POLITECNICO
MILANO 1863

CLup - Customer Line Up

Software Engineering 2 Project 2020/21

Authors

- Francesco Attorre - *10618456*
- Thomas Jean Bernard Bonenfant - *10597564*
- Veronica Cardigliano - *10627267*

Deliverable:	DD
Title:	Design Document
Authors:	Francesco Attorre, Thomas Jean Bernard Bonenfant, Veronica Cardigliano
Version:	1.0
Date:	10-January-2021
Download page:	https://github.com/FrancescoAttorre/softeng2-attorre-bonenfant-cardigliano
Copyright:	Copyright © 2021, Francesco Attorre, Thomas Jean Bernard Bonenfant, Veronica Cardigliano - All rights reserved

Contents

1 Introduction

A Purpose

The goal of this Design Document is to provide a general view of the architecture of the software, adding more technical details to the information provided in the RASD document. The architecture will be explained in terms of hardware and software components, the interfaces they provide and the interactions between them. The document is going to face also the integration and testing plans, and the main design patterns to be exploited. Information related too much on the implementation will not be dealt in detail since this document aims to the description of an high level architecture of the system.

B Scope

CLup is an application which aims to manage crowds in a period of global pandemic. It allows store managers to monitor the entrances in a building and to avoid the the formation of queues in front of stores. CLup will allow people to save time and be safer, managing both the accesses of people who use the application directly, and of people who go physically to the store, using store managers as intermediaries. CLup can be accessed by customers, store managers and activities which will register stores available to be visited. Customers using the application, moreover, will be notified when they're supposed to leave for the store, and, when registered, they will have the possibility to book a visit to a store, being informed on alternative time slots or similar stores when nothing is available for their choices. Booking a visit, customers can decide how much time they want spend in the building and, optionally, which specific departments they want to visit, in order to allow a greater number of people to enter when possible.

C Definitions, Acronyms, Abbreviations

C.1 Definitions

C.2 Acronyms

- RASD: Requirements Analysis and Specification Document
- GPS: Global Positioning System
- CLup: Customers Line-up
- DD: Design Document

C.3 Abbreviations

- Rn: requirement number n

D Revision History

E Reference Documents

- Specification Document: "R&DD Assignment AY 2020-2021.pdf"
- Slides of the lectures
- UML diagrams: https://www.uml-diagrams.org/

F Document Structure

This DD in composed by 7 main sections:

- SECTION 1 is the introduction, containing the scope and the purpose of the system, together with Definitions, Acronyms, Abbreviations, the revision history of the document deployment, the reference documents and the document structure.
- SECTION 2 contains the architectural design ...
- SECTION 3 contains the user interface design, in particular some mockups show the main mobile and web application interfaces.
- SECTION 4 contains requirements traceability showing how the requirements described in the RASD map the design components identified in this document. This table clearly shows if all the components cover at least one requirement and if each requirement is met by at least one component.
- SECTION 5 concerns the implementation, integration and testing. Here it is defined how the subcomponents should be implemented and integrated and which kinds of tests should be carried out on them.
- SECTION 6 contains a table with the effort spent by each member of the group.
- SECTION 7 references/tools.

2 Architectural Design

A Overview: High-level components and their interaction

The architectural style chosen to develop the system is a three layer architecture, with a layer of Presentation, one of Business logic/Application and one of Data. This style has been chosen since it allows an easy decoupling of logic and data and of logic and presentation. The presentation level is the one which handles interactions with users, with the interfaces to communicate with them. The Business logic consists in the functions provided to the users. Moreover, this layer handles the communication between the other two layers. The data access layer, instead, manages the access to the database both for storing and retrieving data for the other layers. The hardware architecture chosen is the three-tier one. An advantage of this architecture is that the client tier doesn't communicate directly with the DBMS, so the middle tier guarantee a major level of security. Moreover, in this way the connection with the DBMS will be persistent and consequently less expensive. These application layers are divided into three physical dedicated machines. A mobile device/pc, basing on the type of user, that is a personal computer for the activities and a mobile phone for customers and store managers, is used to interface with the user. The Business logic, instead, is the application server which communicates with the DatabaseServer.

B Component view

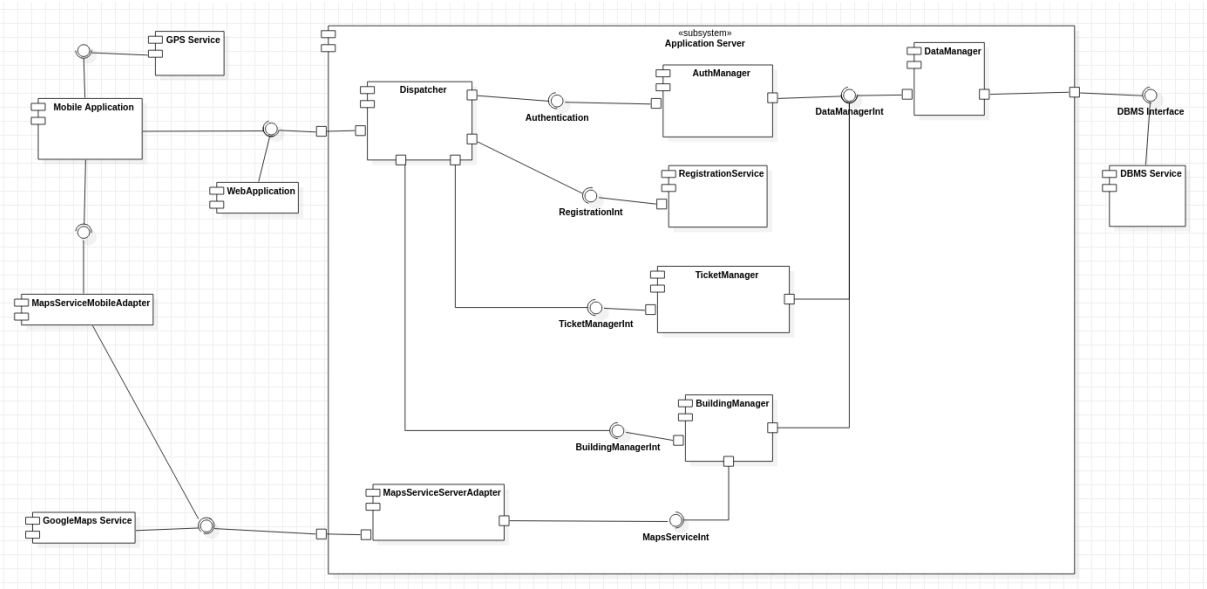

Figure 1: Component Diagram

TicketManager
This component provides services to deliver new tickets, change their state and communicates with BuildingManager to associate them with a Building.
AuthManager
This component offers services to authenticate users and deliver them tokens that authorize to use the other services. The dispatcher component will ask the Auth Manager to check the token validity before forwarding requests to other Components.
BuildingManager
Provides services to add buildings to the system, track the number of people in specific buildings, append and pop line up tickets from queues and add booking tickets to a building.
RegistrationService

Allows UnregisteredAppCustomers and Activities to register to CLup through its service.
GoogleMapsService
External service component that provides web mapping. Used to locate Buildings nearby.
Dispatcher
This component manages the requests received from MobileApp and Web Server components and redirects each one to the correct component able to handle it.
Mobile Application
Acts as a Client sending requests to the Dispatcher component. It is a fat client since it does contain Applicative logic.
Web Application
Web Client sending requests to the Dispatcher like the Mobile Application. It will be thinner than the Mobile Application taking part only at the Presentation Level since it does not contain any Applicative logic.
DataManager
Manages data structures and objects to persist to the database.
DBMS Service
Represents the DBMS and the service he offers. It communicates with the DataManager
GPS Service
Positioning system used to locate a device.
MapsServiceMobileAdapter
Adapter that offers to the MobileApplication component an interface for using Mapping services. Useful if the external Mapping service changes the Mobile Application will always used the same interface.
MapsServiceServerAdapter
Like previous adapter but for the BuildingManager component. Having 2 different Adapters offers the possibility to use different Mapping services for MobileApplication and BuildingManager

C Deployment view

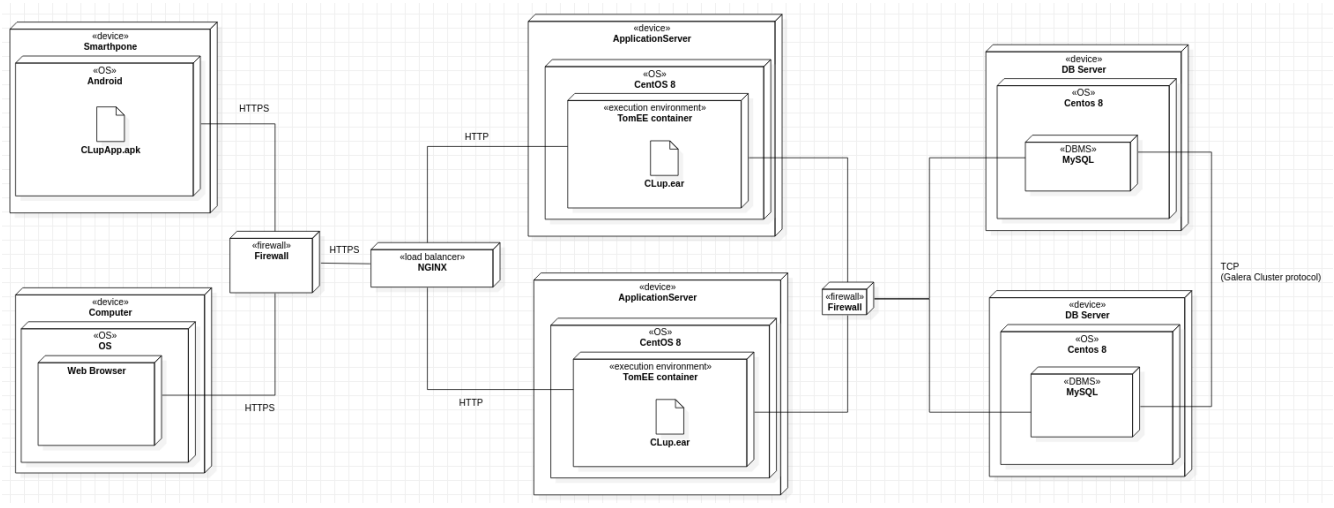

Figure 2: Deployment Diagram

This diagram shows the overall deployment architecture. Here is a description of every element:

- **Smartphone**: Device used by AppCustomers and StoreManagers. It will have to contain the CLup App executable in order to use the system.

- **Computer**: Device used by Activities in order to use the WebApp.
- **Firewall**: Provides safety access to the internal network of the system as part of the safety of the system against external attacks.
- **Nginx**: Nginx server used to balance the load on the multiple Application servers. It is responsible to establish an encrypted connection with the clients through Transport Layer Security (TLS). From this point on the connection will be unencrypted as we consider the internal network as secure.
- **ApplicationServer**: Contains most of the Application Logic. Because of the stateless behavior of the server side logic Application Servers can be easily added to scale out and maintain high availability and reliability. The CLup.ear artifact is the implementation of the ApplicationServer subsystem found in the Component Diagram.
- **DB Server**: It implements the DBMS Service in the Component Diagram. It is part of a Galera Cluster, a synchronous replication solution, transparent to the Application Servers to improve availability and performance of the DBMS Service. The Cluster will be deployed in a distributed load balancing configuration: each Application server will have a JDBC connector configured for load balancing; this avoids to have between the Application Servers and the Database Servers a separated load balancer that could be a single point of failure (if an ulterior backup load balancer is not added) and a potential bottleneck.

D Runtime view

Here are proposed sequence diagrams in order to describe the way components interact with each other to accomplish specific tasks.

Premise

It's not explicitly stated in every sequence diagram that a token checking is performed for every request done from Clients to the Server. The Dispatcher, before processing the request just received, will perform on AuthServer a token validity check, which comprises a verification on the funcionality that a Client can request on the Server.
Instead the identification action performed in some diagrams has the task to retrieve the User information, usually an identifier.

D.1 Authentication Sequence Diagram

Two different authentication procedures can occur, the first is an authentication on a RegisteredAppCustomer, instead a daily access to CLup services can be requested by UnregisteredAppCustomer.

Description

Both of the following process lead to gaining a token, that is a temporary access key for Server functionalities. The second one is Daily in the sense that every operation on server would stay for at most one day.

account based - an authentication request is made by the MobileApplication, after correctly forwarding this request to AuthManager an account check is done. If the user exists and has no token already associated with its account, then a new one will be created. MobileApplication receives a response containing his token.

daily - a daily authentication request is made by the MobileApplication, after correctly forwarding this request to AuthManager an access check is done. If the client can get a token in order to access server basic functions a new association between a specific MoibleApplication and a new token is made, else a negative response is sent back to the client.

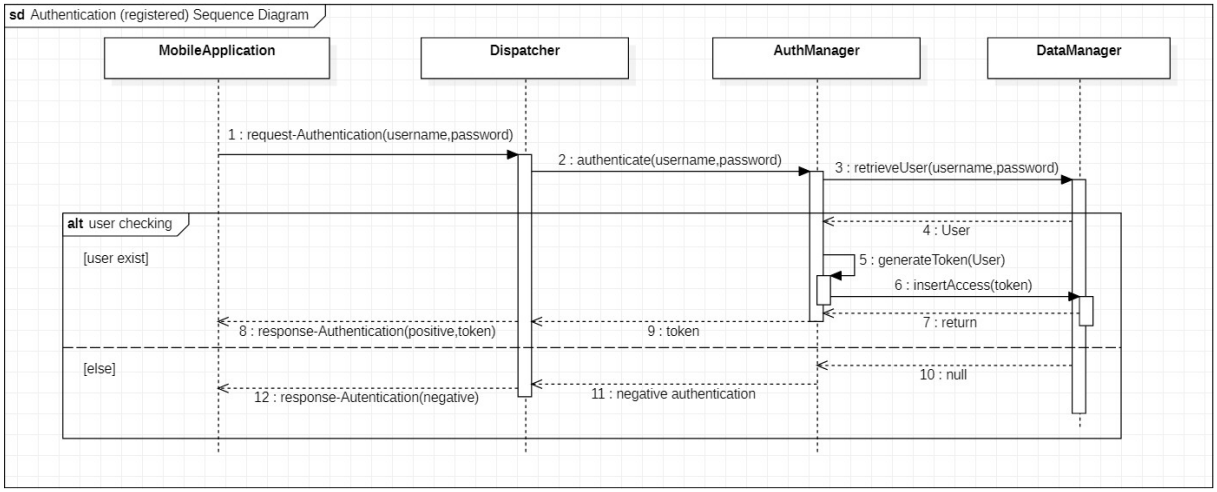

Figure 3: Authentication (registered) Sequence Diagram

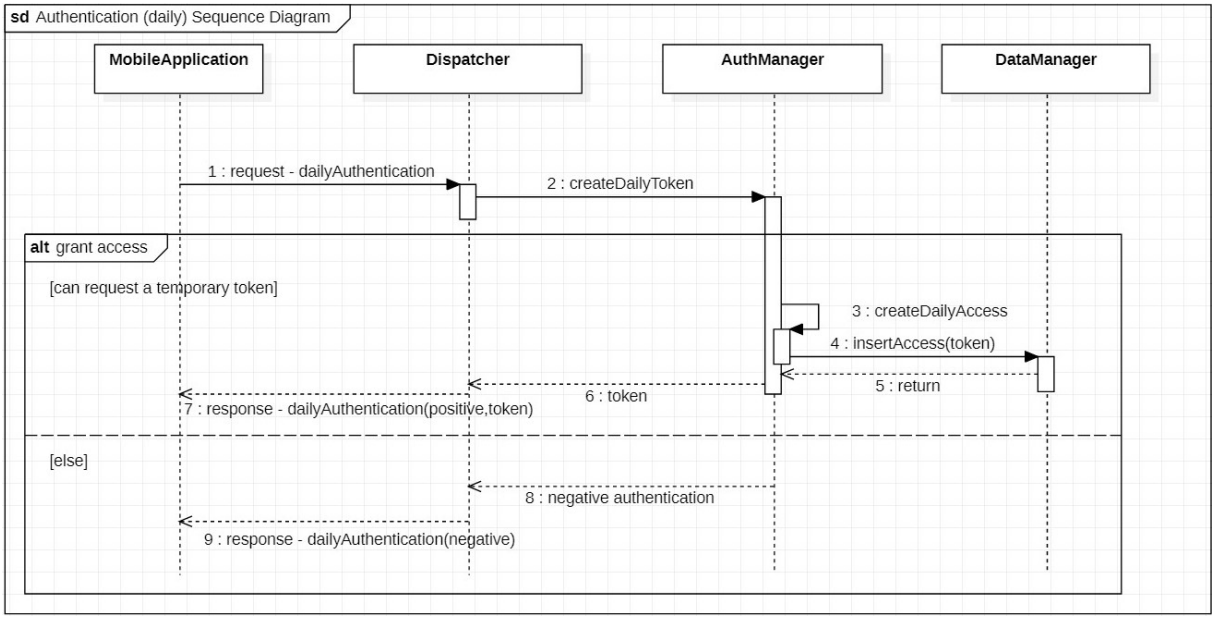

Figure 4: Authentication (daily) Sequence Diagram

D.2 Acquire a LineUpDigitalTicket Sequence Diagram

This diagram explains how an AppCustomer can acquire a LineUpDigitalTicket.

Description

MobileApplication retrieves a position (manually or through GPS) and sends a request in order to receive a list of buildings from which one would be selected. Building retrieval is effectively done by the DataManager. Whereas the MapsServiceAdapter (communication with a maps API) provides to BuildingManager useful information to correctly pick only reachable buildings.

Once the building is selected, MobileApplication sends another request in order to gain a ticket. An identification is performed on AuthManager to correctly retrieve the UserID (associated to the token) needed in ticket acquisition. Afterwards, to complete the process,if the building has at least one void space the ticket state is setted to Valid, else the ticket is inserted in queue. If ticket insertion goes wrong a negative acknowledgment is sent back.

Note that only after a Discovery request from the MobileApplication, this ticket will produce a notification (as described in Discovery diagram).

Variant

MobileApplication ticket request performed by a StoreManager works equally, but building list is not requested because of it is constrained by the one for which StoreManager is signed in.

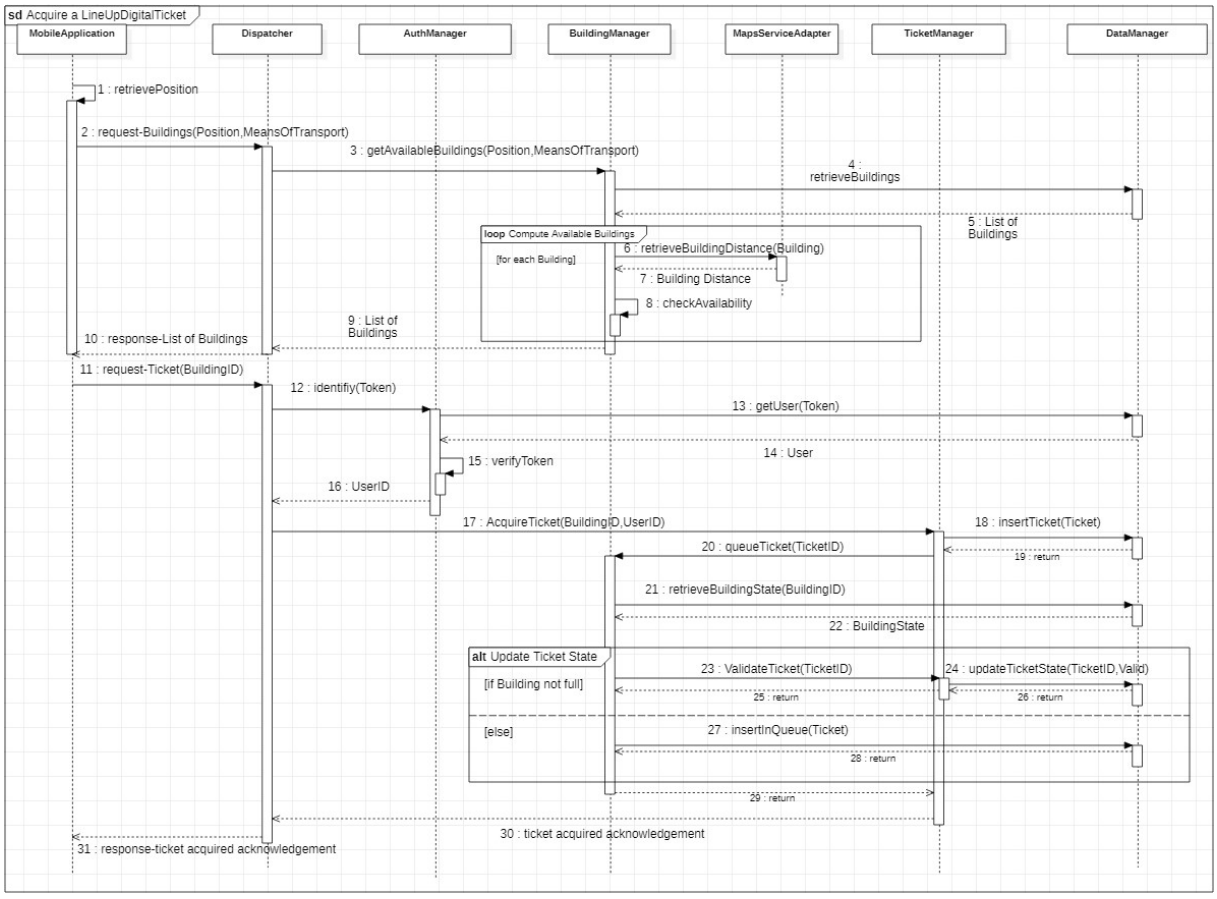

Figure 5: Acquire a LineUpDigitalTicket Sequence Diagram

D.3 Acquire a BookingDigitalTicket Sequence Diagram

This diagram explains how a RegisteredAppCustomer can acquire a BookingDigitalTicket
Description
MobileApplication retrieves a position (manually or through GPS) and sends a request with the purpose to receive a list of buildings from which one would be selected. Building retrieval is effectively done by the DataManager. Whereas the MapsServiceAdapter (communicating with a maps API) provides BuildingManager useful information to correctly pick only reachable buildings.
Once the building is selected, MobileApplication sends another request in order to have a list of TimeSlots. Dispatcher forwards this request to the BuildingManager that primarily retrieves Building which capacity information is needed to compute TimeSlots availability for a specific date.
Lastly MobileApplication requests the Server to gain a ticket. An identification is performed on AuthManager to correctly retrieve the UserID (associated with the token) needed in ticket acquisition. Then a general check validity is performed by BuildingManager, non valid tickets will be not acquired. An acknowledgement is sent back to MobileApplication.

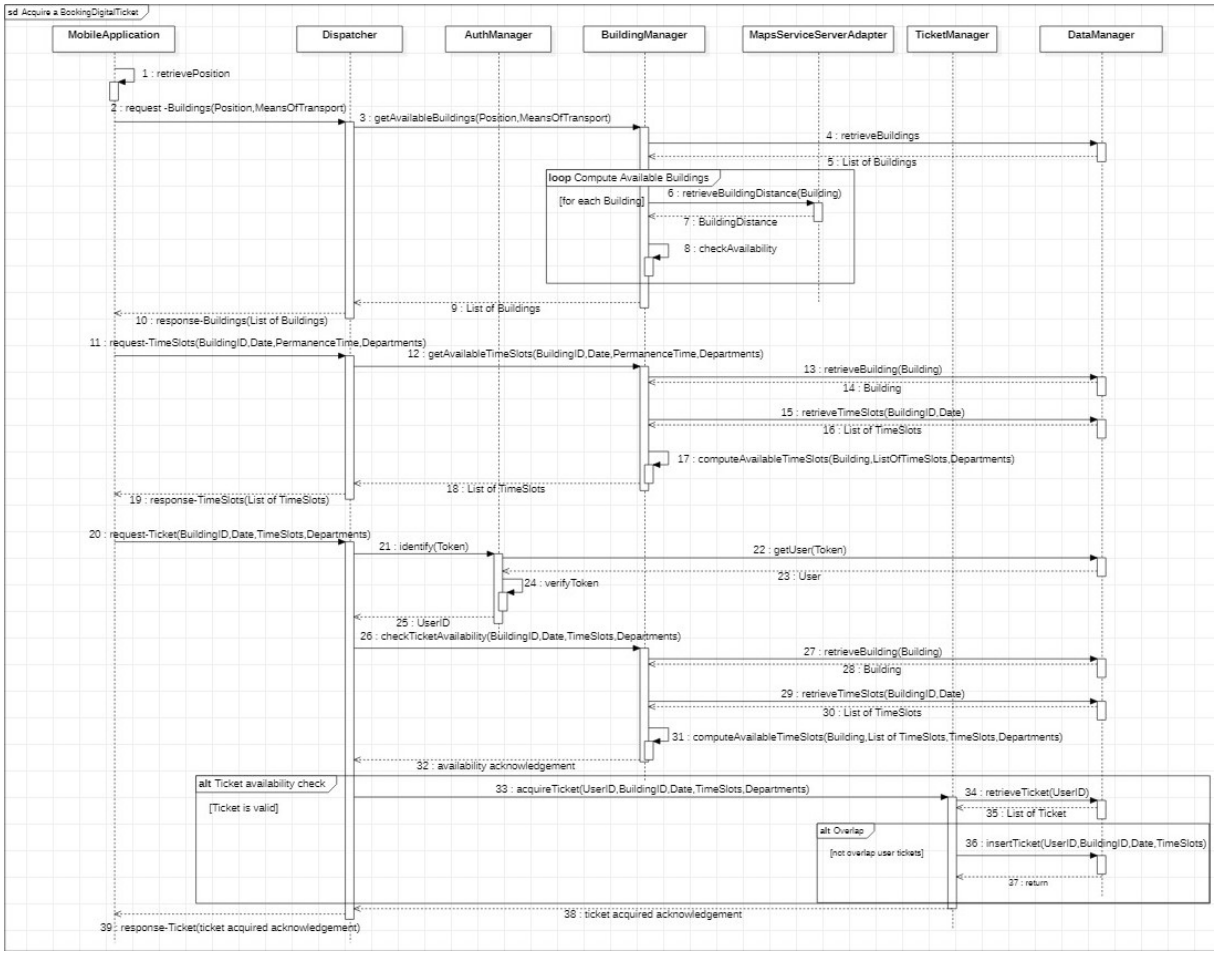

Figure 6: Acquire a BookingDigitalTicket Sequence Diagram

D.4 Discovery Sequence Diagram

This diagram explains how an AppCustomer can be notified about when to start reaching a Building and about its ticket validity or a StoreManager when to inform validity of some PhysicalCustomers's Tickets.

Description

A Discovery request is a polling request made by a MobileApplication in order to receive updates about Tickets. The Dispatcher first identifies the User and then get updated waiting time estimation from TicketManager.

The waiting time computation process is made by means of a statistic (peculiar of each Building and kept up to date on every exit event) and the queue (which defines the ticket position) in the waiting time computation service. Waiting time depends also on exiting delays which would have a particular rounding. Ticket and estimated waiting time for each ticket are sent back to Mobile Application, that will start checking all of them.

For AppCustomer a notification will be received based on the actual position and on ticket estimated waiting time, instead a StoreManager will check only for just validated tickets.

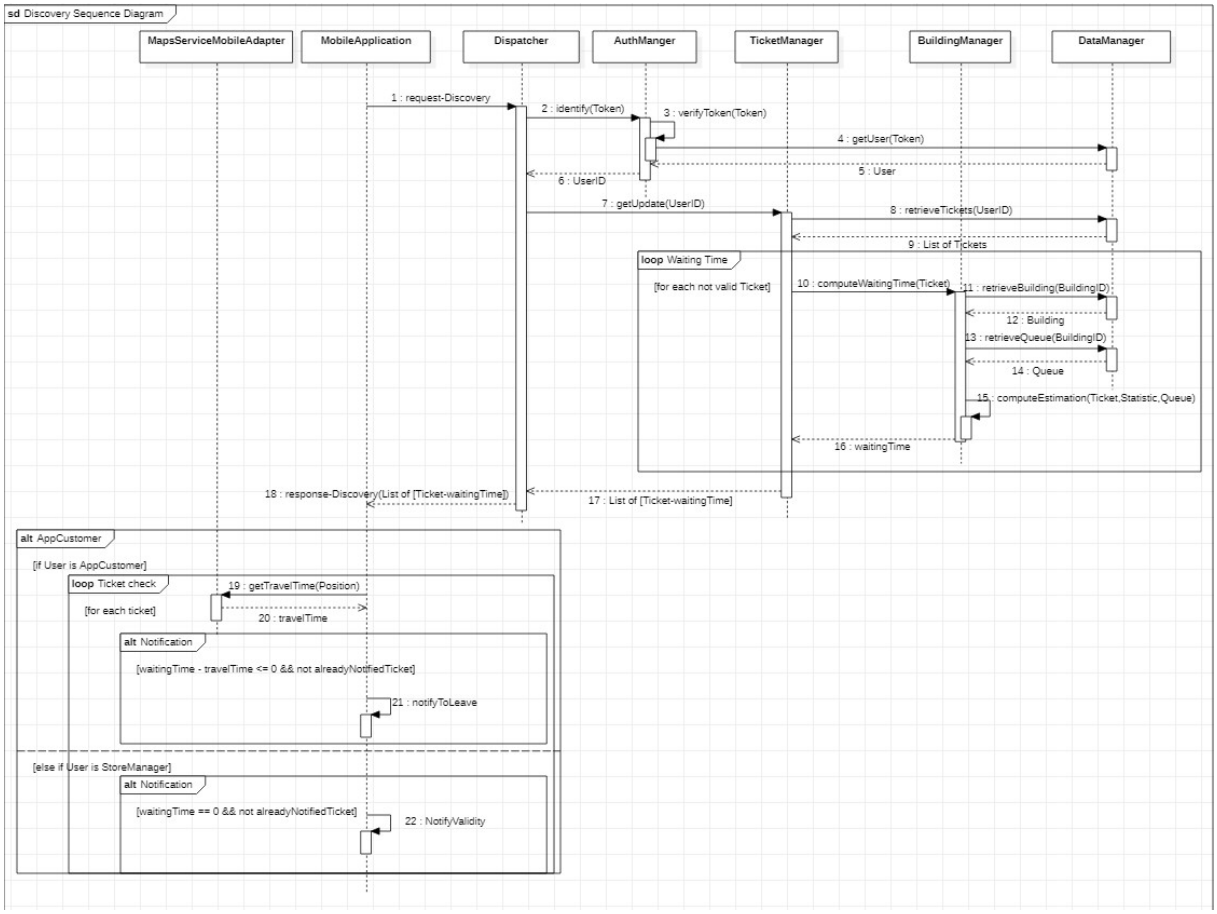

Figure 7: Discovery Sequence Diagram

D.5 Customer Exit Building Sequence Diagram

This diagram explains how a StoreManager can notify CLup about a Customer left the Building.
Description
MobileApplication exit request is sent to Dispatcher that identify User(StoreManager associated with the Building for which he is signed up for) then the request is forwarded to BuildingManager.
It will keep up to date statistic (means of delta customer leave) and last exit time for that specific buildin, the nthe validation process will start. Queue is retrieved from DataManager and the next ticket in queue (if there is one) will be validated. An acknowledge for successful processing is received by Dispatcher and then MobileApplication.

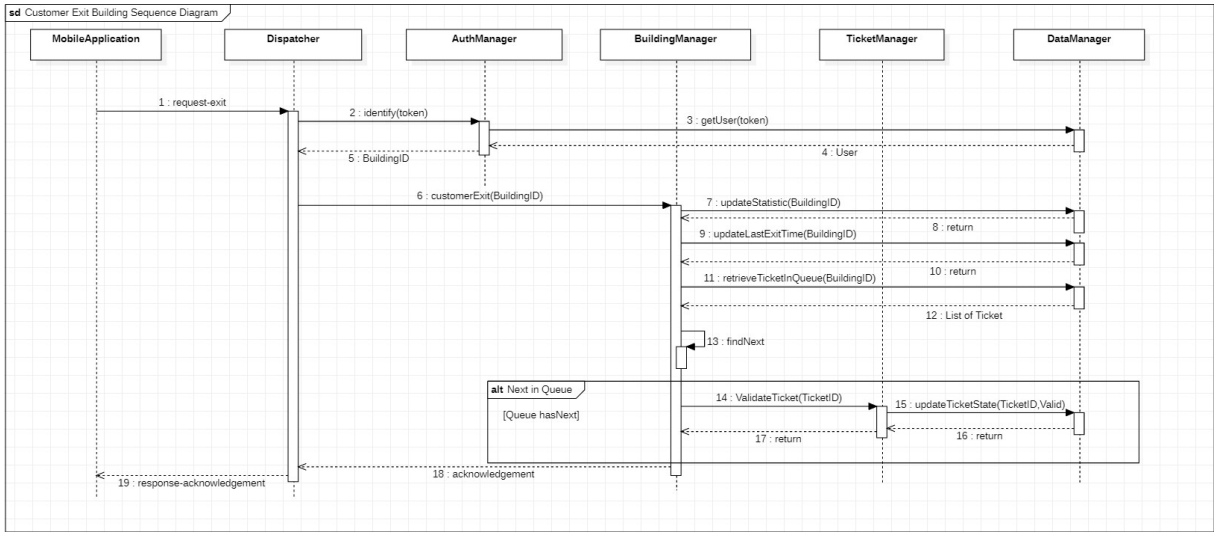

Figure 8: Customer Exit Building Sequence Diagram

D.6 Activity Insert Building Sequence Diagram

This diagram explains how an Activity can insert a new Building in CLup.

Description

WebApplication requests to insert a building providing some parameters, and optionally departments with surplus capacity. Dispatcher receives the request, it firstly identifies the user (that should correspond to an Activity) and then forward the insertion request to BuildingManager, that in order to correctly check if the building can be inserted,it retrieves a list of all buildings and the location,through the MapsServiceServerAdapter, given by the address. At this point if the Building is valid it is persisted. An acknowledgement is sent back to WebApplication.

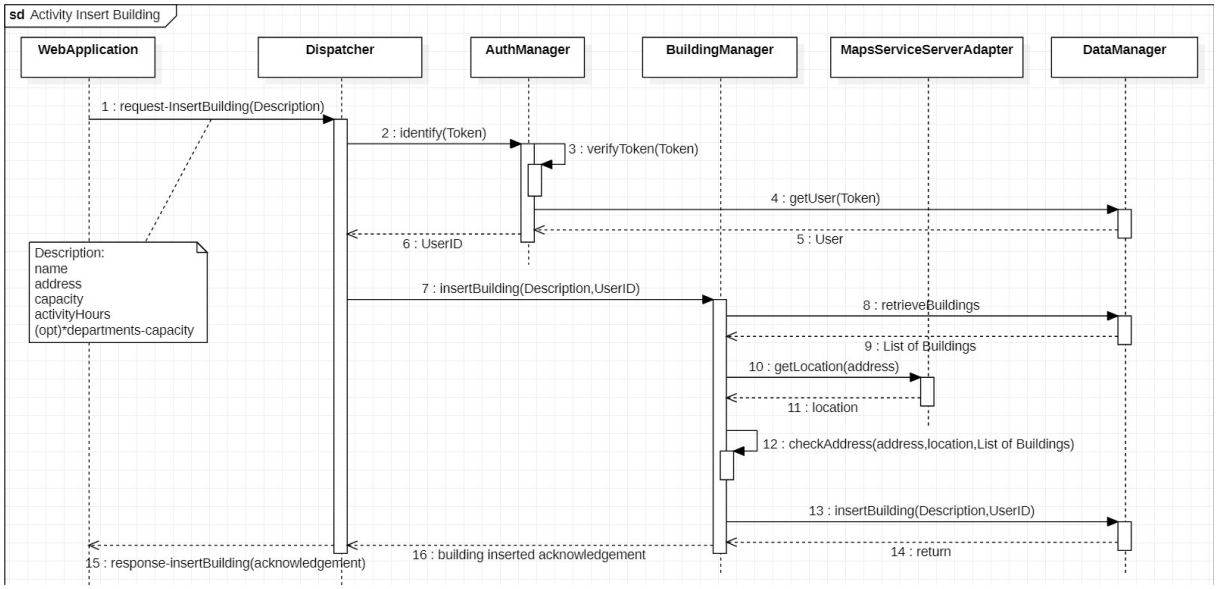

Figure 9: Activity Insert Building Sequence Diagram

E Component Interfaces

The following diagram shows all the component interfaces already exploited in the sequence diagrams together with the dependencies between the various components.

F Selected architectural styles and patterns

The architectural style selected is a three-tier client-server architecture, in order to have a good decoupling of logic, data and presentation, increasing reusability, flexibility and scalability. Moreover, components in the application server have to be developed mainly with low coupling among modules in order to make the system more comprehensible and maintainable. About the components, they have been designed to maintain a stateless logic as much as possible, that is, they should not contain an internal state, but refer to the database to get the necessary information. This is important since instances of components can fail and nothing must go lost in this eventuality. In this context the scalability of the database is so important, and DataAccessManager will play a leading role. The protocol used to send requests is HTTP, which is a good choice to implement a RESTful architecture to meet the above objectives of having a stateless and low coupling system. Other advantages are that it would be cacheable and with a uniform interface. To ensure a secure and reliable communication between client and server, HTTPS is used with TLS encryption. Data are transmitted in JSON, which is one of the simplest and most easily customizable protocol. It is also easy readable and allows fast parsing.
Finally, we've decided to use some design patterns in order to exploit existing models to solve recurrent problems. This benefits the reusability and maintainability of the code, as well as making it easier for designers to understand how the system works. Below the patterns used:

F.1 Model View Controller - MVC

MVC is a widely used pattern, particularly suitable for the development of applications written in object-oriented programming languages such as java. MVC is based on three main roles which are: the Model that contains all the methods to access the data useful to the application, the View that visualizes the data contained in the model and deals with the interaction with users and the Controller which receives the commands of the user and executes them them modifying the other two components. In CLup, the Controller logic is in the Dispatcher component, the Model is represented by services offered by the other application server's components, plus the DBMSAccessService, instead the view is in the Mobile and Web Applications.

F.2 Adapter pattern

Adapter is a structural pattern that aims to match interfaces of different classes. The interface of the Adapter is interposed between the system and the Adaptee, that is the object to be adapted. In this way, whoever has to use a method of the Adaptee sees only an interface (or an abstract class) which would be implemented according to the component to be adapted. In the case of CLup, the components MapsServiceMobileAdapter and MapsServiceServerAdapter act as the adapter for the mobile application and the server, and GoogleMapsService is the component to be adapted. In this way, even assuming that the external service is changed, the internal system will not undergo any changes, since the new API will be handled by these components.

F.3 Facade pattern

Facade is also a structural pattern, which consists in a single class representing the entire subsystem. In the case of CLup, the Dispatcher takes all the requests from the client and then directs them to the specific component of the AppServer. The aim of this component is to mask the complexity of the entire subsystem, with which you can communicate via a simple interface.

G Other design decisions

G.1 Thin Client

Having most of the business logic on server side, our client can be defined thin, although there are small pieces of business logic also in the client side. In fact, the client has a MapsServiceMobileAdapter to access the external component in order to compute independently the travel time in a real time way, in case it has decided to activate the GPS services. An advantage of this choice is a lighter communication with the server, which is not constantly updated on the position of the client when not necessary. Also the control on when to notify the client to leave for the building is made client side, given such information about position and time to reach it, avoiding to overload the server. Another advantage of having a thin client with a little of business logic is that, having an application which works mainly online, in any case it will always be connected to the server having the main logic, but even if the connection would be interrupted, some services encapsulated in the client will still remain available.

G.2 Database

...

3 User Interface Design

A UserMobileApp Interfaces

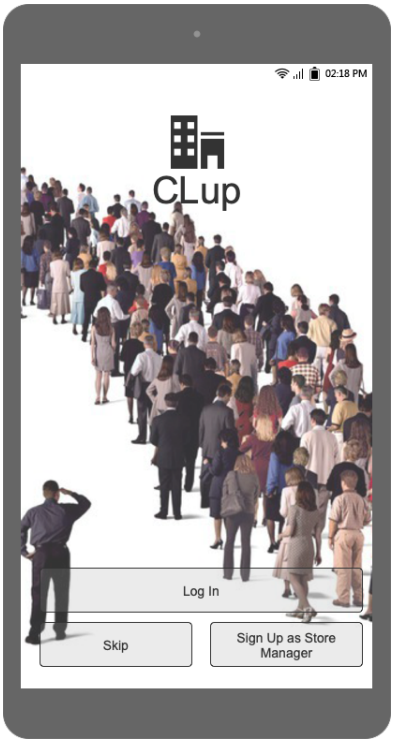

Figure 10: Starting page

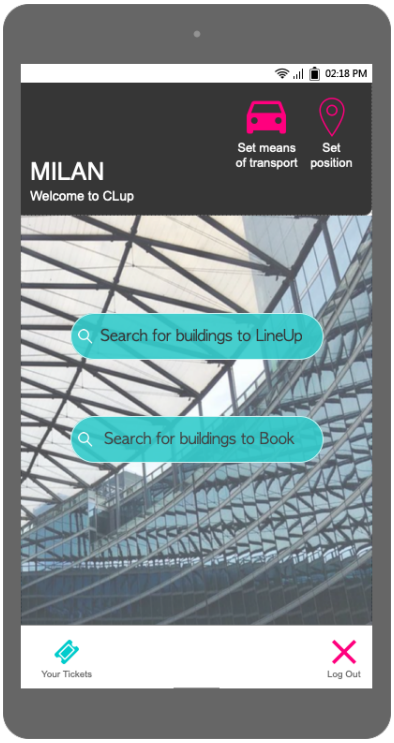

Figure 11: General home page of a registered customer

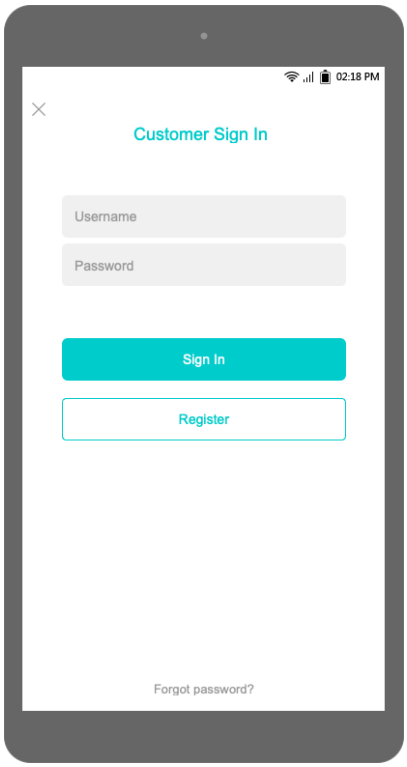

Figure 12: Interface for customers that want to sign in

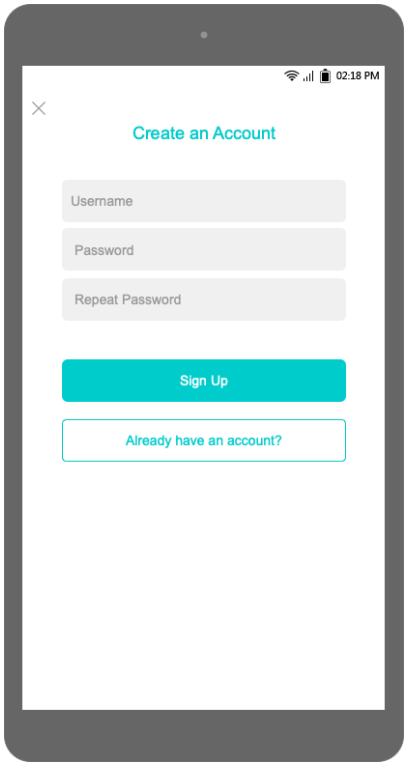

Figure 13: Interface for customers that want to register to CLup

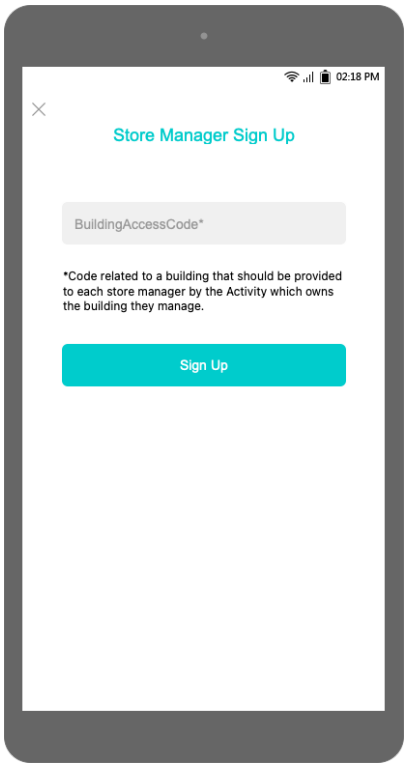

Figure 14: Interface for store managers to sign up

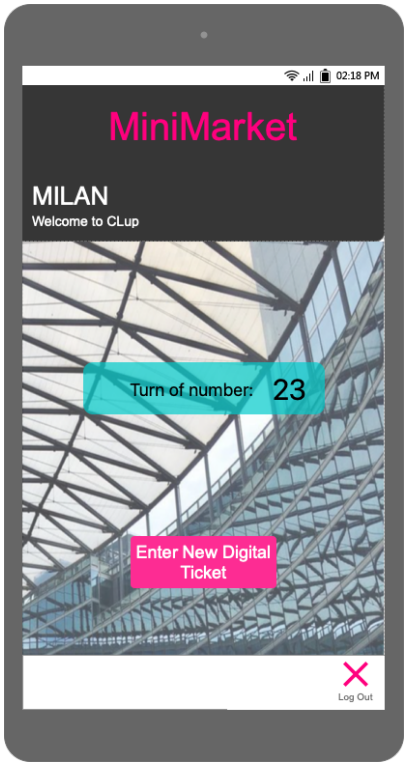

Figure 15: Home Page for store managers that have signed up

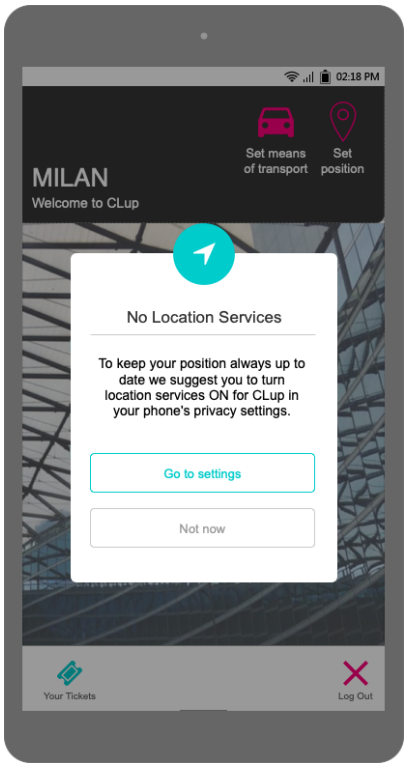

Figure 16: Pop-up to activate GPS services

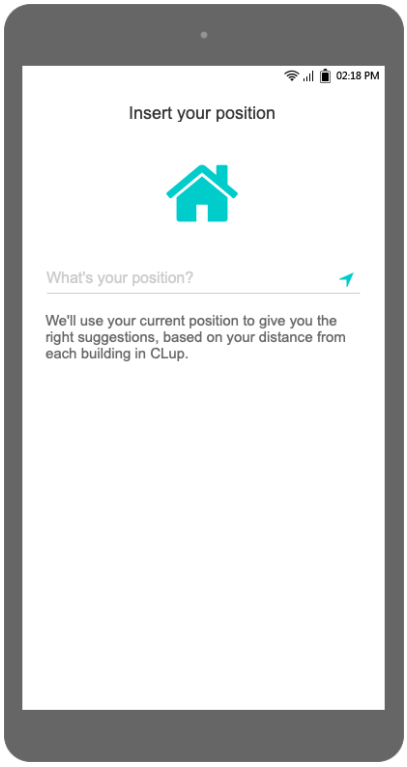

Figure 17: Setting page to set a position manually

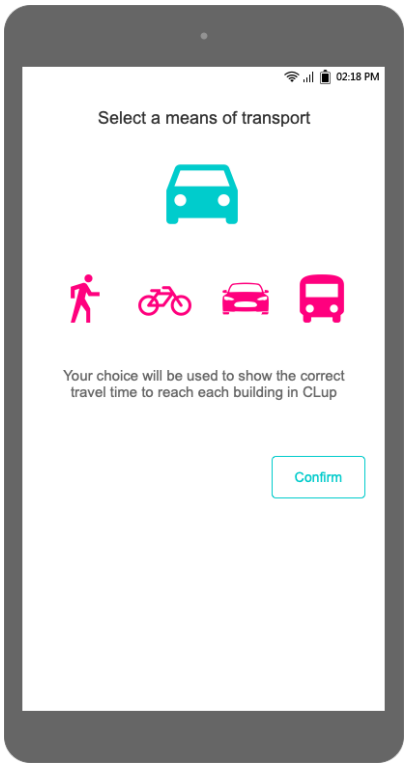

Figure 18: Setting page to choose a means of transport

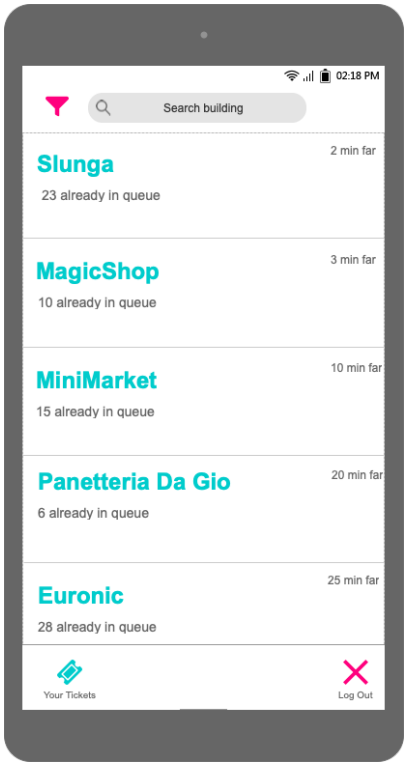

Figure 19: List of available buildings

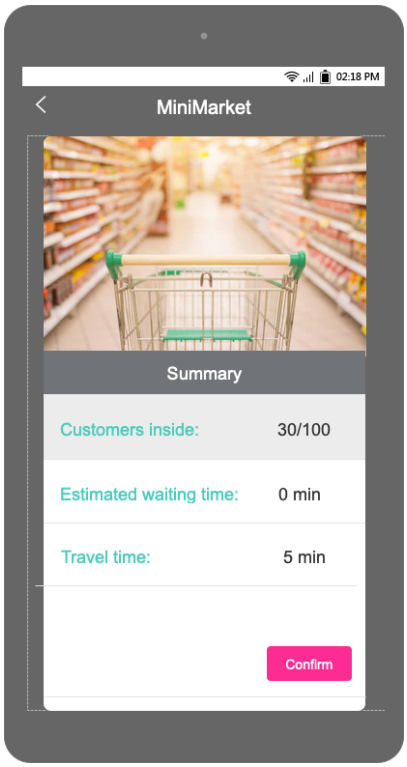

Figure 20: Summary page after the choice of a building to line up for.

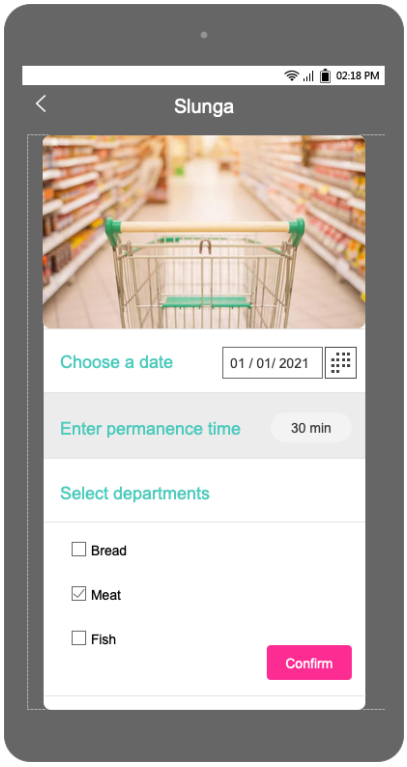

Figure 21: Options to set in order to book a time slot for the chosen building

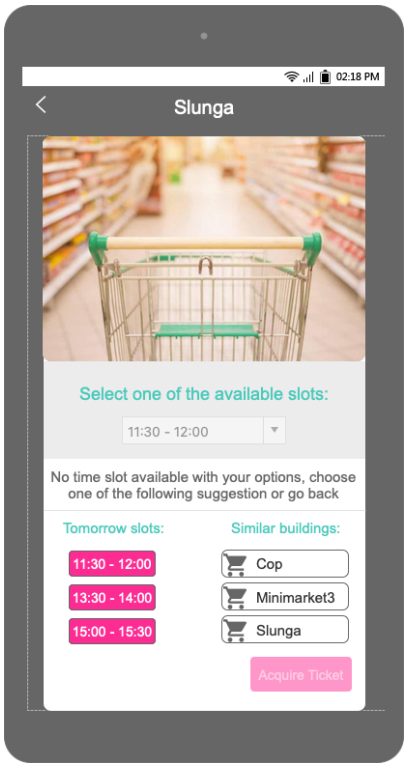

Figure 22: Interface to decide one of the available time slot/see alternative suggestions

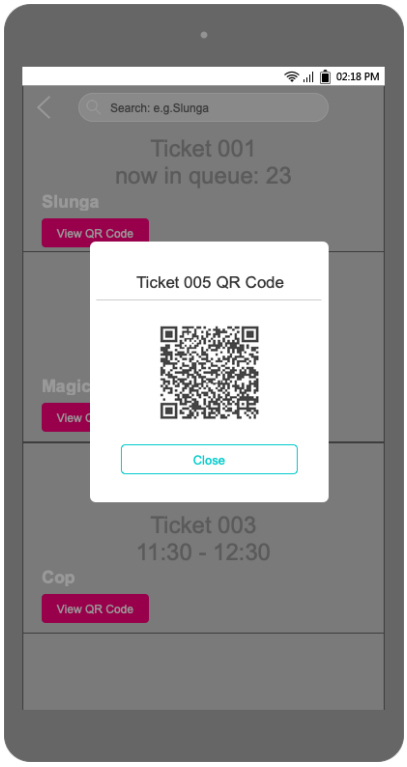

Figure 23: List of tickets of a customers, with the possibility to view the related QR code

B WebApp Interfaces

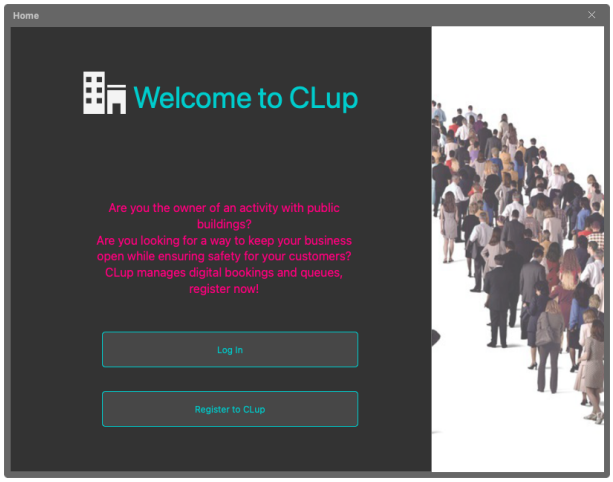

Figure 24: List of tickets of a customers, with the possibility to view the related QR code

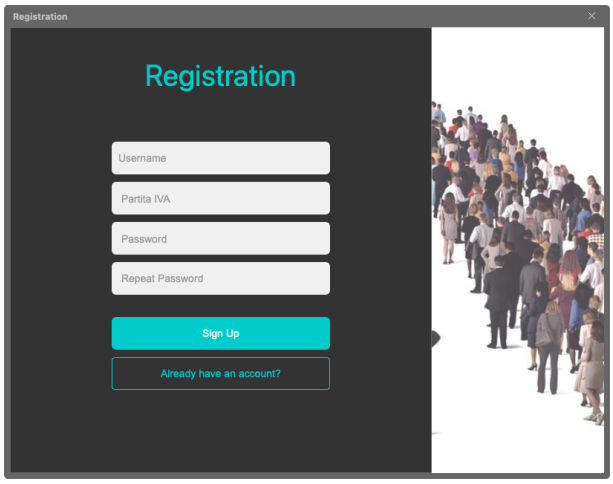

Figure 25: List of tickets of a customers, with the possibility to view the related QR code

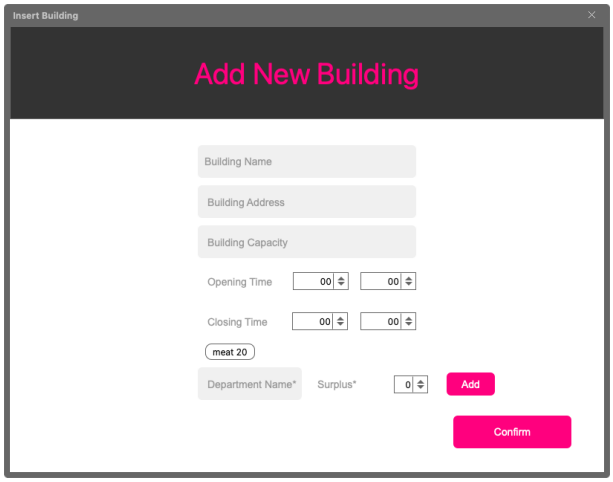

Figure 26: List of tickets of a customers, with the possibility to view the related QR code

4 Requirements Traceability

A External Interface Requirements

-	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14
R1						X								
R2						X								
R3				X										
R4				X	X									
R5								X						
R6								X						
R7								X						
R8								X						
R9								X		X				
R10										X				
R11										X				
R12											X			
R13											X			
R14											X			
R15											X			
R16											X	X		
R17					X									
R18							X							
R19							X							
R20							X							
R21							X							
R22							X							
R23									X					
R24							X							
R25							X							
R26							X							
R27							X							
R28													X	
R30									X					
R31														X
R32														X
R33														X
R35									X					

-	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14
R36	X													
R37		X												
R38		X												
R39		X												
R40		X												
R41		X												
R42		X												
R43		X												
R44		X												
R45		X												
R46		X												
R47		X												
R48		X												
R49		X												
R50		X												
R51		X												
R52		X												
R53		X												
R54		X												
R55		X												
R56		X												
R57			X											
R58			X											
R59			X											
R60			X											
R61			X											
R62			X											
R63			X											
R64			X											
R65			X											
R66			X											
R68							X							
R69	X													
R70	X													
R71	X													
R72									X					
R73									X					
R74			X											
R76			X											
R77									X					
R78									X					

5 Implementation, Integration and Test Plan

A Implementation

B Integration

C Test plan

6 Effort Spent

Task	*Name*	*Time spent*
x	Francesco Attorre	h
x	Thomas Jean Bernard Bonenfant	h
x	Veronica Cardigliano	h
x	Francesco Attorre	h
x	Veronica Cardigliano	h
x	Veronica Cardigliano	h
x	Francesco Attorre	h
x	Thomas Jean Bernard Bonenfant	h
x	Veronica Cardigliano	h
x	Veronica Cardigliano	h
x	Thoma Jean Bernard Bonenfant	h

comments

7 Used Tools

Tools used to create this RASD document:

- StarUML: for all the UML diagrams
- LaTeX: to create the pdf
- GitHub: for the repo of the project
- GoogleDoc: for a shared editing of the document