

M.Sc. Computer Science and Engineering

Software Engineering 2 Project

**CLup - Customers Line-Up**

**Design Document**

Leoni Luca, Locarno Silvia, Minotti Luca

10th January 2021

Version 1.0

GitHub Repository: https://github.com/lucagrammer/LeoniLocarnoMinotti

**Contents**

[1 Introduction 3](#_Toc58149793)

[1.1 Purpose 3](#_Toc58149794)

[1.2 Scope 3](#_Toc58149795)

[1.3 Definitions, Acronyms, Abbreviations 3](#_Toc58149796)

[1.3.1 Definitions 3](#_Toc58149797)

[1.3.2 Acronyms 4](#_Toc58149798)

[1.3.3 Abbreviations 4](#_Toc58149799)

[1.1 Revision History 4](#_Toc58149800)

[1.2 Reference Documents 4](#_Toc58149801)

[1.5 Document Structure 5](#_Toc58149802)

[2 Architectural Design 6](#_Toc58149803)

[2.1 Overview 6](#_Toc58149804)

[2.2 Component View 8](#_Toc58149805)

[2.2.1 Application Component 8](#_Toc58149806)

[2.2.2 Web Server Component 9](#_Toc58149807)

[2.2.3 Application Server Component 9](#_Toc58149808)

[2.2.4 Data Components 12](#_Toc58149809)

[2.2.5 External services and infrastructures 14](#_Toc58149810)

[2.3 Deployment View 15](#_Toc58149811)

[2.4 Runtime View 16](#_Toc58149812)

[2.5 Component interfaces 24](#_Toc58149813)

[2.6 Selected Architectural Styles and Patterns 24](#_Toc58149814)

[2.6.1 4-Tier Architecture 24](#_Toc58149815)

[2.6.2 Thin Client 24](#_Toc58149816)

[2.6.3 Model-View-Controller 24](#_Toc58149817)

[2.6.4 Data Access Object 24](#_Toc58149818)

[2.6.5 Application Logic 24](#_Toc58149819)

[2.7 Other design decisions 24](#_Toc58149820)

[2.7.1 User Authentication and Password Storing 24](#_Toc58149821)

[2.7.2 Chain of Custody 24](#_Toc58149822)

[2.7.3 Relational Database 24](#_Toc58149823)

[3 User Interface Design 25](#_Toc58149824)

[3.1 UX Diagrams 25](#_Toc58149825)

[3.2 User Interface Mockups 25](#_Toc58149826)

[4 Requirements Traceability 26](#_Toc58149827)

[5 Implementation, Integration and Test Plan 27](#_Toc58149828)

[5.1 Development Process 27](#_Toc58149829)

[5.2 Implementation 27](#_Toc58149830)

[5.2.1 Implementation Plan 27](#_Toc58149831)

[5.2.2 Implementation Choices 27](#_Toc58149832)

[5.3 Integration 27](#_Toc58149833)

[5.3.1 Entry Criteria 27](#_Toc58149834)

[5.3.2 Elements to be integrated 27](#_Toc58149835)

[5.3.3 Integration Sequence 27](#_Toc58149836)

[5.4 Test Plan 27](#_Toc58149837)

[6 Effort Spent 28](#_Toc58149838)

[6.1 Leoni Luca 28](#_Toc58149839)

[6.2 Locarno Silvia 28](#_Toc58149840)

[6.3 Minotti Luca 28](#_Toc58149841)

[7 References 29](#_Toc58149842)

**Chapter 1**

# 1 Introduction

## Purpose

This document constitutes the Design Document (DD). It provides a more technical overview of the Requirement Analysis and Specification Document (RASD) of the system-to-be, describing the main architectural components, their communication interfaces and their interactions.

It will also present the implementation, integration and testing plan. This type of document is mainly addressed to developers since it provides a guide during the development process through an accurate vision of all parts of the software-to-be.

## Scope

As explained in the RASD, CLup aims to manage the queues to access stores in the Coronavirus era in a simple but effective way. Avoiding dangerous gatherings and unnecessary waste of time, CLup allows you to queue at a store directly from the application and receive a notification when it is time to reach the store, taking into account the time required to get to the shop using the indicated means of transport. Of course, fall-back options are available for people who don't have access to the required technology.

Moreover, the CLup Application also allows you to book a visit to a store in advance by indicating the date, time and product categories you are interested in. In addition, in case the desired slot is full, CLup provides useful suggestions for other slots available at the selected store or similar less crowded stores at the indicated date and time.

Furthermore, CLup can periodically notify you of the availability of slots in the day/time range in which you usually shop.

On the other hand, supermarket managers can easily keep access data under control and effortlessly ensure compliance with safety regulations.

## 1.3 Definitions, Acronyms, Abbreviations

### 1.3.1 Definitions

* **CLup System (or “The System”)**: refers to the whole system to be developed.
* **CLup Services (or “Services”)**: refers to the functionalities offered by the CLup System, such as the queue management mechanism and the booking service.
* **CLup Application (or “The Application”)**: refers to the application that makes CLup Services available everywhere.
* **QR** **Code**: quick response code, a type of matrix barcode.
* **Reservation ID**: a code that unequivocally represents either a position in the queue or a slot reservation.
* **Physical Ticket Dispenser**: a computer connected to the CLup System that distributes paper tickets. It acts as a proxy for Guests.
* **Guest**: a person who has not access to the CLup Application but still uses the CLup Service to access stores through Physical Ticket Dispensers.
* **Customer**: a person that uses the CLup Application and its services to access stores and book visits.
* **Client (or Store Client)**: either a Customer or a Guest.

### 1.3.2 Acronyms

* **RASD**: Requirement Analysis and Specification Document.
* **DD**: Design Document
* **UML**: Unified Modelling Language.
* **API**: Application Programming Interface.
* **PTD**: Physical Ticket Dispenser.
* **GPS**: Global Positioning System.
* **API**: Application Programming Interface.
* **ETA**: Estimated Time of Arrival

### 1.3.3 Abbreviations

* **[R.i]**: i-th requirement.

## Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Authors** | **Summary** |
| 1.0 | --/--/2020 | Leoni Luca  Locarno Silvia  Minotti Luca | First release |

## Reference Documents

* Specification document: Project Assignment A.Y. 2020-2021.pdf
* RASD of CLup
* Software Engineering 2 course slides
* Previous project examples:
* Specification document: Project Assignment A.Y. 2019-2020.pdf
* DD to be analyzed.pdf
* IEEE Standard on Requirement Engineering (ISO/IEC/IEEE 29148)

## 1.5 Document Structure

This document is structured as follows:

1. ***Introduction*** - A general introduction of the system-to-be. It aims at giving general, but exhaustive, information about what this document is going to explain.
2. ***Architectural Design -*** An overview of the high-level components and their interactions, with a focus on both static and dynamic view, helped by diagrams.
3. ***User Interface Design -*** A representation of how the User Interface will look like.
4. ***Requirements Traceability* *-*** An explanation about how the requirements defined in the RASD map to the design elements defined in this document.
5. ***Implementation, Integration and Test Plan* *-*** Identification of the order in which the sub-components of the system should be implemented, integrated and tested.
6. ***Effort spent*** - Effort spent by all team members shown as the list of all the activities done during the realization of this document.
7. ***References*** - References to documents that this project was developed upon.

**Chapter 2**

# Architectural Design

## Overview

The figure below represents a high-level description of the main components which constitute the System. They are organized in a 4-tier architecture thus facilitating maintainability and scalability. In particular, the *Web Server* and the *Application Server* have been separated mainly for security reasons. Further details about the architectural choices can be found in sections 2.6 and 2.7.

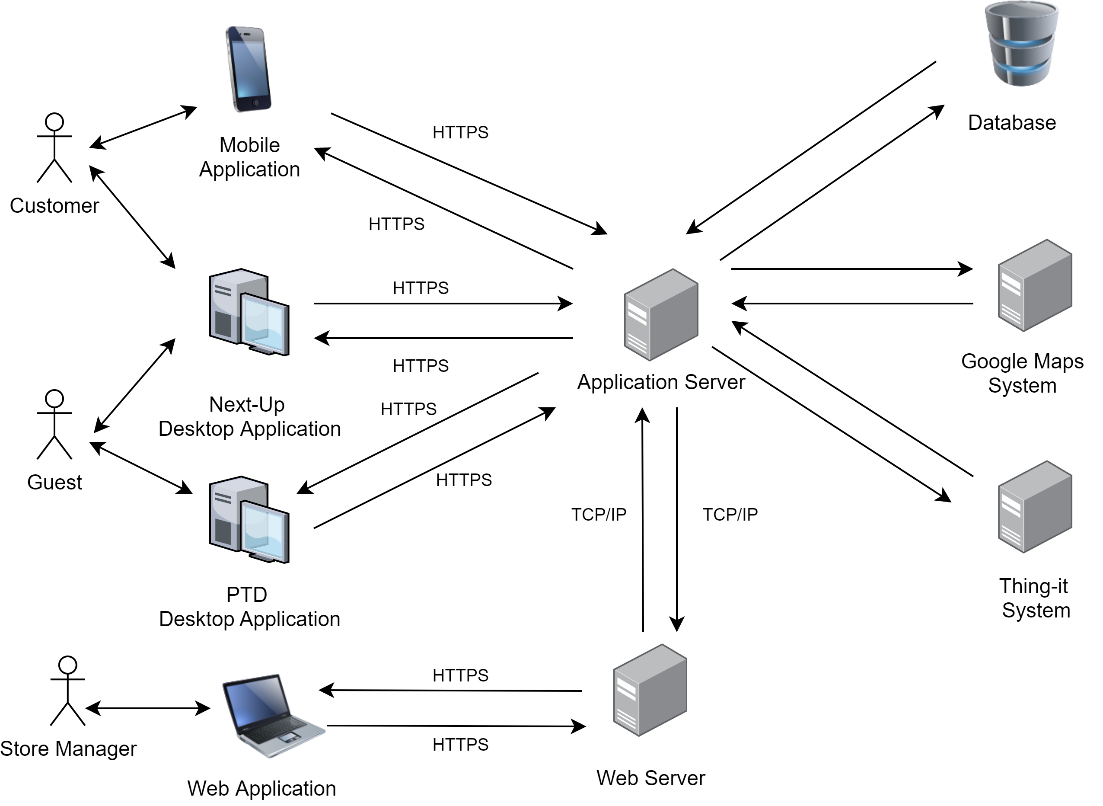


Figure 1: Overall architecture of the system

The main components of the Systems are the following:

* **Web Application**

A web application accessible through the Store Manager’s browser that allows him to access CLup Services. The web app will work with the most modern internet browsers, which communicate with the System by sending requests to the *Web Server*.

* **Customer Mobile Application**

An application installed on the Customer’s smartphone that allows him to use CLup Services anywhere. It communicates with the System forwarding every request to the *Application Server*. The mobile application will be available for both iOS and Android devices.

* **PTD Desktop Application**

An application installed on a touchscreen device located near the entrance to each store that allows Guests to use the “Line Up” Service. It communicates with the System forwarding every request to the *Application Server*. This application will be available for both macOS and Windows devices.

* **Next Up Desktop Application**

An application installed on a device located near the entrances of each store that allows both Customer and Guests to know which Reservation IDs are allowed to access the store. It communicates with the System through the *Application Server*. This application will be available for both macOS and Windows.

* **Web Server**

It is the back-end component of the *Web Application* that communicates with the Store Manager’s browser on one hand and with the components of the *Application Server* on the other (i.e. the Business logic tier).

* **Application Server**

It is the main back-end component of the System on which the business logic of the application takes place: it elaborates the requests coming from the end-user applications, it interacts with the data layer and the web layer and it communicates with the various external systems.

* **Database**

It is the component responsible for data storage. It can only be accessed by the Application Server.

* **External Systems**

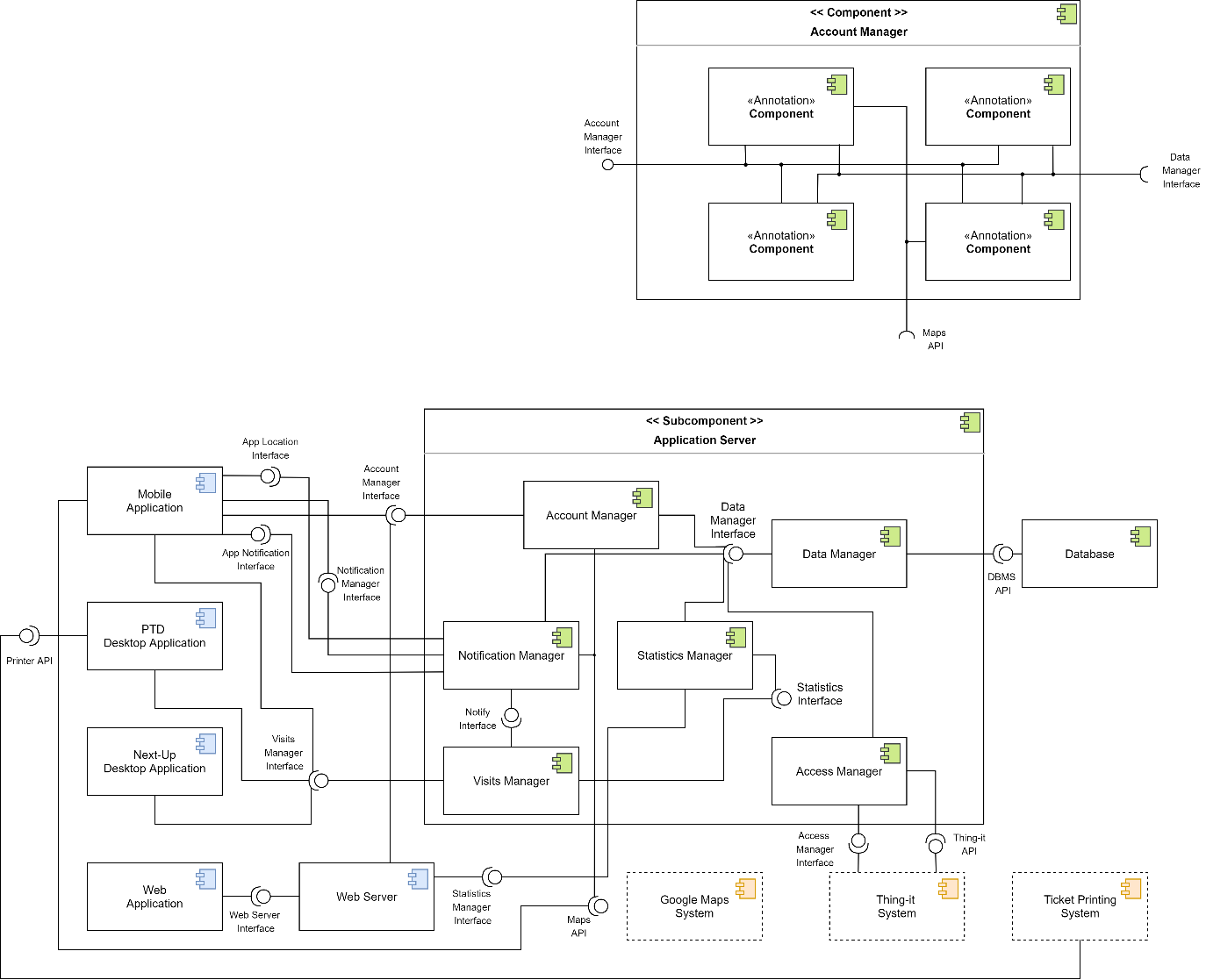
These are systems that provide functionalities that are not internally developed. *External Systems* interact mainly with the *Application Server*.

* **Google Maps System**: this system is responsible for providing the Map Services necessary for computing the ETA and supporting the identification of stores close to a Customer.
* **Thing-it System**: this Smart-Office System allows CLup to remotely control the sliding doors of the store and communicate with the QR code readers.
* **Ticket Printing System**: this system is responsible for printing paper tickets. It’s directly connected to the device running the *PTD Desktop Application.*

## Component View

In this section, every high-level component is analysed in terms of its subcomponents. External Systems, such as *Google* *Maps System, Thing-it System* and *Ticket Printing System*, are presented as black-boxes that expose only the interfaces used by CLup.

Further details about the component interfaces can be found in section 2.5.



Component Diagram 1

### 2.2.1 Application Component

The *Application Component* is the front-end of the System. Its purpose is to render the graphical interfaces (detailed in section 2.3) that allow the end-user to take advantage of the CLup Services.

The only logic incorporated in the *Application Component* is the ability to perform basic checks, such as the detection of incomplete forms. Indeed, all the computational tasks are carried out by the *Application Server* and, for this reason, the *Application Component* must be able to continuously communicate with it.

As shown in the figure above, the *Application Component* consists of the following sub-elements, already presented in section 2.1:

* **Customer Mobile Application**

It is the application dedicated to Customers that allows them to line up directly from their smartphone, book visits, access their QR Codes and receive periodic notifications.

* **Next-Up Desktop Application**

It is the application that runs on a device whose screen is visible from outside each store. It shows which Reservation IDs are allowed to access that store at any given time. Both Guests and Customers can verify whether it is their turn to enter the store via this application.

* **PTD Desktop Application**

It is the application dedicated to Guests that allows them to line up directly from the store and receive a paper ticket.

* **Web Application**

It is the application dedicated to Store Managers that allows them to keep access data under control and possibly modify the store information.

### 2.2.2 Web Server Component

A *Web Server* is required in order to provide the *Web Application* for the Store Managers. This component receives HTTPS requests from the Store Manager’s browser and forwards them to the *Application Server* to collect all the data required to generate the dynamic web pages.

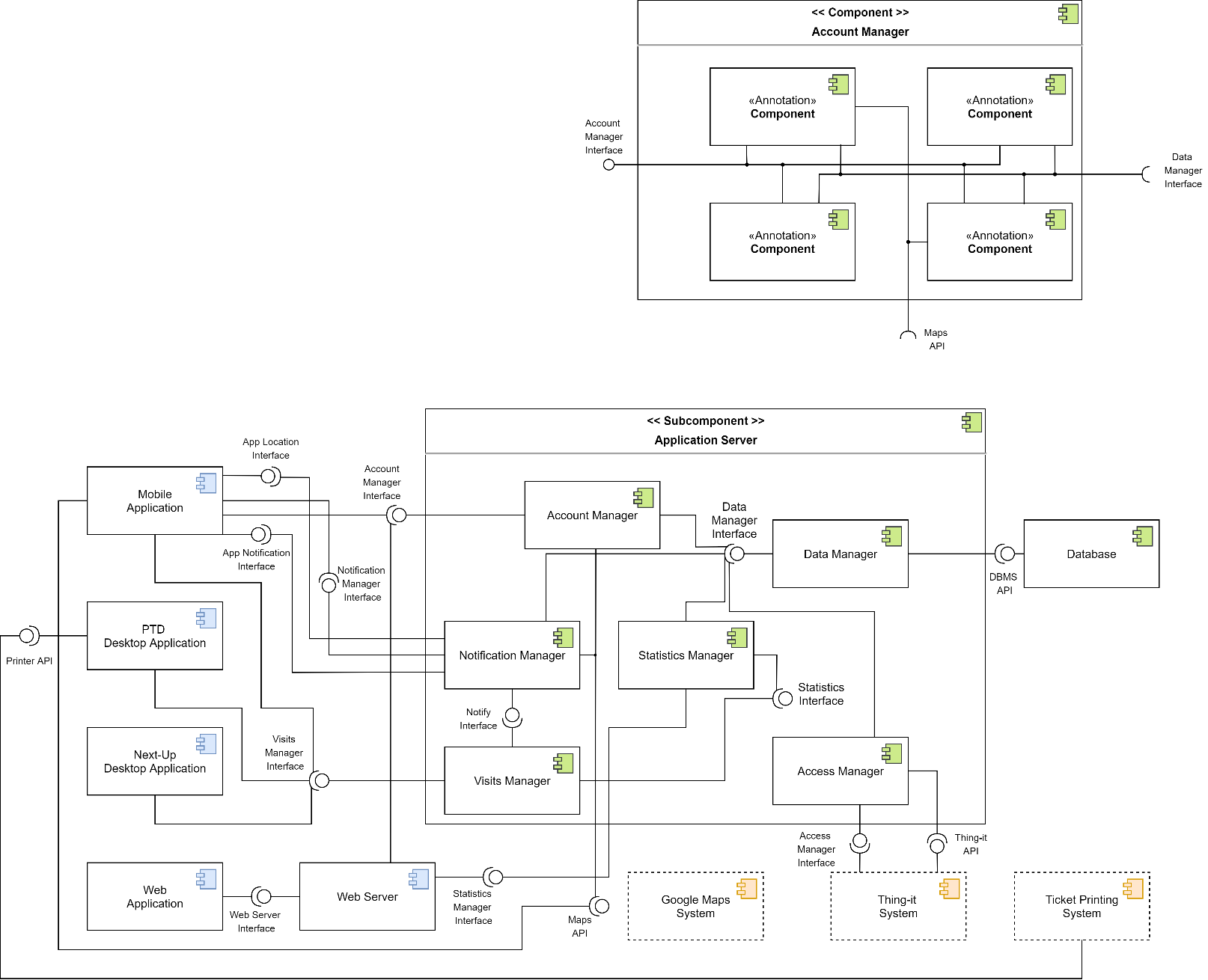
### 2.2.3 Application Server Component

The *Application Server* realizes the business logic, its role is to compute all the needed data and coordinate the flow of information between the application layer and the data layer.

As shown in the component view, the *Application Server Component* consists of the following sub-elements:

* **Account Manager**

It handles all the operations related to accounts of Store Managers and Customers. It communicates with the *Data Manager* in order to verify, access, store and delete account information. Indeed, this component is responsible for account creation and for the authentication process.

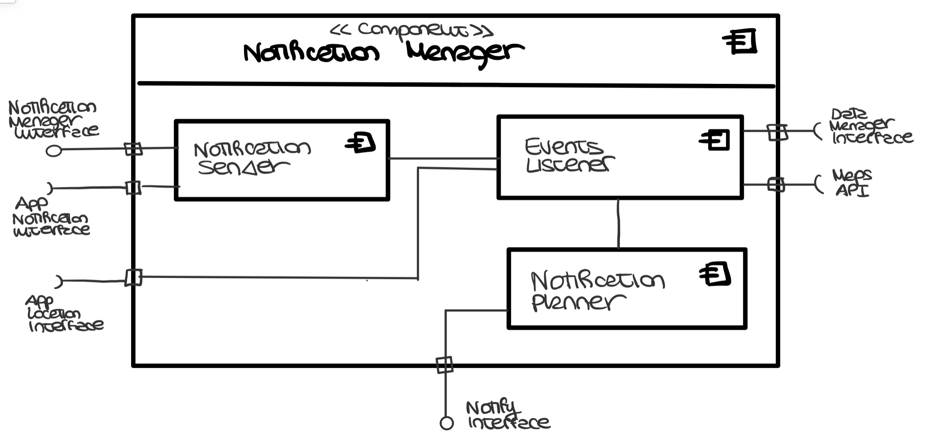


Sub-Component Diagram 1: Account Manager

It consists of the following subcomponents:

* **Store Registration Manager***:* it manages the registration of new stores and, in particular, it checks that the indicated Store ID is not already taken. It also interacts with the *Google Maps System* to obtain the coordinates of a store from its address.
* **Customer Registration Manager***:* it manages the registration of new Customers.
* **Authenticator**: this subcomponent handles the authentication by checking the user’s credentials.
* **Updater:** it is responsible for account data updates and account deletion.
* **Notification Manager**

It handles notifications to Customers. Specifically, it exposes an interface that allows changing notification preferences. A further interface is made available to the *Visit Manager* to allow the registration of new events that will trigger the sending of notifications. To manage these events, the component needs to interface with the *Data Manager* and the *Google Maps System.*

****

Sub-Component Diagram 2: Notification Manager

It consists of the following subcomponents:

* + **Notification Planner***:* it manages the registration of the events that will trigger the sending of notifications.
  + **Notification Sender:** it takes care of actually sending notifications to the *Mobile Application.*
  + **Events Listener:** it takes care of detecting the occurrence of the events that trigger the sending. It also manages the functionality of periodic notifications.
* **Visits Manager**

It manages all the operations related to visit-scheduling. It communicates with the *Data Manager* in order to access, store and delete visit information. This component also interacts with the *Notification Manager* to notify the Customer when it is time to reach the store.



Sub-Component Diagram 3: Visits Manager

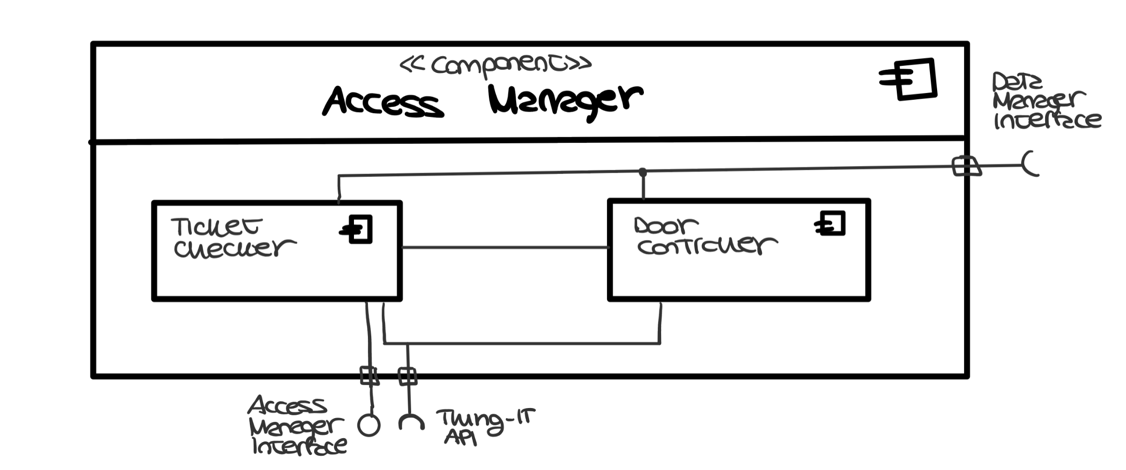
It consists of the following subcomponents:

* **Visit Scheduler:** it plans all the visits possibly assigning them an expected duration calculated by the *Statistics Manager*.
* **Waiting Time Estimator:** it computes the estimated waiting time based on the scheduled visits stored in the database
* **Statistics Manager**

It is responsible for computing the access statistics based on the data stored in the database, which is accessed through the *Data Manager.*

* **Access Manager**

It manages all the access requests to the stores. In particular, it receives the data coming from the QR Code readers and verifies whether the ticket is allowed to access or not. Depending on this, it interacts again with the *Thing-it System* to control the status of the sliding doors.

****

Sub-Component Diagram 4: Access Manager

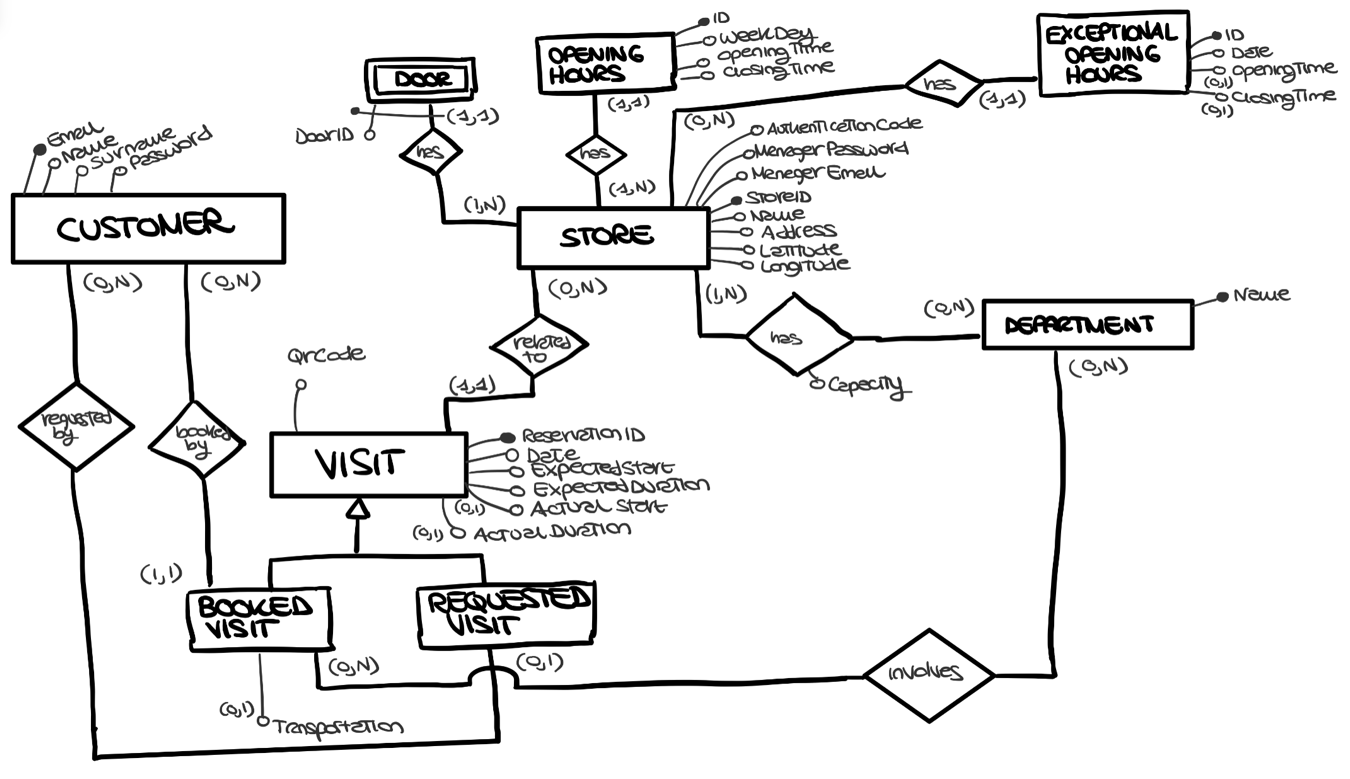
It consists of the following subcomponents:

* **Ticket Checker**: it checks whether the read QR Code is associated with a visit scheduled for the current date and time.
* **Door Controller**: it manages the requests to the Smart-Office System for opening and closing the doors
* **Data Manager**

It is the only component that interacts with the data layer. Indeed, it receives requests coming from the other components for reading, storing and deleting data from the database.

### 2.2.4 Data Components

The data layer is composed of a single relational database, which is managed by a DBMS that optimizes and performs the queries. The image below represents the Entity-Relationship (ER) diagram of the database.



CLup Entity-Relationship Diagram

Relevant aspects of the ER schema are the following:

* The *Visit* entity refers to all the scheduled visits, both those booked in advance and those requested through the “Line Up” Service. Both are assigned a Reservation ID, a QR Code and a time slot, possibly estimated by the System. The actual entry and exit time data are also stored to allow the computation of statistics. Finally, any information relating to the product categories, in which the Customer is interested, is also stored. If this information has not been specified, the System will assume during the computation that all departments are visited by that Store Client. Note that the visits requested by the Guests will not be connected to any Customer.
* The *Store* entity refers to the stores registered to CLup. For each of them, the data relating to departments and opening hours (regular and exceptional) are saved. Other stored information are the coordinates, to allow CLup to retrieve the stores close to the Customer, and an authentication code provided by the Store Manager. The latter allows CLup to take control of doors and code readers remotely via the *Thing-it System* (further details will be provided in subsection 2.2.5). The data relating to these devices are also stored in the CLup database.

From the previous ER schema, it is possible to derive the following logical model:

***Customers*** **(** Email, Name, Surname, Password **)**

***Store*** **(** StoreID, Name, Address, Latitude, Longitude, ManagerEmail,

ManagerPassword,AuthenticationCode **)**

***Doors*** **(** StoreID, DoorID **)**

***OpeningHours* (** ID, WeekDay, OpeningTime, ClosingTime, StoreID **)**

***ExceptionalOpeningHours* (** ID, Date, OpeningTime\*, ClosingTime\*, StoreID **)**

***Departments* (** Name, Keywords **)**

***StoreDepartments*** **(**StoreID, DepartmentName, Capacity**)**

***Visits*** **(** ReservationID, QrCode, Date, ExpectedStart, ExpectedDuration,

ActualStart\*, ActualDuration\*, Transportation\*, StoreID, CustomerEmail\***)**

***BookedDepartments*** **(** DepartmentName, ReservationID **)**

### 2.2.5 External Systems

* **Google Maps** **System**

This external system communicates with the *Application Server* and the *Mobile* *Application* via APIs. In particular, it provides an interface to insert a map view, to translate an address into geographical coordinates and to calculate the ETA to reach the store.

* **Thing-it System**

This external system communicates with the *Application Server* via APIs. In order to use these APIs and thus being able to remotely control doors and QR Code readers of a store, an authentication code is required. This code, provided by the Store Manager during the registration, allows CLup to take control of the devices.

* **Ticket Printing System**

This external system exposes APIs that are used directly by the *PTD Desktop Application* for printing paper tickets.

## Deployment View

This section describes the configuration of the processing nodes and the components that live on them. In the figure below there is a representation of the Deployment diagram for the entire System.

## Runtime View

In this section the dynamic behaviour of the System is described by showing the major interaction processes between the various components. Further details about the component interfaces can be found in section 2.5.

* **Customer Registration**

In the diagram below is presented the workflow that is executed during the registration of a Customer through the *Mobile Application*. Once the form has been submitted, the application performs a basic check to verify that the required fields have been filled in. If so, it proceeds by forwarding the account creation request to the *Application Server*. Here the *Account Manager* checks the data entered by the Customer and in particular verifies if the email is not already in use. If the checks pass, it proceeds by storing this data.

Diagram

Description automatically generated

Runtime View 1: Customer Registration

* **Store Manager Registration**

The following diagram represents the work stream that occurs during the registration of a Store Manager through the *Web Application*. Once the form has been submitted, the *Browser* proceeds by forwarding the registration request to the *Web Server*. Here the call to the *Account Manager* takes place. The latter verifies the data entered and in particular checks if the chosen StoreID is not already in use. If the checks pass, it proceeds by connecting to the *Thing-It System* using the authentication code. If successful, the *Account Manager* proceeds by subscribing to events relating to the QR code readers. From now on, when a QR code is read by the scanner, CLup will be notified of it. Finally, all the needed data is stored in the database.



Runtime View 2: Store Manager Registration

* **Store Manager Login**

In the sequence diagram below is detailed the workflow that occurs during the login of the Store Manager. Once the form has been submitted, the *Browser* sends the request to the *Web Server* which forwards it to the *Account Manager*. This component verifies the credentials by interacting with the *Data manager*. If correct, the *Web Server* redirects the Store Manager to the homepage otherwise it shows an error page.

Diagram, timeline

Description automatically generated

Runtime View 3: Store Manager Login

* **Store Manager Login**

In the image below is presented the workflow that is executed during the login of a Customer through the *Mobile Application*. Once the login form has been submitted, the application performs a basic check to verify that the fields have been filled in. If so, it proceeds by forwarding the request to the *Application Server*. Here the *Account Manager* checks the credentials and returns the result.

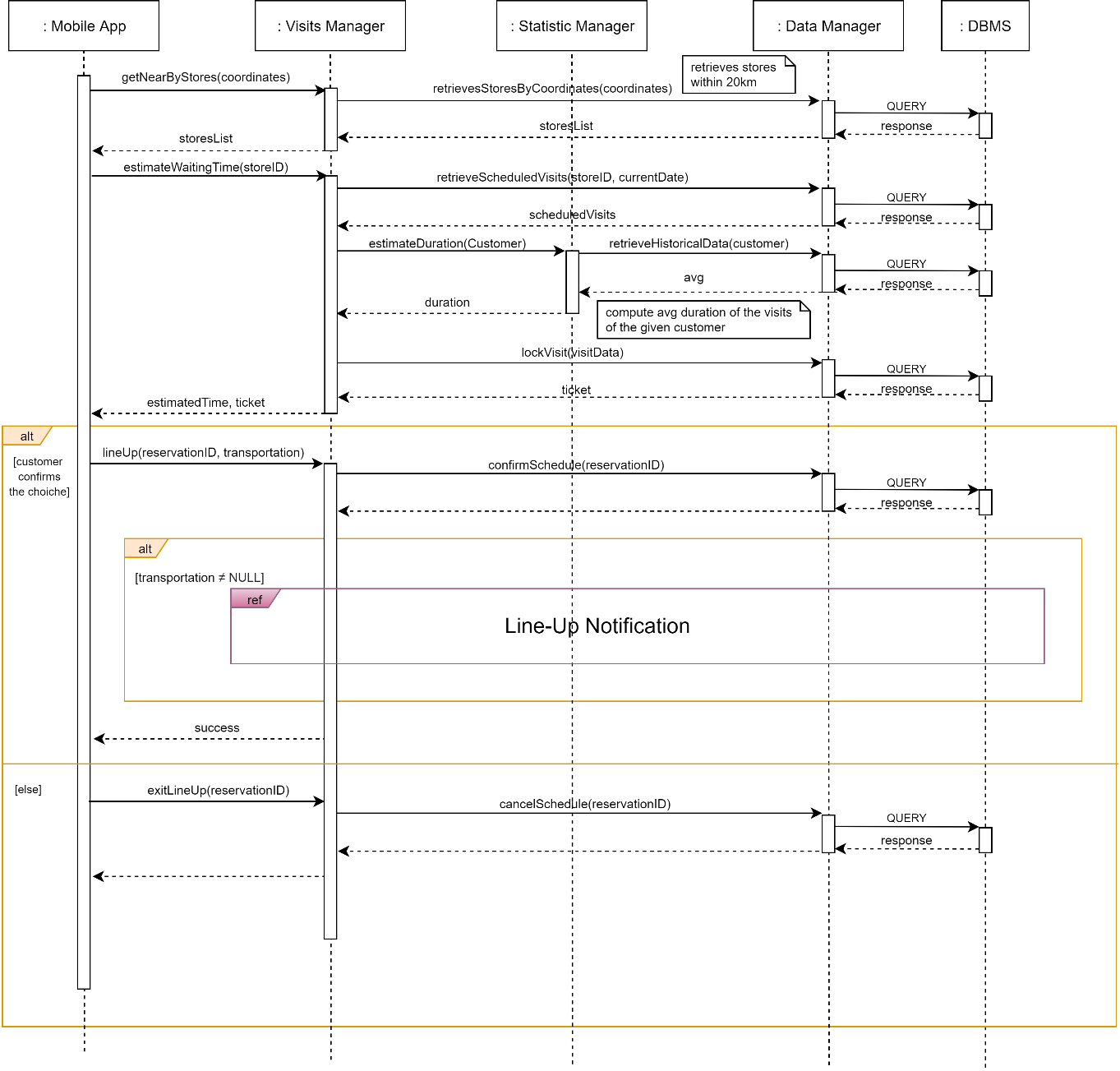
Diagram

Description automatically generated

Runtime View 4: Customer Login

* **Customer Line-Up**

The following diagram represents the work stream that occurs when a Customer tries to line up at store from the *Mobile Application*. When the Customer opens the line-up interface, the application retrieves the nearby stores by forwarding a request to the *Visits Manager*. Once the store has been selected, the application asks the *Application Server* to estimate the waiting time. The *Visit Manager* then makes the estimation. First it retrieves the data of the already scheduled visits, then estimates the average duration of the Customer’s visit through the *Statistic Manager*. This component analyses the historical data relating to the Customer. Once the estimation has been computed, a visit to the shop is provisionally reserved for the Customer. If the Customer decides to proceed, the visit is effectively reserved otherwise the reservation is cancelled. If the Customer has also indicated the means of transport he will use to reach the store, the process specified in the Runtime View 10 starts.



Runtime View 5: Customer Line-Up

* **Customer Books a Visit**

In the diagram below is presented the workflow that is executed during the booking of a visit through the *Mobile Application*. When the Customer opens the booking interface, the application retrieves the nearby stores by forwarding a request to the *Visits Manager*. Once the store and the slot have been selected, the application asks the *Application Server* to book the slot. The *Visit Manager* then checks the availability of the slot. First it retrieves the data of the visits already scheduled and the opening hours of the store, then estimates the average duration of the Customer’s visit through the *Statistic Manager*. This component analyses the historical data relating to the Customer. Once the estimation has been computed, a visit to the shop is booked for the Customer. If the slot is not available, the Visit Manager compute the available alternatives.

Diagram

Description automatically generated

Runtime View 6: Customer Books a Visit

* **Guest Line-Up**

In the following image is presented the workflow that occurs during the line-up of a Guest through the *PTD Desktop Application*. When a Guest reaches the touchscreen monitor, the application contacts the *Application Server* to compute the estimated waiting time. This is done by the *Visit Manager*. First, it retrieves the visits already scheduled, then estimates the average duration of the visits for that shop. When the Guest decides to queue, another request is sent to the *Visit Manager* that, after calculating the slot to be assigned, stores the assignment through the *Data Manager*. At this point the *PTD Desktop Application* orders the *Ticket Printing System* to print the paper ticket.

Diagram

Description automatically generated

Runtime View 7: Guest Line-Up

* **Store Manager Analyse Access Data**

In the diagram below is presented the workflow that is executed when the Store Manager wants to analyse the access data through the *Web Application*. The *Browser* sends a request to the *Web Server* to visualize the statistics report that is generated by the *Statistics Manager*. Finally, the report is included in the web page that is returned to the *Browser.*

Graphical user interface, text, application

Description automatically generated

Runtime View 8: Store Manager Analyse Access Data

* **Periodic Notification**

The sequence diagram below explains how the System works to periodically notify the Customer and provide him useful suggestions. Periodically, the *Notification Manager* is activated to check for the presence of available slots that could be of interest to a Customer. In order to do this, it interacts with the *Visits Manager* that, after computing the slots of interest to the Customer, checks their availability. Once the computation is complete, a notification is sent to the *Mobile Application*.

A picture containing graphical user interface

Description automatically generated

Runtime View 9: Periodic Notification

* **Line-Up Notification**

The following diagram explains how the System works to notify the Customer when it is time to reach the store. The *Visits Manager* delegates the notification management to the *Notification Manager*. This component, once retrieved the position of the Customer and that of the store, contacts the *Google Maps System* to estimate the departure time. If the application detects that the Customer has moved, it communicates the new position to the server and the calculation is repeated. When the *Notification Manager* detects that it is time to notify the Customer, it sends a notification to the *Mobile Application*.

Diagram

Description automatically generated

Runtime View 10: Line-Up Notification

* **Update of “Next-Up” Interface**

The following diagram explains how the System works to periodically update the Next-Up Interface that shows the Reservation IDs that are allowed to enter a given store. When the application is started, it contacts the *Application Server* to subscribe to updates related to visits for that store. In this way, once the Notification Manager has detected that other Customers can access the store, a notification is sent to the application and the new Reservations IDs appear on the interface.

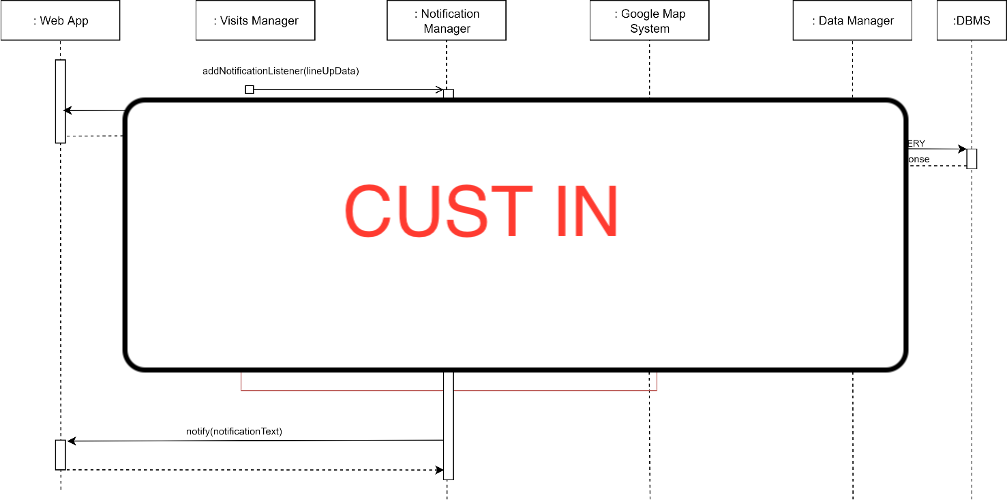
Immagine che contiene testo

Descrizione generata automaticamente

Runtime View 11: Update of "Next-Up" Interface

* **Customer/Guest Enters a Store**

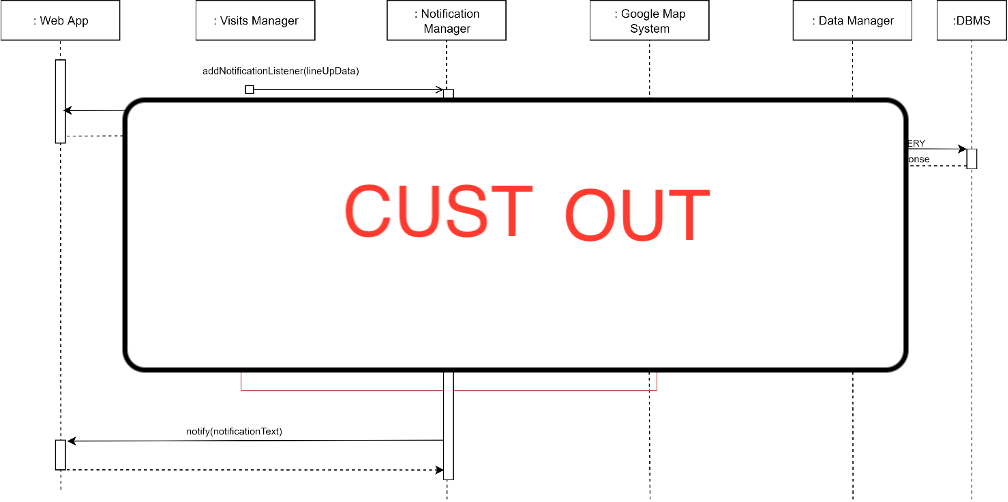
The runtime view below explains the workflow that occurs when a Customer or a Guest tries to enter a store. When a code is scanned by the QR reader, the *Thing-It System* notifies the *Application Server* by forwarding the read data. Here, the Access Manager checks if the code is assigned to a ticket of a scheduled visit. If so, it allows access and updates the queue data, otherwise it denies access.



Runtime View 12: Customer Enters a Store

* **Customer Leaves a Store**

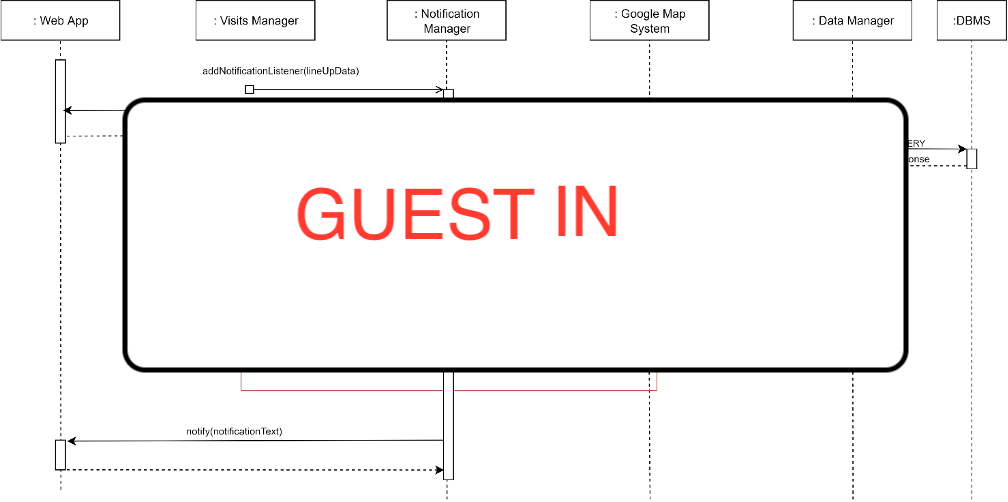
The following diagram details the workflow executed when a Customer leaves a store.



Runtime View 13: Customer Leaves a Store

* **Guest Enters a Store**

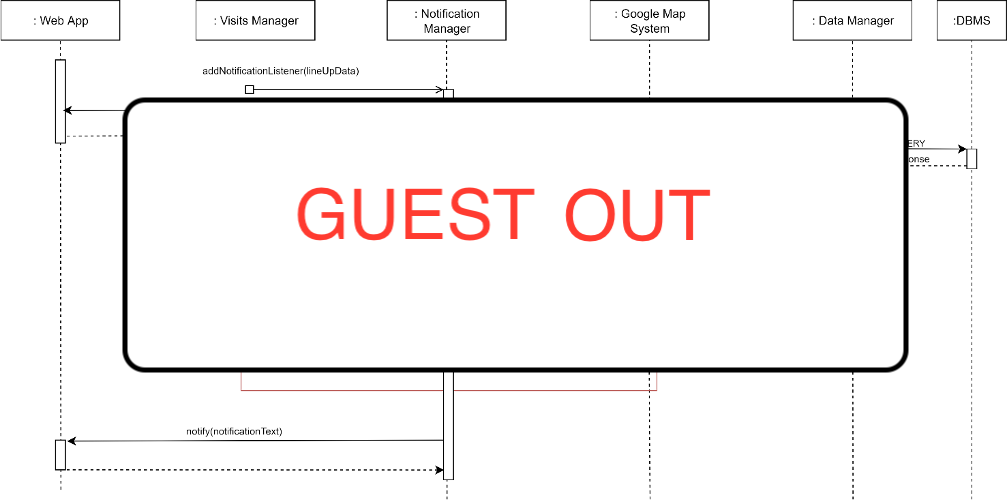
The following runtime view explains the workflow that occurs when a Guest tries to enter a store.



Runtime View 14: Guest Enters a Store

* **Guest Leaves a Store**

The following diagram details the workflow executed when a Guest leaves a store.



Runtime View 15: Guest Leaves a Store

## Component interfaces

## 2.6 Selected Architectural Styles and Patterns

### 2.6.1 4-Tier Architecture

### 2.6.2 Thin Client

### 2.6.3 Model-View-Controller

### 2.6.4 Data Access Object

### 2.6.5 Application Logic

## 2.7 Other design decisions

### 2.7.1 User Authentication and Password Storing

### 2.7.2 Chain of Custody

### 2.7.3 Relational Database

**Chapter 3**

# User Interface Design

## 3.1 UX Diagrams

## 3.2 User Interface Mockups

**Chapter 4**

# Requirements Traceability

**Chapter 5**

# Implementation, Integration and Test Plan

## 5.1 Development Process

## 5.2 Implementation

### 5.2.1 Implementation Plan

### 5.2.2 Implementation Choices

## 5.3 Integration

### 5.3.1 Entry Criteria

### 5.3.2 Elements to be integrated

### 5.3.3 Integration Sequence

## 5.4 Test Plan

**Chapter 6**

# Effort Spent

The following tables summarize the effort spent by each member of the team to create the DD document.

## 6.1 Leoni Luca

|  |  |
| --- | --- |
| **Description of the task** | **Hours** |
| Document Introduction: Purpose and Scope Summary | 3 |
| Identification of External Services, Patterns and Architectural styles | 3 |
| Definition of the General Architecture and Components Identification | 4 |
| General definition of Runtime Views: identification and description | 9 |

## 6.2 Locarno Silvia

|  |  |
| --- | --- |
| **Description of the task** | **Hours** |
| Document Introduction: Purpose and Scope Summary | 3 |
| Identification of External Services, Patterns and Architectural styles | 3 |
| Definition of the General Architecture and Components Identification | 4 |
| Architecture Overview and Component View Diagrams | 1.5 |
| General definition of Runtime Views: identification and description | 9 |
| Actual creation of Runtime View Diagrams | 5 |

## 6.3 Minotti Luca

|  |  |
| --- | --- |
| **Description of the task** | **Hours** |
| Document Introduction: Purpose and Scope Summary | 3 |
| Identification of External Services, Patterns and Architectural styles | 3 |
| Definition of the General Architecture and Components Identification | 4 |
| Detailed Description of the Components and their Composition | 5 |
| General definition of Runtime Views: identification and description | 9 |
| ER schema definition | 1 |

**Chapter 7**

# 7 References

* E. Di Nitto. Lecture Slides. Politecnico di Milano.
* E. Di Nitto. Project Assignment AY 2019-2020. Politecnico di Milano.
* ISO/IEC/IEEE 29148:2011. Standard on requirement engineering. https://standards.ieee.org/standard/29148-2011.html.