1. INTRODUCTION

1.1. Purpose

CLup aims to provide chains of stores with a reliable solution to the problem of people gathering inside and outside the shops.

To face the problem, the application focuses on its principal causes, which are the management of people inside the store, that often leads to overcrowding, the effectiveness of standard queuing systems and the way people are allowed to visit the stores. Moreover, the system aims to provide a useful tool for store managers in order to help them in administering stores and monitoring their status.

In particular, the main goals that CLup aims to achieve, summarized in the table below, are the following:

- Prevent the store from being overcrowded, in order to avoid indoor gatherings while maximizing its occupancy, by means of an access management system
- Reduce gatherings of people waiting to enter outside the store, providing a way to virtualize queues
- Provide a more efficient way to access stores, reducing the time customers waste while waiting to do it
- Help store managers in monitoring the status of the store and regulating the influx of people

More details on how CLup is supposed to fulfill these goals are in the Requirement Document.

1.2. Scope

During the current situation of emergency, it is fundamental to prevent contacts among people. For this reason, governments impose strict rules concerning social distancing, both for indoor and outdoor contexts.

However, crowding management inside stores like supermarkets and grocery shops could be challenging. Currently, stores limit the maximum number of people allowed, and therefore long queues arise: entering a store for a few minutes might even require hours. Moreover, customers who see a crowded store might avoid lining up to save time and prevent contact with others.

CLup fits into this context allowing customers to remotely line up in a queue of a given store and to be notified when they should head toward it. Furthermore, it allows the customer to book a visit for a store on a specific day and time, which grants him priority over the queued customers.

Users can interact with CLup thanks to two distinct interfaces: one is an easy-to-use application designed for the customers, while the other one is an administrative tool that allows store managers to monitor their stores and modify their parameters.

Moreover, CLup also provides physical proxies outside the stores as a fallback option for users who want to line up but do not have access to the application.

1.3. Definitions, Acronyms, Abbreviations

AMS	Access Management System
TAS	Turn Announcement System
CLup	Also known as the system. It is the software to be developed. From a design-oriented point-of-view, the term is also used to refer to the mobile application, the administrative tool and the server all together
Customer application	Also known as application. It is used to access the functions provided by CLup
Administrative tool	The tool provided to store managers in order to administer stores
Proxy	The physical fallback option for customers that want to use CLup but cannot use the application. It is placed outside the store it belongs to
Turn Announcement System	An external system which informs customers about who has been allowed by CLup to enter the store it belongs to
Access Management System	An external system which regulates physical entrances and exits to the store it belongs to by interacting with CLup
App-customer	A customer who uses CLup functions through the application
Proxy-customer	A customer who uses CLup functions through the proxy
User	Either a customer or a store manager
Long-term customer	With respect to a certain store, a customer who already used CLup to visit it
Current occupancy	Also known as occupancy. It can be referred to the store or one of its sections. It is the number of people inside it
Maximum occupancy	Refers to the store or one of its sections. It is the maximum number of people allowed to be in that area

Virtual queue	Also known as access queue or simply queue. It represents the set of customers who lined up through the app or the proxy
Line up	With respect to a customer and a store, it is the event of joining the queue
Visit request	A customer's request to visit a store. It can be either a line-up request or a booking request
Line-up request	A request made by the customer to line up for a store
Booking request	A request made by the customer to book a visit to a store
Visit	The realization of a visit request which takes place when a customer enters the store. After the customer exits the store, we talk about *completed visit*, otherwise it is a *visit in progress*.
Visit token	A unique token bound to a visit request. It allows the Customer to enter and exit the store
Pending request	A customer's visit request that does not have a completed visit associated with and is not allowed to enter the store it is associated with
Ready request	A customer's visit request that does not have a completed visit associated with and is allowed to enter the store it is associated with
Fulfilled request	A customer's visit request that has an associated *visit in progress*
Completed request	A customer's visit request that has an associated *completed visit*
Active request	A customer's visit request that is not a completed request (thus it is either a pending, a ready or a fulfilled request)

1.4. Revision history

1.0 - First version of the document (DATA FINALE)

1.5. Reference Documents

IEEE standard for Software Design Descriptions, IEEE 1016-2009

R&DD Assignment AY 2020-2021

CLup Requirements Analysis and Specification Document (RASD)

Teaching material provided by professors Matteo Rossi and Elisabetta di Nitto

1.6. Document Structure

The reference structure used for the document is an adapted version of the one suggested by professor Matteo Rossi of Politecnico of Milan. It is derived from the IEEE standard, which is used as a reference document (IEEE standard for Software Design Descriptions, IEEE 1016-2009).

Chapter 1 is an introduction to the software to be designed and developed and to the problem that it addresses. It presents the goals that should be achieved and an analysis of the context in which the system will be placed.

Chapter 2 defines the system architecture. It includes a view of the system components and of their interfaces, a view about the deployment choices and some views about the runtime behaviour of the system. It also explains all the other design decisions.

Chapter 3 focuses on the design of the user interfaces. It also illustrates the users interactions with the system through mockups.

Chapter 4 better details the connections between goals and requirements already mapped in the RASD, taking into account the system components identified in Chapter 2

Chapter 5 focuses on future plans about the implementation, the integration and the testing of the system components.

Chapter 6 contains a report on the effort spent by all the members of the group while writing the current document.

2. ARCHITECTURAL DESIGN

2.1. Overview

The following sections are about the architecture of CLup. In order to better understand the whole document and to make it more self-contained, some recalls from the CLup Requirements Analysis and Specification Document follow, including an updated class diagram that considers a more design-oriented point-of-view.

Table of functional requirements

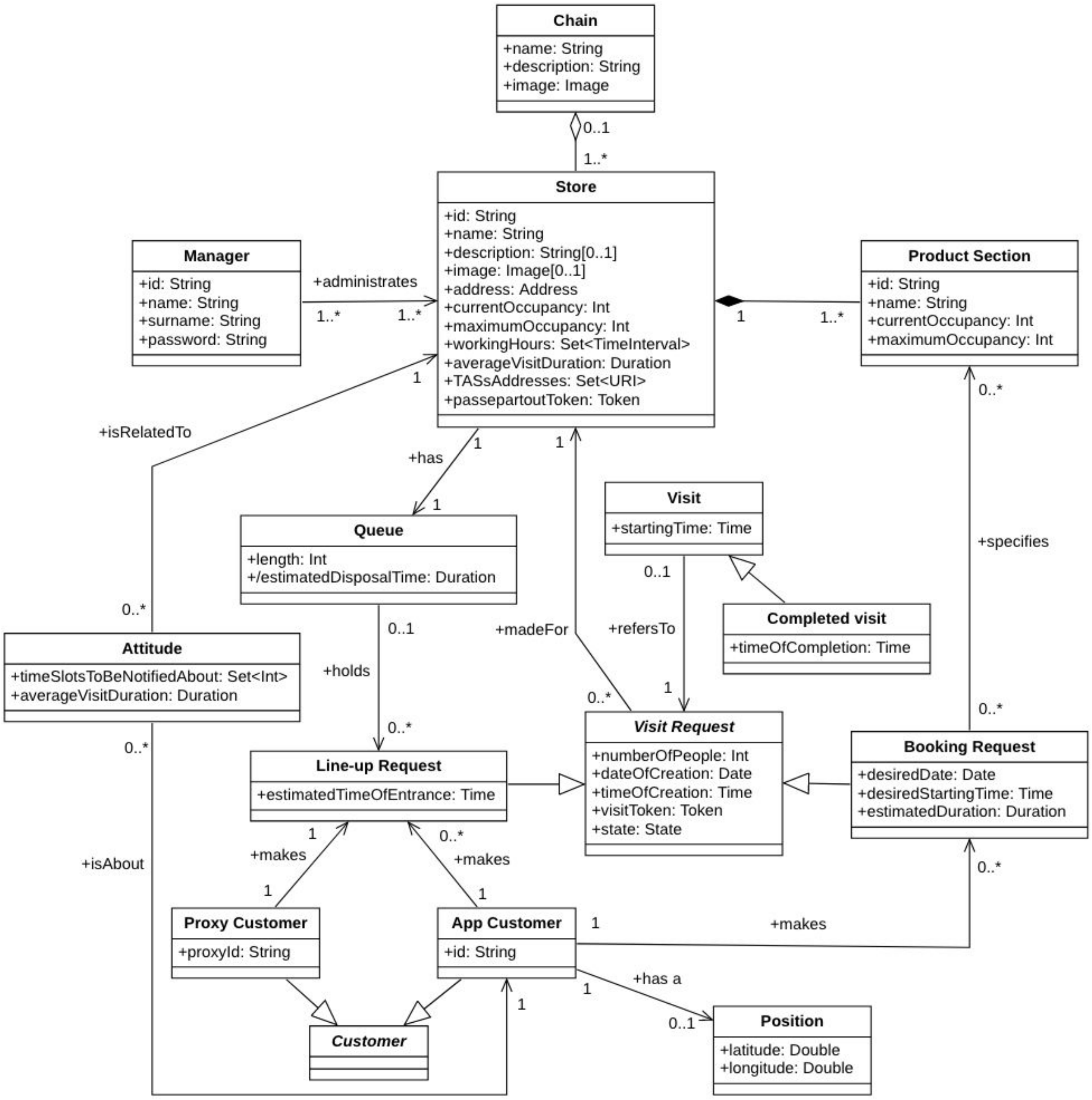
R1	The system shall allow managers to specify the store parameters
R2	The system shall allow managers to monitor entrances
R3	The system shall allow managers to monitor exits
R4	The system shall authorize accesses to the store
R4.1	The system shall authorize customers to enter if and only if the store would not exceed the maximum number of people allowed inside it
R5	The system shall provide a way to line up in the virtual queue of the store
R6	The system shall provide a way to exit the queue before entering the store
R7	The system shall alert the app-customer when it is time to reach the store
R8	The system shall provide the possibility to book a time interval for visiting the store
R9	The system must not allow customers to book a visit in a time interval if, over its duration, bookings by other users already maximize store occupancy
R9.1	The system must not allow customers to book a visit in a time interval if, over its duration, bookings by other users already maximize at least one of the product sections specified in the booking request
R10	When booking a visit, the system shall allow customers to specify what kind of products they intend to buy
R11	The system shall provide the possibility to cancel a booked visit before entering the store
R12	While making a booking request, the system shall suggest alternative time intervals if the demand of the chosen one is too high
R13	While making a booking request, the system shall suggest alternative stores of the same chain if the demand for the chosen time interval in the selected store is too high
R14	The system shall allow managers to regulate entrances
R15	The system shall allow managers to regulate exits

R16	The system shall notify a customer when, during a specific time interval, a specified store is reaching its maximum occupancy
R17	The system shall keep track of the average duration of a generic visit to the store
R18	The system shall manage the case in which customers do not show up when it is their turn to enter the store
R19	The system shall inform customers when they are allowed to enter the store

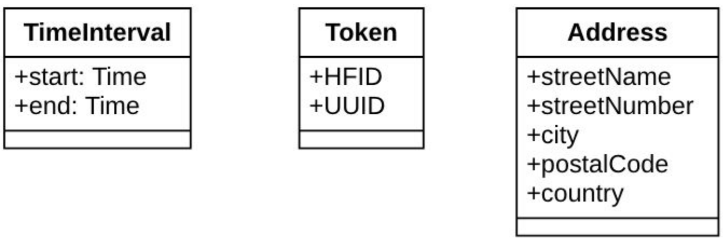
Table of nonfunctional requirements

NF1	The system, in normal operating conditions, should be able to handle all line-up and booking requests of each customer.
NF2	The system responses and notifications must be sent within 3 seconds from the triggering event.
NF3	The software should be GDPR compliant
NF4	CLup should provide a high degree of reliability
NF5	CLup should be available the 99% of the time
NF6	The system should be protected against malicious attacks
NF7	The system must be well-documented and adaptable to changes
NF8	The CLup customer application should be compatible with most of the smartphones currently on the market
NF9	CLup should be platform independent
NF10	The application should be easy to use
NF11	CLup shall give priority to booking requests over line-up requests
NF12	Customers can remotely line up in a store's queue only if they are not in the queue of any store at that moment
NF13	Customers can book a visit to a store for a specific time interval only if they have not booked any other visit which overlaps with that time interval
NF14	Customers can book a visit to a store for a specific time interval only if it starts after the current queue disposal time of that store.

Class Diagram



For the sake of simplicity and readability of the diagram, complex types of attributes are detailed below and not included in the diagram itself.



Furthermore, *Image* and *Duration* are abstract types representing respectively an image and a time duration, while *State* is an enumeration of the possible states of a request (pending, ready, fulfilled, completed).
Details on the attribute mentioned are provided through the following sections.

As evidenced by the following sections, CLup is modelled according to a multi-tier 3-layer architecture. The software components to be developed are the server, the mobile application and the administrative tool, which in the current document are mentioned all together as *CLup* or *system*.

2.1.1. Selected architectural styles and patterns

The whole system (i.e. CLup itself) is designed as a multi-tier 3-layer architecture. Thus, it allows the decoupling of presentation, logic and data layers, which are hosted on different devices.

- The **presentation layer** includes all the devices which customers and managers interact with in order to use the services offered by CLup;
- The **application** (logic) **layer** includes all the components needed to guarantee the complete implementation of CLup functions;
- The **data layer** includes the DBMS in charge of managing all the persistent data of CLup.

The application layer also includes third-part components which the system interacts with and which allow it to properly offer its services.
The only noticeable exception in the CLup 3-layer architecture is represented by the interaction between the mobile application and the NotificationService component. Indeed, the MobileApplication offers an interface, which depends on the development technology stack, that guarantees the feasibility of remote push notifications service offered by CLup. More details on this design choice are provided in Section 2.6.

The interfaces between the presentation layer and the application layer are designed according to the REST (Representational State Transfer) architectural style and are based on the HTTPS protocol. Thus, each interaction between those layers is stateless and follows a client-server approach.

Since REST applies the “separation of concerns” (SoC) paradigm, it allows an independent development and replacement of client-side logic and server-side logic as long as the interfaces are not changed.
Furthermore, this choice results in an improvement of the portability (and the freedom of implementation choices) of the user interfaces across multiple platforms, of the scalability of the system (by simplifying the server components) and of the maintainability.

Regarding the functionalities that involve the notification of app-customers, it was decided to use existing providers of remote push notifications.

No constraints concerning the provider of the push notification service (e.g. Google Firebase, Apple Push Notification service) and the one of the maps service (e.g. Google Maps, OpenStreetMap) are imposed on the developers. Thus, across the entire document, the NotificationService component and the MapsService component are considered as dependent on the chosen service provider.

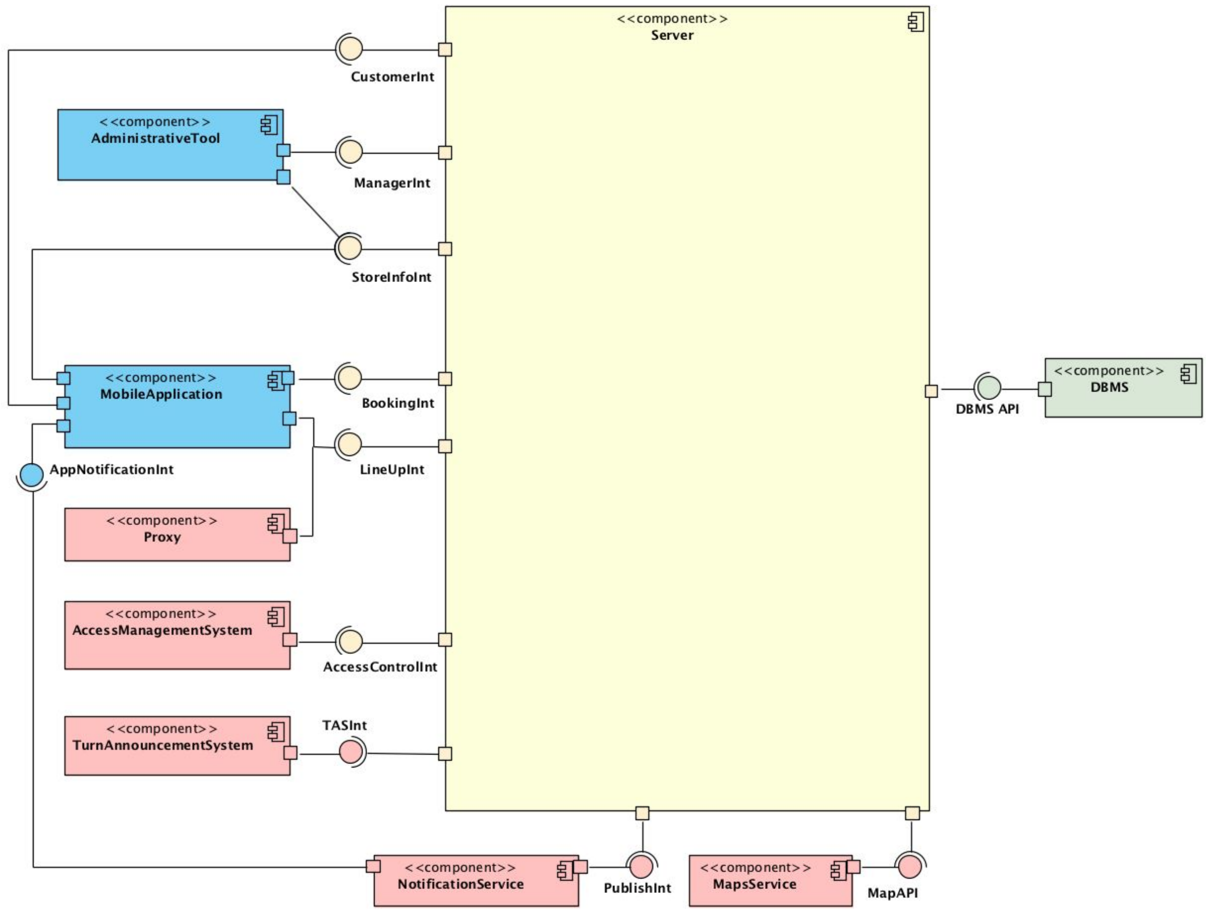
While the server, the mobile application and the administrative tool are the software components to be developed:

- the proxy and the AMS are existing components that will be configured to interact with the system through the interfaces provided by CLup. Thus, they are supposed to be able to communicate with CLup over HTTPS;
- the TAS, the NotificationService and the MapsService components are existing components which CLup interacts with through the interfaces they provide. They all offer an interface through the HTTPS protocol;
- Also the DBMS component is an existing component which offers an interface to CLup. Their communication is based on TCP/IP.

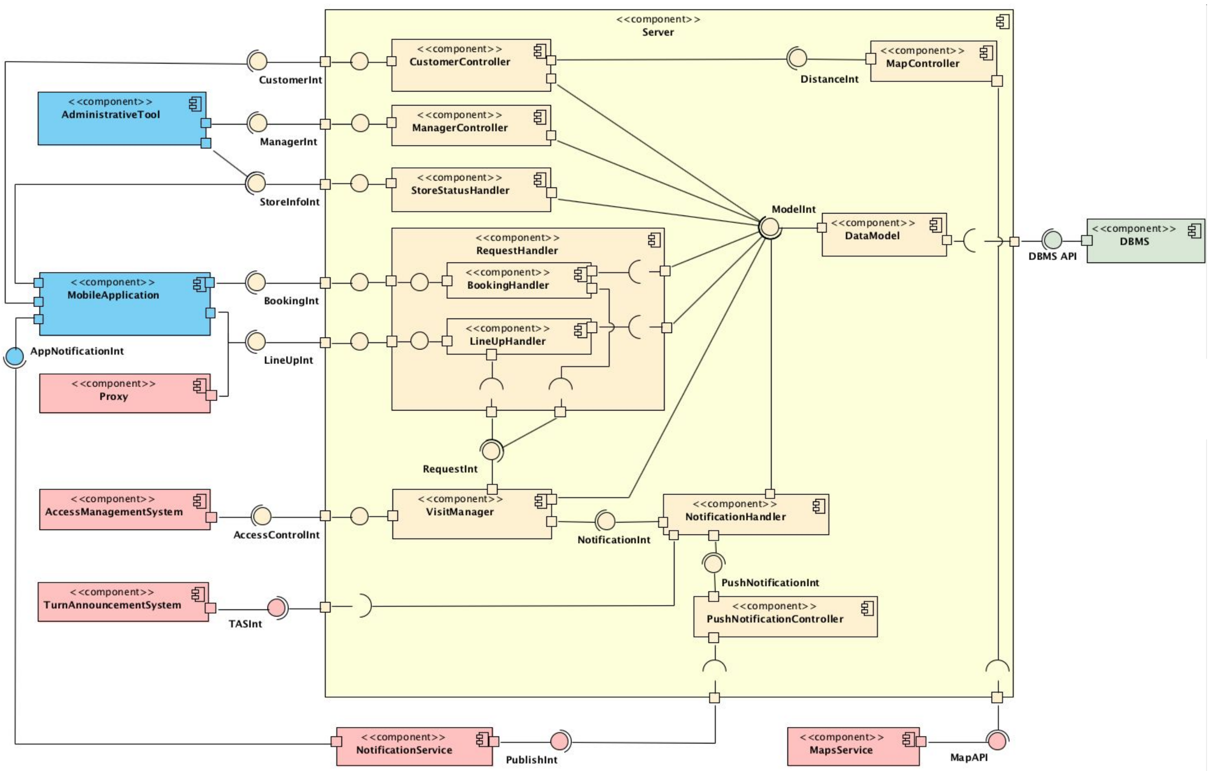
In order to reach a low level of coupling, the server internal logic has been the object of a functional decomposition into subcomponents and has been designed considering the proxy design pattern when dealing with external components. Indeed, the PushNotificationController and the MapController components act as interfaces respectively to the external NotificationService and MapsService components.
Furthermore, the DataModel component, which also acts as an interface to the data storage, represents the repository of the homonym pattern.
Further details on the components and on their interaction follow.

2.2. Component view

2.2.1. High-level component view



2.2.2. A more detailed component view



2.2.3. Components description

High-level components:

- **MobileApplication**: it is the smartphone application used by customers to interact with CLup. In order to allow customers to use all the functions provided by the system, it interacts with the server as a client via several interfaces. It also offers an interface to the NotificationService that allows CLup to send remote notifications to the customer.
- **AdministrativeTool**: it is the desktop application used by store managers in order to monitor and administer their stores.
- **Proxy**: it is the component that provides a fallback option to customers who cannot use the MobileApplication. Each instance of this component is associated with a single store managed by CLup and allows customers to line up in the store's queue.
- **AccessManagementSystem**: the component in charge of letting customers enter and exit the stores with respect to their visit token . Each instance of this component is associated with a single store managed by CLup, and contacts the VisitManager via the AccessControlInterface to determine whether a customer is allowed to enter the store. It also informs CLup when the customer actually enters and exits the store, specifying how many people actually enter and exit it.
- **TurnAnnouncementSystem**: the component in charge of informing customers outside the stores when they are allowed to enter the store. Each instance of this component is associated with a single store managed by CLup and provides an interface to be notified whenever a new customer is allowed to enter the store. It can also provide

information about the store's queue disposal time, its length, or the average duration of a visit in its store.

- **NotificationService**: it is the external component which manages the dispatching of push notifications to the MobileApplication. It offers an interface to the CLup in order to collect notifications, and sends them to the MobileApplication of the receiver.
- **MapsService**: it is the external component used to perform the geospatial requests made by NotificationHandler component. It provides an interface contacted by the MapController, which mediates between the two components.
- **Server**: it is the component implementing the system logic.

Internal server components:

- **RequestHandler**: the component is the one in charge of collecting customers' visit requests, accepting or rejecting them, and of cancelling the same requests whenever the customer who placed them asks for it. The request handler is composed of two sub components.
  - **LineUpHandler**: the component in charge of collecting, accepting and rejecting LineUp requests made by all customers. When a request is accepted, the LineUpHandler informs the DataModel and notifies the VisitManager via its dedicated interface. The rejection is performed taking into account the status of the store they refer to, retrieved from the DataModel. This component is also the one in charge of dealing with customers' requests to cancel their previously made line-up requests, informing the DataModel and notifying the VisitManager.
  - **BookingHandler**: the component in charge of collecting, accepting and rejecting booking requests made by app-customers. It also balances out the affluence to the desired store in each time interval. To do so, when a request is placed, if the store is likely to reach its maximum occupancy in the specified time interval, before accepting it the component suggests alternative time intervals to the customer. When a request is accepted, the component informs the DataModel and notifies the VisitManager via its dedicated interface. A request is rejected taking into account the status of the store it refers to in the specified time interval, retrieved from the DataModel. When this happens, the BookingHandler is also in charge of suggesting the customer alternative time intervals or stores of the same chain in which to perform the booking. This component is also the one which deals with customers' requests to cancel their previously made booking requests, informing the DataModel and notifying the VisitManager.
    Moreover, the BookingHandler is in charge of providing customers with information about the estimated time of entrance associated with their booking request if it is the booking desired time but the customer is not allowed yet to enter the store.
- **CustomerController**: the component in charge of retrieving and providing app-customers with their information, such as their active line-up and booking requests. It also manages customers' requests to be notified about stores becoming unavailable in some specific time intervals. These time intervals can both be specified

by the customer or automatically inferred by the component, based on the previous customers' visits to the selected store. Moreover, for each app customer, it periodically makes an estimation on the average duration of a visit in each store already visited by each app customer.
Furthermore, it informs app-customers when they should start heading to the store they lined-up for in order to arrive on time. To do so, it periodically receives the current position of a customer and delegates the MapController component to get the time he would need in order to reach the store he lined-up for by walk and by car.

- **ManagerController**: the component in charge of handling with managers authentication and providing them with information about their profile, such as the list of stores they manage. Furthermore, it is the component in charge of allowing managers to administer their stores, modifying their relative parameters. Moreover, the ManagerController component is also the one in charge of allowing store managers to add already existing managers to their stores, or even add new ones.
- **StoreStatusHandler**: this component is the one in charge of providing all the information concerning the stores managed by CLup, such as their opening time, their current occupancy, their product sections and relative occupancies, their current queue disposal time. Also, it periodically estimates and updates the average duration of a visit to each store, and provides an estimate on the most requested time intervals for each store, in which the store is most likely to reach its maximum occupancy.
- **VisitManager**: this component is the one in charge of coping with visits and visit requests to the stores managed by CLup. In fact, it regulates the order in which customers are allowed to visit the stores they made visit requests for, and also manages the situation in which customers do not show up when it is their turn. Moreover, when a visit request is in its *ready state*, it communicates this information to the NotificationHandler via its dedicated interface. The VisitManager also provides an interface to allow entrances and exits to the stores managed by CLup, by checking if given a visit token and a store the token is associated with a visit request in ready state for the selected store. In addition, this component is the one in charge of informing the NotificationHandler when a given store is likely to reach its maximum occupancy.
- **NotificationHandler**: this component is the one in charge of notifying customers when they are allowed to enter the store. When this happens, the NotificationHandler contacts the TurnAnnouncementSystem, providing it with the visit tokens of the ready requests and with other information about the store. Moreover, in case of visit requests placed by app-customers, the NotificationHandler interacts with the PushNotificationController in order to send remote push notifications to the customers' mobile application.
  The NotificationHandler is also in charge of notifying app customers whenever the stores they asked to be notified about are likely to reach their maximum occupancy in a given time interval. In fact, when a store is reaching its maximum occupancy in a certain time interval, the NotificationHandler, which has been informed by the VisitManager about that event, retrieves from the DataModel the app-customers to be notified and eventually notifies them via the PushNotificationController.
- **PushNotificationController**: this component is the one in charge of mediating between the system and the NotificationService. It receives requests from the

NotificationHandler that include the identifier of the customer to be notified and the content of the notification. It then forwards the request to the NotificationService.

- **MapController**: this component is the one in charge of mediating between the system and the MapsService. In fact, it receives geospatial requests from the CustomerController component and performs them exploiting the MapsService.
- **DataModel**: this component is the one that defines the data model of the system and that manages the actual data. It also interacts with the DBMS in order to ensure data persistence. All the other components of the system interact with it in order to read and update data.