requests for transferring web pages of a client.
  - Web Server
- **Business Logic:** this is the central layer that incorporates all the application logic of the system. It is composed of all the components that are needed to provide the functionalities of the software. All components listed below are all the components inside the server and previously described.
  - Login Manager
  - Registration Manager
  - Redirector
  - Service Handler
  - Management Manager
  - Notification Manager
  - BookAVisit Manager
  - LineUp Manger
  - Map Manager
  - Ticket Manager
  - Supermarket Manager
  - Data Manager
- **Data:** this is the data layer that provides all the services needed for the data management by the server.
  - DBMS

As we can see in the Figure 19 presentation logic is distributed among the clients and the Web Server, the application logic is distributed among the Business component, whereas the data layer is completely centralized in the DBMS.

### 2.6.3 RESTful Architecture

Now we are going to justify the type of connection between external components. As we can see from the Deployment Diagram (Fig. 6) we have established a HTTPS connection between the manager client and Web Server, the system and the Web Server, the system and the Ticket Machine and also between the system and the external APIs. The term REST represents a system of transmission of data over HTTP. This design choice comes from all the design decisions already explained; indeed, a **RESTful architecture** is useful and coherent with the design strategies described and applied so far. This way of invoking services has advantages in terms of performance, modifiability and simplicity. REST systems do not have the concept of session, they are stateless. This helps the central server to manage a huge number of clients and a huge number of requests.

## 2.7 Other Design Decisions

In this section we want to give further explanations about other design choices for our software. In the previous section (Section 2.6) we have already explained what kind of architecture, how many tiers are involved and what type of connections they are linked with. Now we want to focus on the DBMS: we have adopted a relational DBMS and the reasons why are clarified in the next subsection.

### 2.7.1 Relational Database

For what concerns the decisions about the DBMS we have adopted a **Relational DB**. This implementation choice is given only by the simplicity of management of a relational database. This model represents the data through the use of tables and allows queries to be carried out using well-known and simple query languages based on SQL. Furthermore, the main advantages of using a relational database are:

- Simple data model: relational databases rely on a relatively simple data model to implement and manage. Thanks to relational database models, it is easy to retrieve the numerous information contained inside, such as user data or stored requests.
- Reduced data redundancy: thanks to the normalization relational databases provide well-defined rules for eliminating and avoiding data redundancy. In this way it is easier to maintain and manage a large amount of data.
- High data consistency: normalized relational databases allow to avoid data with contradictions. Moreover, relational databases systems also provide rules that automatically define and verify integrity constraints. All these features guarantee a high level maintenance of data consistency.

In the following picture we summarize all the design choices adopted for our CLup software through the Design Space.

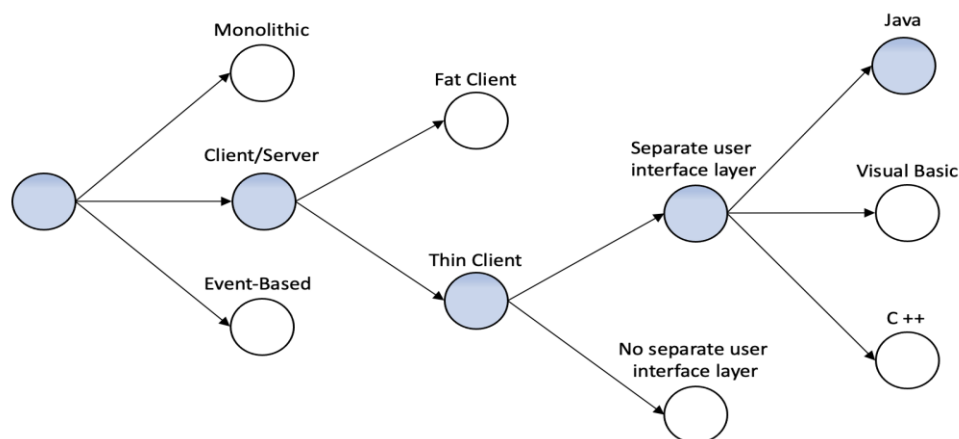


Figure 20: Design Space

### 3 User Interface Design

In order to explain how the key functionalities will be given, this section describes precisely the mockups already presented in the Requirements Analysis and Specification Document. The emphasis of the first subsection is on the Usage of two diagrams to demonstrate the flow of both mobile and web apps and how all the different functionalities are presented to the customer.

In the following subsections will be presented separately the mockups related to the customer's functionalities and the ones related to the manager's functionality.

These mockups are intended to describe specifically how customers can communicate with the system to take advantage of its functions (Line Up, Book A Visit, Notification and Manager Function).

The design of the mockups is still simplified not to restrict the developers' creativity too much and, in addition to that, to be compatible with both iOS and Android systems.

Obviously, the paradigm used for the communication between the applications and the system is a Client-Server one, as already explained in the previous sections. All the business methods are executed on the server side while the management of the graphical interface is assigned to the Client side.

#### 3.1 UX Diagrams

In the following part two different diagrams will be presented.

The first one (Fig. 21) is related to the different possible flow of events inside the customer application. It is quite complex and, in case of poor readability it is available at this [link](#).

In this diagram all the functionalities are analyzed and all the mockups presented in the Requirements Analysis and Specification Document are connected highlighting the interactions between the different screens.

In the second diagram (Fig. 22), it is possible to see how the different screens of the web applications, dedicated to the managers, are connected.



### 3.1.1 Customer Application Flow Diagram

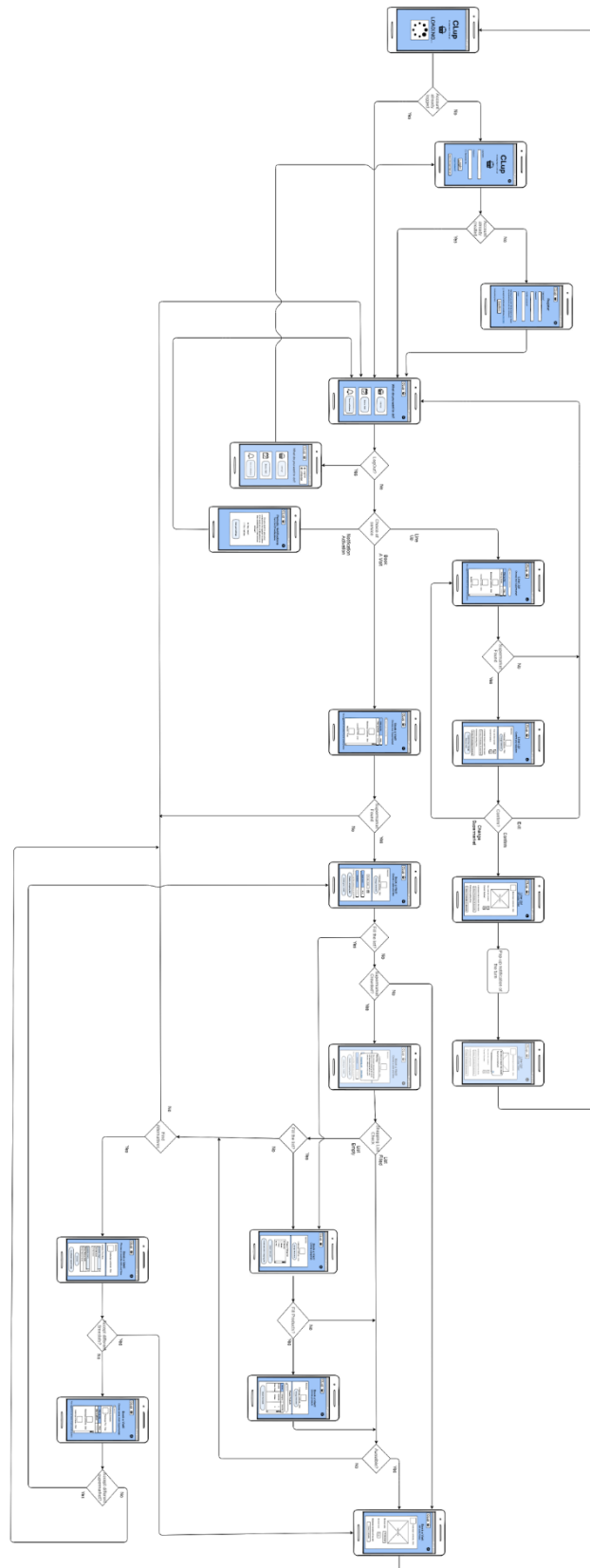


Figure 21: Customer Mobile Application UX Diagram

### 3.1.2 Manager Application Flow Diagram

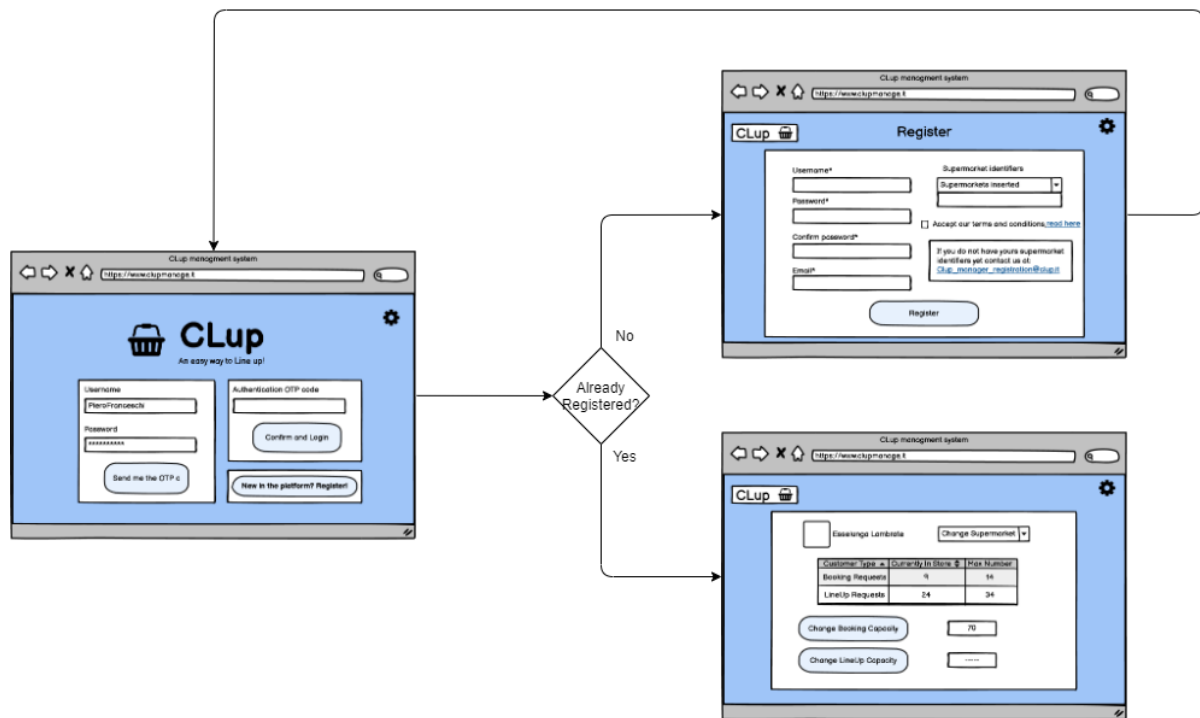


Figure 22: Manager Web Application UX Diagram

## 3.2 Customer Application

In the following subsection, the different screens of the mobile application will be presented. All of them are briefly explained to better understand their functioning. Obviously, as regards the “esthetic design” choices, they are only guidelines for the developers. One important thing to take into account is the fact that the customer can be a person of every age and the interface must be intuitive and clean. The graphical interface has to adapt its size to the device in which the application is in use, granting that the size of the QR Code present in the ticket remains correct and readable from the External Reading Service.

### 3.2.1 Login

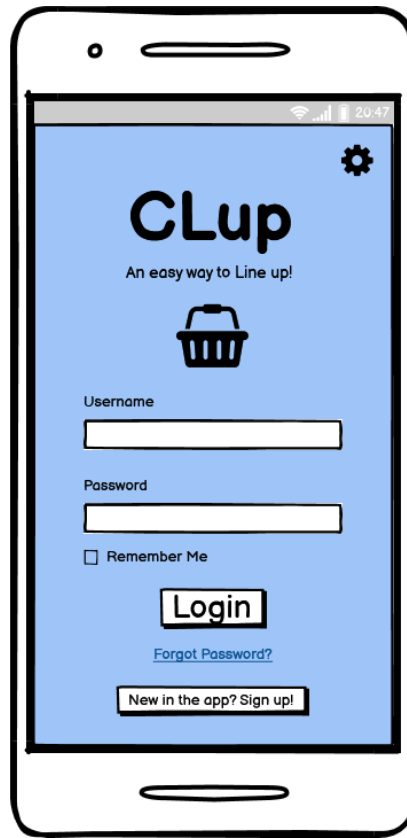


Figure 23: Customer Login

In this picture (Fig. 23) it is possible to observe the first page that a customer opening the application must see (obviously if it is not considered the case of an already logged-in customer with the “Remember Me” option can be activated).

In the case of a first access to the application the customer has to push the button where is written “New in the app? Sign Up!” in order to create an account and, in that case, the customer moves to the screen described by the next mockup (Fig. 24).

If the customer is already registered, he needs to login in order to access the application, he has to insert his “Username” and “Password” in the correct fields. In the unlucky case of a forgotten password he can push the link “Forgotten Password” for reobtaining it but only if during the registration he has also inserted the email. In that case a new password will be sent to his email and the customer can access the application.

The “gear button” in the top-right of the screen can be a useful addition for the management of the language switch and other options of the application. In this document this feature will be not explained in detail but in that case the “gear button” is an example of a possible additional feature implementation.

### 3.2.2 Registration

CLup

Register

Username\*

Password\*

Confirm password\*

Email

Warning: if you don't fill this field if you forgot your password you can not retrieve it.

☐ Accept our terms and conditions, [read here](#)

Confirm

\*compulsory field

Figure 24: Customer Registration

In this picture (Fig. 24) it is possible to observe the page dedicated to the creation of an account. In this page minimal information is required to the customer, in order to permit all the customers to use the application. For this reason, the only compulsory fields are the “Username”, “Password” and “Confirm Password”.

Obviously in the “Username” field the user has to insert a unique username. In the “Password” and “Confirm Password” fields the user has to insert two times the same password, in order to avoid errors in typing.

The “Email” field is not compulsory, and it is used for the reset of the password in the case of a forgotten one.

At the end of the page is also compulsory to accept the “Terms and Conditions” of the application. It is possible also to read them using the link present in the “read here” written part. The last step is pushing the “Confirm” button that confirms the registration and permit to store all the information in the system.

### 3.2.3 Home Page

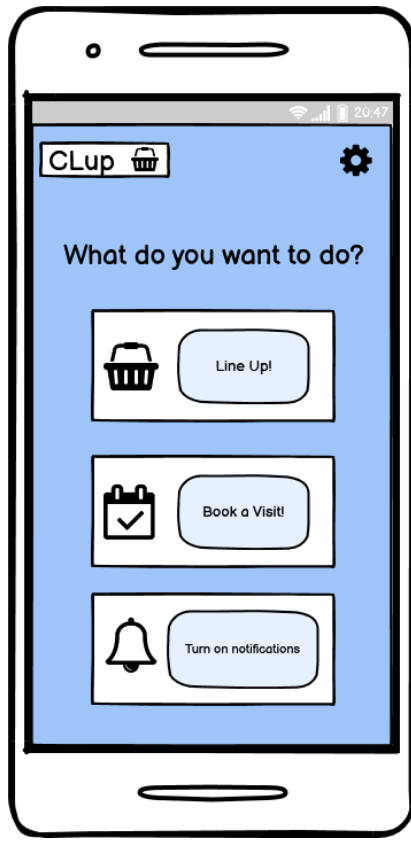


Figure 25: Home Page

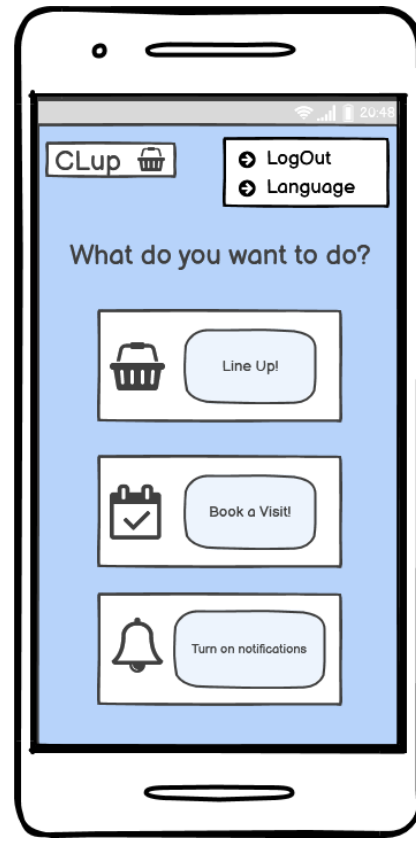


Figure 26: Home Page (LogOut)

In this picture (Fig. 25) it is possible to observe the Home Page of the application. From this page it is possible to choose the service required by the customer or turn on the periodic notification function.

The design is minimal in order to be user-friendly and there are only three buttons with a correlated picture.

- The first one is used to access the **LineUp Service**.
- The second one is used to access the **BookAVisit Service**.
- The third one is used to turn on the **Periodic Notifications**.

One possible addition in this page, related to the “gear button”, is the possibility to logout using it. In that case the customer is moved to the login page.

### 3.2.4 Supermarket Selection (LineUp)

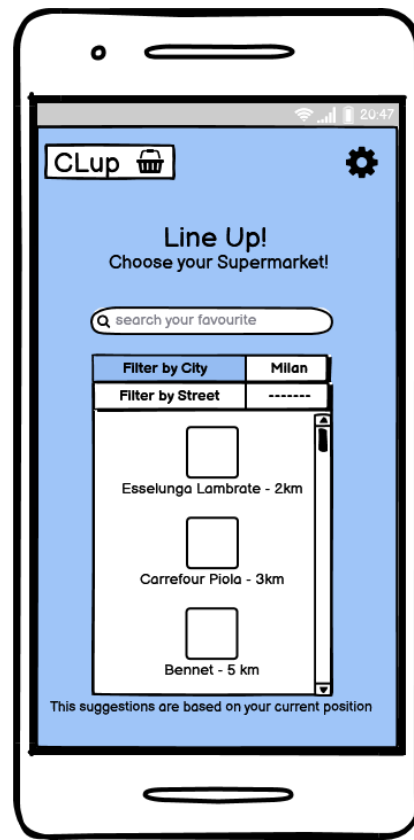


Figure 27: Choice of Supermarket (LineUp)

In this picture (Fig. 27) it is possible to observe the selection of a specific supermarket during the use of the “Line Up” functionality.

In this page it is possible to search for a specific supermarket using the “search bar” in the upper part of the page. It is also possible to filter the different supermarkets by city or, for a more accurate filtering, by street. This second case is more useful in big cities with many supermarkets of the same chain.

The supermarkets presented without the insertion of a specific filter are the ones which are closer to the customer’s position. For the selection of a supermarket the customer must click on the icon of the desired one.

### 3.2.5 Ticket Preview (LineUp)

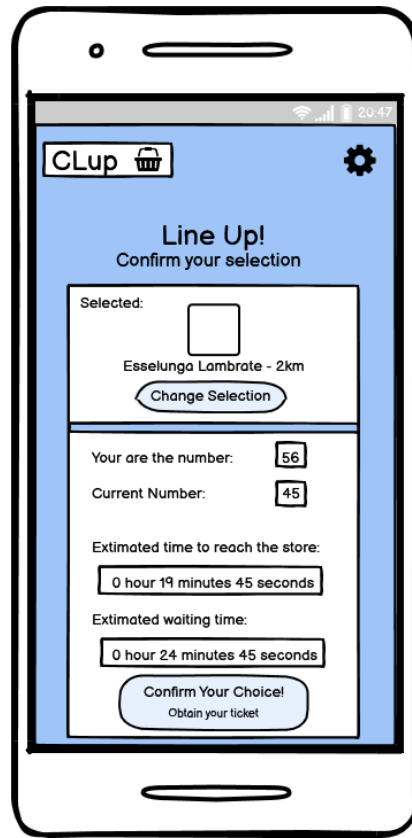


Figure 28: Ticket Preview (LineUp)

In this picture (Fig. 28) it is possible to observe the ticket preview, containing all the important information of a Line Up reservation before the confirmation.

This preview can be useful for the customer in order to know in advance how much time is needed to enter the supermarket. In this page is possible to see:

- The number of the last customer that has entered the supermarket.
- The number of the reservation (in case of confirmation).
- The estimated time to reach the store.
- The estimated waiting time.

It is also possible to change the previous choice of the supermarket in the case the customer changes ideas.

If the customer wants to confirm his reservation he has only to click the “Confirm Your Choice! Obtain your ticket!” button.

### 3.2.6 Ticket (LineUp)

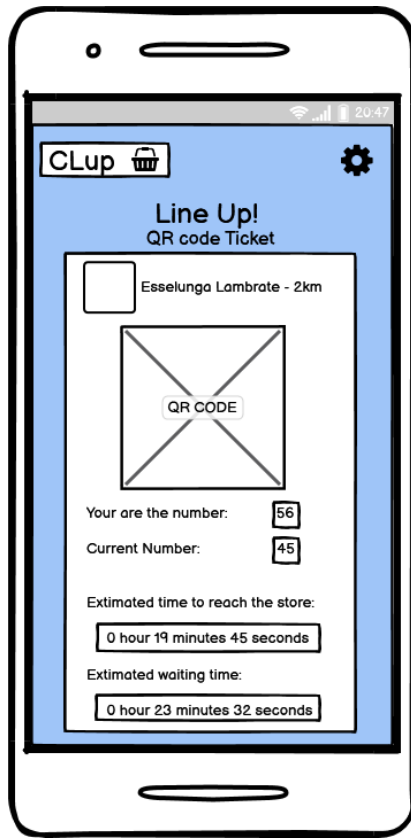


Figure 29: Ticket (LineUp)

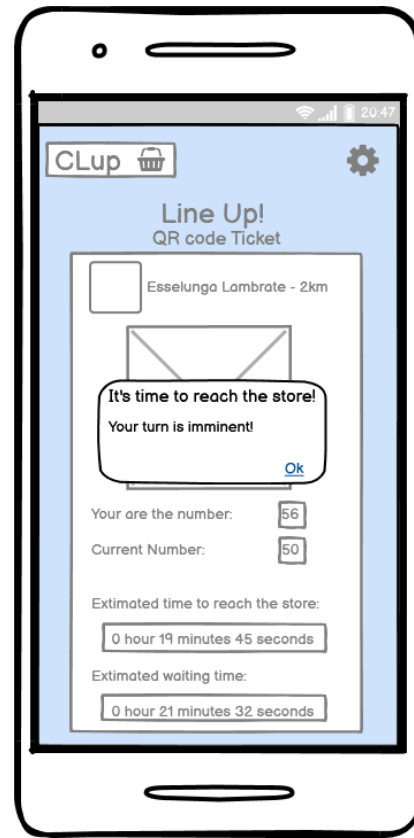


Figure 30: Ticket Notification (LineUp)

In these pictures (Fig. 29 and Fig. 30) it is possible to observe the ticket related to a specific Line Up reservation and the notification shown when it is almost the turn of the customer.

The pop-up is shown in advice, in order to let the customer reach the store in time and avoid crowding outside the store. All the important information is present inside the ticket are:

- The QR Code used to access the store.
- The number of the last customer that has entered the supermarket.
- The number of the reservation.
- The estimated time to reach the store.
- The estimated waiting time.
- The supermarket chosen.

Once the ticket is used, the system moves the customer to the initial screen.



### 3.2.7 Ticket Machine Interface

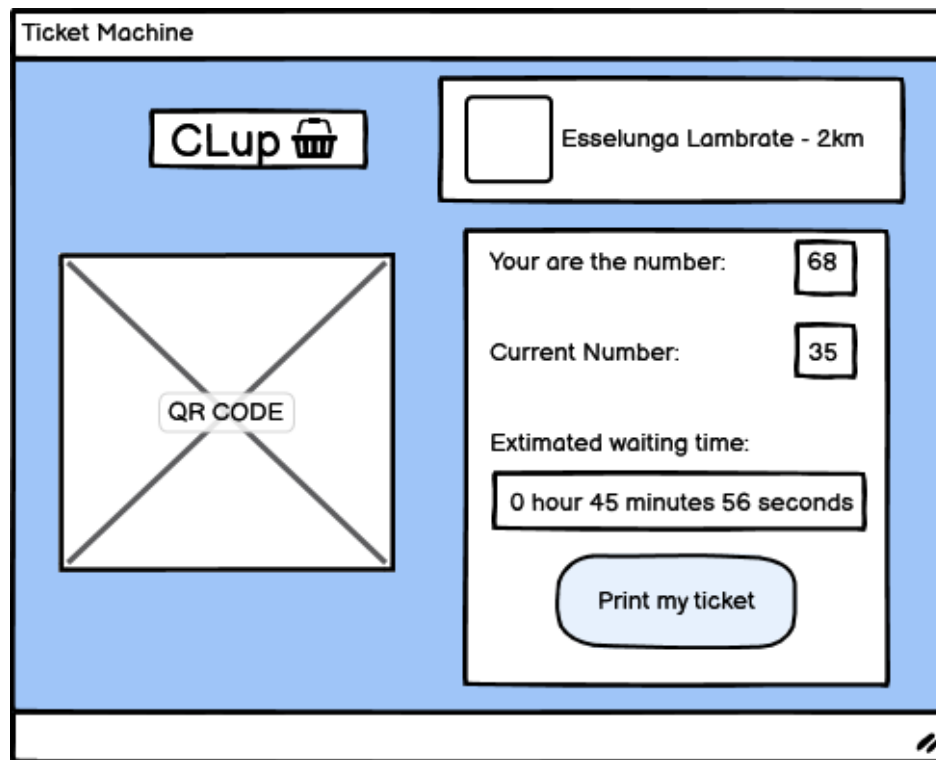


Figure 31: Ticket Machine Interface

In this picture (Fig. 31) it is shown the graphical interface of the ticket machine placed at the entrance of every supermarket. This ticket machine, as already explained in the RASD, is used to guarantee all the customers, also the customers without digital devices, to have access to the supermarket. Using the ticket machine, it is possible to print only the tickets related to the Line Up service.

The information reported on the screen are:

- The QR Code used to access the store.
- The number of the last customer that has entered the supermarket.
- The number of the reservation.
- The estimated waiting time.
- The supermarket which has the ticket machine.

The only button present is “Print my ticket” that sends a lineup request to the system and prints the QR Code of the reservation using a printer connected to the machine.

### 3.2.8 Supermarket Selection (BookAVisit)

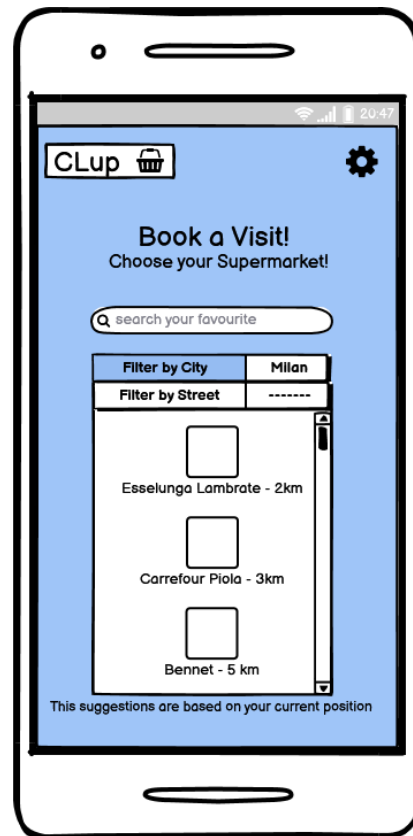


Figure 32: Choice of Supermarket (BookAVisit)

In this picture (Fig. 32) it is possible to observe the selection of a specific supermarket during the use of the “Book A Visit” functionality.

In this page it is possible to search for a specific supermarket using the “search bar” in the upper part of the page. It is also possible to filter the different supermarkets by city or, for a more accurate filtering, by street. This second case is more useful in big cities with many supermarkets of the same chain.

The supermarkets presented without the insertion of a specific filter are the ones which are closer to the customer’s position. For the selection of a supermarket the customer must click on the icon of the desired one.

### 3.2.9 Time Slot Choice (BookAVisit)

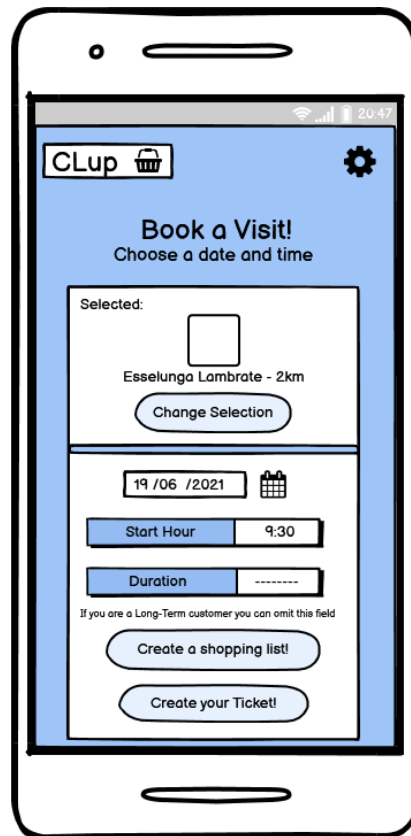


Figure 33: Time Slot Choice (Book A Visit)

In this picture (Fig. 33) it is possible to observe the selection of a time slot during the use of the “Book A Visit” functionality.

The first choice is the one related to the “Date of the visit” to the supermarket, the customer has to select it in a pop-up menu with a calendar. Then the customer has to select a “Start Hour” for the visit, another mandatory field. The last field is the selection of the “Duration” of the visit that is possible to omit in the case of a long-term customer. In that case the system can estimate it through an external service.

It is possible also to change the supermarket if the previous selection is wrong with the button “Change Selection” present in the upper part of the screen.

At the bottom of the screen is possible to push two different buttons, the first one allows to add a shopping list (useful for the customer and used by the Internal Crowd Estimation System) and the second one allows to obtain the ticket in the case of availability of the selected slot.

### 3.2.10 Fully Booked Notification (BookAVisit)



Figure 34: Fully Booked Notification (Book A Visit)

In this picture (Fig. 34) it is possible to observe the pop-up notification provided by the system to the user in case of choice of a fully booked time slot. In this case the customer has two choices (if he has not fulfilled the shopping list yet) identified with two links in the pop-up:

- **Create a Shopping List:** in this case, after the filling of the list, the external service “Internal Crowd Estimation Service” can try to check if the items selected by the customer are not in crowded departments in the store and, if this check is successful, the reservation continues.
- **Choice of other Time Slots:** in this case the customer can check if there are other time slots when the supermarket is available that are good for him. In that case he can continue the reservation without problems.

If the customer is not satisfied by both of these choices, he can change supermarkets with one suggested by the application or stop the reservation.

### 3.2.11 Choice of Product Categories

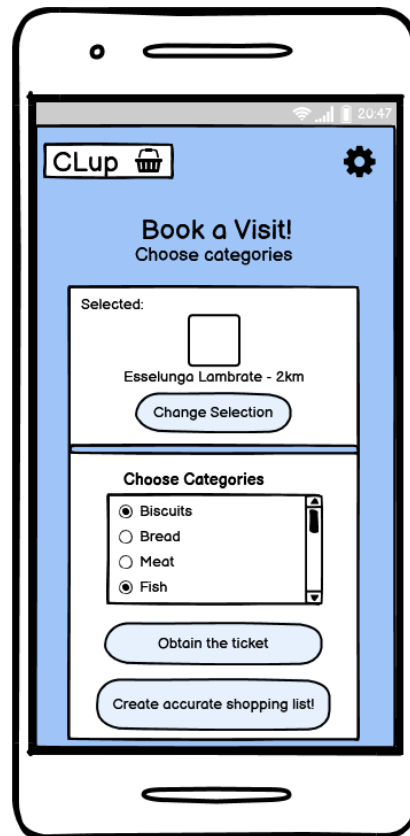


Figure 35: Choice of Product Categories

In this picture (Fig. 35) it is possible to observe the page where the customer can choose the categories of the products that he would like to buy in the store. This possibility is provided by the system in order to permit more customers to access the store if they are not going to crowded departments. With the selection of the categories the customer can try to obtain a reservation without fulfilling a specific shopping list.

At this point of the booking, it is possible for the customer to select all the categories with a scrollable menu and a press-to-select list. Once selected the categories the customer can try to obtain his ticket by pushing the button “Obtain the ticket”.

If the customer wants to change his previous selection of the supermarket, he can do it by pushing the button “Change Selection” on the upper part of the screen.

If the customer wants to fulfil a more complete Shopping List with the name of all the items, he can do it by pushing the button “Create accurate shopping list”.

### 3.2.12 Shopping List

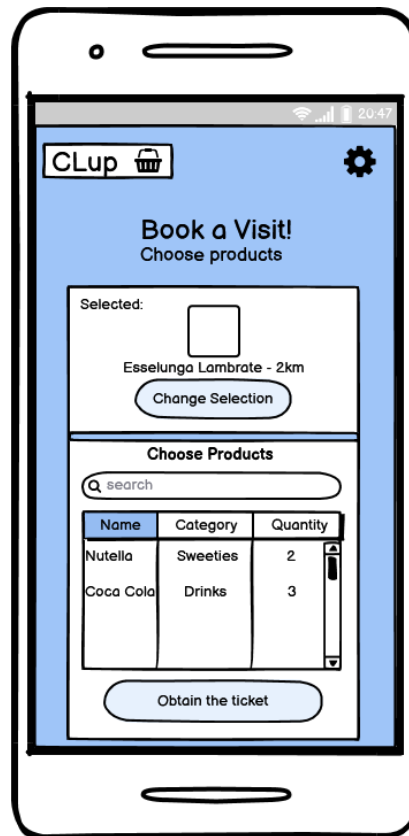


Figure 36: Shopping List

In this picture (Fig. 36) it is possible to observe the page where the customer can choose the products that he would like to buy in the store. This possibility is provided by the system in order to permit more customers to access the store if they are not going to crowded departments.

At this point of the booking, it is possible for the customer to select all the products with a search bar and a list where it is also possible to insert the amount of product needed. Once selected the products the customer can try to obtain his ticket by pushing the button “Obtain the ticket”. If the additional information can infer that the customer will not pass in crowded areas the booking request is accepted, else the request is rejected and the customer is moved to the “Suggested Time Slots” page.

If the customer wants to change his previous selection of the supermarket, he can do it by pushing the button “Change Selection” on the upper part of the screen.

### 3.2.13 Suggested Time Slots

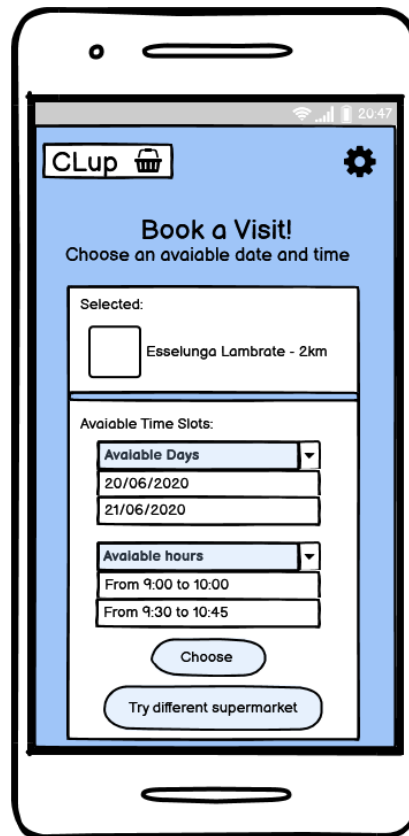


Figure 37: Suggested Time Slots

In this picture (Fig. 37) it is possible to observe the selection of the different time slots suggested by the application if the check of the shopping list was not successful for granting the access to the supermarket or the customer chooses to not fulfil it.

For the customer it is possible to select one available date for the visit and one available time slot in that specific date. The two selections are done by selecting one record in the list that informs of all the possibilities.

If the customer finds one alternative of his taste, he can obtain the ticket by pushing the button “Choose” and terminate his reservation.

If the customer is not satisfied with the suggested time slots, he can look for different supermarkets related to the previous selected one by pushing the button “Try different supermarkets”.

### 3.2.14 Alternative Stores

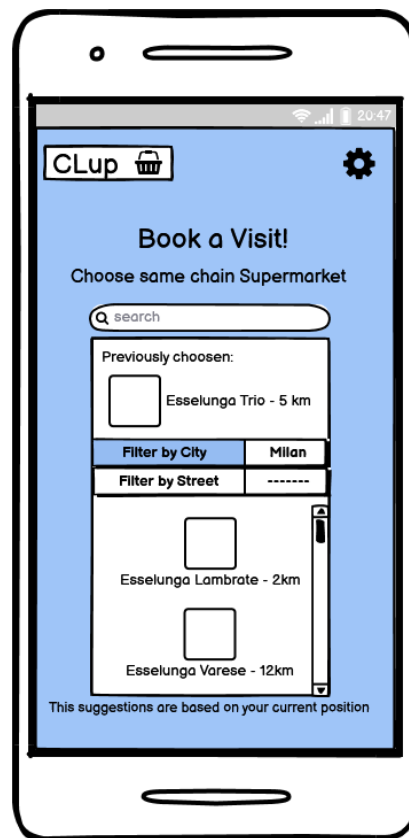


Figure 38: Alternative Stores

In this picture (Fig. 38) it is possible to observe the selection of a suggested supermarket related to the one selected during the previous reservation.

In this page it is possible to search for a specific supermarket using the “search bar” in the upper part of the page. It is also possible to filter the different supermarkets by city or, for a more accurate filtering, by street.

The supermarket presented without the insertion of a specific filter are the ones which are closer to the customer’s position and related to the supermarket selected during the first phase of the booking. For the selection of a supermarket the customer must click on the icon of the desired one.



### 3.2.15 Ticket (Book A Visit)

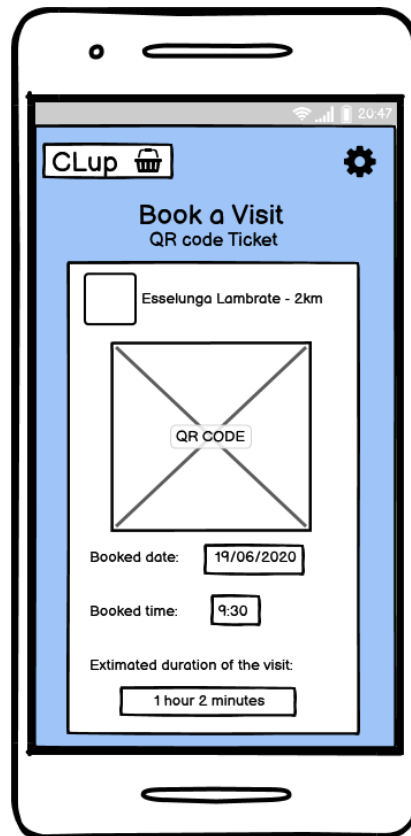


Figure 39: Ticket (Book A Visit)

In this picture (Fig. 39) it is possible to observe the ticket related to a specific Book A Visit reservation. Differently from the ticket of the LineUp functionality no pop-up notification is shown when the turn of the customer is imminent.

All the important information present inside the ticket is:

- The QR Code used to access the store.
- The date of the reservation.
- The starting hour of the reservation.
- The (estimated) duration of the visit to the store.
- The supermarket chosen.

### 3.2.16 Notification Management

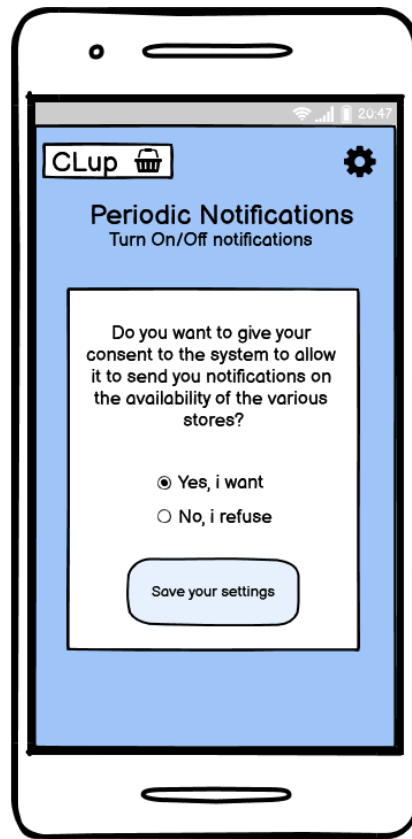


Figure 40: Notification Management

In this picture (Fig. 40) it is possible to observe the selection of the permission for the periodic notifications provided by the system to the customer.

The customer can change the permission every time he wants, by selecting one of the two possibilities and confirming his choice by pushing the button “Save your settings”.

The customer can update that setting by pushing the “Turn on notification” button in the Home Page (Fig. 25).

### 3.3 Manager Application

#### 3.3.1 Login Page

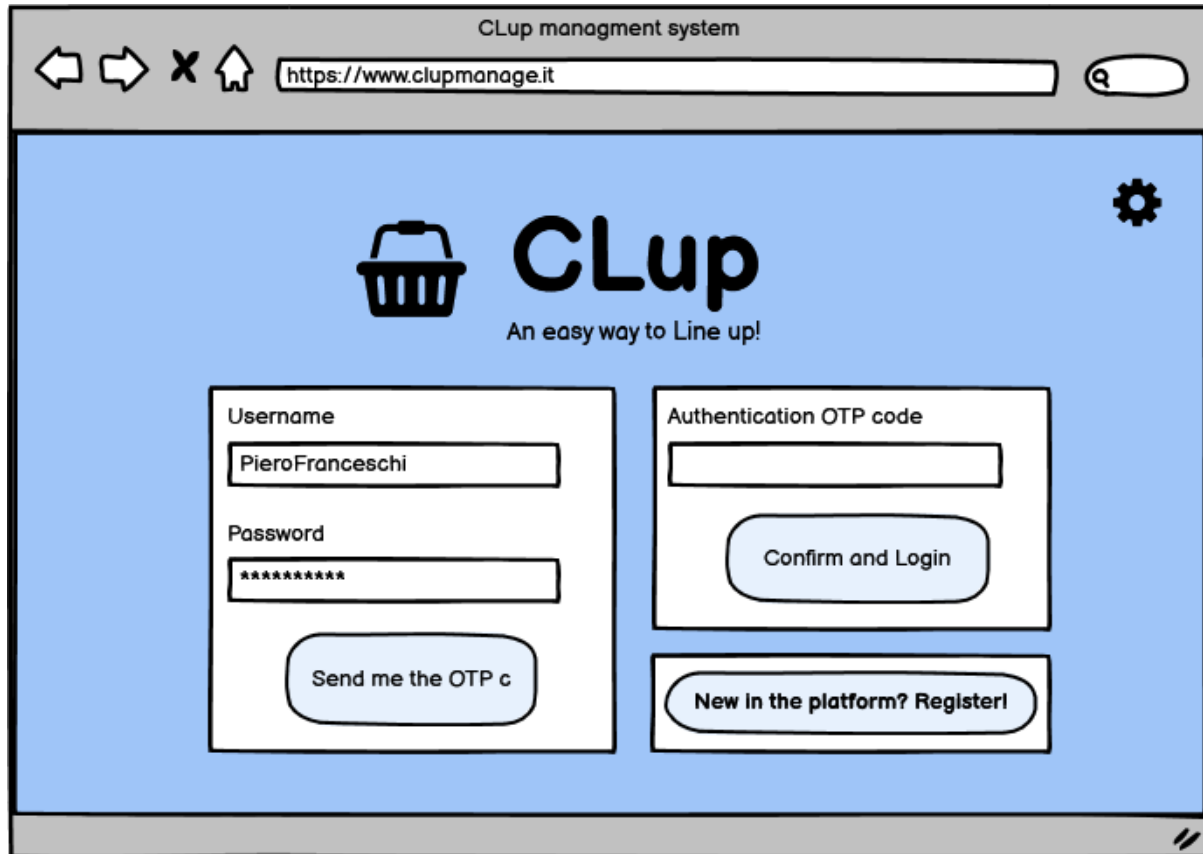


Figure 41: Manager Login Page

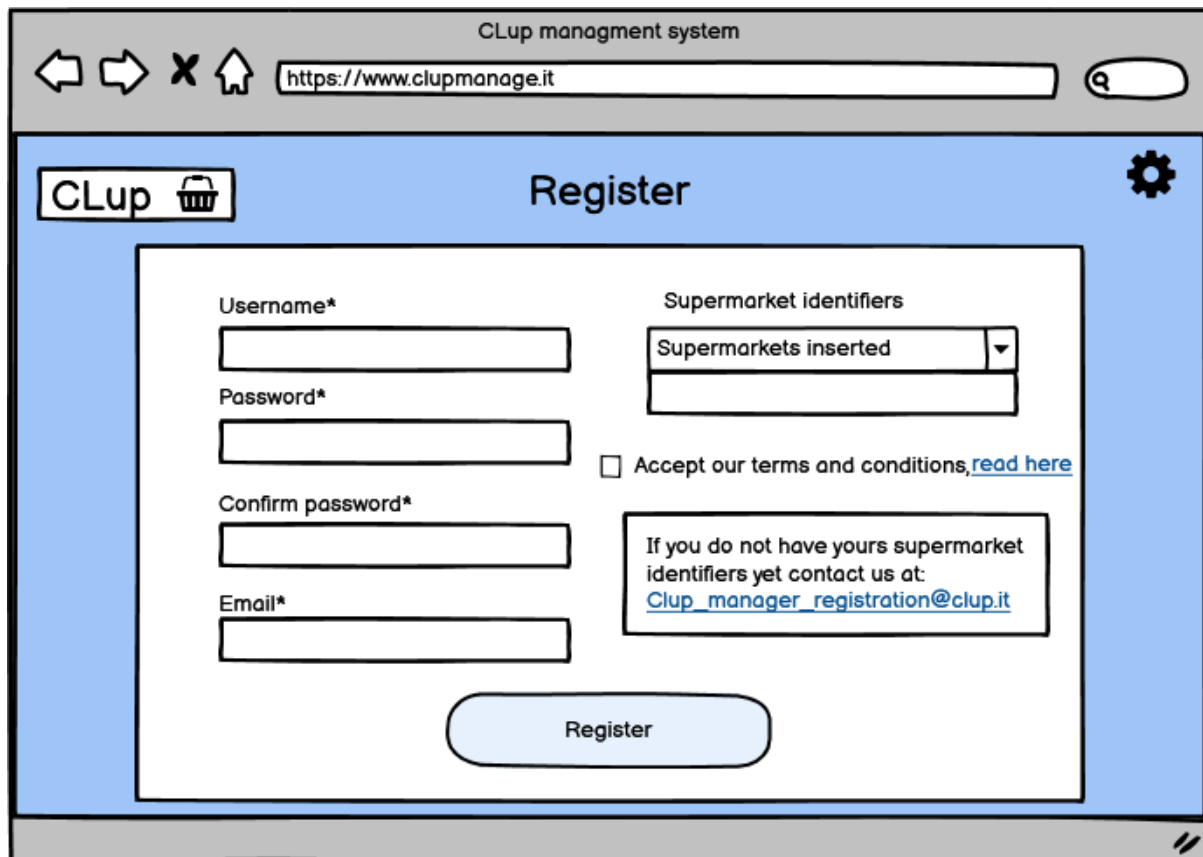
In this picture (Fig. 41) it is possible to observe the first page accessed by a manager when he wants to use the services granted by the system for the management of the supermarket.

If the manager is not already registered in the system, he has to click on the “New in the platform? Register!” button and then he is moved to the Registration Page (Fig. 42).

If the manager already has an account, he has to insert all his personal information (Username and Password) in the two empty fields. At this point he has to push the “Send me the OTP” button. The system at this point sends an Email to the manager’s email indicated during the registration with the OTP code. Inserting the code in the “Authentication OTP code” and by pushing the button “Confirm and Login” the manager sends back the OTP and, once verified by the system, he can access the services of CLup management.

Thanks to the OTP code a higher level of security is reached for the managers’ accounts.

### 3.3.2 Registration Page



The screenshot shows a web browser window titled "CLup management system" with the address bar displaying "https://www.clupmanage.it". The page has a blue header with the "CLup" logo and a shopping cart icon on the left, and a gear icon on the right. The main heading is "Register". The registration form is centered and contains the following fields and elements:

- Username\***: A text input field.
- Password\***: A text input field.
- Confirm password\***: A text input field.
- Email\***: A text input field.
- Supermarket identifiers**: A dropdown menu with "Supermarkets inserted" as the selected option.
- ☐ **Accept our terms and conditions**, [read here](#)
- A box containing the text: "If you do not have yours supermarket identifiers yet contact us at: [Clup\\_manager\\_registration@clup.it](mailto:Clup_manager_registration@clup.it)"
- Register**: A large, rounded button at the bottom center.

Figure 42: Manager Registration Page

In this picture (Fig. 42) it is possible to observe the Registration Page where a manager can register in order to use the services provided by the system.

The first step is the insertion of one unique username, a password and the confirmation of the password. Another mandatory field in this case is also the Email. It is mandatory because it is used by the OTP service for safer access. The manager has to insert one or more unique identifiers that represent the supermarket or supermarkets he runs. These codes are provided by the account services of CLup, in this way only real supermarkets are inserted in the system after a specific check of proof of ownership sent to the email "[Clup\\_manager\\_registration@clup.it](mailto:Clup_manager_registration@clup.it)" (obviously the email is a placeholder).

Once inserted all the information in the fields the manager has to accept the terms and conditions and terminate the registration by pushing the button "Register".

### 3.3.3 Home Page

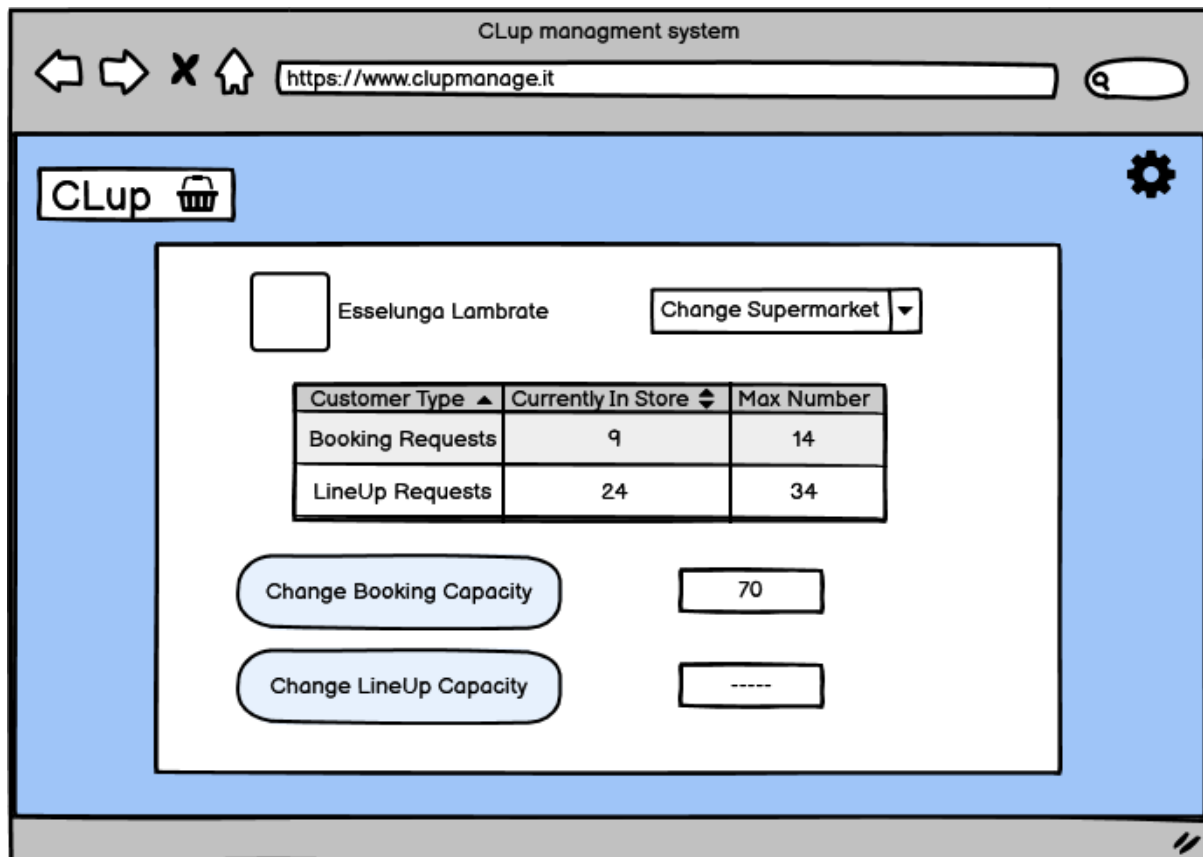


Figure 43: Manager Home Page

In this picture (Fig. 43) it is possible to observe the Home Page of the manager where he can perform all the functions provided by the system.

In the upper part of the page the manager can select the supermarket desired among those he runs. He can also add/remove a supermarket using the identifier. It is possible to monitor the situation of the customers currently in the store and the maximum capacity of both the functionalities.

The possible actions that the manager can perform are the modification of both the capacity related to the two different services:

- By the insertion of the new Booking capacity in the blank space and by pushing the “Change Booking Capacity” button.
- By the insertion of the new LineUp capacity in the blank space and by pushing the “Change LineUp Capacity” button.

## 4 Requirements Traceability

In this section it is possible to observe the mapping between the components, described in Section 2, and the requirements identified in the RASD. Some requirements are reached with the help of external services that provide algorithmic operations which we have chosen to not include in the CLup system as better explained in Section 2.2.

**R1:** The system must allow customers to register.

**Registration Manager**

**R2:** The system must allow managers to register.

**Registration Manager**

**R3:** The system must allow customers to log in.

**Access Manager**

**R4:** The system must allow managers to log in.

**Access Manager, OTP Manager**

**R5:** The system must guarantee that each username is unique.

**Registration Manager**

**R6:** The system must save customers registration data.

**Registration Manager, Data Manager**

**R7:** The system must prevent users from using special characters in their username.

**Registration Manager**

**R8:** The system must retrieve the GPS position while the user makes a reservation.

**Customer App, Map Manager**

**R9:** The system must allow users to insert in which store they want to go.

**Customer App,**

**For LineUp: LineUp Manager,**

**For Book A Visit: Reservation Manager**

**R10:** The system must allow users to insert a shopping list of items available in the store.

**Customer App, Shopping List Manager**

**R11:** The system must allow users to choose categories of products.

**Customer App, Shopping List Manager**

**R12:** The system must provide users with the QR code.

**Ticket Manager**

**R13:** The system must provide users with the ticket number.

**Supermarket Manager**

**R14:** The system must allow users to insert visit date on the booking.

**Customer App, Reservation Manager**

**R15:** The system must allow users to insert visit time on the booking.

**Customer App, Reservation Manager**

**R16:** The system must be able to send notifications to the user.

**Notification Manager**

**R17:** The system must allow customers to filter by city.

**Customer App, Map Manager**

**R18:** The system must allow customers to filter by street.

**Customer App, Map Manager**

**R19:** The system must estimate the waiting time before customers' turn.

**LineUp Manager (external service)**

**R20:** The system must infer the usual duration of the visit to the store of a long-term customer.

**Reservation Manager (external service)**

**R21:** The system must estimate the necessary time to arrive at the store.

**Map Manager (external service)**

**R22:** The system must notify users in advance about their turns.

**LineUp Manager**

**R23:** The system must allow users to indicate the approximate expected duration of the visit.

**Customer App, Reservation Manager**

**R24:** The system must allow managers to know how many “line up customers” are inside.

**Supermarket Manager**

**R25:** The system must allow managers to know how many “book a visit customers” are inside the shop.

**Supermarket Manager**

**R26:** The system must allow managers to add a shop in the managed ones.

**Manager WebApp, Management Manager**

**R27:** The system must allow managers to remove a shop in the managed ones.

**Manager WebApp, Management Manager**

**R28:** The system must allow managers to change the maximum booking capacity of the managed shops.

**Manager WebApp, Management Manager, Supermarket Manager**

**R29:** The system must allow managers to change the maximum line up capacity of the managed shops.

**Manager WebApp, Management Manager, Supermarket Manager**

**R30:** The system must allow customers to be informed about the availability of different time slots.

**Advice Manager**

**R31:** The system must allow customers to be informed about the availability of different supermarkets of the same chain.

**Advice Manager**

**R32:** The system must allow customers to know if the store is in one of its closing days or not.

**Supermarket Manager**

**R33:** The system must save managers registration data.

**Registration Manager, Data Manager**



## 5 Implementation, Integration and Test Plan

So far in the Section 2 we have described each component our software is composed of (Subsection 2.2), how these different components are deployed among devices (Subsection 2.3), then we have clarified the dynamic behavior and how these components interact with each other (Subsection 2.4), we have also pointed out all the interfaces involved (Subsection 2.5) and at the end we have justified all the decisions and strategies applied. In Section 3 we have shown all the mockups and in Section 4 we have highlighted the correspondence between components and requirements through the requirements traceability. Now, in this section, we are ready to focus on the implementation, integration and test plan of the CLup system, the last step before the implementation and the development of the code.

### 5.1 Overview

Before going into the details, we want to specify some important criteria that are needed to the developers in order to follow a right order and a right plan that can help the implementation of the software and also some points that help to understand the whole section better.

- One important criterion is that the testing activity needs to be done in parallel with the implementation. This allows to maintain a high level of consistency of the code and reduce the probability to meet bugs and errors. In fact, doing so it is probable to discover as soon as possible issues and then solve them. Furthermore, solving a recently introduced problem is much easier than solving one that covers most of the code.
- It is important to use the **incremental strategy** in order to facilitate bug tracking, therefore coherently with what we have said before, integration and testing need to be executed in parallel. Every time a first version of a module is released, it is tested, if no bug is found, it is integrated with a second module already tested and so on until all the components are tested and integrated with each other.
- It is good to use a **bottom-up approach**, starting from the leaves up to the highest level module through the use of drivers that help us to simulate the module we want to test and then integrate it. It is important to say that for the Implementation and Integration order suggested in this document, there will be the need to generate Stubs for every APIs invoked by the Server subcomponent which is about to be implemented or integrated. Although using Stubs may go against the bottom-up approach, it is evident that since the APIs provide small data, that the server needs to ensure the most important functions, it is quite simple to generate a Stub for every APIs called. Also the top-down approach can be applied.
- The Implementation Plan and the Integration Plan explained respectively in Section 5.2 and 5.3 are just a suggested order but other solutions can be applied as well.

## 5.2 Implementation Plan

As we have already anticipated the strategy chosen is the bottom-up approach. This is mainly due to the fact that we suggest starting from the last component that most of the others refer to and then go up the tree until all the other components have been tested and integrated and step by step recover all the functionalities that CLup offers. Our analysis mainly focuses on the Server and the client-side components implementation, without considering the external APIs, since we have supposed them as already existing systems with which the Server interacts and takes advantage of some of their features useful for CLup and hence they do not need to be implemented.

Before showing the order of the implementation, we want to report here the Call Server Tree (Fig. 44) in which are displayed all the direct calls of each Server subcomponent, in order to better understand the Implementation Plan listed later concerning all the internal Server components. As it is possible to see the root is the Redirector that is the component that deals with all the requests, it manages and routes requests depending on its type. Whereas at the leaf nodes it is possible to see that the most frequent component is the Data Manager that is the component that does call no other component and it is used to answer the majority of the requests.

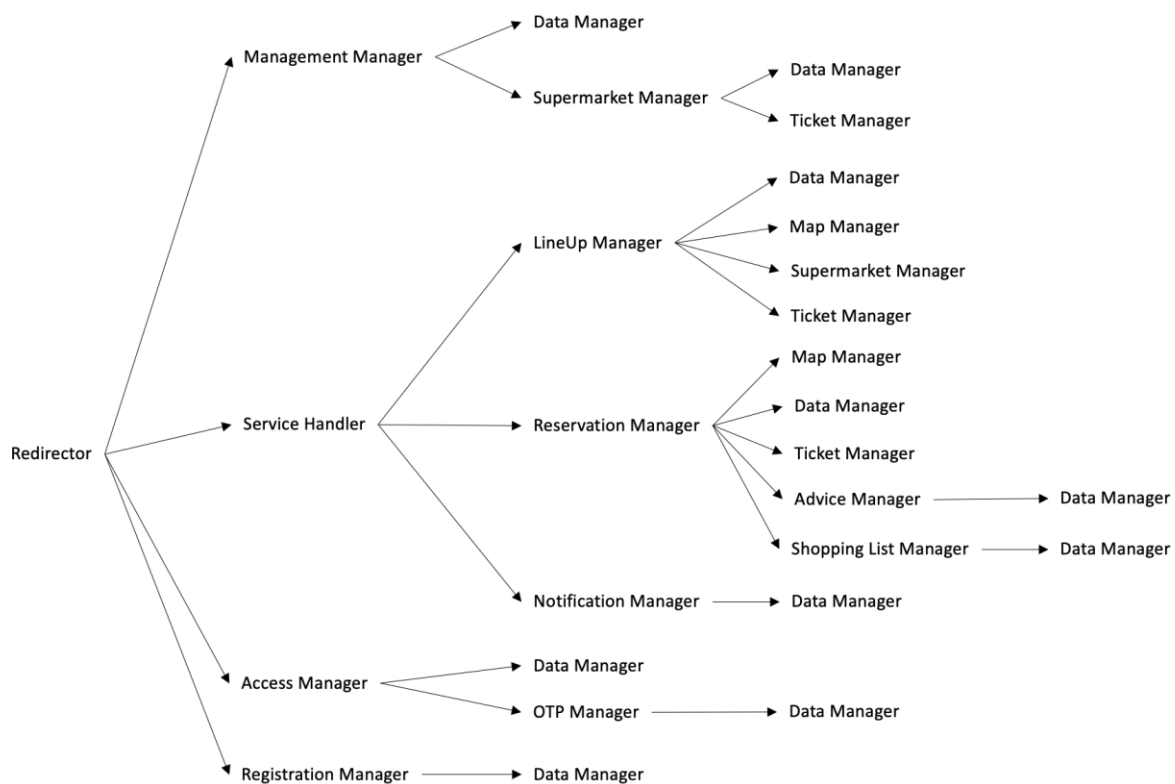


Figure 44: Call Server Tree

It is advisable to start from the lowest part, in order to first implement all the components dedicated to some features of the main functions. The order in which all the Server subcomponents are implemented is performed from particular to general, from the simplest to the most complex functions. When all the Server components are implemented, it is possible to complete the entire system by implementing the client-side components: Web Server, Customer App and Ticket Machine.

The implementation plan is performed in the following order:

1. Data Manager
2. Ticket Manager
3. Supermarket Manager
4. Shopping List Manager, Advice Manager and Map Manager
5. LineUp Manager, Reservation Manager, Notification Manager and Management Manager
6. Service Handler
7. OTP Manager
8. Access Manager and Registration Manager
9. Redirector
10. Web Server, Customer App and Ticket Machine

This order is consistent with the bottom-up approach and with the Call Server Tree (Fig. 44) for what concerns Server components. The Call Server Tree helps to follow the Server Implementation Plan and it should guide in its understanding. Here we are just listing the components since we want to avoid repeating it two times as the order will be the same, but then in Section 5.4 we will explain more in detail every step. The figure below (Fig. 45) shows the entire Implementation Plan: Server and Client-side components. We have omitted some of the direct links that we can see instead on the Call Server Tree, this is to make the figure less complex and more readable. Moreover, the numbers near the components refer to the implementation of all the components at the same level.

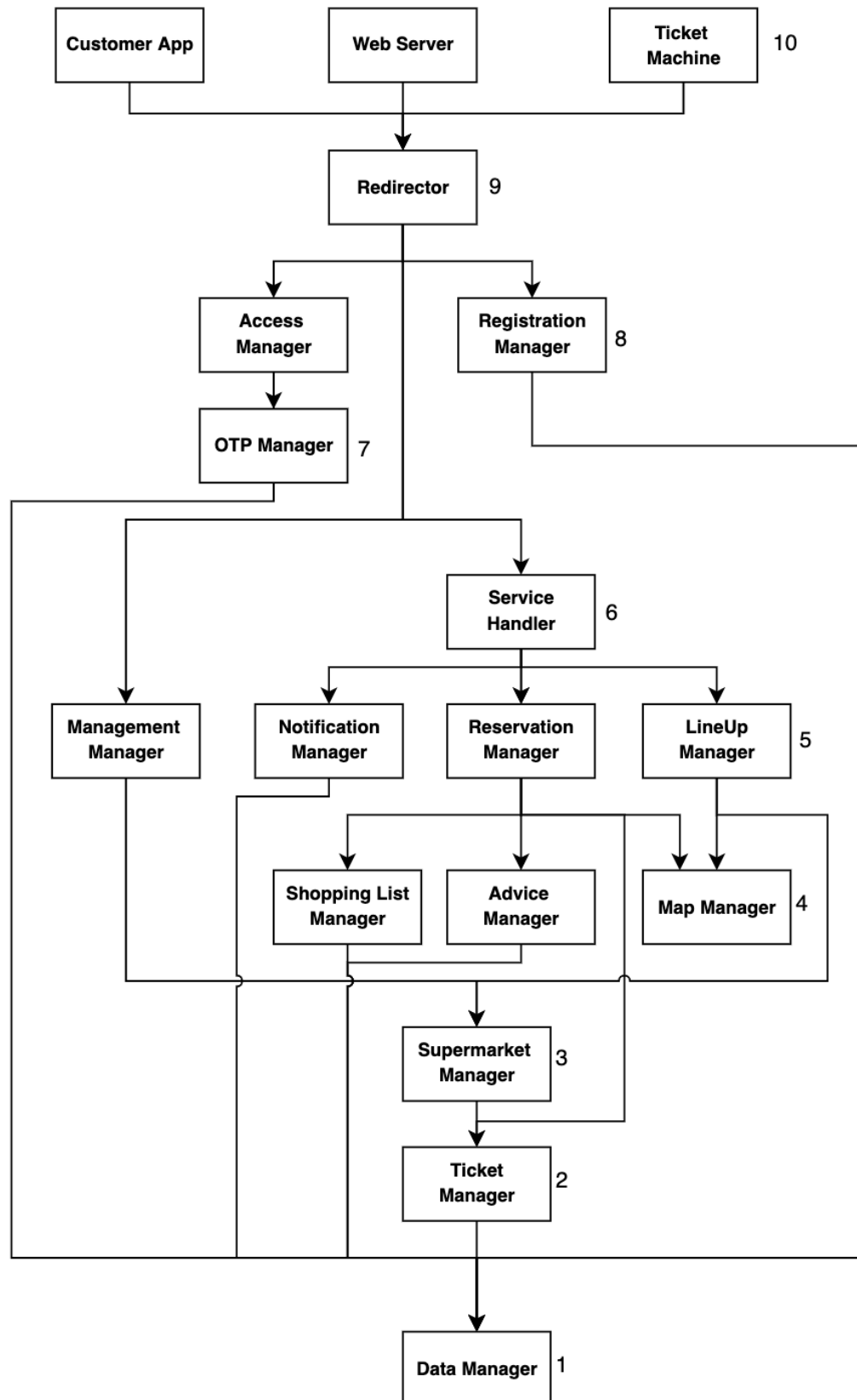


Figure 45: Implementation and Integration Plan

### 5.3 Integration Plan

At this subsection we are going to formulate an Integration Plan for the entire CLup system. First we want to give a high-level plan in order to anticipate an overview approach, later in Section 5.4 we will go into details and we will explain each step.

The Integration Plan we suggest is consistent with the following steps:

- The very first component to be integrated is the DBMS with the DataManager. As we have already said in the implementation plan, these components need to be the first because we need to access the data stored in the DBMS.
- Then there is the integration of the subsystems inside the server. This integration plan follows the implementation plan previously described in Figure 45 since the integration plan runs in parallel to the implementation.
- The next part to be integrated is the one that involves all the external services and all the APIs with which the system relates and requests some data. In this way after having tested and integrated entirely the central server we can complete the server components that rely on the external APIs by requesting some necessary data to provide the core functionalities of CLup.
- Now that all the functionalities of the server are ready, we can now move on to the client-side component: Ticket Machine, Customer App, Manager WebApp and Web Server. In this way we have completed the system and managed the requests sent by the clients, but also the login and registration functionalities.

## 5.4 Plan Definition

We are going to show each step of the Integration Plan but also review the Implementation Plan (already described but here explained in detail). For simplicity in the figures that show the Integration steps, we have only illustrated components that rely directly on the component(s) that is about to be integrated.

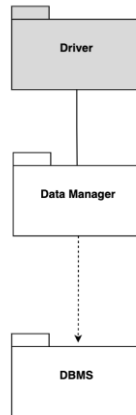


Figure 46: 1 - *Data Manager Integration*

1. The very first component to be implemented is the **Data Manager**. It is the ultimate component in which the other components store and request data in order to provide the functionality each component is dedicated. All the other components rely on it, directly or indirectly, so it is the very leaf component of the entire system. It is tested and then integrated with DBMS.

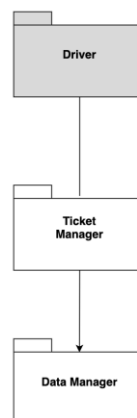
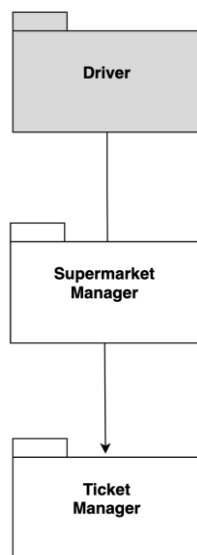
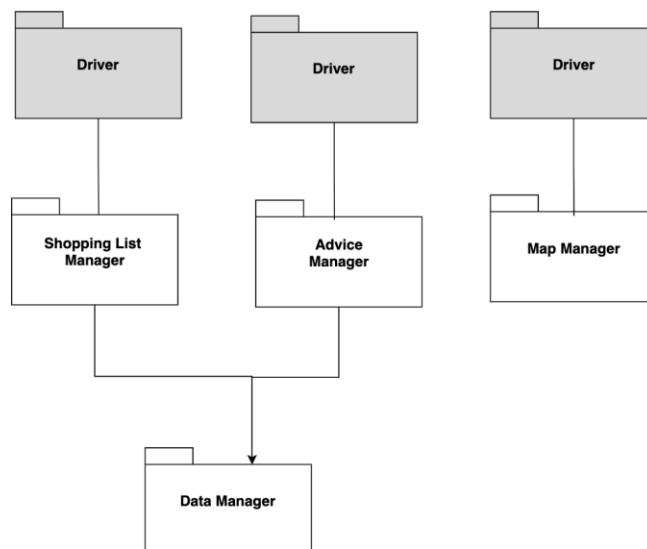


Figure 47: 2 - *Ticket Manager Integration*

2. The second one to be implemented is the **Ticket Manager**, it is tested and integrated with the Data Manager. It is directly connected with the Data Manager, even if in the Call Server Tree it is not shown, since it has to control that ticket data coming from the QR Code Reader API are stored in the Data Manager in order to accept the request or reject the request.

Figure 48: 3 - *Supermarket Manager Integration*

3. The next one is the **Supermarket Manager**, integrated with the Ticket Manager. We have chosen these two because they manage lower level data about tickets and supermarkets that are useful for components that provide the main functions.

Figure 49: 4 - *Shopping List Manager, Advice Manager, Map Manager Integration*

4. It is now possible to move on to the implementation of **Advice Manager**, **Shopping List Manager** and **Map Manager**. The first two ones are subcomponents of the BookAVisit component; indeed they provide advanced features to the booking function. The last one is the one that relies with the Map Interface, it provides data about streets and positions. They are tested and integrated with the system already integrated.

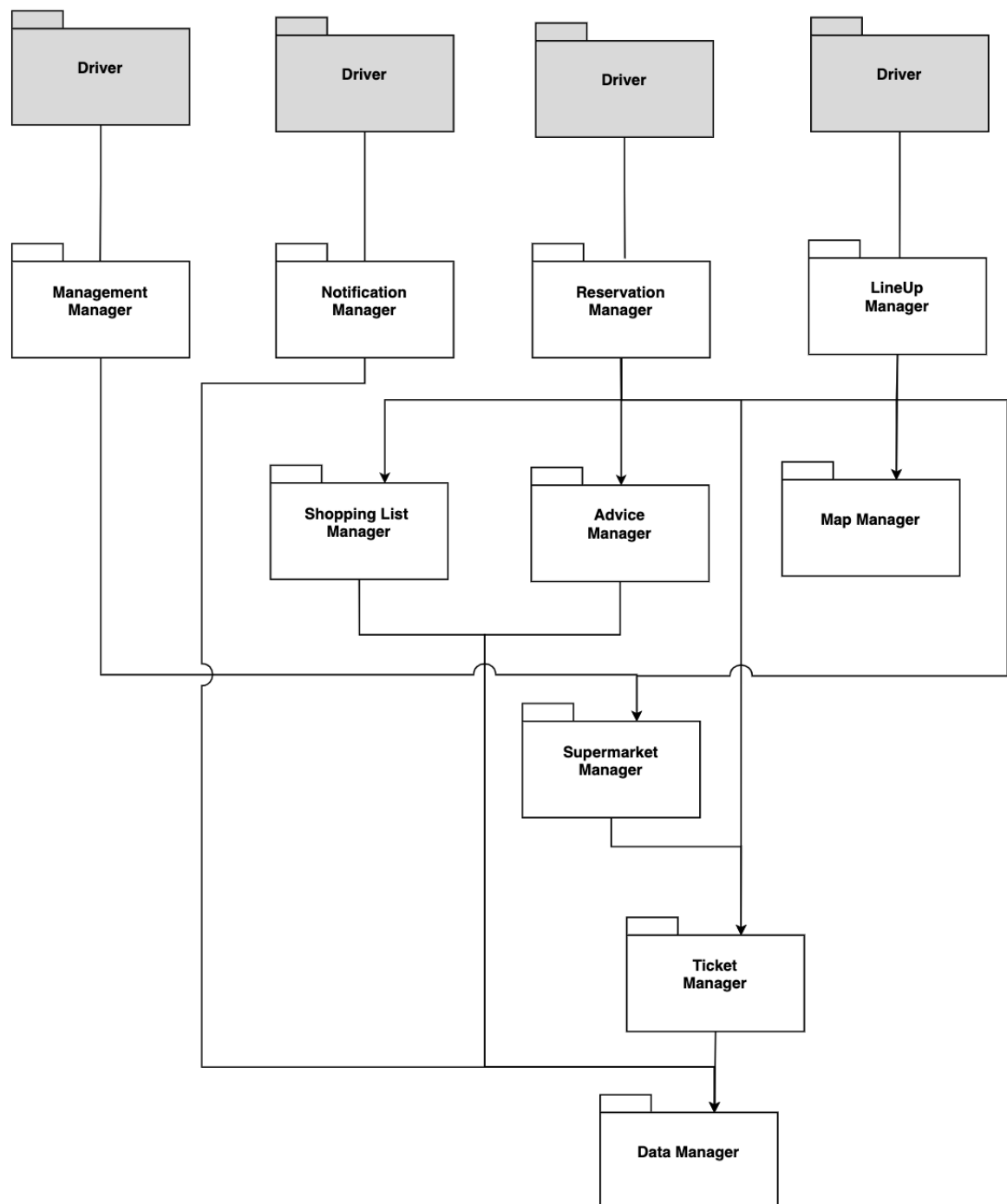
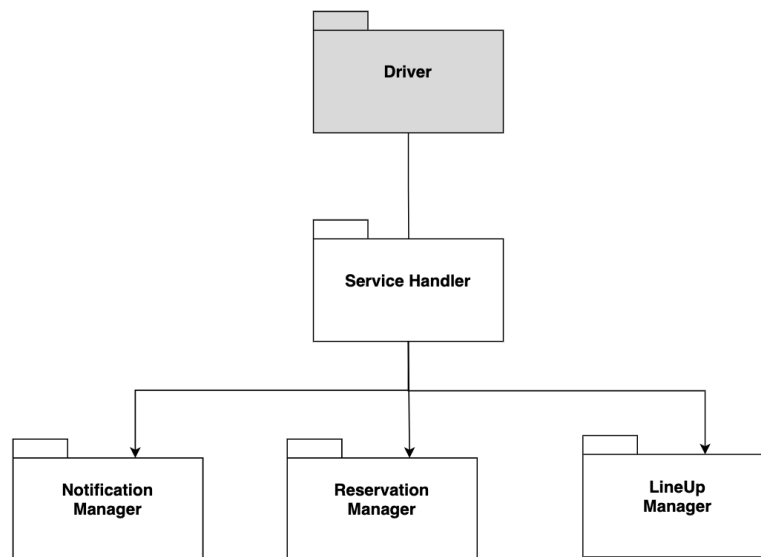


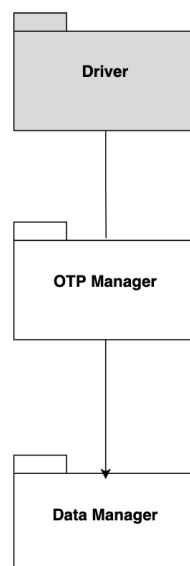
Figure 50: 5 - *Management Manager, Notification Manager, Reservation Manager, LineUp Manager Integration*

- Now it is possible to implement and integrate the components that grant the main functions of CLup: **LineUp Manager**, **Reservation Manager**, **Notification Manager** and **Management Manager**. They correspond respectively to the management of the lineup and book a visit request, notification function and the manager functions. They are tested and then integrated.



Figure 51: 6 - *Service Handler Integration*

6. Now it is possible to implement the **Service Handler** that manages all the functions dedicated to the customer functions previously implemented. It is tested and integrated with these components.

Figure 52: 7 - *OTP Manager Integration*

7. At this stage all the components that help to provide system functions have been tested and integrated, hence all the operative components of CLup. Therefore, it is possible to move on to the components that manage client-side. The first one is the **OTP Manager** that is useful in the manager login.

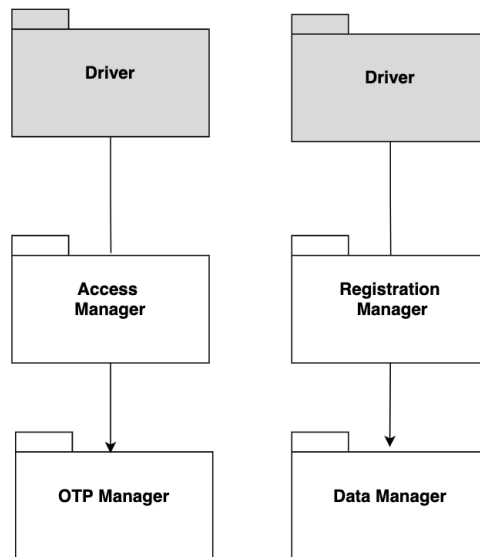


Figure 53: 8 - *Access Manager and Registration Manager Integration*

8. The second ones are **Access Manager** and **Registration Manager**. The Access Manager relies on the OTP Manager. In this the components related to the access functions have been completed. They are tested and integrated with the rest of the system.

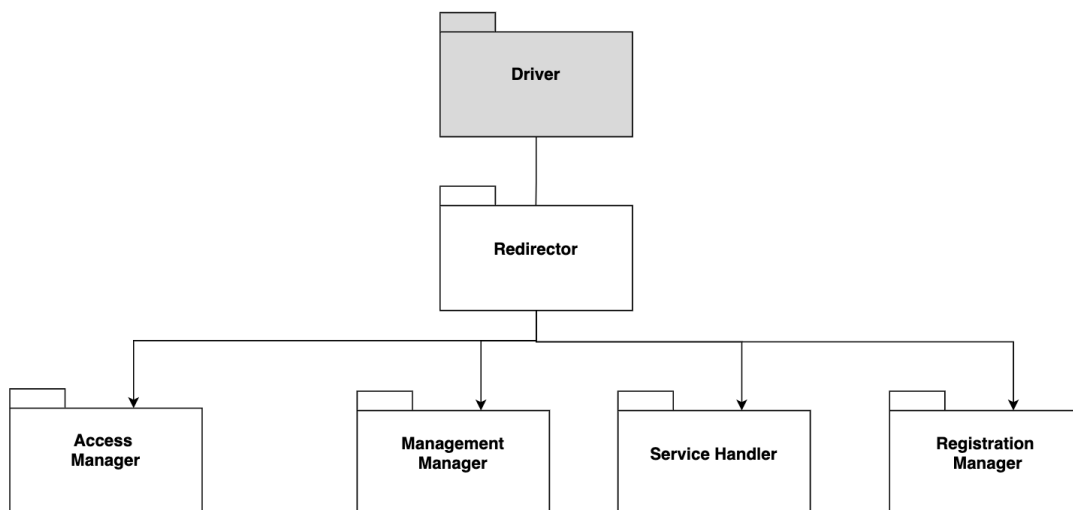


Figure 54: 9 - *Redirector Integration*

9. The next step is the implementation of **Redirector**, it is first tested and then integrated. This component manages and routes all the types of requests and unifies the entire system. At this point we can now move on to the external APIs integration: all the external services are integrated each with the corresponding internal server component that relies on it.

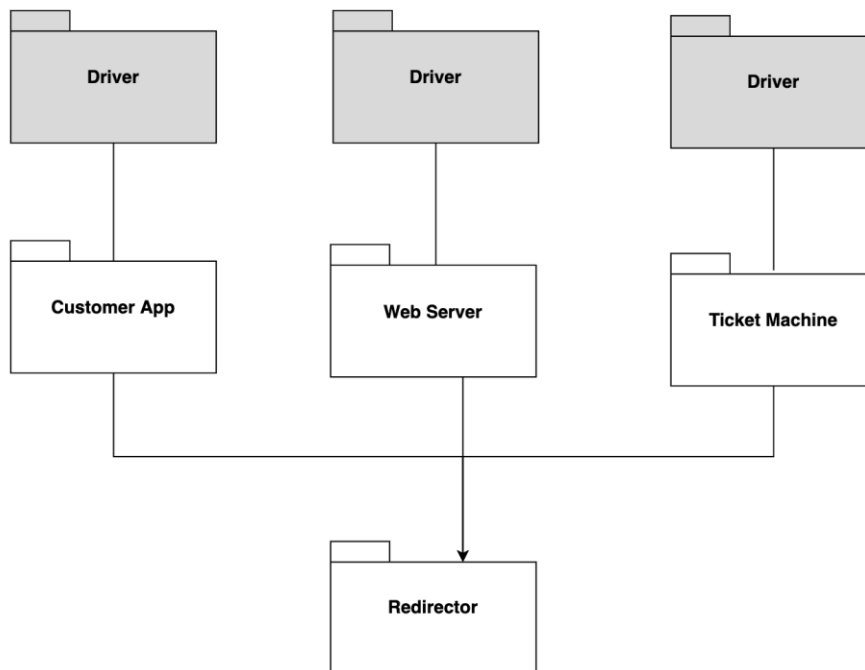


Figure 55: 10 - *Customer App, Web Server and Ticket Machine Integration*

10. The last step is to implement all the components related to the client-side: Customer App, Web Server and Ticket Machine. they can be implemented at the same time because they do not interfere with each other.

At this step, the entire Server components are tested and integrated with the Web Server, Customer App and the Ticket Machine. The Figure 45 represents this situation.

Now, once the Server and the Client-side are both implemented and integrated, it is now possible to integrate the entire system with the external APIs, each API is connected with a corresponding component inside the Server with which it interacts.

## 5.5 Testing

In order to complete the discussion, we now focus on the testing activity, we want to give some more specifications and technicalities. We have already said that unit testing and integration run in parallel, indeed every time a first version of a component is implemented it is also tested before integrating it.

We advise to use a **White-Box Testing** that verifies the internal functioning of a software component, since it requires knowledge and understanding of the internal structure of the software under test, this approach is usually under the responsibility of programmers. It can facilitate finding test cases to discover code issues. Then each unit component is tested also following a **Black-Box Testing** in order to discover issues on the integration between unit modules. A developer of the project is not required for this approach, often it is done by people who are not part of the organization that develops the software. Once the system is completely integrated, we plan also to execute the **System Testing** that mainly focuses on functional and non functional requirements.

We can consider **Java** as the main language used to implement CLup and the testing tools suggested are **JUnit** and **Mockito**, since both are consistent with the Java language. The testing coverage needs to be at least 85% per each component in order to maintain a high level of consistency of the system.

## 6 Effort Spent

### 6.1 Teamwork

Task	Teamwork's Hours
Introduction	2
Architectural Design	35
User Interface Design	8
Requirement Traceability	4
IIT Plan	5

Table 2: Teamwork effort

### 6.2 Individual Work

Task	Cristiano Serafini's Hours
Introduction	0,5
Component View	9
Deployment View	1,5
Runtime View	34
Component Interfaces	6
Styles and Patterns	1,5
Other Design Decisions	1,5
User Interface Design	3
Requirements Traceability	0,5
IIT Plan	2

Table 3: Cristiano's effort

<b>Task</b>	<b>Agnese Straccia's Hours</b>
Introduction	3
Component View	3
Deployment View	10
Runtime View	3
Component Interfaces	1,5
Styles and Patterns	6
Other Design Decisions	4
User Interface Design	2
Requirements Traceability	2
IIT Plan	20

Table 4: Agnese's effort

<b>Task</b>	<b>Stefano Vanerio's Hours</b>
Introduction	0,5
Component View	22
Deployment View	2
Runtime View	5
Component Interfaces	2,5
Styles and Patterns	2,5
Other Design Decisions	2
User Interface Design	11
Requirements Traceability	4
IIT Plan	2,5

Table 5: Stefano's effort

## Appendices

### A Revision History

The full revision history is available at this link: [history](#).

- Version 1.0, online version on Google Docs.
- Version 2.0, final fixes, page enumeration and heading.

### B Software and Tools used

- Git & GitHub as version control systems. The repository of the project is [here](#).
- Google docs as a collaborative work platform.
- Draw.io for the pictures (component, deployment, implementation diagrams).
- Balsamiq for the mockups.
- Visual Paradigm Online for the sequence diagrams.
- Microsoft Powerpoint

## 7 References

- [1] Slides of the lessons, Software Engineering II, Professor Rossi.
- [2] *RASD: Requirements Analysis and Specification Document*, Stefano Vanerio, Agnese Straccia, Cristiano Serafini - Politecnico di Milano - Software Engineering II, 2020.