POLITECNICO
MILANO 1863

CLup

Design Document

Simone Abelli
Stefano Azzone

Deliverable:	DD
Title:	Design Document
Authors:	Simone Abelli, Stefano Azzone
Version:	1.0
Date:	19 December 2020
Download page:	https://github.com/StefanoAzzone/AbelliAzzone
Copyright:	Copyright © 2020, Simone Abelli, Stefano Azzone – All rights reserved

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to provide more technical and detailed information about the software discussed in the RASD document. The Design Document is a guide for the programmer that will develop the application in all its functions. The document will explain and motivate all the architectural choices by providing a description of the components and their interaction. We will also enforce the quality of the product through a set of design characteristics. Finally we describe the implementation, integration and test planning.
The topics touched by this document are:

- high level architecture
- main components, their interfaces and deployment
- runtime behavior
- design patterns
- more details on user interface
- mapping of the requirements on the components of the architecture
- implementation, integration and test planning

1.2 Scope

CLup is a system that allows customers to line up in a virtual first in first out queue, in order to avoid overcrowding outside of stores. Customers can queue up remotely or on premise (by using a device installed outside the store). When a customer queues up remotely he/she can choose to line up immediately or to book a future visit. The system alerts customers when it is time for them to depart to reach the store. The system builds statistics on customer entry and exit in order to provide a better estimation of waiting times. The system allows the store owners to control the occupation of each of their stores. This is just a summary of all the features of the system, for a more detailed description of the software functionalities read the RASD.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

Reservation	Virtual or physical artifact used to identify the position of a customer in a queue
Queue up	Customers are lined up in a FIFO queue
Enqueued	A customer is enqueued when he has provided the system with a means of identification and requested a reservation
Authorized	A customer is authorized when he has been enqueued and is allowed temporary access to the store.
Occupation	Number of customers currently present in the store
Printer	Device that can read a social security card and print tickets that contains a progressive number and an estimate of the waiting time.
User	Either a customer or a store owner.

1.3.2 Acronyms

RASD	Requirement Analysis and Specification Document
GPS	Global Positioning System
S2B	Software to be
UI	User Interface
FIFO	First in first out

1.3.3 Abbreviations

Gn	Goal number n
Rn	Requirement number n

1.4 Revision history

Not yet defined.

1.5 Reference Documents

1. IEEE Std 830-1998 Recommended Practice for Software Requirements Specifications
2. Specification Document: R&DD Assignment A.Y. 2020/2021
3. uml-diagrams.org

1.6 Document Structure

- Chapter 1: gives an introduction about the Design Document, enumerating all the topics that will be covered. Moreover this section contains specifications such as the definitions, acronyms, abbreviation, revision history of the document and the references.
- Chapter 2: contains the architectural design choices, with an in-depth look at the high level components and their interactions. It include several views: the component view, the deployment view and the runtime view. Here are described the interfaces (both hardware and software) used for the development of the application, their functions and the processes in which they are utilized. Finally, there is the explanation of the architectural patterns chosen with the other design decisions.

- Chapter 3: this section provide an overview of how the user interfaces of the system will look like.
- Chapter 4: This chapter maps the requirements defined in the RASD to the design elements defined in this document.
- Chapter 5: here we define the plan for the implementation of the subcomponents of the system and the order in which the integration operations and their testing will be performed.
- Chapter 6: shows the effort which each member of the group spent working on the project.
- Chapter 7: includes the reference documents.

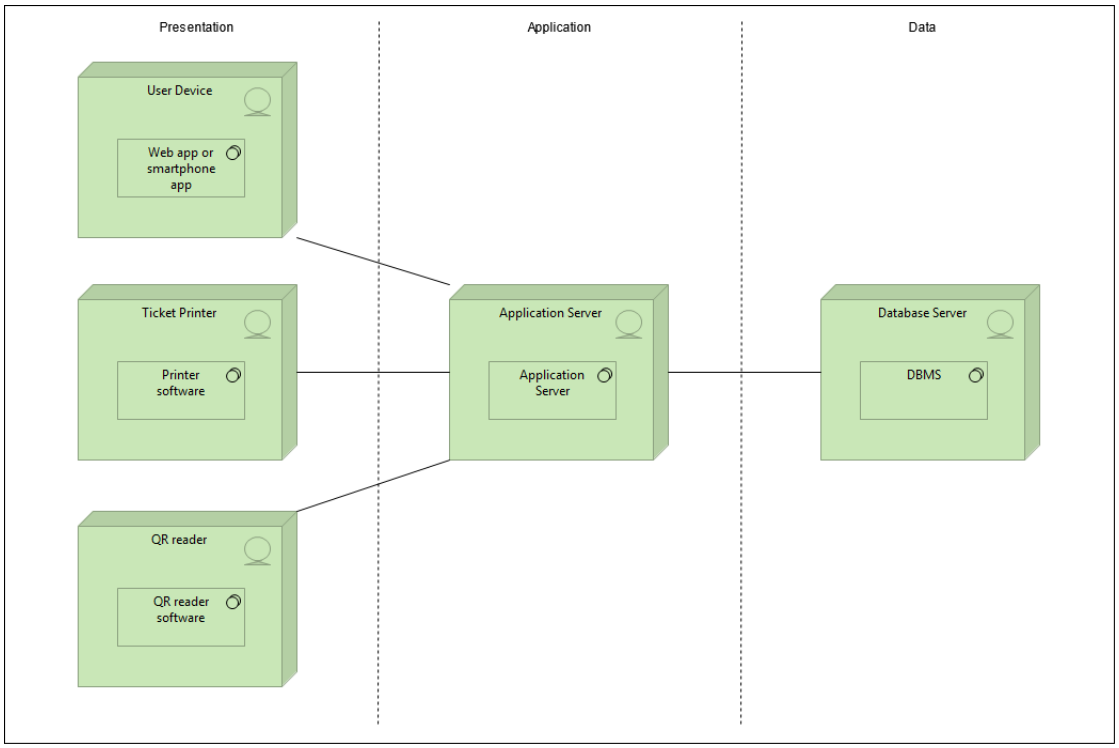
2 Architectural Design

2.1 Overview

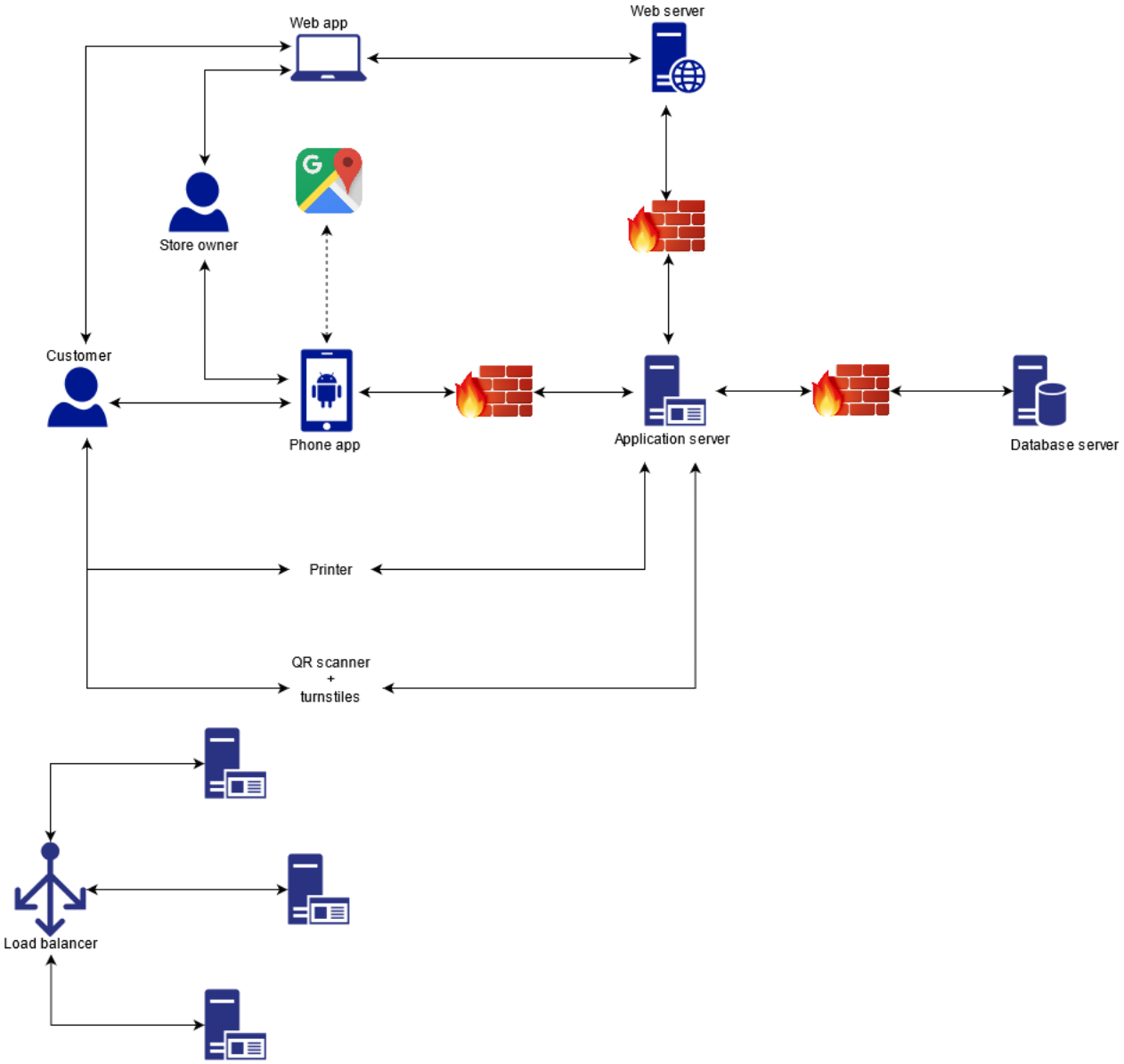
The architecture of the S2B is a distributed client-server architectural design, structured according to three logic layers:

- **Presentation level P**: manages the user interaction with the system. This layer contains the interfaces able to provide the functions of the application to the users.
  To the presentation layer belong the web app, the phone application and the software on the ticket printer and on the QR reader.
- **Business logic or Application layer (A)**: handles the business logic of the application and its functionalities. This layer represent the core of the application logic.
- **Data access layer (D)**: manages information and data, by accessing the database.

Every logic layer can be mapped in an hardware layer.
The presentation layer is composed by the smartphone or the computer of the user, the ticket printer outside the stores, the QR reader and the turnstiles.
The application layer is composed by the application server. The data layer is composed by the database server.

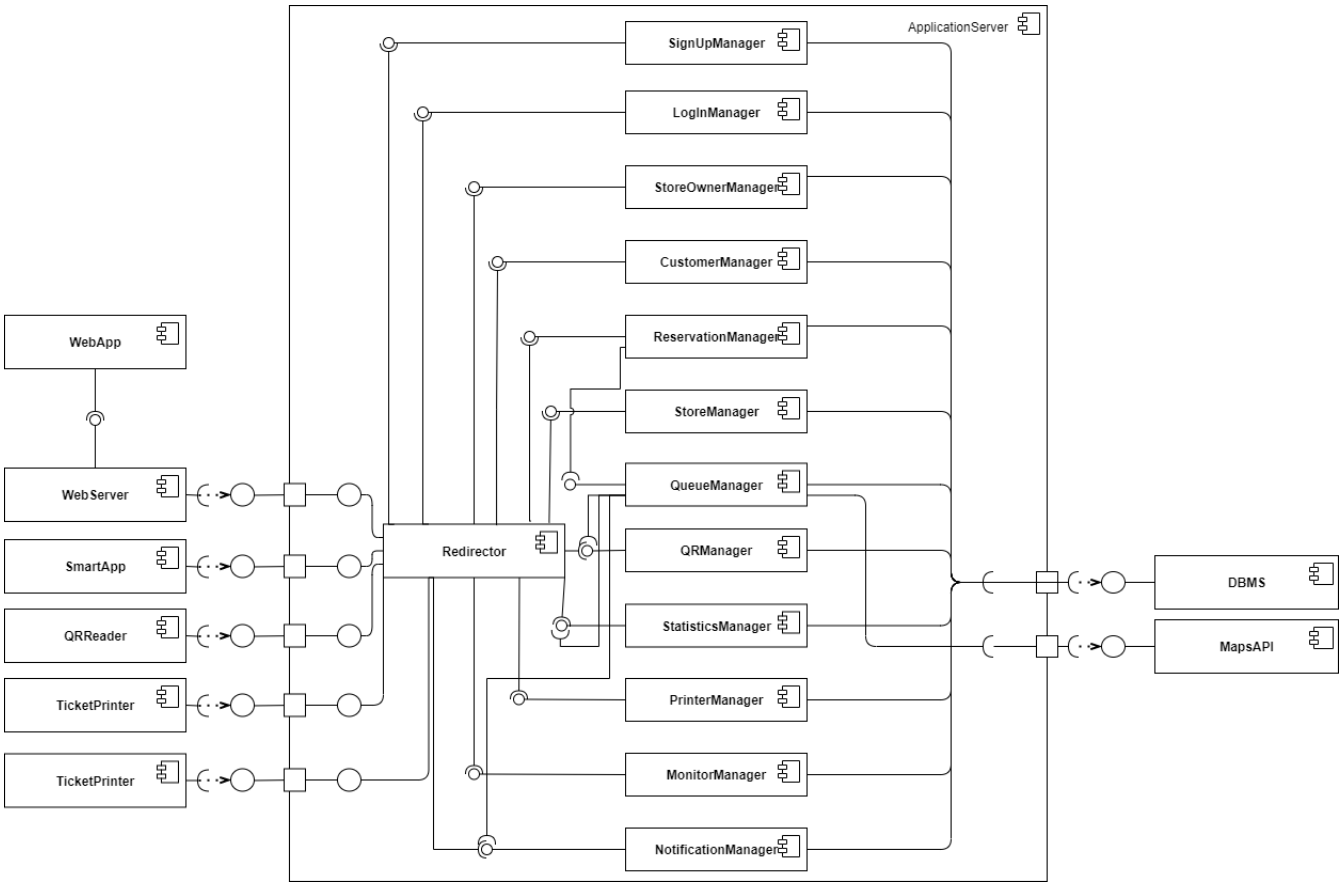


Customers line up at a store trough the web app or the smartphone application and their requests are sent to the



2.2 Component View

Here we display the main component architecture of our S2B. Since the ApplicationServer contains all the business logic, we will describe in detail the structure of its subcomponents.



2.2.1 WebApp

This component allows customer and store owners to access their respective services on a computer. It requires a WebServer.

2.2.2 WebServer

This component communicates directly with the ApplicationServer and serves web pages to implement the WebApp component.

2.2.3 SmartApp

This component allows customers and store owners to access their respective services on a smartphone, by interfacing with the ApplicationServer.

2.2.4 QRReader

This component reads user provided QRCodes and sends them to the ApplicationServer, thus enabling authentication for turnstiles.

2.2.5 TicketPrinter

This component accepts a user document and after having validated it through the ApplicationServer, it prints a reservation ticket with a QR.

2.2.6 StoreMonitor

This component updates periodically calling the ApplicationServer, which provides it with the number of the last authorized reservation, thus notifying customers in front of the store when they can enter.

2.2.7 Redirector

This component provides an external interface for the previously described components, and allows them to communicate with the components that are located within the ApplicationServer, that we describe below.

2.2.8 SignupManager

This component allows customers and store owners that provide a valid identification document to register to the service, thus gaining access to its functionalities, provided that they log in. It is called through WebApp or SmartApp, and needs to access the DBMS to search for existing users with same identification document (to avoid duplication), and to create a new user.

2.2.9 LoginManager

This component allows customers and store owners to log in the service, thus gaining access to its functionalities. It is called by WebApp and SmartApp components, and needs to access the DBMS to verify user credentials against the stored ones.

2.2.10 StoreOwnerManager

This component is accessed through WebApp or SmartApp and allows store owners to edit their credentials (obtained during sign up process), thus it needs access to DBMS.

2.2.11 CustomerManager

This component is accessed through WebApp or SmartApp and allows customers to edit their credentials (obtained during sign up process), thus it needs access to DBMS.

2.2.12 ReservationManager

This component is accessed through WebApp or SmartApp and allows customers to send a reservation for a specific store, to delete an existing reservation, or to view status of non expired reservations (QR code, position in queue, status) by accessing QueueManager. It needs to access DBMS to retireve the list of the departments of the store that the reservation targets and, in case of an immediate reservation, to verify that the store is open at the current time. It needs access to DBMS also to fetch the list of non expired user reservation along with their information, and to delete them if required.

2.2.13 StoreManager

This component is accessed through WebApp or SmartApp, and allows store owners to view owned stores, add new stores, delete existing stores or update information of existing stores, such as the list of departments and their respective maximum occupation. It needs to access the DBMS to execute the previous functions.

2.2.14 QueueManager

This component is accessed by the ReservationManager to add or remove a reservation from a queue of a specific store; whenever a reservation is added, this component accesses QRManager to generate an appropriate QR code for the reservation. It needs access to DBMS to get access to the current status of the queue of each store. It checks periodically the number of customers currently present, their location and the maximum occupation for every store and computes an estimated time that customers have to wait before gaining authorization to enter a specific store if such store has already reached maximum occupation. It checks periodically through MapsAPI an estimate of the time the customer needs to reach the store with his selected means of transport. If such time becomes lesser or equal to the time he/she needs to wait before entering the store, a notification is sent to the customer through NotificationManager if he/she uses a mobile application. This component accesses StatisticsManager to improve the computation of the time a customer needs to wait before being authorized to enter the store.

2.2.15 QRManager

This component is accessed by the QueueManager during the insertion of a reservation in the queue to produce the QR that will be used to identify the customer that created such reservation. It needs access to the DBMS: when QRReader reads a QR code, it accesses this component to retrieve from the data base the reservation (if any) to which this QR corresponds, and opens turnstiles depending on the state of the reservation.

2.2.16 StatisticsManager

This component is accessed by QueueManager to improve the estimate of the time that a customer needs to wait before being authorized to enter a specific store. This component is accessed by a store owner through WebApp or SmartApp to visualize statistics. This component accesses the DBMS periodically to obtain the data with which to generate the statistics, and to save such statistics.

2.2.17 PrinterManager

This component allows the store owner with WebApp or SmartApp to register a TicketPrinter. This component also allows customer identification through TicketPrinter by validating a provided identification document. This component needs to access the DBMS to register or unregister instances of TicketPrinter.

2.2.18 MonitorManager

This component allows to update periodically a StoreMonitor. It retrieves from the DBMS the number of the last authorized reservation.

2.2.19 NotificationManager

This component is accessed by the SmartApp to retrieve notifications. This component is sent notifications by the queue manager, whenever the time to reach the store becomes greater or equal to the estimate of the time to wait before being authoruzed to enter the store. This component saves unsent notifications in DBMS until it is able to dispatch them.

3 User Interface Design

4 Requirements Traceability

5 Implementation, Integration and Test Plan

6 Effort Spent

Provide here information about how much effort each group member spent in working at this document. We would appreciate details here.

6.1 Simone Abelli

Introduction	3.5
UML class diagram	4
Product perspective	3
State charts, External interface requirements	4.5
User interfaces	4.5
Requirements	6.5
Sequence diagrams	3.5
Scenarios	0.5
Alloy	7

6.2 Stefano Azzone

Introduction	3
UML class diagram	3
Product perspective	3
Product functions, user characteristics, domain assumptions	2.5
State charts, External interface requirements	3
Use cases	3.5
Alloy	11.5
Requirements	6.5
Performance Requirements, Design constraints, Software system attributes, first two scenarios	1

References

- **drawio.org** was used to draw diagrams
- **alloy.mit.edu** was the reference for alloy model
- **uml-diagrams.org** was the reference for uml diagrams

