L.M. Geoinformatic Engineering

AA 2020/2021

DD

Design Document

Version: First Version

Date: 06/01/2021

Authors: Marianna Alghisi
Gabriele D'Ascoli
Martina Pasturensi

Table of Contents

1. Introduction 4

1.1. ***Purpose*** 4

1.2. ***Scope*** 4

1.3. ***Definitions, Acronyms and Abreviations*** 5

1.3.1. Acronyms 5

1.3.2. Abbreviations 5

1.4. ***Revision History*** 5

1.5. ***Reference Documents*** 5

1.6. ***Document Structure*** 5

2. Architectural Design 7

2.1. ***Overview*** 7

2.1.1 High Level Components 8

2.2. ***Component View*** 10

2.2.1 Additional Specifications 12

2.3. ***Deployment View*** 13

2.4. ***Runtime View*** 15

2.4.1. Login Request 15

2.4.2. Book A Visit Request 16

2.4.3. Queue Request 17

2.4.4. Delete A Visit Request 18

2.4.5. Statistics Request 19

2.5. ***Component Interface*** 20

2.5.1 Internal Component Interface 20

2.5.2 External Component Interface 22

2.6. ***Selected Architectural Style and Patterns*** 22

3. User Interface Design 24

4. Requirements Traceability 30

5. Integration and Testing 35

5.1. ***Overview*** 35

5.2. ***Integration and Testing Strategy*** 35

5.3. ***System Testing*** 40

5.5. ***Additional specification*** 40

6. Effort Spent 42

7. References 43

1. Introduction

1.1. Purpose

The Design Document is intended to provide a deeper functional description of the CLup – Costumer Line UP system-to-be. The document aims to guide the developers to implement the architecture of the project, nevertheless the target audience of this document are not only developers but also project managers, testers and Quality Assurance.

Its aim is to provide a functional description of the main architectural components, their interfaces and their interactions. The relations among the different modules are pointed out using UML standards and other useful diagrams showing the structure of the system. In addition, it contains also the testing techniques to be used to integrate and verify the system.

1.2. Scope

The required application, named Costumer Line-Up – CLup, is aimed to help people in facing the sanitary emergency due to Covid-19. In fact, the goal of this project is 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, helps the costumers in handling and saving time spent for going out of home for essential needs, avoiding crowds and long lines out of the shop. In particular, the application focuses on grocery shopping and supermarkets, by the way it could be extended to all kind of shops. More precisely:

1) ***Basic Function***: the costumer should be able to access the application and line-up for a selected supermarket among the five closest to him (according to his position) and then wait until it is his turn. The costumer should be able to check through the application the number of people currently in line for the five closest supermarkets. If the costumer is lining for a certain supermarket, the application should inform the in-line costumer about the number of people in line before him and the approximate time of his turn. When the costumer asks to be put in-line for the selected supermarket receives through the application a QR code. Once his turn has arrived, before entering the supermarket his QR code will be scanned by an operator through an appropriate device, that update the status of the queue for the entering the supermarket.

2) ***Advanced Function 1***: The costumer should be able to reserve a visit to a selected supermarket, by choosing the day and the time slot, through the available ones, in which he is going to visit the shop and indicate the approximate duration of the visit. The costumer will receive a specific QR code that allow the costumer to enter in the supermarket only in the selected day and time.

3) ***Advanced Function 2***: The application provides to the shop a statistical analysis of the per hour affluences, in such a way that the shop can better organize the employees shifts.

1.3. Definitions, Acronyms and Abbreviations

1.3.1. Acronyms

RASD	Requirement Analysis Specification Document
GPS	Global Positioning System
UML	Unified Modeling Language

1.3.2. Abbreviations

Gx	Goal
Rx	Requirement
UCx	Use Case
MVC	Model View Controller
DBMS	Data Base Management System

1.4. Revision History

Date	**Modifications**
06/01/2021	First Version

1.5. Reference Documents

The production of this Requirement Analysis has been obtained with the support of the following documents:

- Specification Document: “CLup Assignment”
- RASD CLup
- Slides of the lectures

1.6. Document Structure

Chapter 1 – Introduction

The first chapter gives an introduction of the purpose and the scope of the software to develop, it contains specifications about the definitions, acronyms and abbreviations used for the drafting of the document, information about the revision history and the reference documents.

Chapter 2 – Architectural Design

The second chapter focuses on the architectural design choices, it includes all the components, the interfaces, the technologies (both hardware and software) used for the development of the application. It also includes the main functions of the interfaces and the processes in which they are utilised (Runtime view and component interfaces). Finally, there is the explanation of the architectural patterns chosen with the other design decisions.

Chapter 3 – User Interface Design

A preview of how user interface should be on the mobile and web application.

Chapter 4 – Requirements Traceability

This chapter is the connection between the RASD and the DD: for each goal of the software system are shown the involved requirements and the components of the architecture that execute the goal.

Chapter 5 – Integration and Testing

Fifth chapter introduces and clarifies the validation activities of the software system. In particular it focuses on the integration and testing strategy and the system testing strategy to perform before the User Acceptance test.

Chapter 6 – Effort Spent

Contains the effort spent by each member of the group in hours.

Chapter 7 – References

States the reference documentation.

2. Architectural Design

2.1. Overview

The architecture of the application is structured according to three logic layers:

1. Presentation level (P) handles the interaction with users. It contains the interfaces able to communicate with them and it is responsible for rendering of the information. Its scope is to make understandable the functions of the application to the customers.
2. Business logic or Application layer (A) takes care of the functions to be provided for the users. It also coordinates the work of the application, making logical decisions and moving data between the other two layers.
3. Data access layer (D) cares for the management of the information, with the corresponding access to the databases. It picks up useful information for the users in the database and passes them along the other layers.

The architecture has to be made in client-server style. Client and server are being allocated into different physical machines and their communication takes place via other components and interfaces located in the middle of the structure, composed by hardware and software modules.

The process begins with the invocation of a method to provide any functionality to the client, for instance sending a QueueRequest or requiring some information about the influx of customers in the supermarket. Then, the invocation of a specific method is caught by the server and its behaviour depends on the required function.

2.1.1. High Level Components

The hardware architecture chosen for the distributed application is Three-Tier. The three application layers are subdivided, therefore, among many dedicated machines, i.e.: a tier to interface with client, a middle tier for the application level, and another server for the database management. This approach is beneficial because the middle tier can maintain persistent connection with the DBMS, which is less expensive. Furthermore, having an intermediate machine between client and server can guarantee more security to the access control of the database from the users.

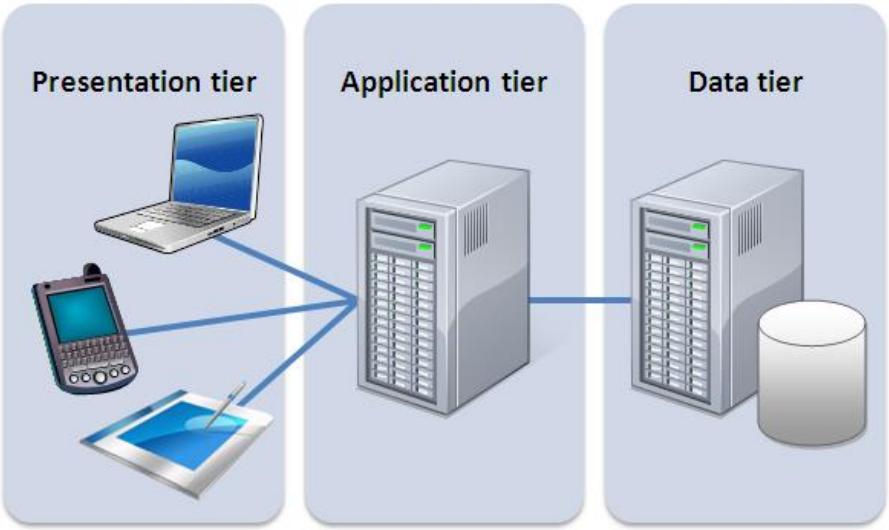


FIGURE 1 - THREE-TIER ARCHITECTURE

The figure below describes very high-level components and their interaction. The client's device can be a computer or a mobile phone for both users: Costumers and Supermarket. Smartphone using the application is connected directly with the application server while personal computer using the web app are managed by the web server.

The two servers that constitute the middle tiers communicate with each other, but the application server is the only one that communicates with the DBMS (database server).

In addition, the application server is responsible for using the external API (provided by external software) that allows to use the maps for the different functions of the application and for Qr code scan and generation.

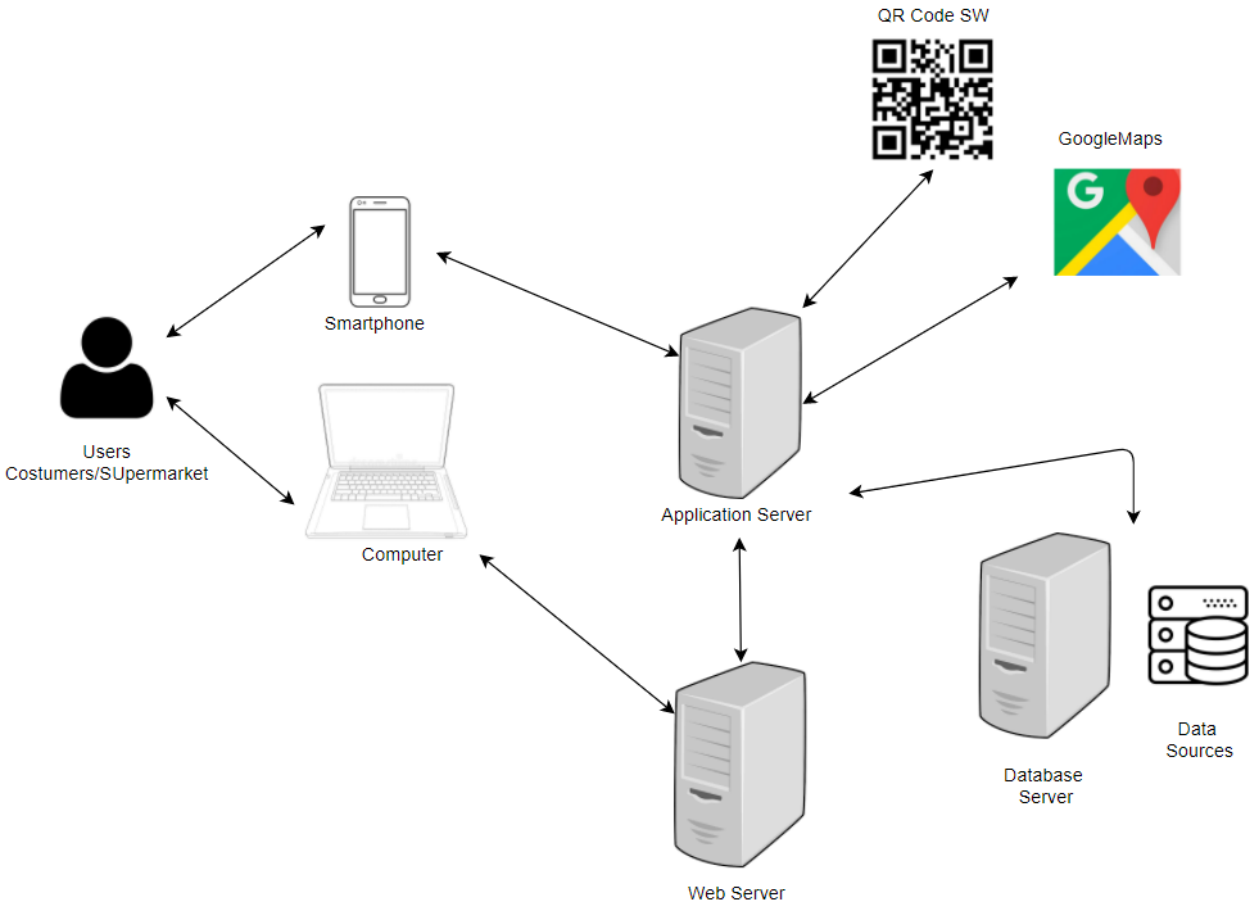


FIGURE 2 - SYSTEM ARCHITECTURE

The figure above does not included details of the architectural choices that have been made for the application. In the next sections the architectural design aspects, chosen to improve the realization, are described more accurately.

2.2. Component View

The following component diagram gives a specific view of the system focusing on the representation of the internal structure of the application server, showing how its components interact. The application server contains the business logic of our software. Other elements in the diagram, besides the application server, have been depicted in a simpler way just to show how the communication is structured among these components and the application server.

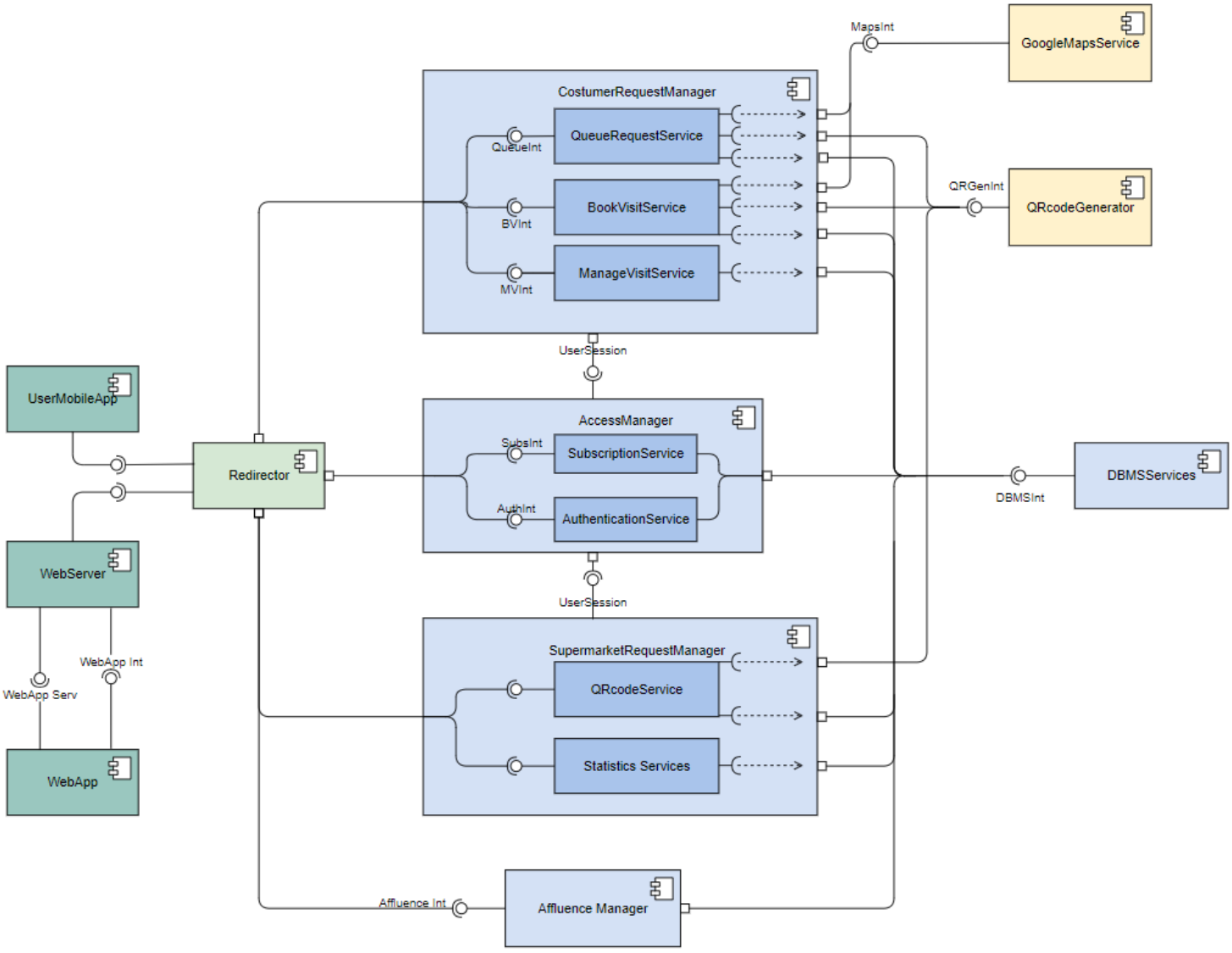


FIGURE 3 - COMPONENT ARCHITECTURE

Main Components:

- **AccessManager**: component that allows to manage subscription and log in to the application and creates user's sessions. It comprises two sub-components:
    - **SubscriptionService**: manages the subscription of a user to the application.

- **AuthenticationService**: it is meant for the already registered users, which access the application and needs to log in to be able to use all the functionalities to which it is subscribed. In addition, it handles the authentication mechanism of the supermarket.

  A certain level of security is needed to be sure to authenticate the right subject and for this purpose some security mechanisms have been chosen to avoid malicious people to have access to personal information. These mechanisms are not mentioned in this document.

- **CostumerRequestManager**: this component manages the requests performed by the customer users; it handles with its sub-components the three types of requests that costumars can do:

  - **QueueRequest**: costumer lines up for a selected supermarket (Basic Function).

  - **BookAVisit:** costumer books a visit for going to the supermarket in a specific day and time slot (Advanced Function 1).

  - **ManageVisit:** costumer can check for his booked visits and modify or delete them (Advanced Function 1).

- **SupermarketRequestManager**: this component manages requests performed by supermarket user, it comprises two sub-components

  - **QrcodeService**: allows to scan QRcodes for enter and exit the supermarket, it also allows to generate QRcode (queue tickets) for the costumers that physically line up to the store and don't have access to the application.

  - **StatisticsService:** used by the store to access to the information about the affluence and statistic analysis of data in the database (Advanced Function 2).

- **AfffluenceManager**: component that manages the notifications of the supermarket that sends information about the affluences occurred in the supermarket itself. These data are then collected in the database.

- **Redirector**: this component simply dispatches the requests and calls to methods from the users and the supermarket to the core of the application server. Every method is redirected to the proper component that can handle it. Also, responses and data sent back pass through this component to reach the applicant.

- **DBMSService:** handles the interaction with the database. The Interface provided by this component contains all useful methods to store, retrieve, update data into the database from different actors. Every internal component of the application server uses some methods of its interface.

The application server, in order to carry out all the requests, needs to interact with two external components, that perform specific functions:

- **GoogleMapsService**: component that allows to visualize the map of a specific location, compute the distance of the costumer (GPS position) with respect to the closest supermarkets and select the five closest shops.
- **QRcodeGenerator**: external component that allows to encode and decode data in QR code.

2.2.1. Additional Specifications

The configuration adopted is called Thick client or Fat client, in which, in addition to the presentation logic, in the client node is allocated also a part of the business logic. This can be an advantage for the mobile application that is able not only to display functions but also to compute functions, for example checking the correct compilation of the fields directly from the client application.

To fasten the communication, cache is used on the client-side. This introduces a good advantage on performance given to the fact that part of the communication over the internet can be avoided when the cache already contains the requested data: this limits the traffic over the network and decreases the load on the server. Obviously, on the other hand, a mechanism must be implemented to invalidate data on the cache when they become obsolete.

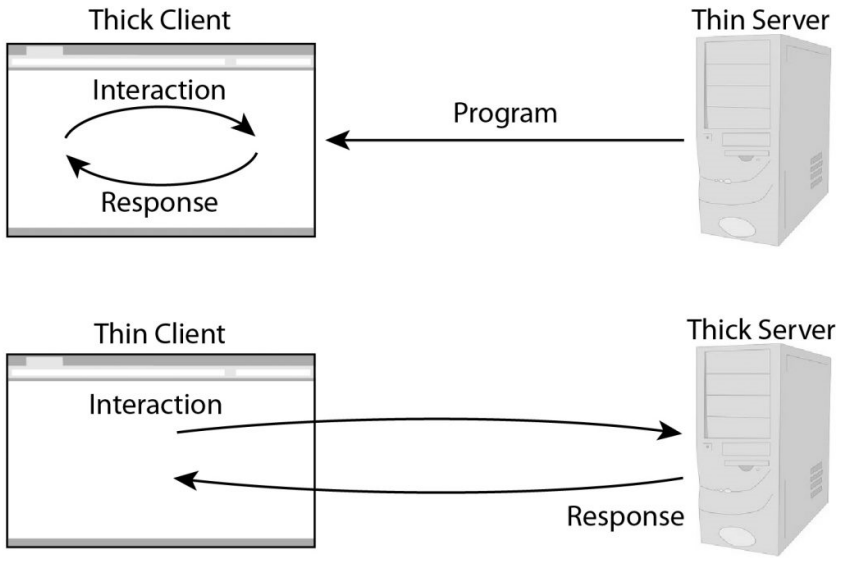


FIGURE 4 - THICK CLIENT

The Application Server and the Web server have been replicated two times to guarantee the reliability of the application if one of them goes down.

In addition, also the Redirector component has been replicated to avoid overload of requests at the same time. In the diagrams, nevertheless, it is reported like a unique element for the sake of simplicity.

2.3. Deployment View

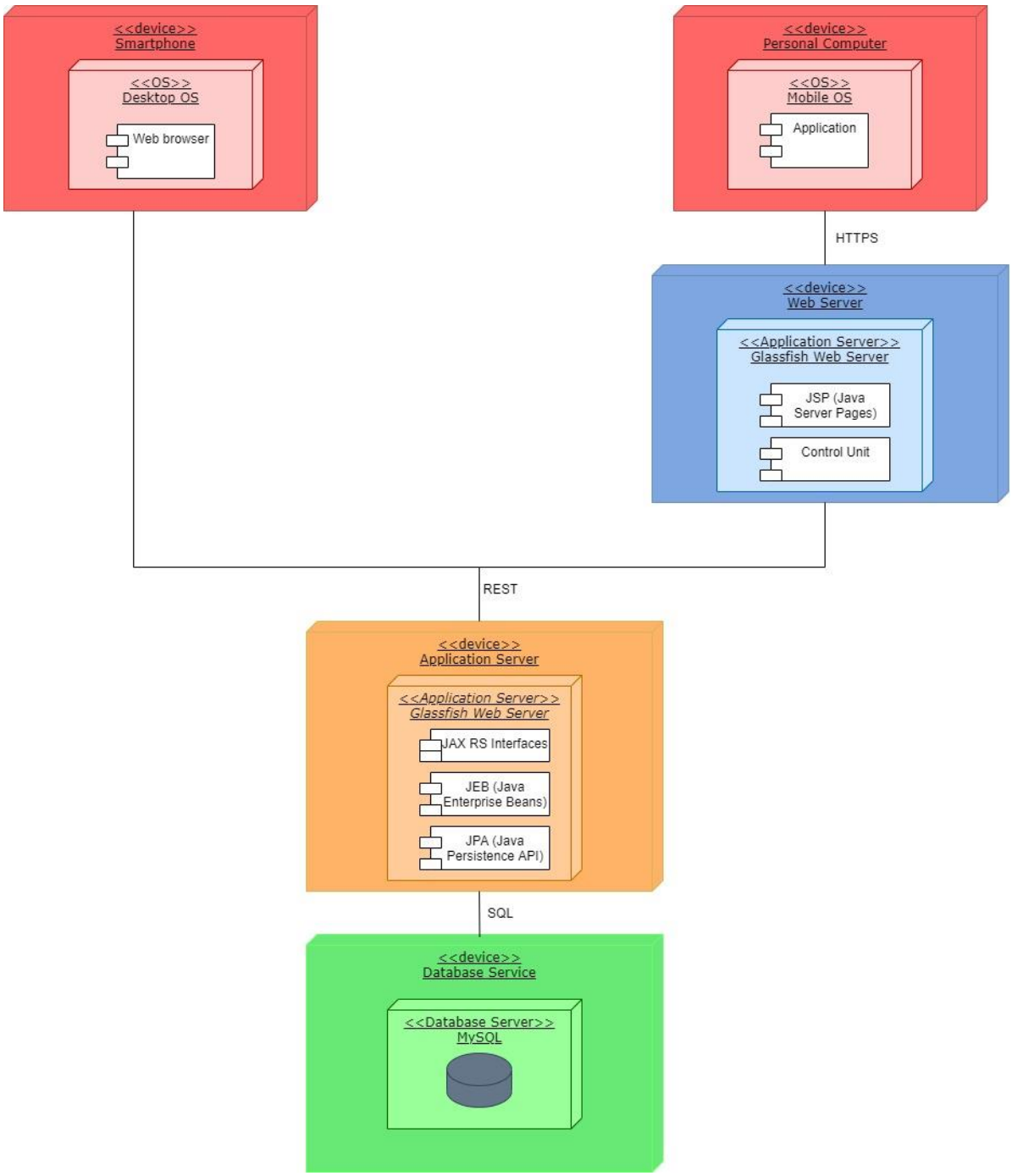


FIGURE 5 - DEPLOYMENT DIAGRAM

The deployment diagram shows the architecture of the system. A better explanation of components follows here:

- **Smartphone (or tablet)**
  Mobile phone applications communicate directly to the application layer. This could be used both from the customer user -to book an entrance ticket- and supermarket user -to scan an entrance ticket and view statistics analysis.

- **Computer**
  Web application could be used by customer user to book an entrance ticket provided that he prints a paper copy of the QR code. Web application could be also used by supermarket user, but only for statistics analysis.
  It does not communicate directly with application server, but it goes through web server.

- **Web Server**
  It is able to manage requests from the client or the browser. Communication between server and client takes place via the HTTP protocol. We use the Glassfish Web server. Glassfish is the open source Java EE Reference Implementation.
  In fact, we use **JSP** (Java Server Pages) that is a collection of technologies in Java; this is helpful to create dynamically generated web pages based, for example, on HTML.
  All data are transferred to Application server thanks to REST (transmission systems based on HTTP).

- **Application Server**
  It provides functionalities to develop and execute applications. Enterprise JavaBeans are helpful for the logic presentation of the application: they define different properties of the system like persistence, integrity or security. Instead, JPA deals with the management of persistence of DBMS' data.

- **Database Management System (DBMS)**
  It is the software that connects end user, applications and database itself in order to collect and mange data such as usernames, password, booking, or statistical analysis. In particular, we choose MySQL.

2.4. Runtime View

2.4.1. Login Request

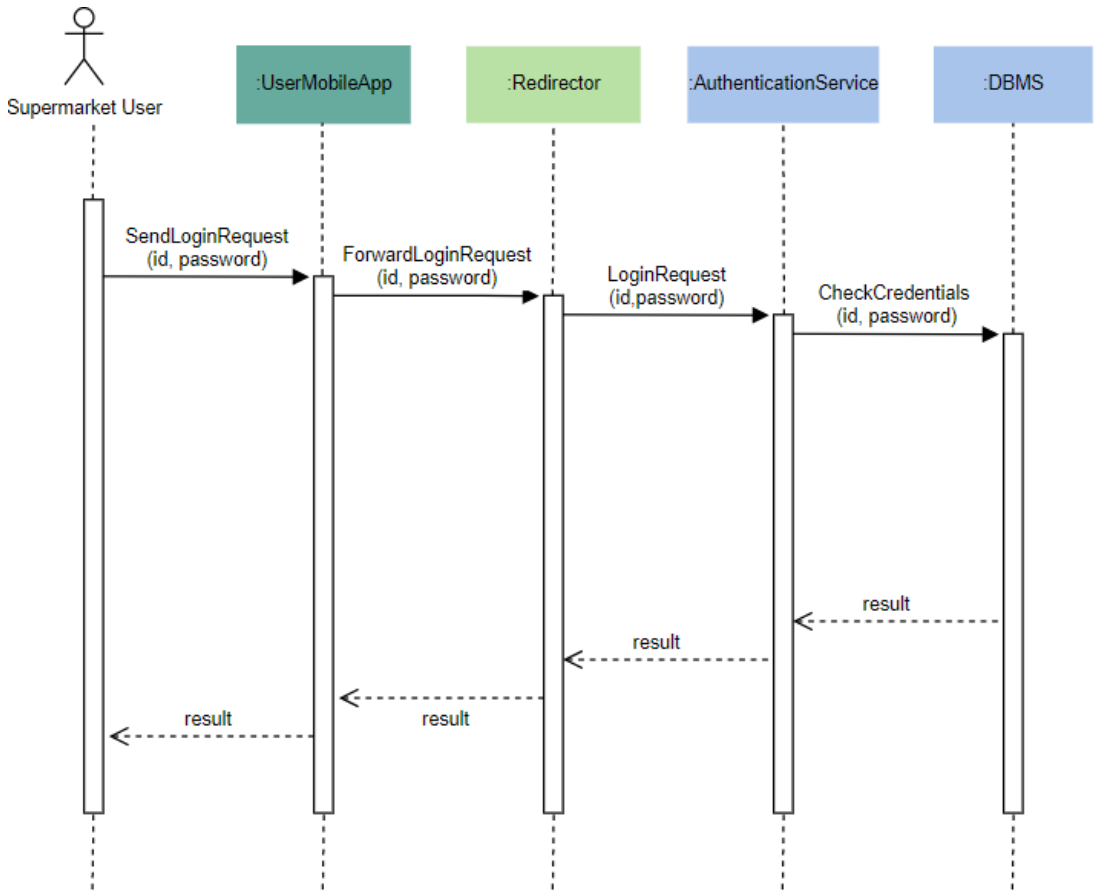


FIGURE 6 - RUNTIME, LOGIN

The diagram shows the login operation done by the supermarket user. We assume that the user is an employee of the supermarket, who has to scan all entrance ticket with the application, so he uses a mobile device.

The user fills empty spaces with the identification code (one for each supermarket) and the password associated.

When the login request is sent, the Redirector will forward it to the Authentication Server; it controls if the identification code exist and if the password is correct according to data stored in DBMS.

The result of this operation is propagated back to the user. If the result is positive the user can enter into the application and see the home page, otherwise, he has to repeat the login operation.

The login operation done by the customer user is very similar because the component involved are the same; the only difference is that the user has to insert his email and password (there is not an identification code for him).

2.4.2. Book A Visit Request

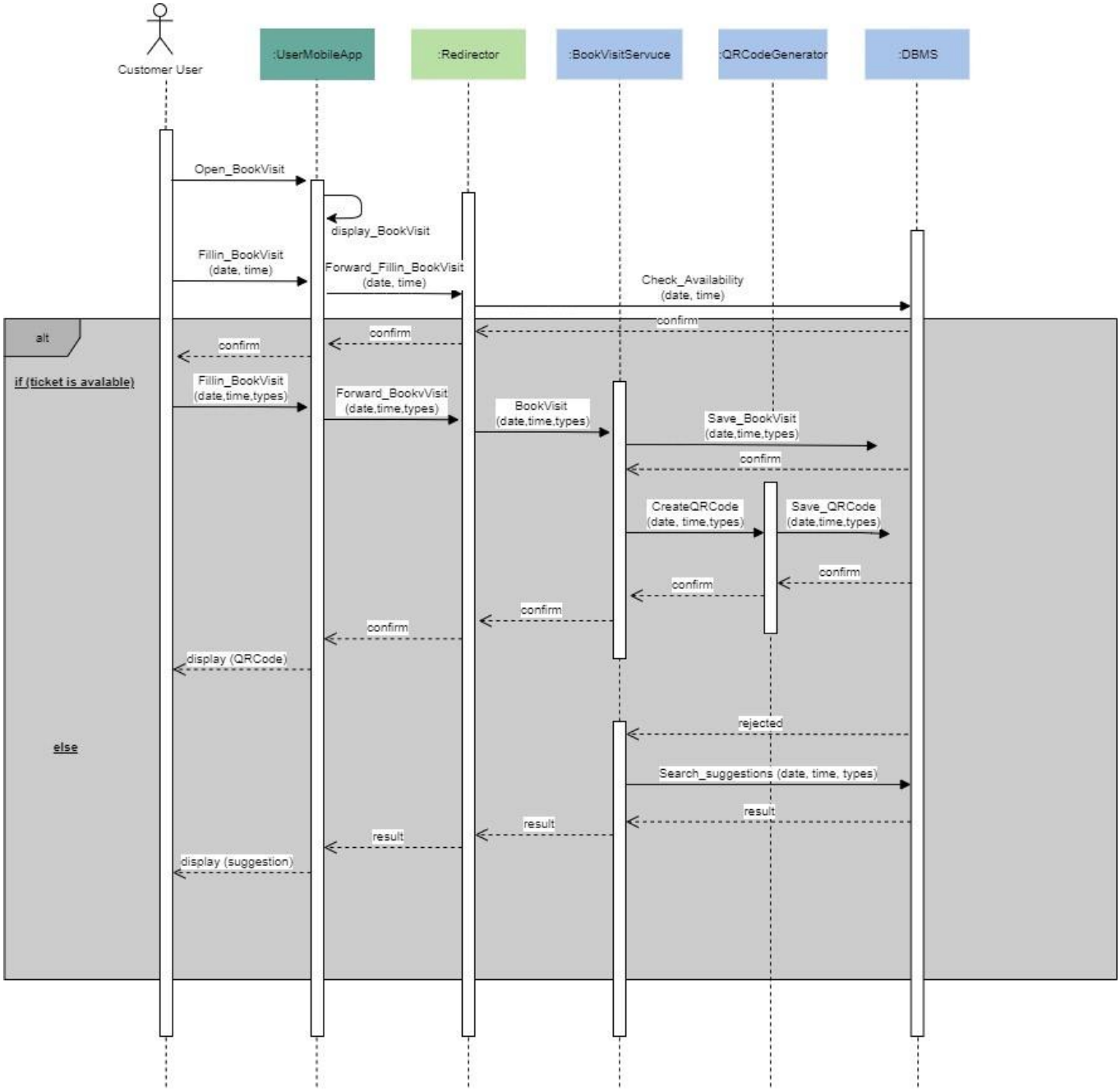


FIGURE 7 - RUNTIME, BOOK A VISIT

The diagram shows the booking operation done by the customer user. He can use both mobile application and web application, but in this case, we assume he uses the mobile one.

Once he is done the login and he has selected one of the nearest supermarkets, the user will visualize the “book a visit” button.

He has to fill empty spaces with date and time so he sends a request; the Redirector will forward it to the DBMS to check if the date and time are available.

If the operation is successful, the user has to insert also grocery types that he wants to buy, then the Book Service saves the visit into the DBMS. After a confirm message, the QR Code Generator creates a QRCode for the customer (he will use it as an entrance ticket). The confirm message and the QR Code will be visualized on the display.

If the operation is not possible, suggestions about different free slots appear on the display with the message of rejection; the user has to repeat the booking operation (insert the suggested date and time if he wants).

2.4.3. Queue Request

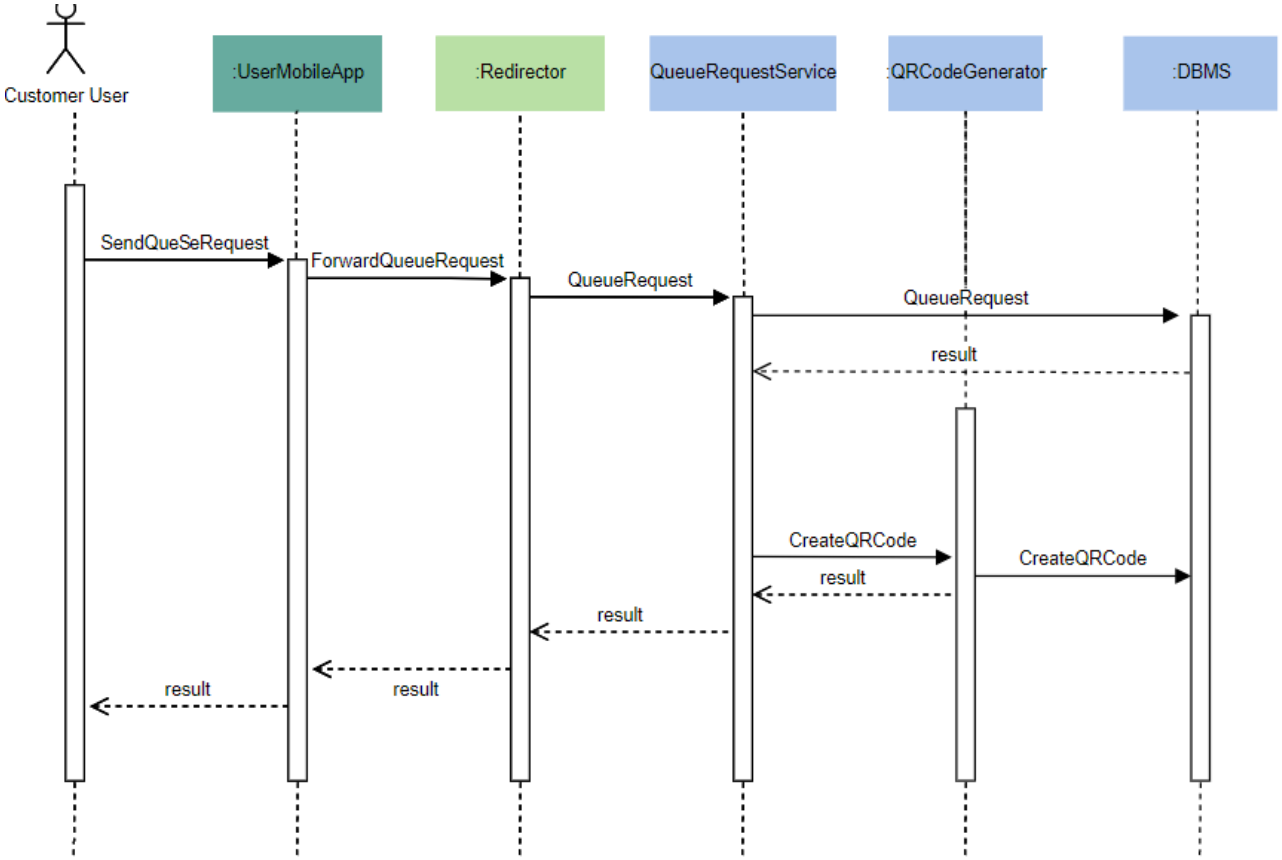


FIGURE 8 - RUNTIME, QUEUE REQUEST

The diagram shows the Queue Request done by the customer user. He can use both mobile application and web application, but in this case, we assume he uses the mobile one.

Once he is done the login and he has selected one of the nearest supermarkets, the user will visualize the “Line up” button. He sends the Queue Request through that and the Redirector will forward it to the Queue Request Service. The Queue Request Service saves the request into the DBMS.

Then the QRCode is created. The result of this operation (with the QR Code) is propagated back to the user.

2.4.4. Delete A Visit Request

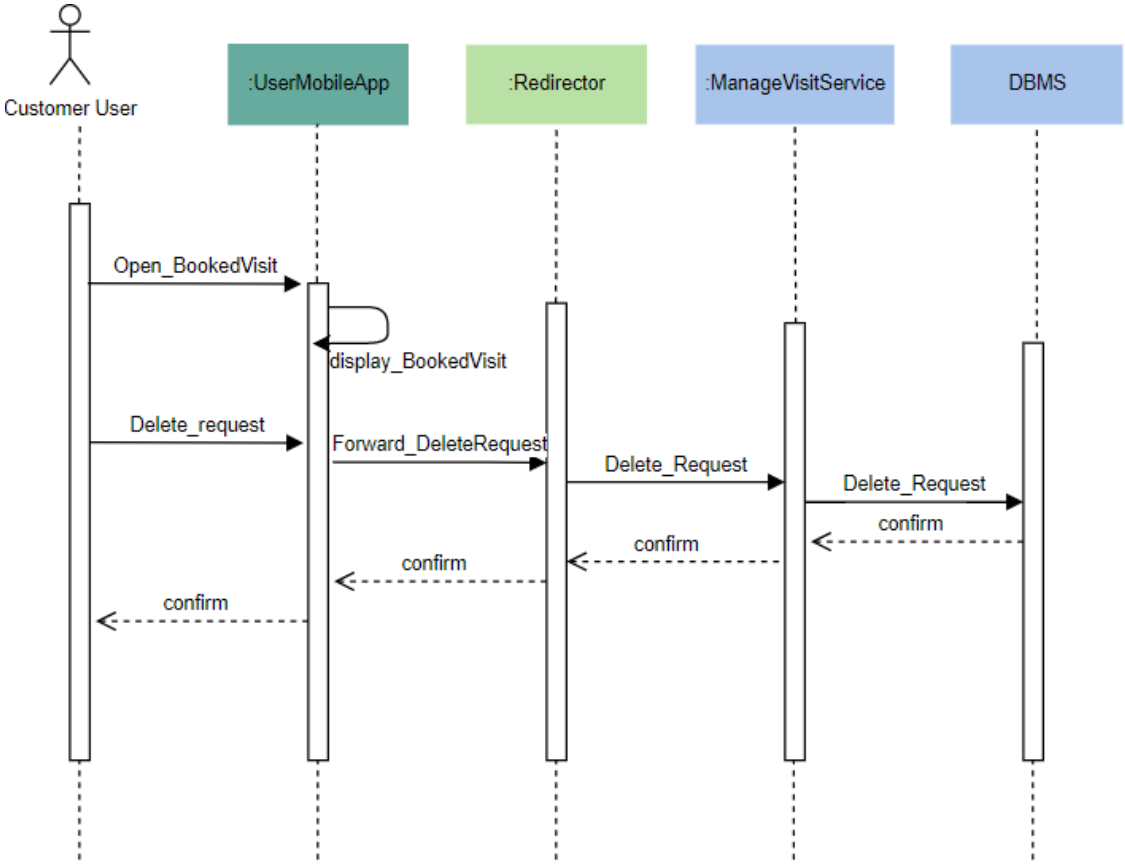


FIGURE 9 - RUNTIME, DELETE

The diagram shows the removal of a booked visit by a customer user. He can use both mobile application and web application, but in this case, we assume he uses the mobile one.

Once he is done the login and he has selected one of the nearest supermarkets, the user will visualize the “Booked visits” button.

He will visualize a list of booked visits with two possible operation buttons: delete operation and update operation. The user will send the Delete Request and the Redirector will forward it to the Manage Visit Service. The Manage Visit Service removes the visit from the DBMS. The confirmation of the transaction is propagated back to the customer user.

2.4.5. Statistics Request

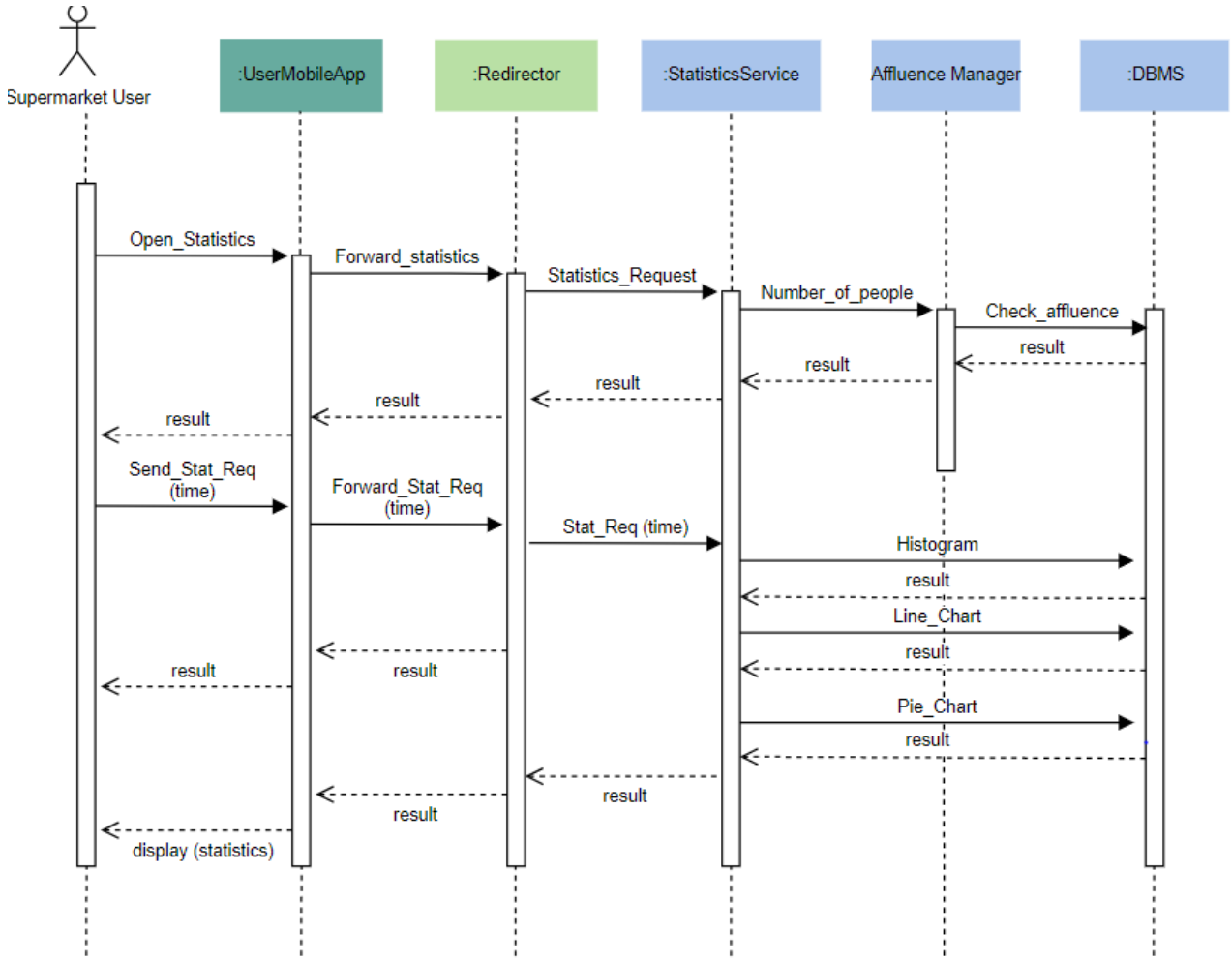


FIGURE 10 - RUNTIME, STATISTICS

The diagram shows the Statistics Request done by a Supermarket employee to monitor the affluence.

Once the user has done a login as supermarket, he clicks on “statistics” button, so he sends a Statistics Request.

The Redirector will forward the request to the Statistics Service which, thanks to the Affluence Manager, counts the number of people into the market at that moment.

When the user visualizes the Statistics Home Page, he has to select a time interval; the system will calculate analysis according to that interval.

When the user sends the statistical request in a specific time interval, the Statistics Service starts three different transactions:

- The creation of a histogram showing the mean number of costumers for each day.
- The creation of a line chart that shows the per hour affluences.

- The creation of a pie chart showing the most requested types of goods

At the end, the result of all these operations is propagated back to the customer user who visualizes Statistics Analysis on his display.

2.5. Component Interface

This paragraph shows the main methods belonging to the interfaces of the components. Every component has been previous described in chapter 2.2. (component diagram). A lot of methods have been previous seen in chapter 2.4 (runtime diagrams).

2.5.1. Internal Component Interfaces

AccessManager:

SubscriptionService:

`NewAccountRequest`: this procedure is used to request the creation of a new account. This takes as parameters all the personal user data required to register:

- Name, surname, email, telephone number and password for a customer user
- Id code of the supermarket (unique), address and password for supermarket user.

The return value for the procedure is an error in case of invalid data submission or a confirmation message in case of success.

AuthenticationService:

`LoginRequest`: This function allows any registered user to log into the system using his email (or IDcode) and password. If the credentials are correct, the user enters the application; otherwise, an error is returned.

`DeleteProfileRequest`: This procedure is used to permanently delete a user account from the system; this takes only a user-identifying token to recognize the requester; the return value for this procedure is a message of confirm.

`EditProfileRequest`: This function allows users to edit their profile information. This takes a user-identifying token to recognize the requester and, also, the password. The return value for the procedure is an error in case of invalid data submission or a confirmation message in case of success.

CostumerRequestManager

QueueRequest:

`QueueRequest`: This function is used to lines up for a selected supermarket. If the operation is successful, the function returns the QRcode ; otherwise, an error is returned.

BookAVisit:

`BookaVisit`: This procedure allows user to book an entrance ticket for a selected supermarket in a specific date and time. If the booking is successful, the function returns the QRcode. Otherwise, the system suggests other time slot to the user who has to repeat the operation.

ManageVisit:

`DeleteRequest`: This function is used to delete a previous booked entrance ticket. The return value for this procedure is a message of confirm.

`EditRequest`: This function is used to modify a previous booked entrance ticket. The same process of BookVisit will be followed.

SupermarketRequestManager

QRCodeService

`QRcodeEntrance`: This procedure is used to scan a QR Code of a customer that is entering in the supermarket, so he will be counted as "person into the shop". The return value for this procedure is a message of confirm.

`QRcodeExit`: This procedure is used to scan a QR Code of a customer who is leaving the supermarket, so he will be removed from the counted "people into the shop". The return value for this procedure is a message of confirm.

StatisticsService

`StatisticRequest`: This function can be used only by supermarket user to see how much people there are into the supermarket. The return value for the procedure is a "statics analysis" home page in which there is the number of people into the market at that moment. Once the user insert a valid time interval the system returns:

- `Histogram`: This function creates a histogram showing the mean number of costumers for each day .
- `LineChart`: This function creates a line chart that shows the per hour affluences.
- `PieChart`: This function creates a pie chart showing the most requested types of goods.

AffluenceManager

`CheckAffluence`: This procedure counts the number of people that are into the supermarket at a precise moment, using information contained into the database.

`SendNotification`: This procedure sends notifications about the affluence to the supermarket user.

2.5.2. External Component Interfaces

QRcodeGenerator

`CreateQRCode`: This function creates a unique QR code using that contains information about date and time of the booked entrance ticket. If the procedure is successful, it returns a QR Code to the user; otherwise, it returns an error message.

GoogleMapsService

`SearchSupermarkets`: This function searches the five nearest supermarkets with respect to the user position (known thanks to GPS on his device). The result of this operation is a map in which all the nearest supermarkets are visible, so user can select one of them.

2.6. Selected Architectural Styles and Patterns

The following architectural styles and patterns have been used:

- **Client and server**

  The client-server model is used at different levels:

  - The mobile applications are clients with respect to the Application Server.
  - The user's browser (client) communicates with the Web Server (server that provides the requested web page).
  - The Web Server is a client when it communicates with the Application Server in order to process user's requests.
  - The Application Server plays the role of a client when it queries the Database.

- **Three-Tier Architecture**

  It is a well-established software application architecture that organizes applications into three logical and physical computing tiers:

  - The presentation tier, or user interface.
  - The application tier, where data is processed.
  - The data tier, where the data associated with the application is stored and managed.

  Thanks to this separation, each tier runs on at least one dedicated server. The services of each tier can be customized and optimized without impact the other tiers. This, also, improvers security, scalability and reliability.

- **Model-View-Controller**

  The web, the mobile and the on-board applications follow the Model-View-Controller software design pattern.

  MVC is particularly adapted for the development of both web and mobile application in an object-oriented style of programming as java. This allows to separate the application into three

communicating and interconnected parts; thanks to MVC, it is possible to create components independently of each other and simultaneous development is simplified.

- **Thick Client**

  See 2.2. paragraph.

3. User Interface Design

The following mockups show how it should look the mobile application will look both from Costumer User's and Supermarket User's point of view. The last mockups show how the Web Application accessed from laptop should like.

When the user opens the application, he will be asked to declare if he intends to use the application as a Costumer User or as a Supermarket User, then he will be asked if he wants to register o log in to the application.

FIGURE 11 - REGISTRATION AND LOG IN

3.1. Costumer User Interface

Since the user logs as a customer, he will visualize Costumer User Home Page, where are available different functionalities.

FIGURE 12 - COSTUMER USER HOME PAGE

The costumer selects “Supermarket Around” option, retrieves the five closest supermarkets and selects one of the available supermarkets.

FIGURE 13 - SUPERMARKET AROUND

The costumer user asks to Book a Visit for the selected supermarket.

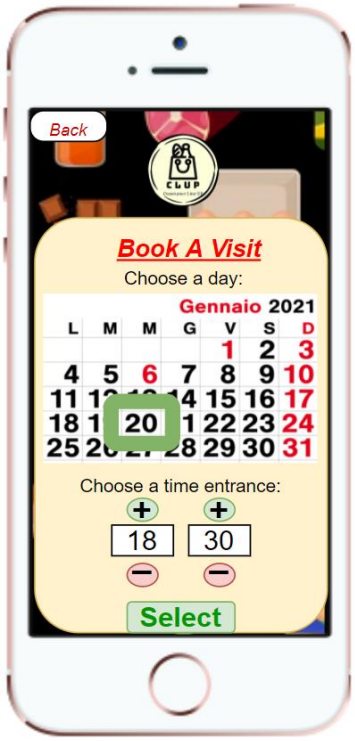


FIGURE 14 - COSTUMER SELECT DATE AND TIME

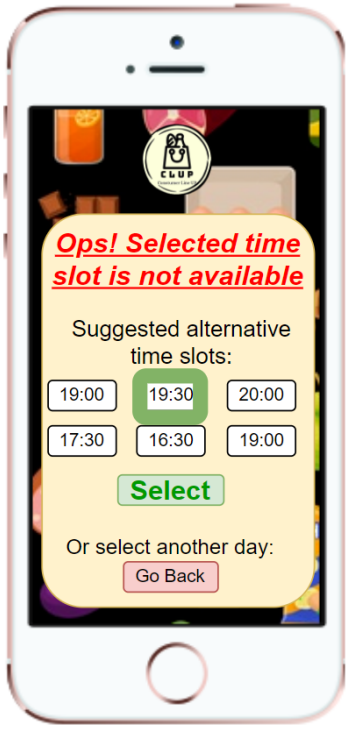


FIGURE 15 - SYSTEM PROVIDES SUGGESTIONS IF TIME SLOT IS NOT AVAILABLE

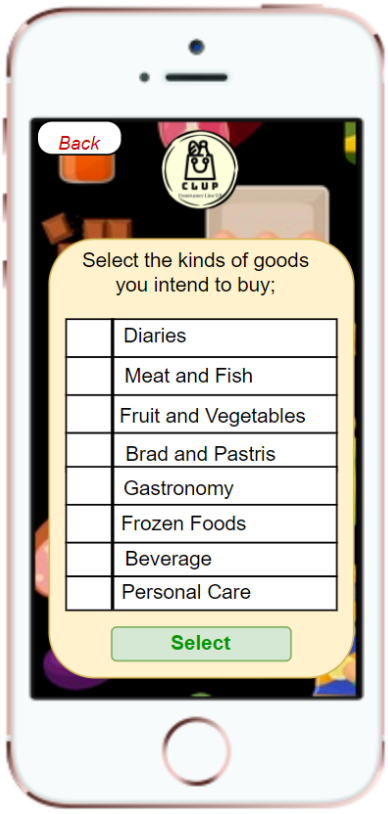


FIGURE 16 - COSTUMER DECLARES THE GROCERY TYPES

Costumer asks to be put in line for the selected supermarket; before commit the operation he will select the types f grocery goods he intends to buy.

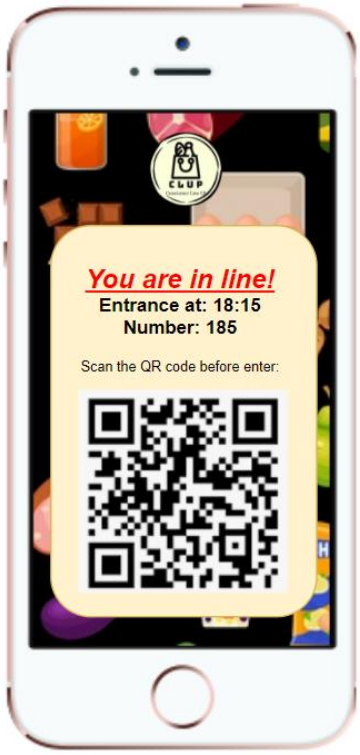


FIGURE 17 - QUEUE REQUEST

The Costumer User selects “Booked Visits” options in the home page.

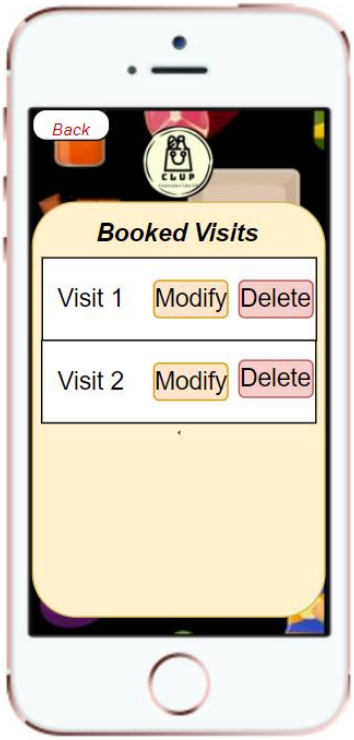


FIGURE 18 - BOOKED VISITS

3.2. Supermarket User Interface

FIGURE 19 - SUPERMARKET USER HOME PAGE

Supermarket User selects Statistic Analysis function.

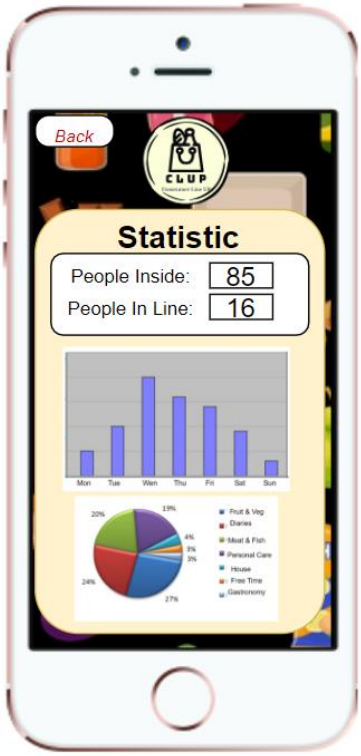


FIGURE 20 - STATISTIC ANALYSIS

3.3. Web Application Interface

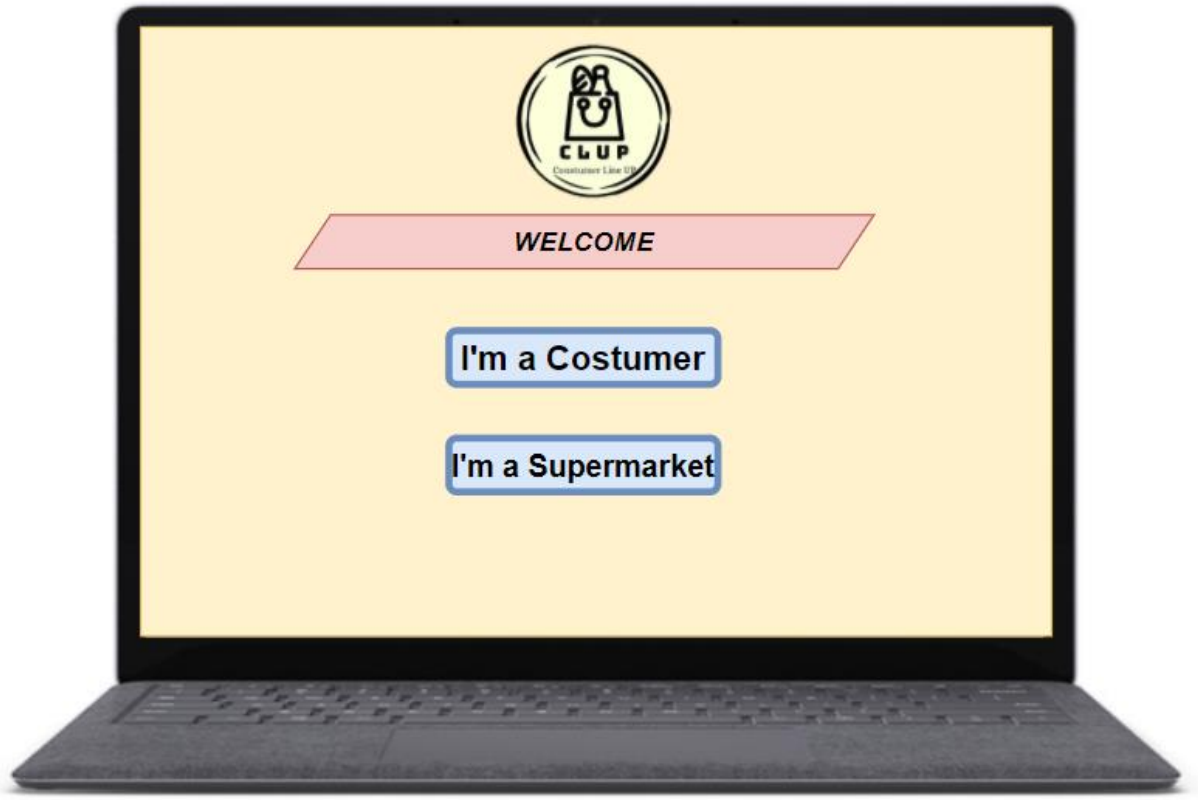


FIGURE 21 - WEB APP HOME PAGE

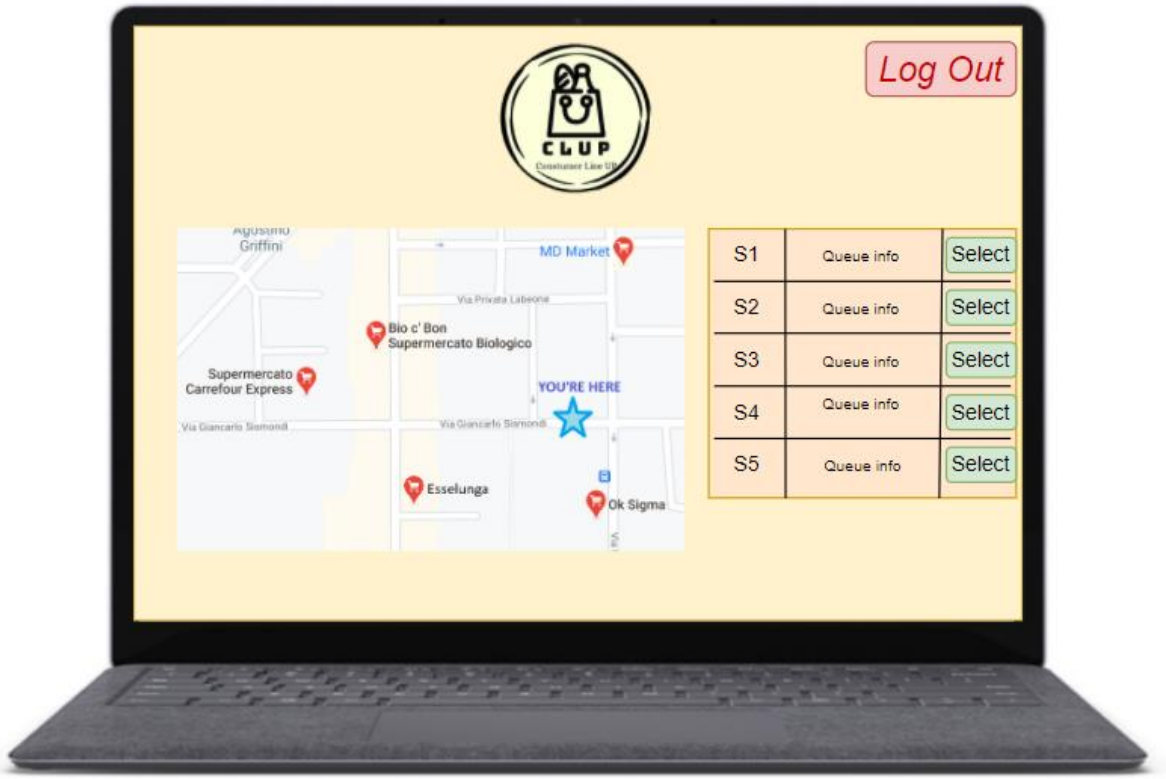


FIGURE 22 - WEB APP SUPERMARKET AROUND PAGE

4. Requirements Traceability

This section binds the goals specified in RASD with design components.

G1	**Allow user to check for the number of people in line for the five closest supermarkets**
	Requirements: R1 – R2 – R12 – R19 – R20
	Components: • UserMobileApp • AccessManager [SubscriptionService] • CustomerRequestManager[QueueRequestService] • GoogleMapsServices • DBMSServices

G2	**Allow user to know the waiting time for accessing the supermarket**
	Requirements: R1 – R2 – R4 – R20 – R21
	Components: • UserMobileApp • AccessManager [SubscriptionService] • CustomerRequestManager[QueueRequestService] • GoogleMapsServices • DBMSServices

G3	**Allow user to line up for a selected supermarket**
	Requirements: R1 – R2 – R4 – R5 – R11 – R12 – R13 – R15 – R18
	Components: • UserMobileApp • AccessManager [SubscriptionService] • CustomerRequestManager[QueueRequestService] • QRcodeGenerator • GoogleMapsServices • DBMSServices

G4	**Allow user to book a visit to the supermarket in the following days**
	Requirements: R1 – R2 – R4 – R6 – R7 – R8 – R11 – R12 – R13 – R15 – R16 – R18
	Components: • UserMobileApp • AccessManager [SubscriptionService] • CustomerRequestManager[BookVisitService] • QRcodeGenerator • GoogleMapsServices • DBMSServices

G5	**Allow user to delete a visit to the supermarket**
	Requirements: R1 – R2 – R10 – R11 – R25
	Components: • UserMobileApp • AccessManager [SubscriptionService] • CustomerRequestManager[ManageVisitService] • DBMSServices

G6	**Allow user to modify a visit to a supermarket**
	Requirements: R1 – R2 – R7 - R8 - R9 - R11 - R13 - R15 - R25
	Components: • UserMobileApp • AccessManager [SubscriptionService] • CustomerRequestManager[ManageVisitService] • DBMSServices

G7	**Inform the supermarket about the number of people currently inside the shop**
	Requirements: R2 - R3 - R14 - R17 – R24 - R26 - R27
	Components: • WebApp • WebServer • AccessManager [AuthenticationService] • SupermarketRequestManager[QRcodeService] • AffluenceManager • DBMSServices

G8	**Inform the supermarket about the number of people in line**
	Requirements: R2 – R3 – R14 – R19 – R20 – R24 – R26
	Components: • WebApp • WebServer • AccessManager [AuthenticationService] • SupermarketRequestManager[QRcodeService] • AffluenceManager • DBMSServices

G9	**Menage the entries in such a way that the number of people currently inside the supermarket does not exceed the legal safety limit**
	Requirements: R2 – R3 – R14 – R17 – R24 – R26 – R27 – R28
	Components: • WebApp • WebServer • AccessManager [AuthenticationService] • SupermarketRequestManager[QRcodeService] • AffluenceManager • DBMSServices

G10	**Perform statistical analysis on the information collected in the database**
	Requirements: R23 – R29
	Components: • WebApp • WebServer • AccessManager [AuthenticationService] • SupermarketRequestManager[StatisticsServices] • AffluenceManager • DBMSServices

G11	**Allow the supermarket to visualize the statistical analysis of collected data**
	Requirements: R2 - R3 - R22 – R23
	Components: • WebApp • WebServer • AccessManager [AuthenticationService] • SupermarketRequestManager[StatisticsServices] • AffluenceManager • DBMSServices

R1	The system access user's GPS position
R2	Users register to the application by filling a form with mandatory fields
R3	Supermarkets are certified with an authentication
R4	Costumer, to get their entry ticket to the supermarket, must select the types of grocery goods they're going to buy
R5	System allows customer line up for the shop by putting the costumer in the last place of the queue
R6	Customer selects the date and the time for booking the visit to the supermarket
R7	System retrieves the free time slots according to the date selected by the user, if the chosen time slot is not available (suggestions)
R8	System generates the booked visit according to the date and time selected by the user
R9	Customer users can modify the time slot of previously booked visits
R10	Customer users can delete their booked visit
R11	The system sends notifications to the users (both costumers and supermarket)
R12	The system evaluates the five closest supermarkets, according to user's position
R13	The system generates the QR code
R14	The system reads the QR code
R15	User can retrieve the QR code on the user interface
R16	The system computes the average time of visit according to the grocery types selected by the user
R17	The system computes the exact number of people currently in the shop, according to the number of entries and exits
R18	The system allows user to select one from the five nearest supermarkets
R19	The system updates the queue automatically
R20	The system computes the exact number of people queuing for a supermarket
R21	The system evaluates the mean waiting time to access the supermarket, according to the number of people waiting in line and the mean visit time of those that are inside the supermarket
R22	Supermarket users are allowed to access to statistical analysis
R23	The system builds statistics according to the data contained in the database
R24	The system asks to the supermarket user to access to the camera in order to scan the QR code
R25	The user can retrieve the list of his future booked visits
R26	Customers can access the supermarket only with a valid QR code
R27	Costumers must scan their QR code to exit from the shop
R28	Supermarket declares the maximum number of people that can access the shop
R29	The system stores information about entries, booked visits, affluences and duration of each visit in a database

5. Integration and Testing

5.1. Overview

The following chapter focuses on program integration and testing. Since it is not possible to develop an error-free software, the purpose of this process is to find the maximum number of possible faults and bugs at the beginning of development process, in order to minimize the cost associated to their repair.

Integration and Testing processes work in parallel: while different components are integrated the integration is tested. We assume that each module to be integrated is correct if considered in isolation (the module has already been successfully tested with Unit Testing by the developers) and the integration is performed incrementally as soon as the components are released, in order to facilitate bugs tracking.

5.2. Integration and Testing Strategy

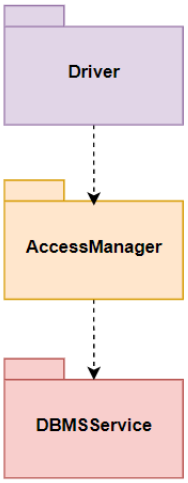
The entire system must be integrated and tested; to achieve this objective a Bottom-Up approach has been chosen. Bottom-Up strategy consists in integrating lower-level modules first. These modules are then further used to facilitate the integration of higher-level modules. The process continues until all modules at top level are integrated. Once the lower-level modules are implemented, tested and integrated, then the next level of modules are formed.

The integration order of the modules of the Application Server (see Figure X, page YZ) is the following:

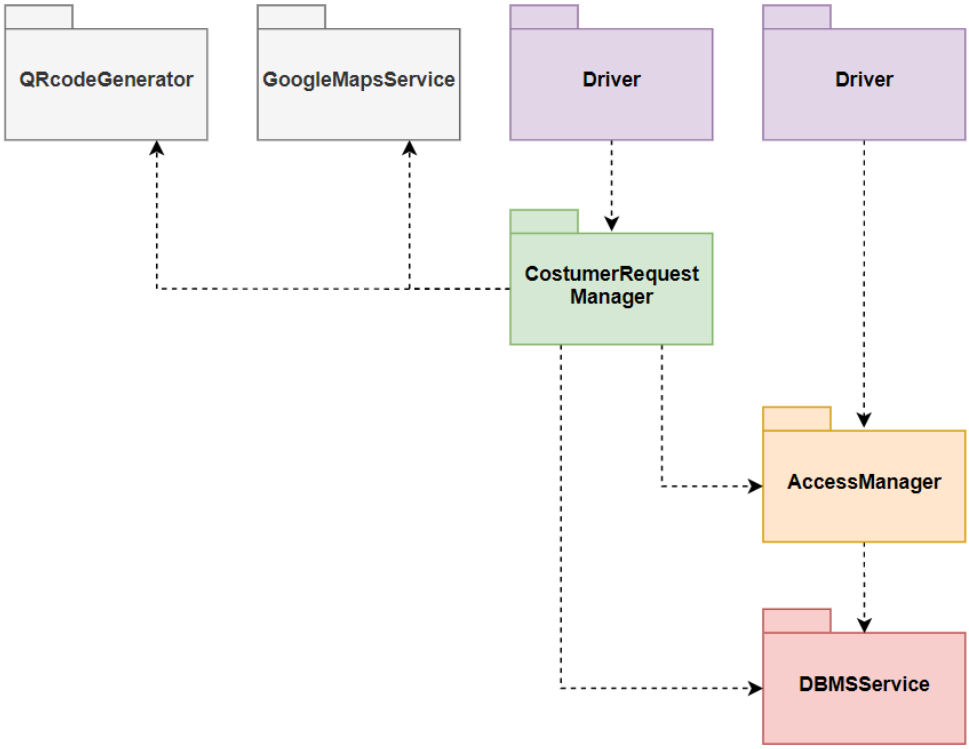
1. DBMSService
2. AccessManager
3. CostumerRequestManager
4. SupermarketRequestManager
5. AffluenceManager
6. Redirector

As the testing goes in parallel with the integration, testing order of the modules is the same. In particular, as we start from the leaves of the hierarchy of the modules, we must construct Drivers for each module as in Unit Testing.

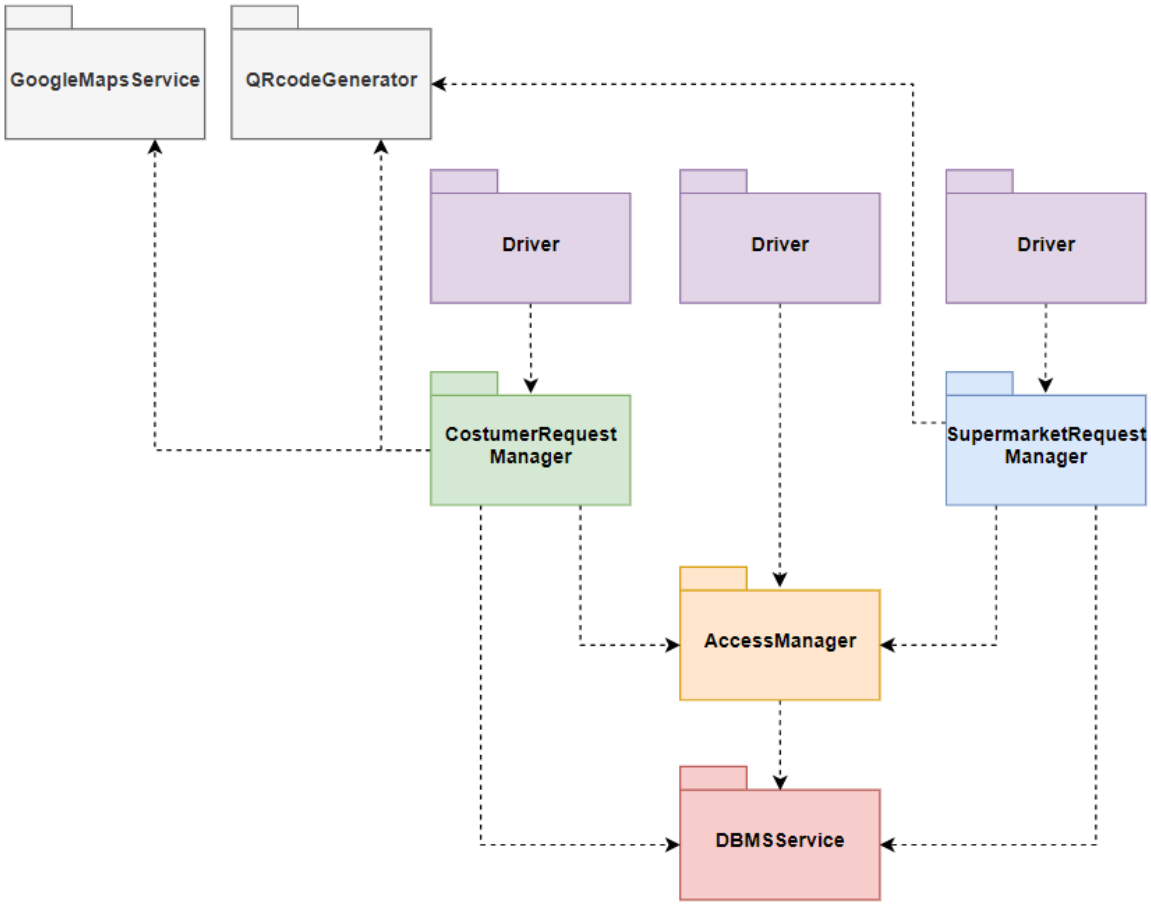
1. First step consists in integrating and testing AccessManager component with DBMSService, which interacts with all other components and allows the access to the Data Base. AccessManager component is fundamental because other components (CostumerRequestManager and SupermarketRequestManager) rely on it to provide some services: in fact, to perform certain functions, the User must be logged in and a User Session must be created.



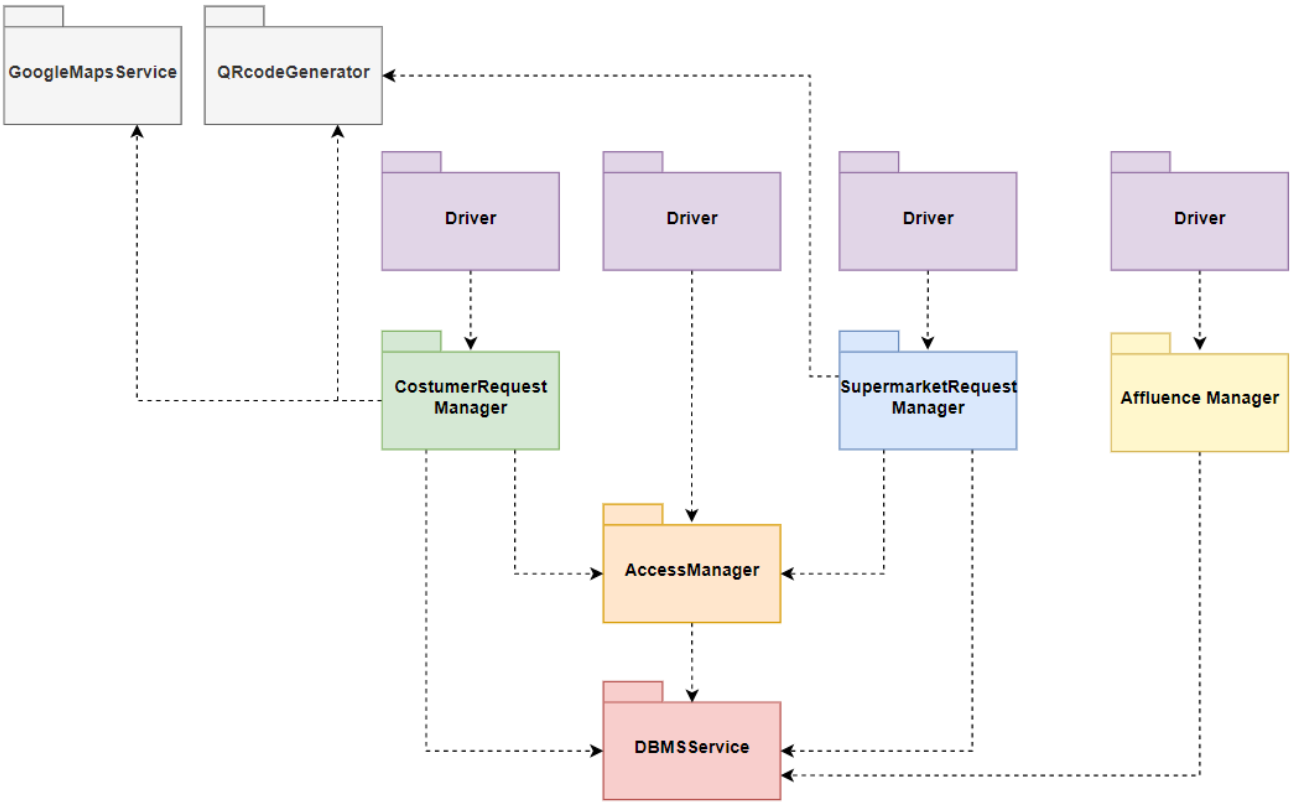
2. Then, we can integrate and test CostumerRequestManger component, this component interfaces with two external software components that allows to perform specific functions. It is important to notice that GoogleMapsService and QRcodeGenerator will not be unit tested (lack of Driver module) because are offered as a service from trusted providers: they only need to be integrated in the system through the CostumerRequestManager.



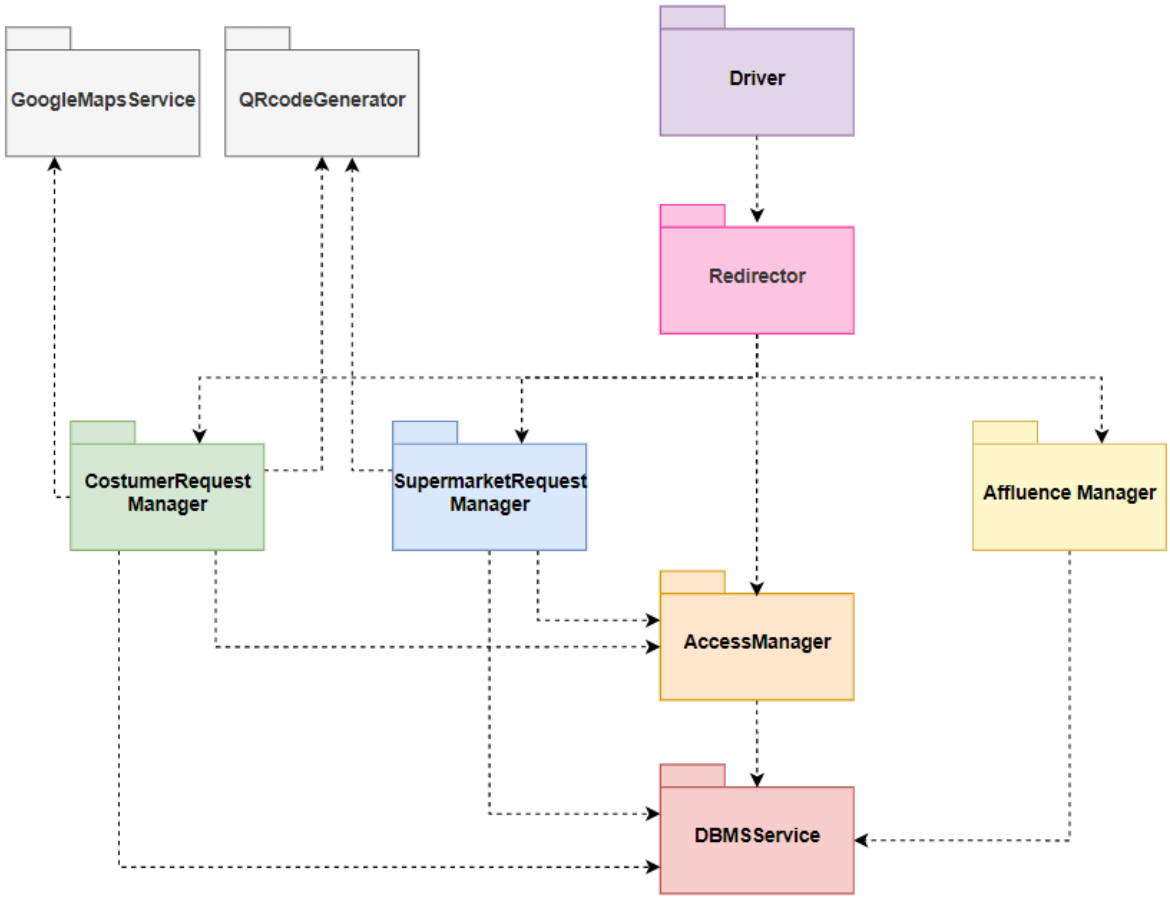
3. Third step consist in integrating and testing SupermarketRequestManager component, that allow supermarket user to perform specific functions. It also interacts with QRcodeGenerator external software to read and create QRcodes.



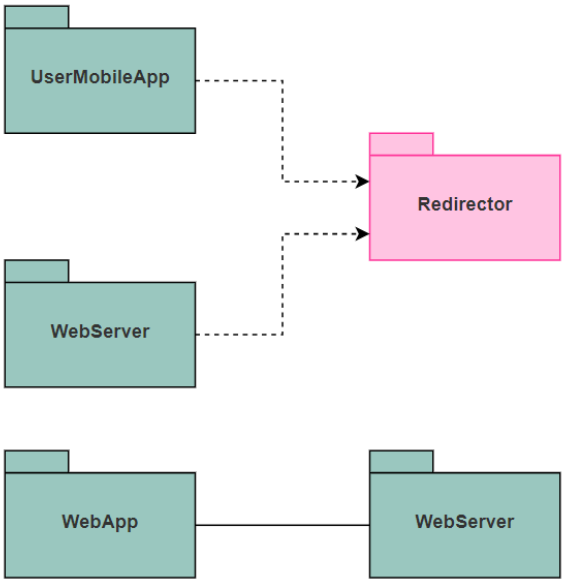
4. Then we can integrate and test AffluenceManager. This component is smaller and communicates only with the DBMSService, it was decided to integrate it later to prioritize the implementation of larger and more complex components (see point 2. 2nd 3.).



5. Then the different drivers in the previous diagram are substituted by the Redirector which must be implemented, unit-tested and integrated with the other components of the application server. As previously said, its role is to allow the flow of called methods from the client to the server.



6. In conclusion, client-side components are integrated and tested in the system.



5.3. System Testing

Once the system is completely successfully integrated, we can proceed with system testing, to establish if the system is fulfilling all the requirements contained in the RASD, both functional and non-functional.

System testing will be performed by an independent team, with respect to the development team. The team will be asked to read the documentation and check, through the software execution, if it achieves all the requirements.

System testing consist of four phases, if the software product passes all four, it is ready to be delivered to the User for the final validation through the acceptance test.

1. Functional Test: proves that the product satisfies all the functional requirements stated in the RASD.

2. Performance test: allows to identify bottlenecks affecting response time, utilization and throughput. Through the performance test we can detect inefficient algorithms, optimise queries and solve hardware o software issues.

3. Load Test: exposes bugs such as memory leaks, mismanagement of memory, buffer overflows and identifies upper limits of components. It is performed by increasing the load until the threshold. Load testing consists in modelling the expected usage of a software program by simulating multiple users accessing the program concurrently.

4. Stress Test: consists in Trying to break the system under test by overwhelming its resources. It allows to make sure that the system recovers gracefully after failure.

5.4. Additional Specifications

Testing activities above descripted are fundamental in order to validate, find many faults and bugs as possible in the executable and to check whether the system fulfils all the requirements and goas stated in the RASD document. This procedure leads us to complete the first release of the product that will be presented to the end user for the User Acceptance Test.

However, it's necessary that all the stages of the software production are controlled: not only the steps concerning the building of the executable system, but also all the specification activities that produce the specification documents.

Analysis activities are foreseen at any stage of the development process. These activities will include:

- Reviews:
    - A review between peers is scheduled every 10 workdays, starting from the beginning of the work. The team members are asked to revise each other's work in order to check the correctness of the product and be always update about the team's work.
    - A formal inspection is required before the final delivery of the product. Formal Inspection will be performed by external professional teams in order to check the correctness of the product according to a checklist.

  In particular, faults found in review activities are not used in personnel evaluation.
  Reviews will be used both for code and documentation production.

- Automated Static Analysis:
  This activity is required in code production, it is based on variable definition and use across the code, to check for possible errors in variables utilization and for code optimization.

In addition, developers are required to write a well-commented code, to make it easier to understand the code for quality assurance activities.

6. Effort Spent

Alghisi Marianna

TOPIC	HOURS
Discussion on first part	2 h
Architectural design	1 h
Discussion on second part	2 h
Component diagram	2 h
Component interface	1 h
Selected architectural styles and patterns	1 h
User interface design	5 h
Requirement traceability	1 h
Integration and testing	5 h
Document Composition	3 h

D'Ascoli Gabriele

TOPIC	HOURS
Discussion on first part	2h
Architectural design	1h
Discussion on second part	2h
Overview & high-level architecture	2h
Component diagram	5h
Description of component diagram	3h
Selected architectural styles and patterns	1h
Requirements traceability	4h
Documents revision	1h

Pasturensi Martina

TOPIC	HOURS
Discussion on first part	2h
Architectural design	1h
Discussion on second part	2h
Component diagrams	1h
Deployment view	2h
Runtime View	6h
Component interface	3h
Selected architectural styles and patterns	2h
Integration and testing	1h

7. References

- All the diagrams have been made with https:// www.draw.io