POLITECNICO
MILANO 1863

CLup project

Design Document (DD)

Revision 1.9 – January 12, 2021

Cristiano Neroni | Davide Pozzi | Maurizio Vetere

Deliverable:	DD
Title:	Design Document
Authors:	Cristiano Neroni, Davide Pozzi, Maurizio Vetere
Version:	1.9
Date:	12-January-2021
Download page:	https://github.com/pollo-fritto/PozziNeroniVetere.git
Copyright:	Copyright © 2021, Neroni| Pozzi| Vetere – All rights reserved

Contents

List of Figures

List of Tables

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. It will represent a strong guide for the programmers that will develop the application considering its different parts: the basic service and the two advanced functions. In this DD we present hardware and software architecture of the system in terms of components and interactions among those components. Furthermore, this document describes a set of design characteristics required for the implementation by introducing constraints and quality attributes. It also gives a detailed presentation of the implementation plan, integration plan and the testing plan. In general, the main different features listed in this document are:

- The high-level architecture of the system
- Main components of the system
- Interfaces provided by the components
- Design patterns adopted

Stakeholders are invited to read this document in order to understand the characteristics of the project being aware of the choices that have been made to offer all the functionalities also satisfying the quality requirements.

1.2 Scope

Clup is an application that aims to avoid users from crowding outside supermarkets when doing grocery shopping in pandemic times.
The application can be used both by store customers and store managers. On one hand users can virtually queue by Clup to enter the supermarket and they are provided with real time information about the line, in this way they can arrive at the entrance only when they are allowed to enter. On the other hand the application monitors and stores the information about people fluxes; this data is then provided to store managers who can take actions depending on the situation. The few paragraphs just read represent an overview of the main functionalities offered by the system: more detailed information can be found on the RASD document.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

[As per the RASD document]

- **The Developers**: we therewith refer to future developers of the CLup application, main target of the present and **CLup: RASD** documents.
- **Check-in procedure**: the process of getting inside the store. It starts from when the user approaches the entrance, includes the QR ticket scan and ends as soon as the turnstile is passed.
- **Reserve entrance**: the process of booking a future entrance (starting from the day next to the current one)

- **Malicious user**: someone committed for any reason to CLup malfunction and/or unavailability, shopping disservices.
- **Quick ticket**: the actual ticket granting access to the stores. We call it «quick» to emphasize the difference between booking and lining up.
- **Totem**: a desktop based PC with advanced input functionalities (touchscreen), with external hard shell protection, stand mount, (optional) integrated printer.
- **Big screen**: a huge screen panel to be located outside the store, in visible placement, used for announcements to offline customers.
- **CLup core system**: CLup innermost back-end functionality providing queueing control, access to already enqueued users, access control, big screens operativeness
- **Affected customers**: customer being main or side target of an event or action taking place inside grocery shopping and CLup's scope.

1.3.2 Acronyms

- **EWT**: Expected Waiting Time
- **ASAP**: As Soon As Possible
- **WRT**: With Respect To
- **PUIF**: Presentation & User Interaction Functionalities
- **FMLBF**: Flow Management & Local Business Functionalities

1.3.3 Abbreviations

1.4 Revision History

- **v1.0**: First version of the document
- **v1.1**: Component revision, general redesign
- **v1.2**: Architectural changes, final architecture definition
- **v1.3**: Sequence diagrams additions
- **v1.4**: Deployment definition
- **v1.5**: Style and formatting changes
- **v1.6**: Minor revisions and cover addition
- **v1.7**: Definitions, acronyms, abbreviations
- **v1.8**: Acronyms additions, section 2.7 revision, sequence diagrams revision
- **v1.9**: Deployment section revision, sequence diagrams descriptions revision.

1.5 Reference Documents

This documents has been designed by taking the **IEEE Std 1016TM-2009** as primary reference.[2]

Other documents of sources have been cited along the way whenever needed.
For additional information over requirements and goals found in this Document, please have a look at the **Requirements Analysis and Specifications Document (RASD)** for the CLup application

1.6 Document Structure

- Chapter 1 describes the scope and purpose of the DD, including the structure of the document and the set of definitions, acronyms and abbreviations used.
- Chapter 2 contains the architectural design choice, it includes all the components, the interfaces, the technologies (both hardware and software) used for the development of the application. It also includes the main functions of the interfaces and the processes in which they are utilised (Runtime view and component interfaces). Finally, there is the explanation of the architectural patterns chosen with the other design decisions.
- Chapter 3 shows how the user interface should be on the mobile and web application.
- Chapter 4 describes the connection between the RASD and the DD, showing the matching between the goals and requirements described previously with the elements which compose the architecture of the application.
- Chapter 5 traces a plan for the development of components to maximize the efficiency of the developer team and the quality controls team. It is divided in two sections: implementation and integration. It also includes the testing strategy.
- Chapter 6 shows the effort spent for each member of the group.
- Chapter 7 includes the reference documents.

2 Architectural Design

2.1 Overview

CLup's architecture is layered as follows:

- **Presentation layer** (P) handles the interaction with users. It contains the interfaces able to communicate with them and it is responsible for rendering of the information. Its scope is to make understandable the functions of the application to the customers.
- **Application layer** (A) takes care of the functions to be provided for the users. It also coordinates the work of the application, making logical decisions and moving data between the other two layers.
- **Data access layer** (D) which takes care of the information management, database access control. It also handles data retrieval and passes them to upper level layers.

The architecture style chosen for CLup is the **multi-tier** one. As previously anticipated in the Requirements Analysis and Specifications Document, there will be at least one server for each one of the following interest areas:

- Bookings
- Queues
- Notifications
- Stores
- Staff members
- Customers

This is mainly done to distribute workload as well as making the overall system more robust. There will be also at least two servers for the two following functionalities:

- Customer related functionalities
- Staff related functionalities

The bookings, queue and notifications databases will be distributed and replicated all over the entire store list. Each store will have its own instance of bookings, queue and notifications database while a central logic server will act as a request redirector towards them whenever needed.

In Figure 1 is represented the high-level architecture of the system: customers clients, through an internet connection, can connect to CLup's App Servers, which represent the A layer of the Clup customer side. Client's operations store o retrieve information from People DB Servers or directly from the Local DB of the store, depending by the kind of information. Staff clients instead are connected through a LAN connection to their Local Web Server, who will redirect their requests to the Staff Operations Manager server. This machine is expected to process the logic of requests and then submit them to the Local Application server who also store or retrieve the information from the Local DB or People DB server depending by the situation. Local Application Server is fundamental also for Totem users, in fact the Totem rely on this server for its requests. Staff clients may need to connect to Clup Centralized Hardware, to do this they need an internet connection to get in contact with the dedicated Central Web server, the P layer who then send the request to the Store Operations Manager, the A layer. This server stores and retrieves information from Stores DB servers

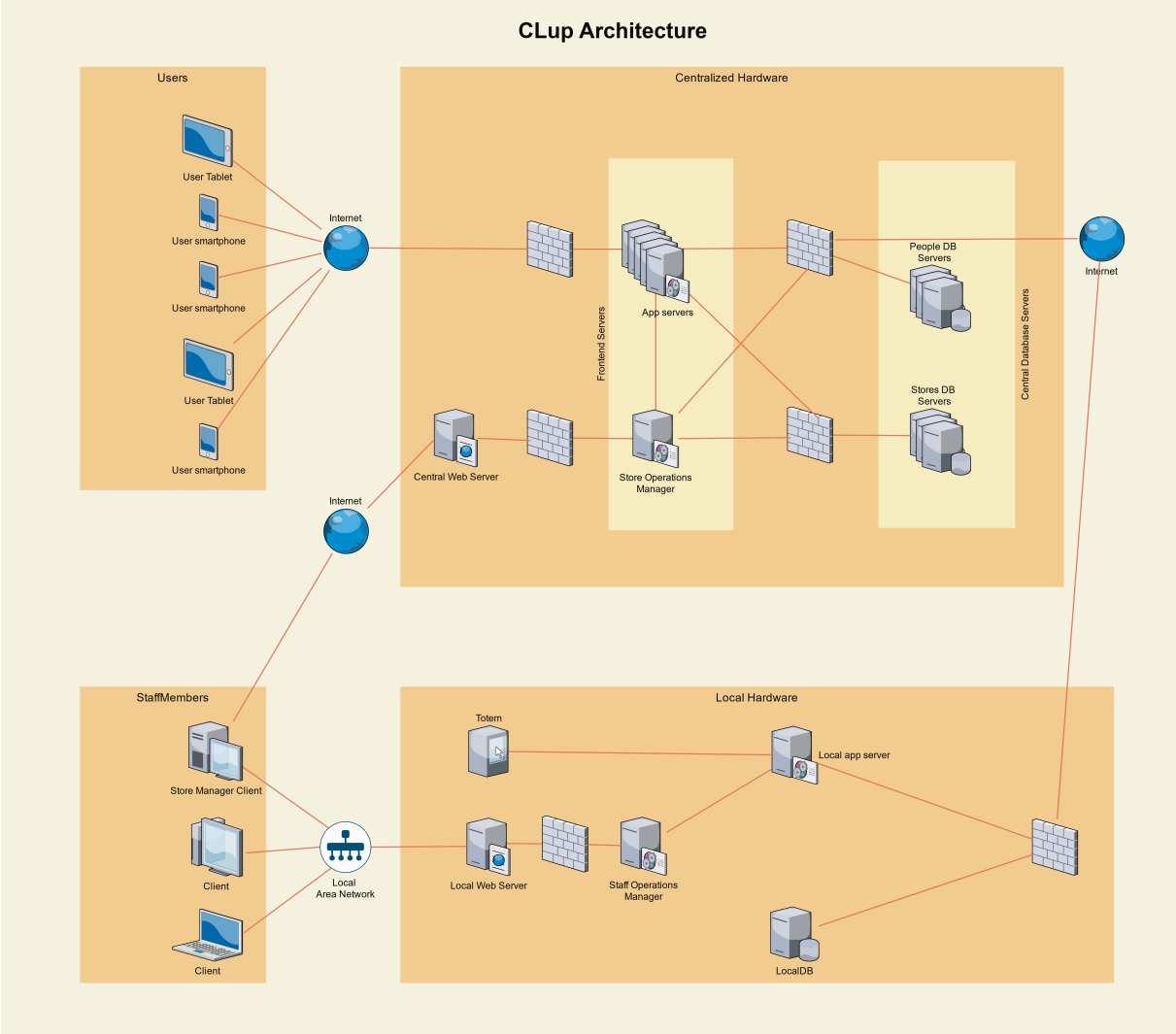


Figure 1: High-level architecture

2.2 Component view

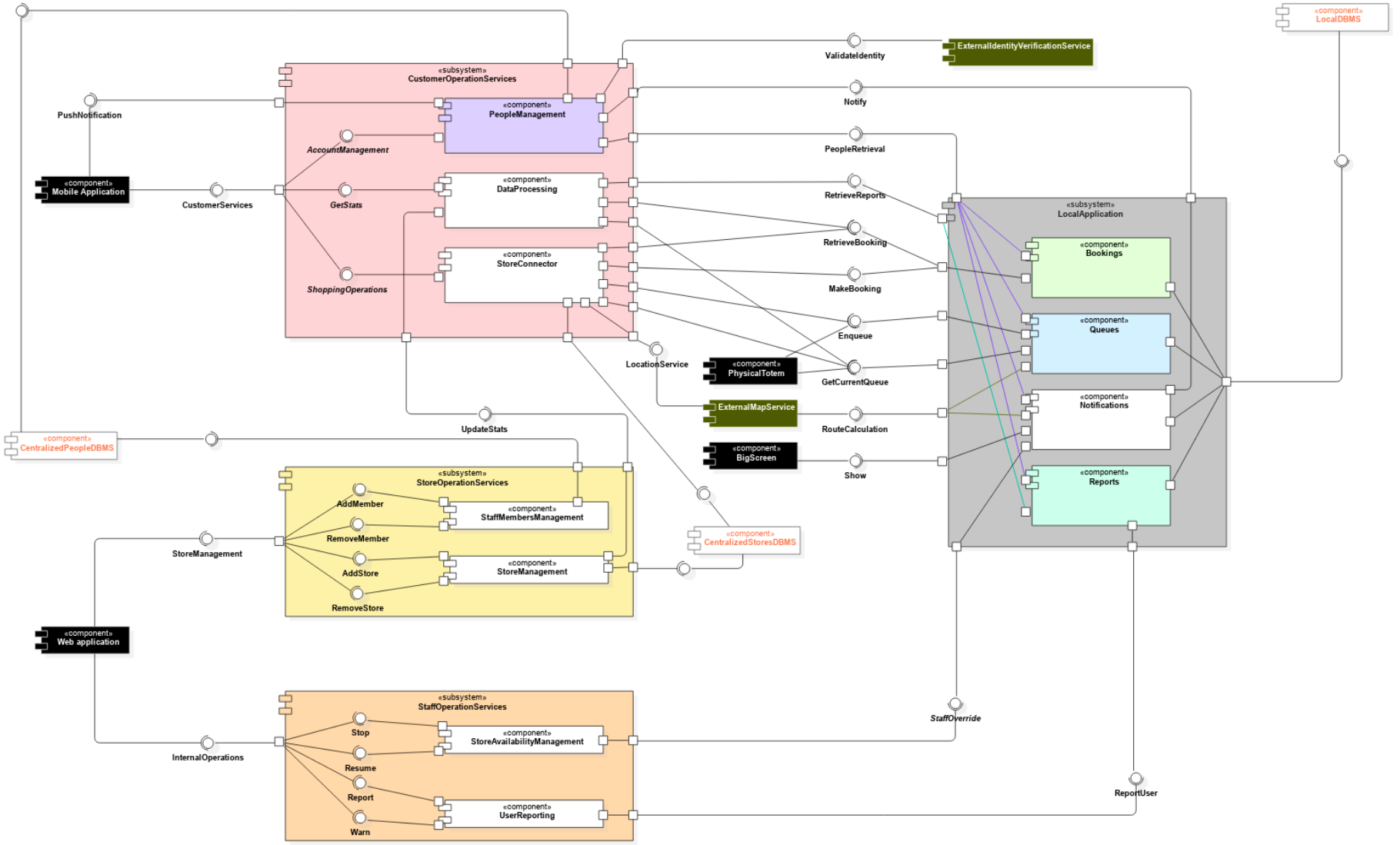


Figure 2: Component Diagram

In Figure 2 is represented the component diagram of the internal structure Application layer, showing how its components and subsystems interact. As previously said, the A layer contains Clup's functional business logic which drives the application's core capabilities, it builds a bridge between the presentation layer and the data layer. What follows is a brief description of every component and subsystem:

- **Staff Operation Services**: This subsystem contains 2 components:
  - **Store Availability Management**: This is the component responsible for both interruption and recovery of availability of a store to accept new virtually queued customers
  - **User Reporting**: This component's function is to take note of customers bad behaviours
- **Store Operation Services**: This subsystem contains 2 components:
  - **Staff Members Management**: A subscribed store can grant access and credentials to CLup's store services through this component
  - **Store Management**: A store can enroll to CLup's network through this component
- **Customer Operation Services**: This subsystem has 2 subcomponents
  - **People Management**: This component is responsible for customers subscription, login t CLup and, thanks to the association device-person, notifications by CLup and Supermarkets
  - **Data Processing**: As the name suggests, the Data Processing component takes care of data about availability of stores, time slots and queues status. It also evaluate time slots, days and store crowdness level and infer better solutions.
  - **Store Connector**:this component acts as an interface towards the distributed part of the application and its components. It allows the reaching of and use of store-hosted components such as Bookings, Queues.
- **Local Application** :
  - **Bookings**: This component makes possible booking an entrance, it takes care of the logic of the booking procedure and retrieves information of a previously made reservation
  - **Queues**: This is the component responsible for the basic function of CLup, it has read-write access to stores queues and allows also to block and unblock them
  - **Notifications**: Here is were all kind of users notifications are generated
  - **Reports**: This is the component that instead generate bad behaviours reports
- **External Map Services**: This is the component that provides the map interface
- **Physical Totem**: This component is a simplified version of the mobile app, it has only what is needed to perform the basic function.

2.3 Deployment view

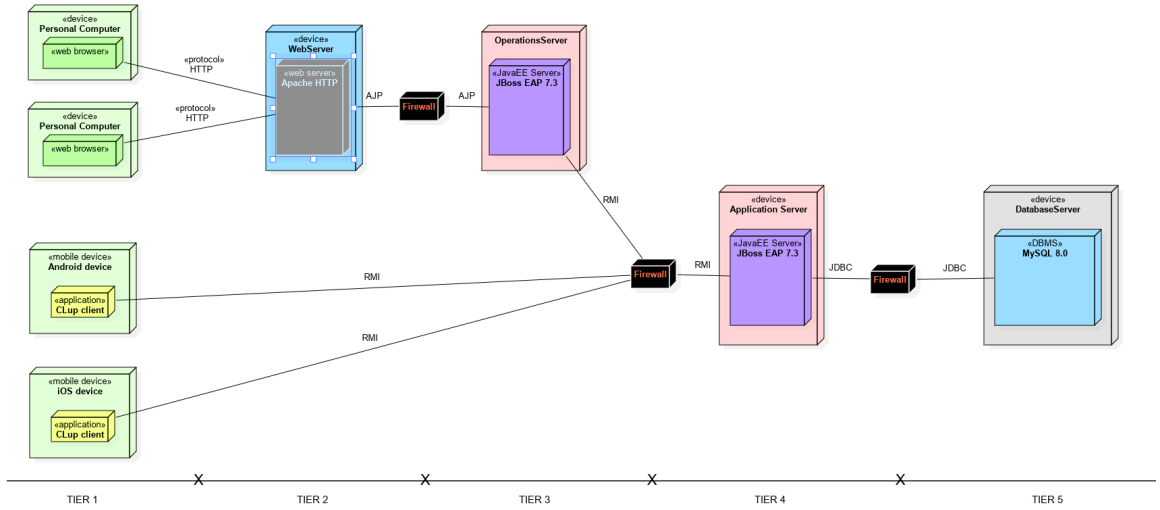


Figure 3: Deployment Diagram

CLup is designed according to a **multi-tier architecture** approach, which is represented by figure 3. Given the heavy distribution adopted for the logical layer of the application, a simplified view is depicted.

- **Tier 1**: the *client tier* includes everything from a smartphone to a desktop computer, mobile devices being the target for end-users while desktop/office PCs for stores staff members.
  - Mobile CLup applications connect to the main application servers via RMI
  - Desktops/office PCs use a standard web application, which offers mainly static content.
- **Tier 2:** the *web tier* offer standard web server functionality. An **Apache** Server takes care of the HTTP requests from the staff's office computers. It then forwards them to application servers by using the **Apache JServ Protocol** (AJP)
- **Tier 3:** the *operations tier* serves as main interface between the outer levels and the inner ones, by accepting incoming requests and redirecting them to the appropriate inner components when necessary. This layer is staff-operations only, and is particularly aimed at reducing main servers' workload by further distributing tasks. A *JBoss EAP server* is run to accomplish this tier's tasks, and it communicates with inner level servers via RMI.
- **Tier 4**: the application tier is the core of CLup functionality. It includes the greatest majority of logical functionality (customer functionalities for the most part) and - architecturally wise - it's distributed across multiple servers: some centralized, some others distributed as one per each physical store. Each of those runs a JBoss EAP server.
- **Tier 5**: the last and innermost tier, the *data tier*, handles data management and retrieval. It's controlled by tiers 3 and 4, and it's invoked by use of the **Java DataBase Connectivity** APIs (JDBC). **MySQL** has been chosen for this Tier.

2.4 Runtime view

The following diagrams (*UML sequence*[1]) aim at giving an overall view about the main components interactions in the real world. Side-interest, particular cases or situations have been left uncovered for clarity and focus reasons. For the same reasons, some *notation abuses* may have been made regarding the main component in each diagram and its lifeline/messages back to the requester.

> You will find reference to undefined methods and interfaces inside the diagrams of this sections. Those diagrams are to be intended as a guideline for the Developers, to help them better visualize the core design choices made. They do **not** represent the final implementation result nor intend to irrevocably specify its characteristics.

Store Operations We can see in these diagrams the whole CLup architecture at work. A qualified member of the store's staff can, after strong authentication (not depicted), invoke the stop and resume of the store's operation. The first meaning no more customers allowed inside the store, no more physical tickets being issued at the totems, and eventually notifications being dispatched to the affected customers[2]; the second implying correct and complete resuming of the store functionality. See figures 10 and 11.

Registration and sign-up Here we can see the different components involved in the signing up process of customer as opposed to stores' staff members, or store themselves. Different components at different locations are used in each case. See figures 6, 7 and 8.

Tickets and Booking The core of CLup's functionality. We can see how these features involve both centralised and remote (localized) servers to work. See figures 4 and 5.
See also 9 regarding offline, physical tickets issuing.

Reporting and Warning Finally, the reporting & warning functionalities of CLup. They allow CLup (automatically) and stores' staff to inform users about relevant events, or even reporting them for misbehaviours. See figures 12, 13.

[1] As per [1]
[2] Refer to 1.3.1 for details

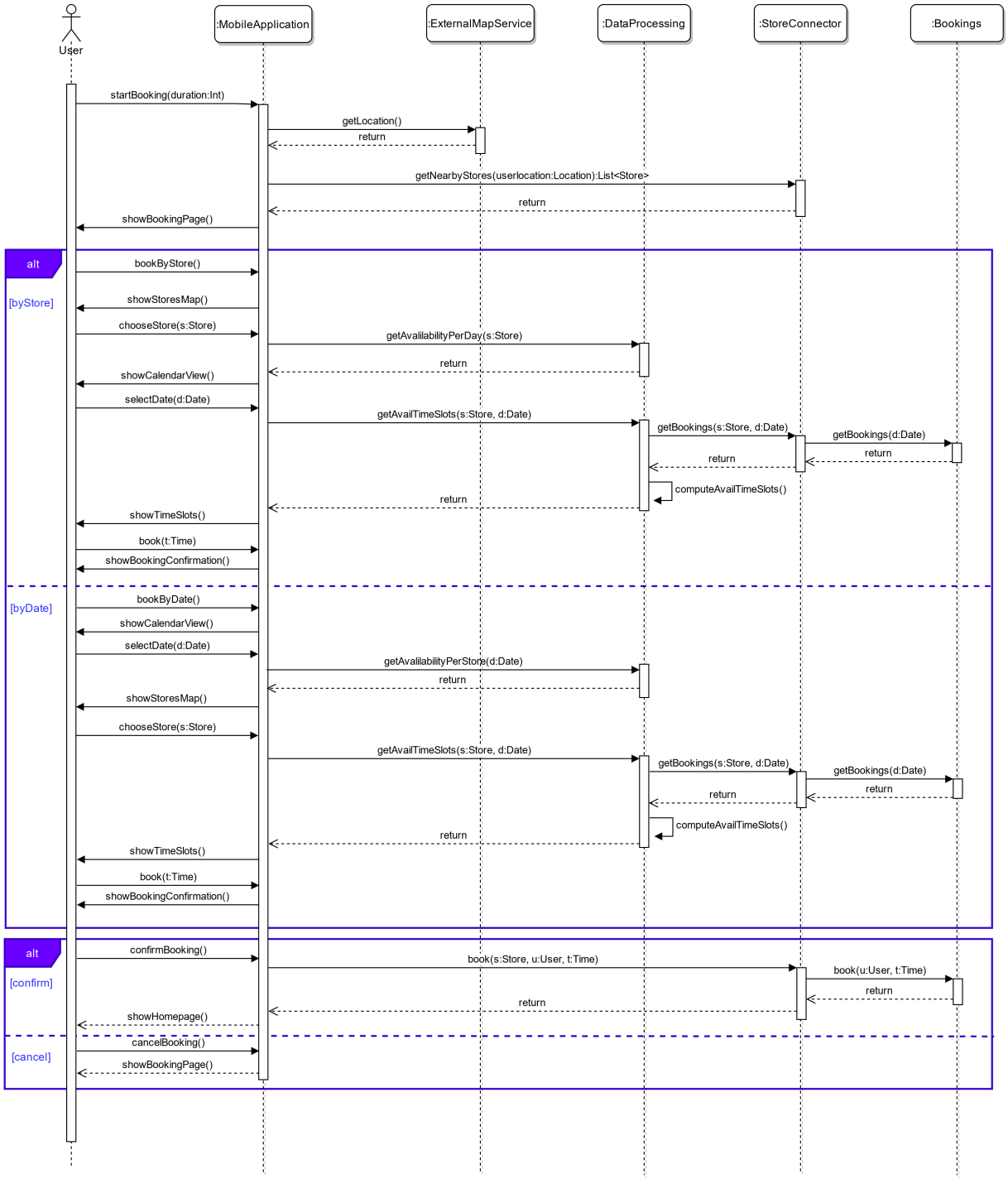


Figure 4: Sequence Diagram: Customer Booking procedure

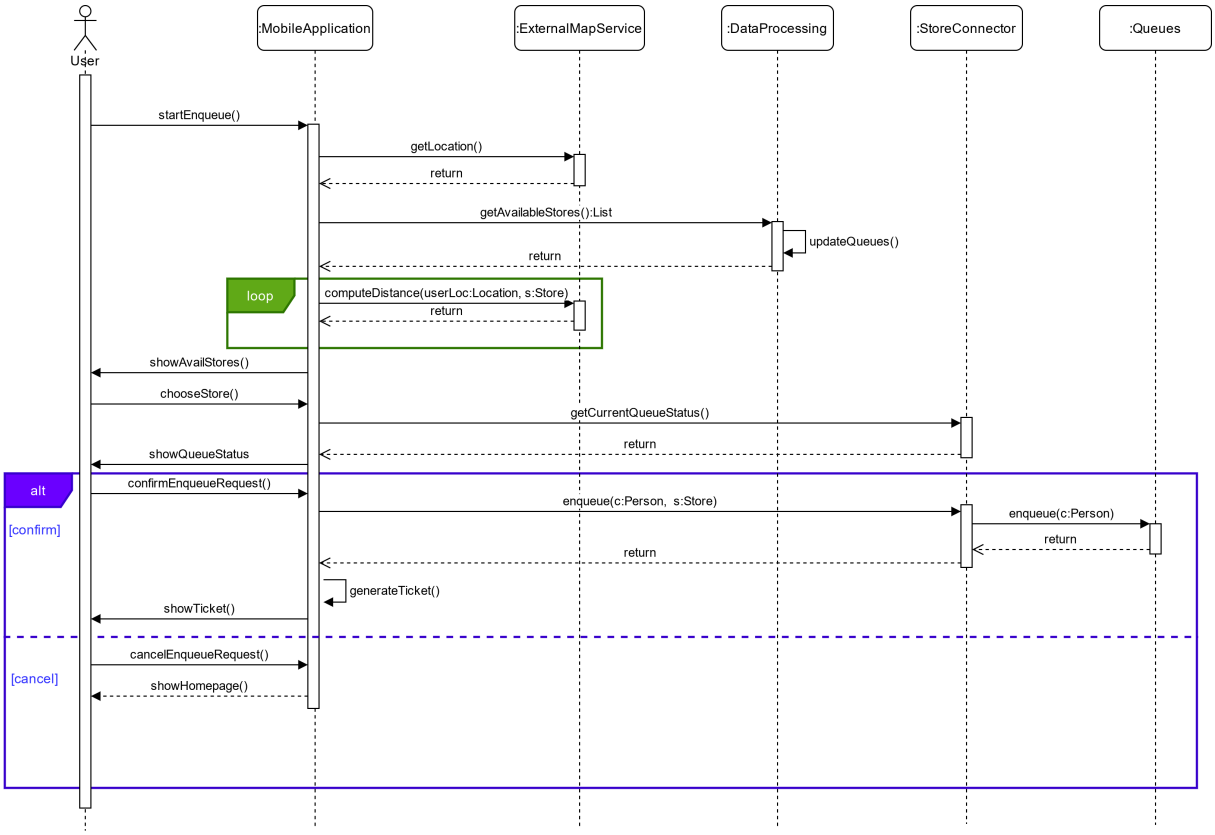


Figure 5: Sequence Diagram: Customer Enqueue (virtual ticket) procedure

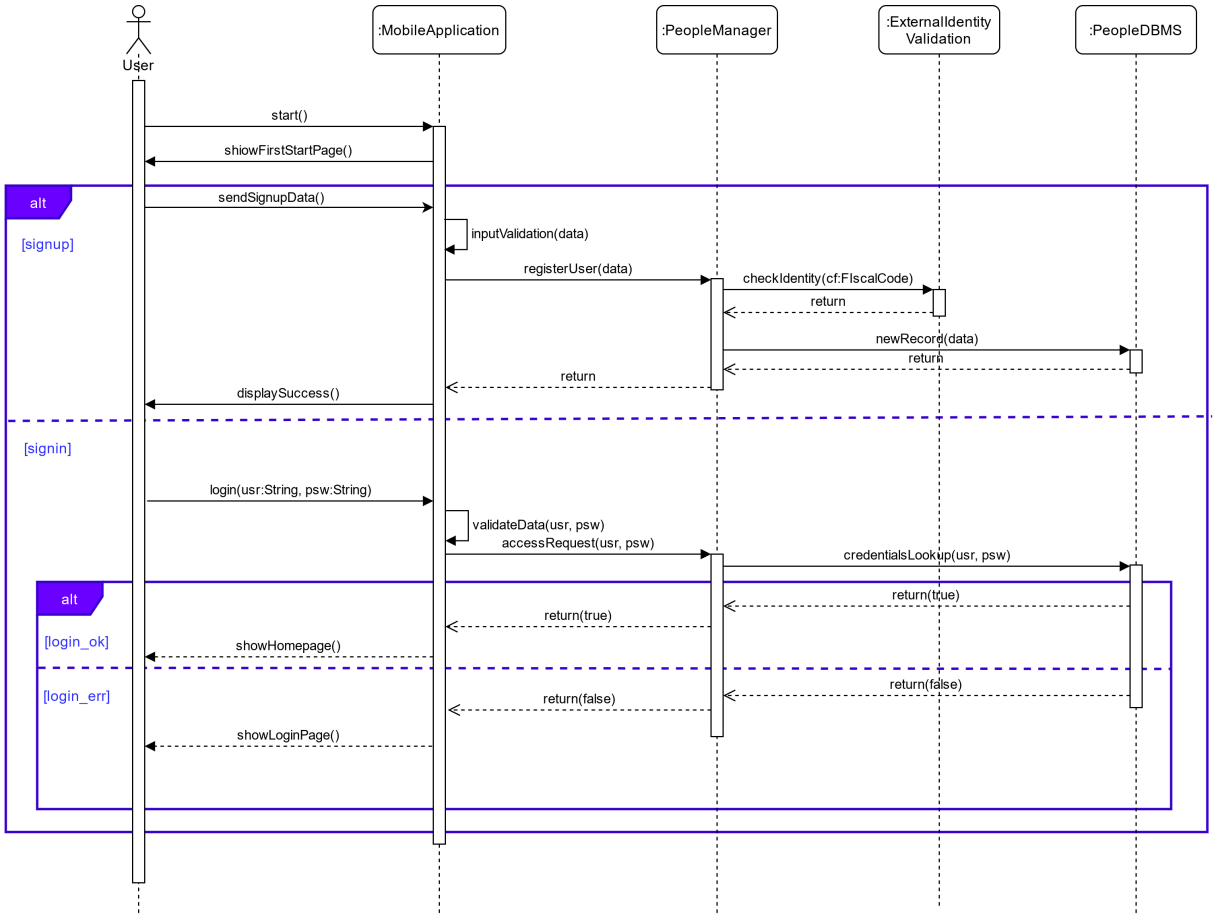


Figure 6: Sequence Diagram: Customer Sign-up procedure

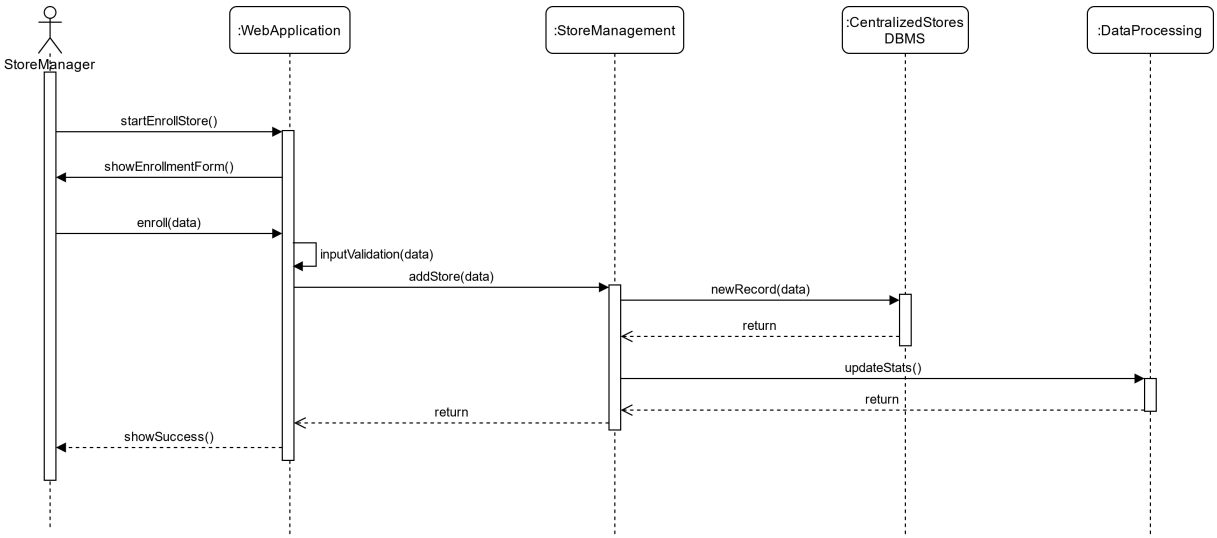


Figure 7: Sequence Diagram: Store Enrollment procedure

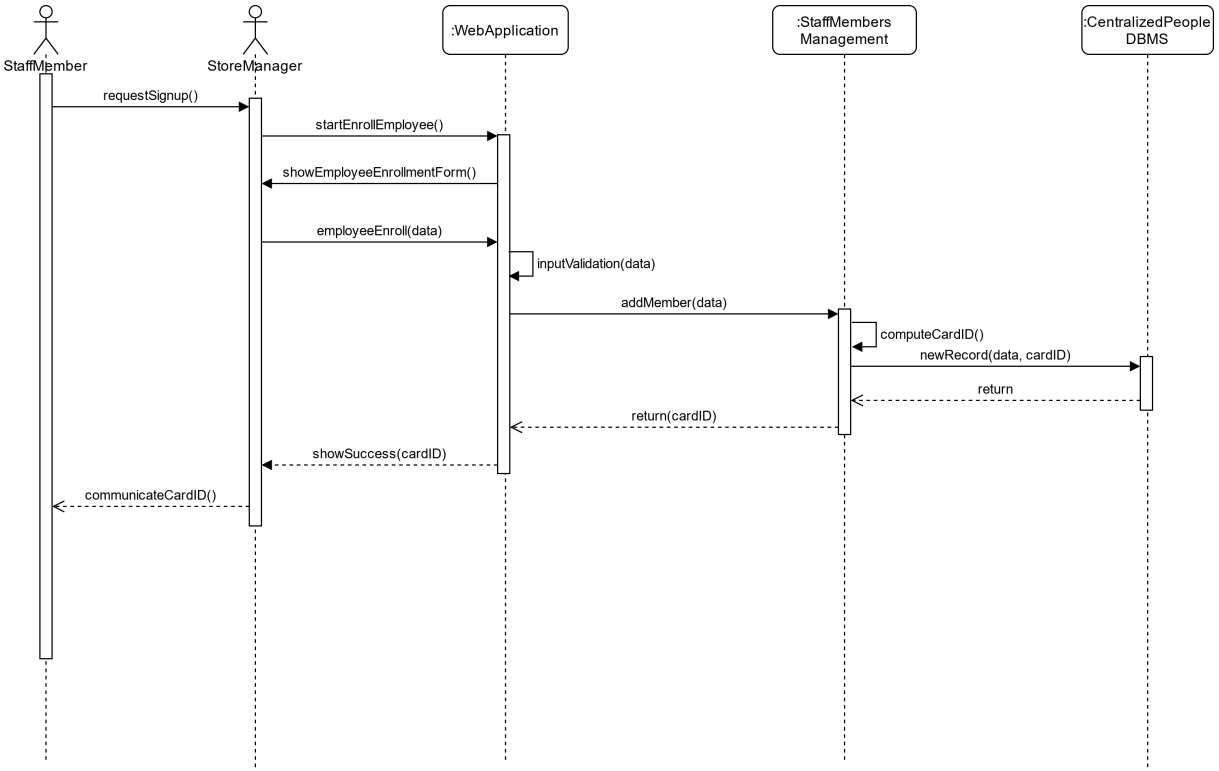


Figure 8: Sequence Diagram: Staff Enrollment procedure

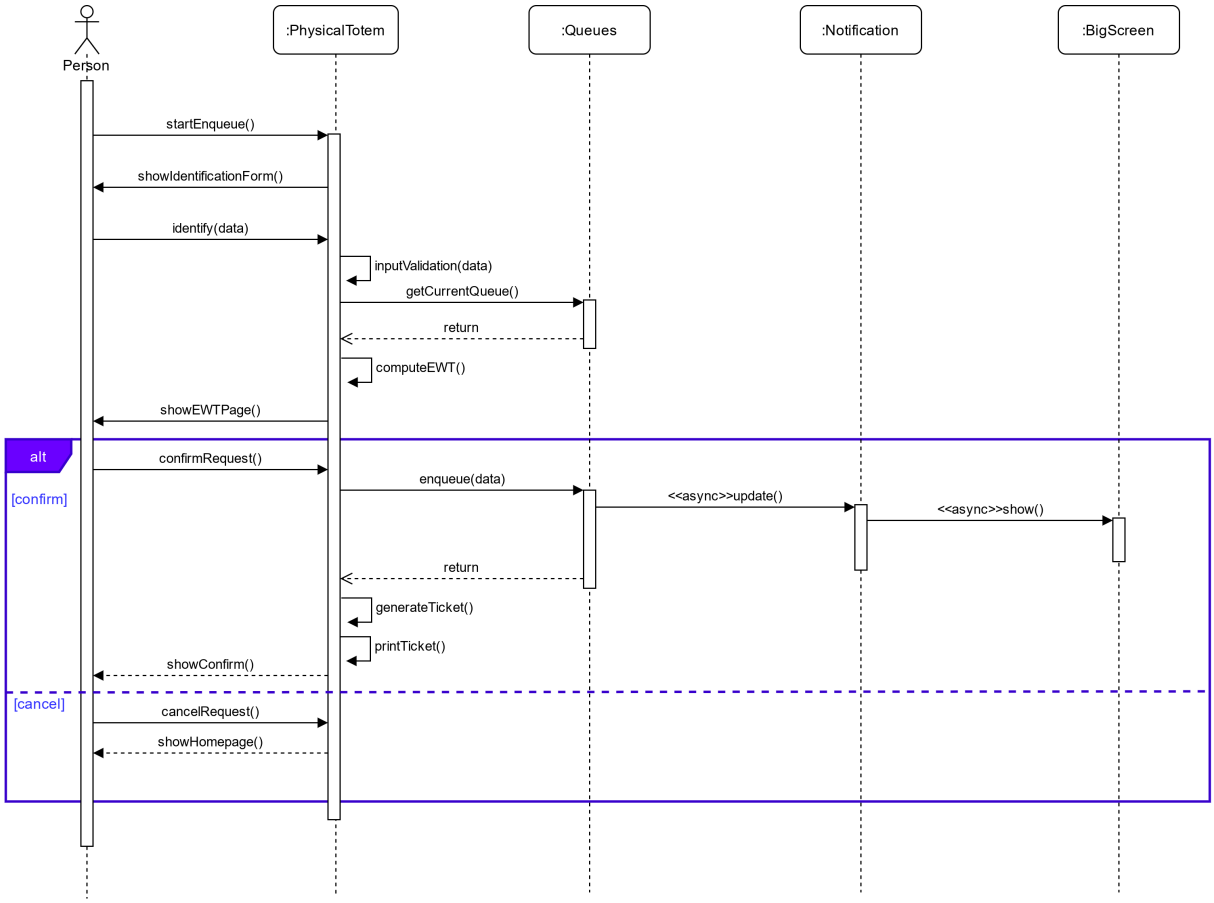


Figure 9: Sequence Diagram: Totem Queueing (physical ticket)

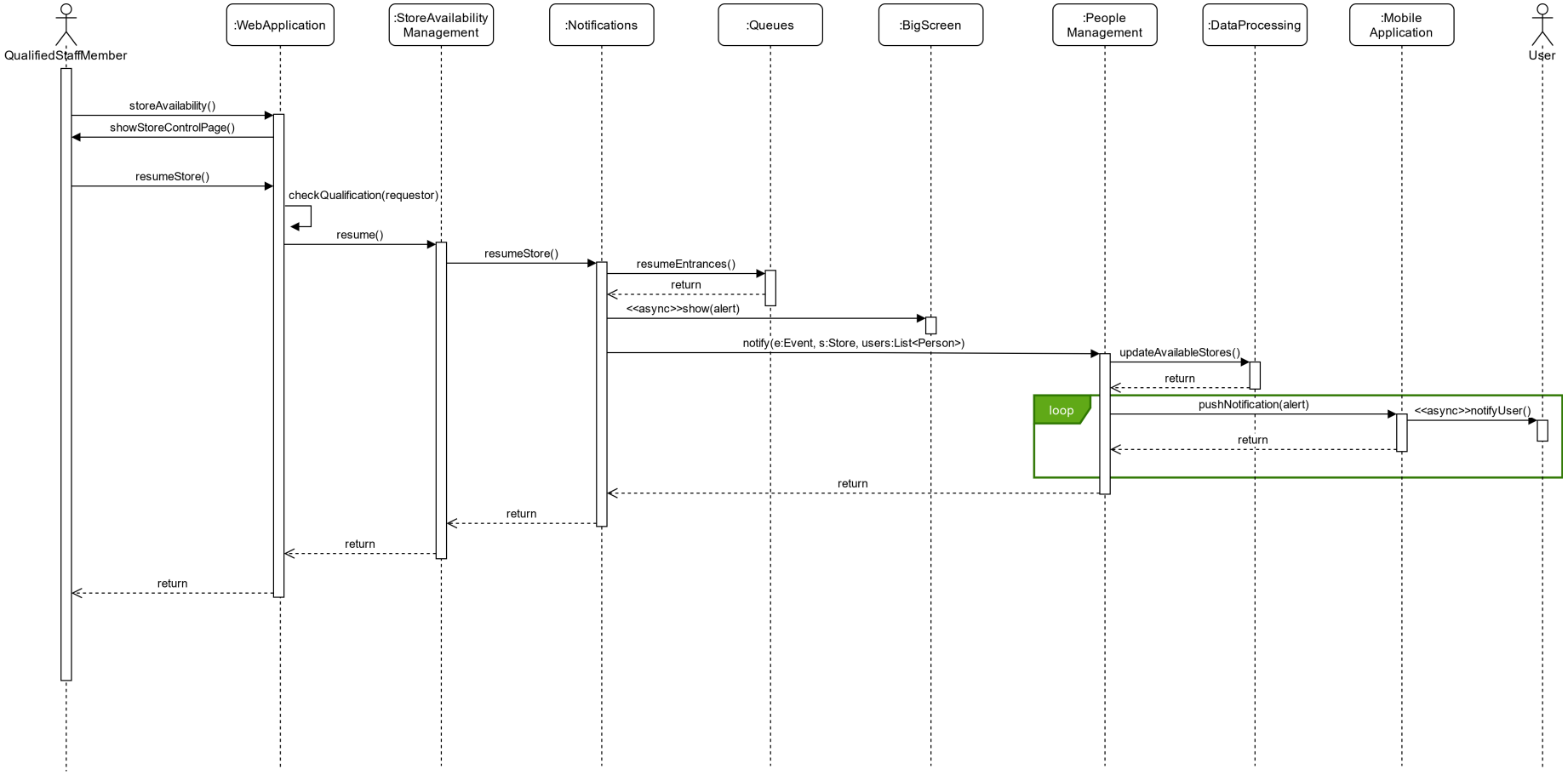


Figure 10: Sequence Diagram: Resuming the Store

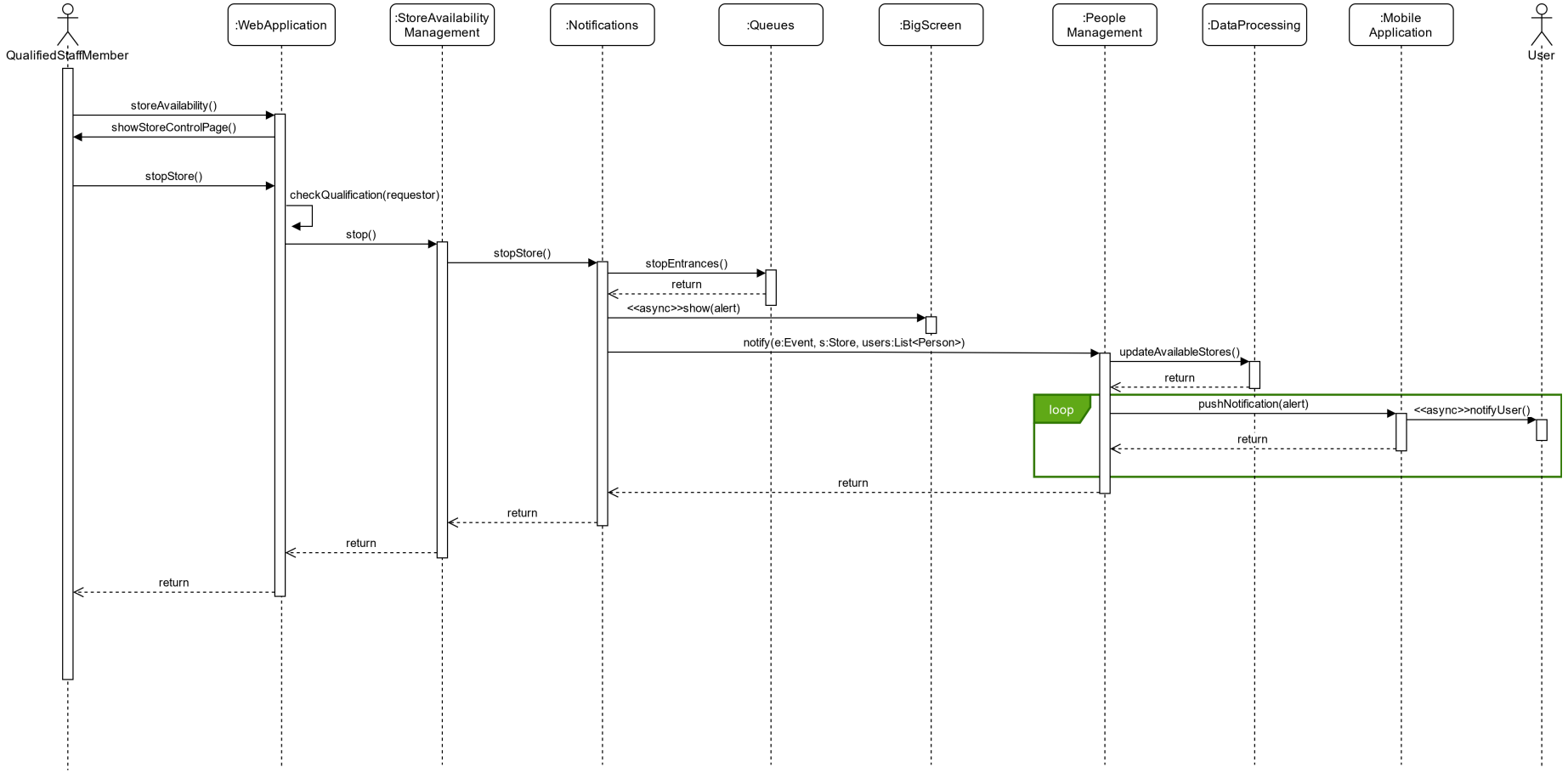


Figure 11: Sequence Diagram: Stopping the Store

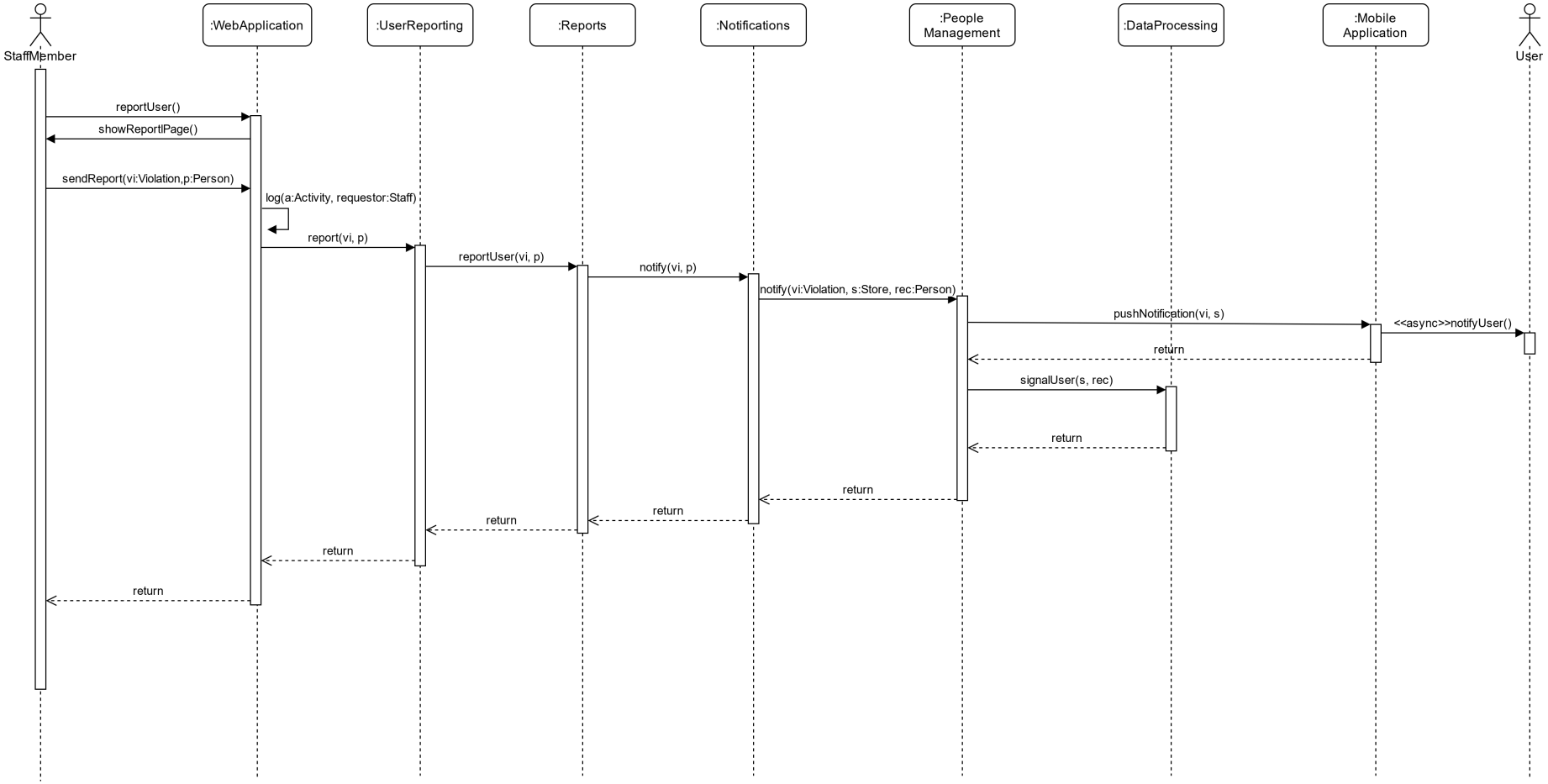


Figure 12: Sequence Diagram: Reporting a User

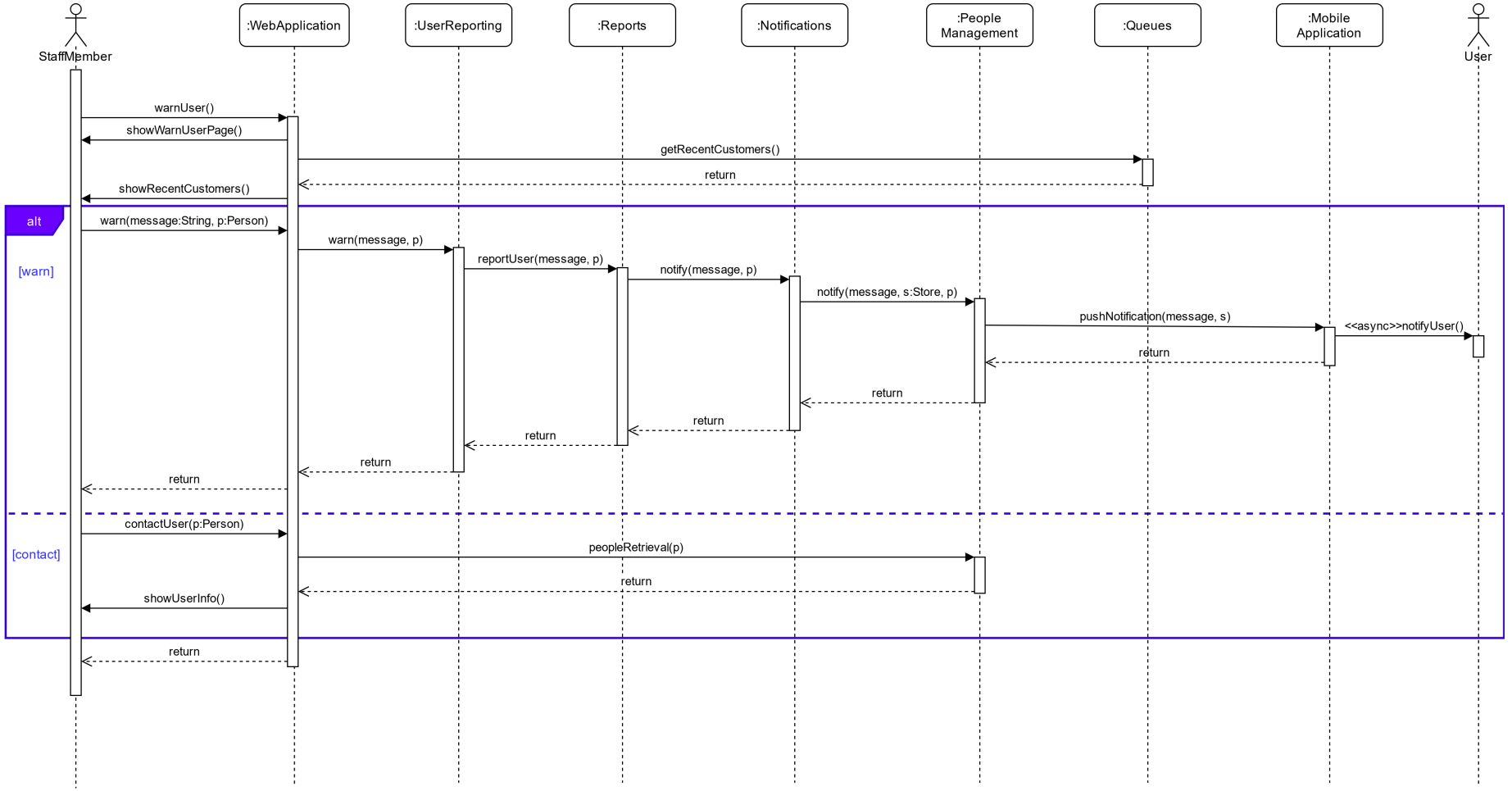


Figure 13: Sequence Diagram: Warning a User

2.5 Component interfaces

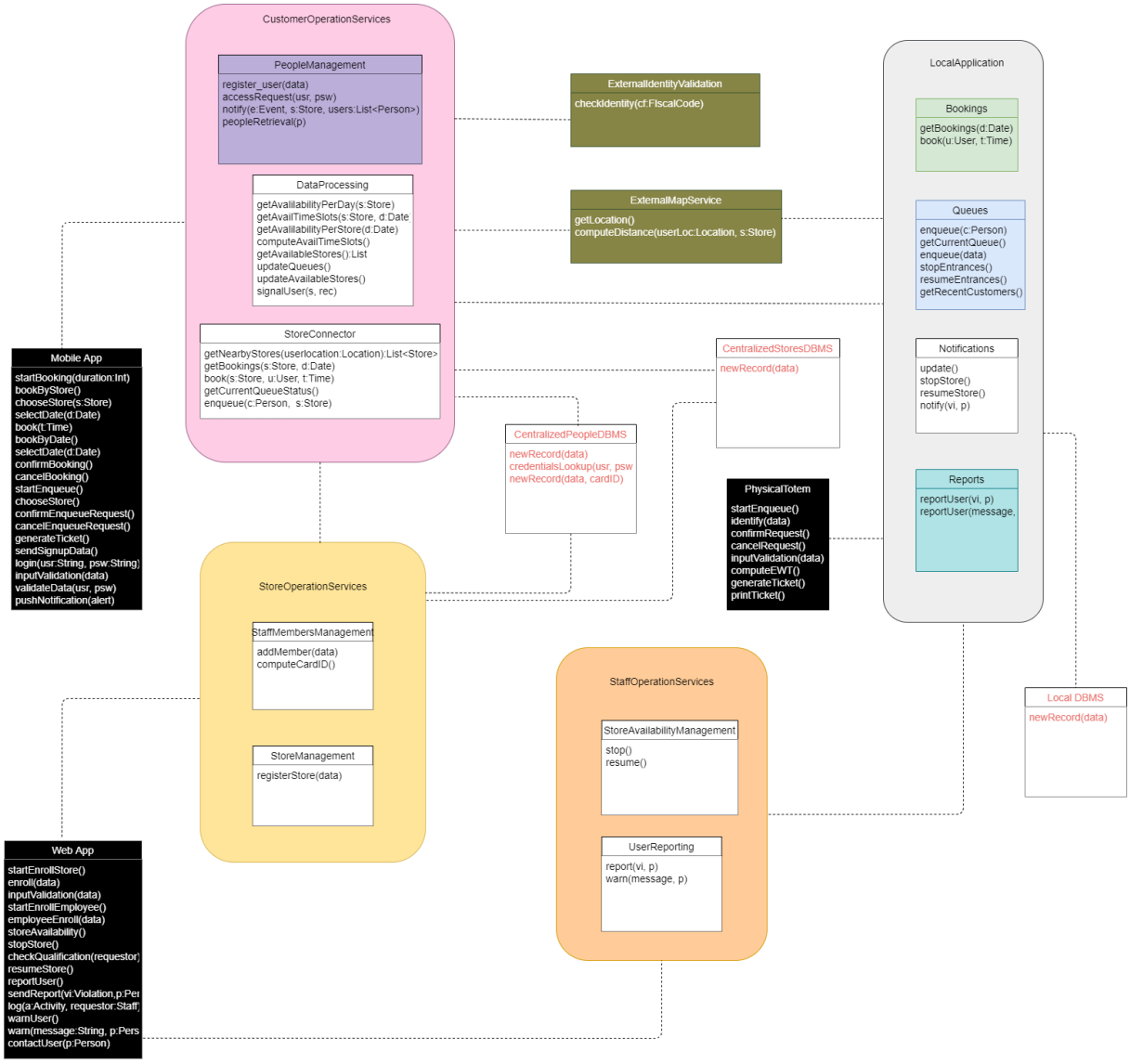


Figure 14: Component Interface Diagram

2.6 Selected architectural styles and patterns

As previously mentioned, a multi-tier client-server architecture has been used to develop CLup in order to promote a major decoupling of the system and to increase the reusability, scalability and flexibility. CLup servers wait for requests to arrive from clients and then respond to them, server provides clients a standardized transparent interface so that they need not be aware of the specifics of the system that is providing the service.
For what concern design patterns, the best suited for this application is MVC: it is a very flexible pattern that massively increase decoupling and simultaneous development through the division of the related program logic into three interconnected elements, Model, View and Controller. Briefly:

- Model: The central component of the pattern, Model is the application's dynamic data structure, independent of the user interface, it directly manages the data, logic and rules

of the application.

- View: This component presents information to the user
- Controller: This component is responsible for interpreting the input and manipulating the model based on that input.

A possible way to implement MVC is a combination of Observer, Composite and Strategy patterns:

- The model implements the Observer pattern to keep interested objects, typically in the View, updated when state changes occur. In this way it is possible to use different views with the same model or even multiple views at once
- GUIs consists of a nested set of a windows, panels and so on. Using Composite, each of those components is a composite or a leaf, when the controller tells the view to update, it only has to tell the top view component and Composite takes care of the rest
- The view and controller implement the classic Strategy Pattern, the view is an object that is configured with a strategy. The controller provides the strategy and the view is concerned only with the visual aspects of the application, it delegates to the controller for any decisions about the interface behavior. Using the Strategy pattern also keeps the view decoupled from the model because it is the controller that is responsible for interacting with the model to carry out user requests

2.7 Other design decisions

As all the diagrams of the previous sections outline, CLup adopts some particular design decisions worth being discussed.
One of those decision is the way in which the application is distributed. According to the aims of this section, CLup's functionality may be summarized as follows:

- Presentation & User interaction
- Flow management & local business functionality.

Let's discuss them in detail.

User interaction The **Presentation & user interaction** set of functionalities (from now on: PUIF) covers anything from the registration of new components/people (customers, staff members, stores etc.) to the making of a booking, the virtual queueing process and so on. Every single interaction taking place between CLup and its users has to be included here.
We have chosen for these functionalities a **cost-saving biased trade-off**. We have decided to sacrifice robustness and - eventually - availability of such functionalities in favour of a more affordable and easy-to-implement architecture: one of CLup's main objectives is to become ubiquitous, so to make increasing the reach of its very primary goal (safer grocery shopping) possible.
In particular, the following aspects have been taken into consideration:

- PUIF usually require a stable, huge **bandwidth** which would have been difficult to get access to in many locations, in case a distributed approach were chosen instead.
- PUIF, still being one fundamental functionality, is less vital wrt. FMLBF[3] (see next paragraph for more on this)

[3] Refer to 1.3.2

- A distributed approach for PUIF would require massive infrastructure updates for the majority of stores willing to join the CLup network. This would in turn reduce the number of potential target stores, thus reducing CLup's main goal reach.
- In the event of connection loss between central and remote (localized) servers, new bookings and tickets would not be issued for the affected stores. Still, **already existing ones would not be affected.**
  - Similarly, in the event of main servers outage, CLup clients would not be able handle the generation of new booking and tickets, but still, they would be able to serve already generated ones - offline.

Local business Completely opposite considerations have been made for **FMLBF**. These functionalities are indeed considered of strong, vital importance and have then been qualified eligible for a distributed architecture requirement. A **robustness/availability biased trade-off** was in fact chosen, also as a consequence of the following aspects:

- FMLBF mainly involve queue management, physical access control to the store, store control, physical ticketing.
- Aforementioned functionalities do not require internet access nor communication with the main servers, provided that the necessary local data structures are put in place.
- As a consequence, the communication between main and remote (localized) servers assumes a new less-vital role, in which it allows part of CLup's functionality (booking, virtual queueing...) but **does not prevent others in the event of failure**.
  - Customers would still be able to **access the store with a valid ticket**, even in case of broadband connection failure
  - Customers would still be able to **exit the store scanning** that same ticket
  - Customers would still be able to get **physical tickets** outside the store, even during a broadband connection failure event
    - This in turns indicates (even though with limitations induced by the number of physical totems present) guarantee of access to the store to a **non-zero number of people** even during such events.
  - People flows would still be guaranteed to be handled correctly and consistently **during and after** the event of broadband connection failure.
  - Within normal operation conditions, CLup's business logic would require either:
    - **Less costs**, in terms of storage (caches would be needed for a centralized approach), or..
    - ..**Less latency** in the processing of customer related events (access, exit etc.).

Databases Given all the considerations above, databases have been split into:

- **Central** databases. Those holding information about general facts, such as the **Stores** Database, or about entities that are not store-specific: like persons (**People** Database).
- **Localized** databases. Those holding information about store-specific facts or entities, such as **Reports, Notifications, Bookings and Tickets** databases.

Each store may **optionally** hold a **cache** for most relevant central databases entries.

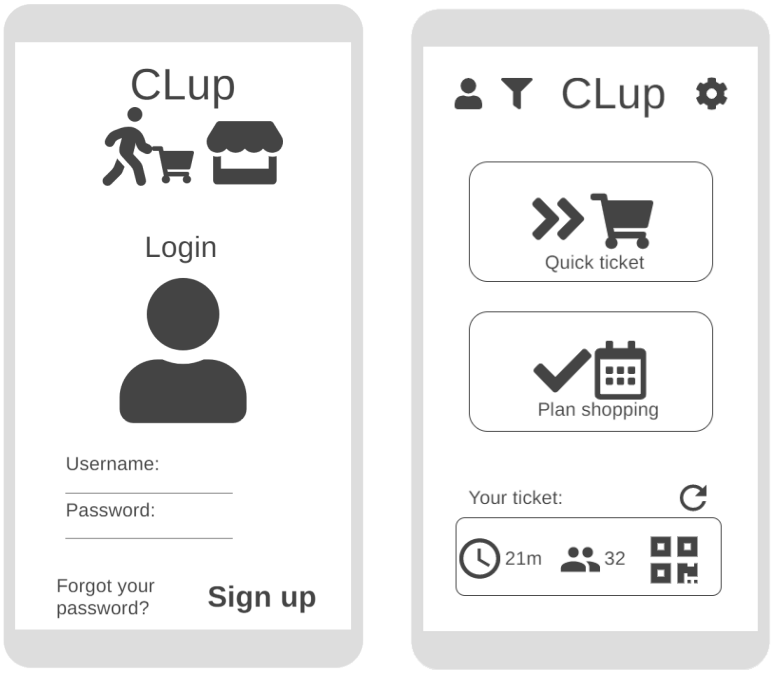
3 User Interface Design

3.1 Overview

CLup is an application aimed at decreasing the probability of contracting COVID-19 (diseases in general) when going shopping to a supermarket. There are two fundamental components: the main one targets customers of supermarkets while the second one is available for store managers.

3.1.1 User Interfaces

End user functionalities Regarding the first target - customers - they will be required to register to the service the first time they use it by inserting their full name, email address, ID card, phone number and a password. The customer will also be requested to specify his physical address, or to enable the GPS, in order to allow CLup to find stores nearby; this last information can be changed anytime the user needs. If the customer is not willing to register or share his address, the service will not be available.



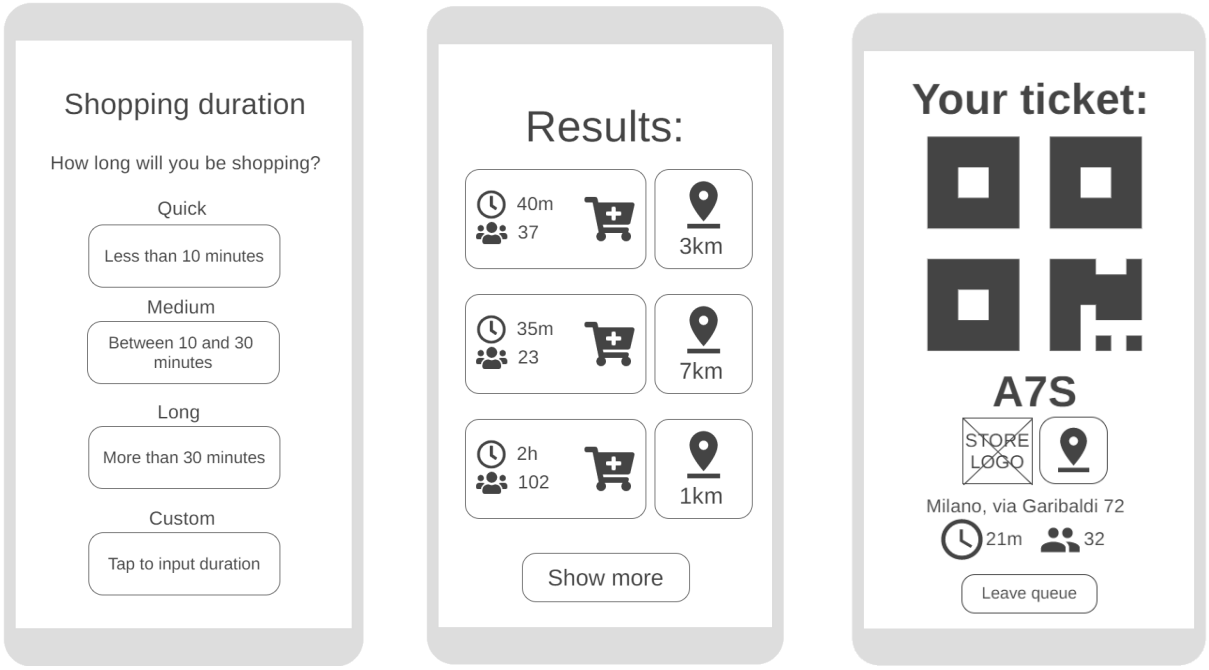
(a) Login screen (b) Main screen

Figure 15: App startup

Once the setup is done, customers will be able to access the homepage of the application, where they can tap on the “Quick ticket” button that will allow them to see a list of stores inside a specified range from their current location: for each store a distance in kilometers from the user position will be outlined, as well as the number of people inside the store and its maximum capacity; whenever a store is full, the current number of people in line and an *EWT* are displayed.
It is also possible to visualize stores on a map and, by tapping on one of them, to see the same information displayed in the list. Now, if the user chooses to reserve a spot in the line, the application will open a confirm dialog specifying EWT and the expiration of the ticket. If the

user refuses nothing happens, if he accepts instead CLup will process the request, show his ticket and the real time evolution of the line; the ticket is also visible from the home screen.

The tickets consist of a QR code and an easy to remember alphanumeric code alternative



(a) Shopping time (b) Store list (c) Your ticket

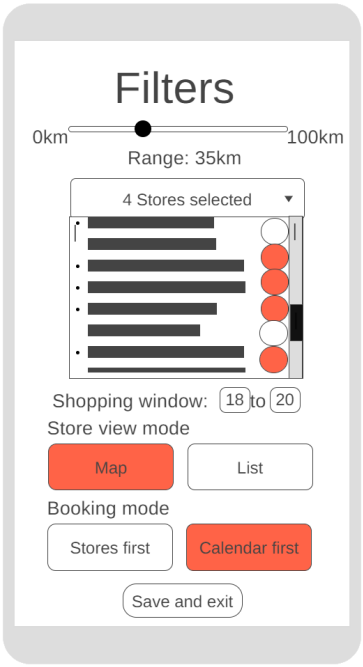
Figure 16: Quick ticket procedure

to enter the store. At the stores entrances there will also be monitors that show the numbers allowed to enter and, eventually, delays.

The distance range in which CLup will look for supermarkets is specified by the user through the filter button in the homepage, this button will in fact open the filter screen in which, among other parameters, a sliding bar controls the distance and a drop-down list allows the user to filter the chains of supermarkets.

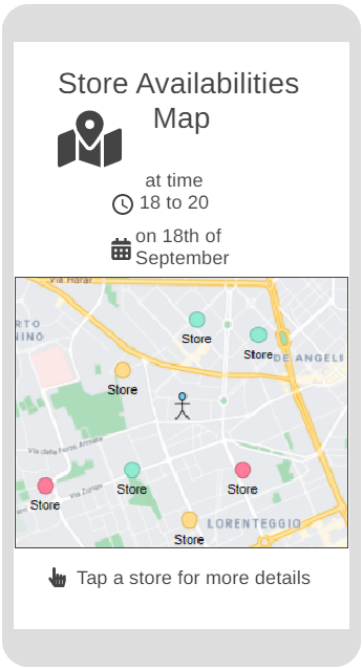
Another important feature is the possibility to book an entrance later in the day or in another day. The user can specify from the filters whether he prefers to choose the day or the store first and he can set the time range in which he wants to book. There is a dedicated button in the app's main screen that redirects the user to either the list/map of supermarkets or the calendar, and once the user chooses he will be respectively shown the calendar or the list/map, this time with colours to indicate the average crowdedness of stores/days given the set time range. When the user chooses the day and supermarket combination, a timetable spanning the chosen time range is shown, divided in 15 minutes time slots each one marked to show its availability. The user will be able to check his reservation on the home page exactly like previously for the quick ticket, and near the entrance time he will be provided an actual ticket.

The access at the supermarket is restricted by turnstiles with QR code readers, a staff member is expected to verify that nobody waits his turn in front of the entrance, jumps the turnstile or does anything irresponsible. During the shopping the user will still be able to view his ticket in order to use it to unlock the turnstile upon exiting the store. Customers who, for any reason, do

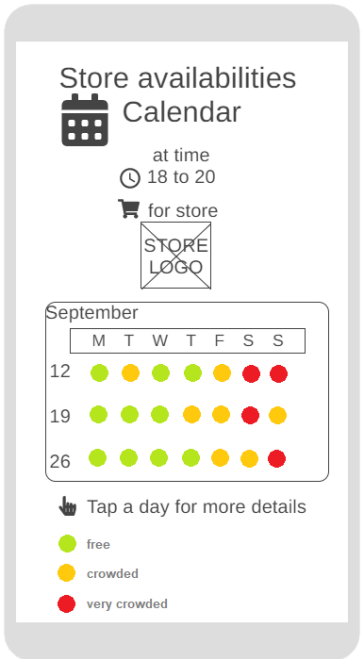


(a) Filters screen

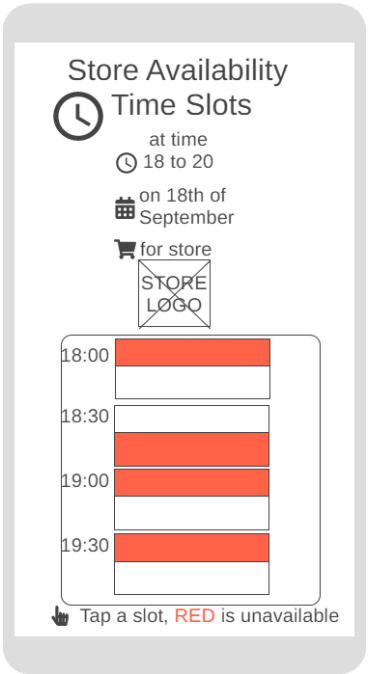
Figure 17: Filters



(a) Map screen



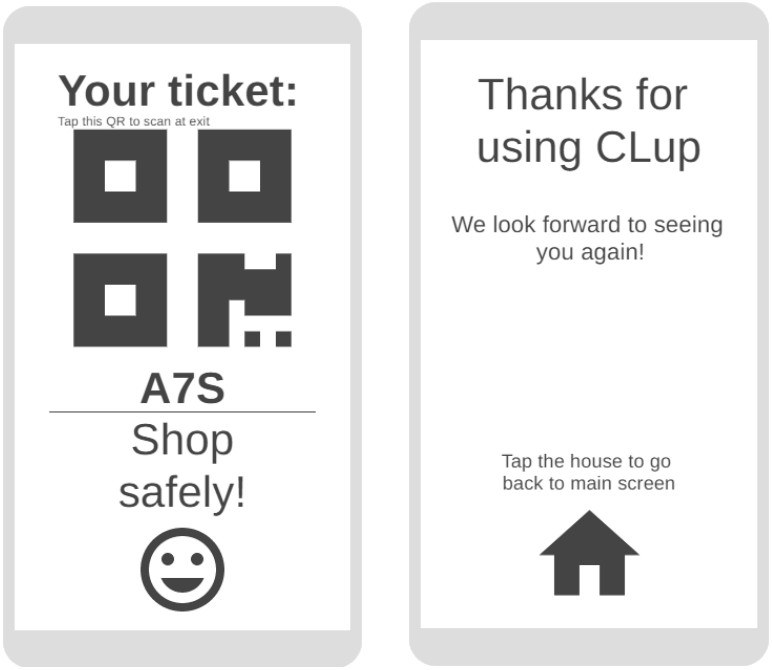
(b) Calendar screen



(c) Time slot choice

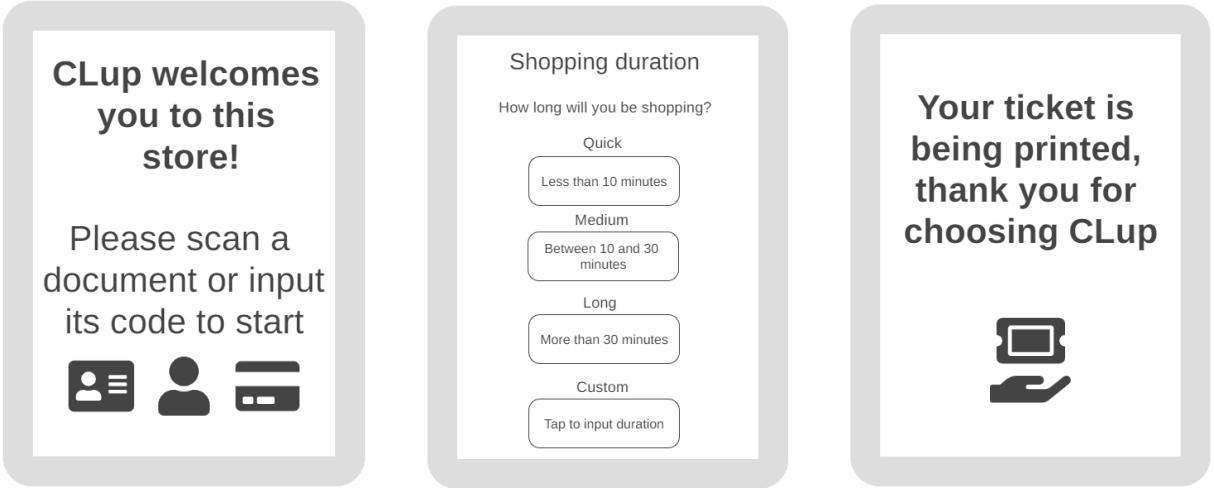
Figure 18: Booking procedure

not use the app will still be able to queue in CLup supermarkets by obtaining a printed ticket from a physical totem located near such stores; the functioning of the application will be similar to the “Quick ticket” app function with the difference that the user can only obtain a ticket for the current totem’s store.



(a) Shopping screen (b) Exit screen

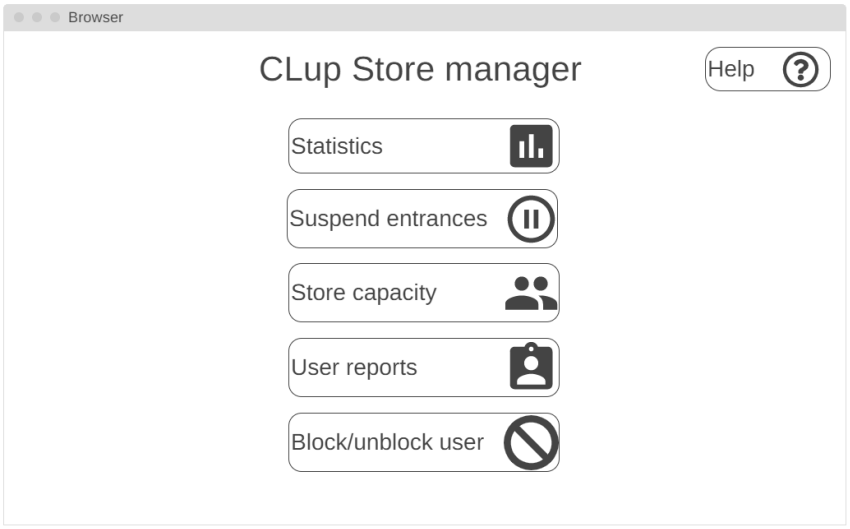
Figure 19: Shop and exit



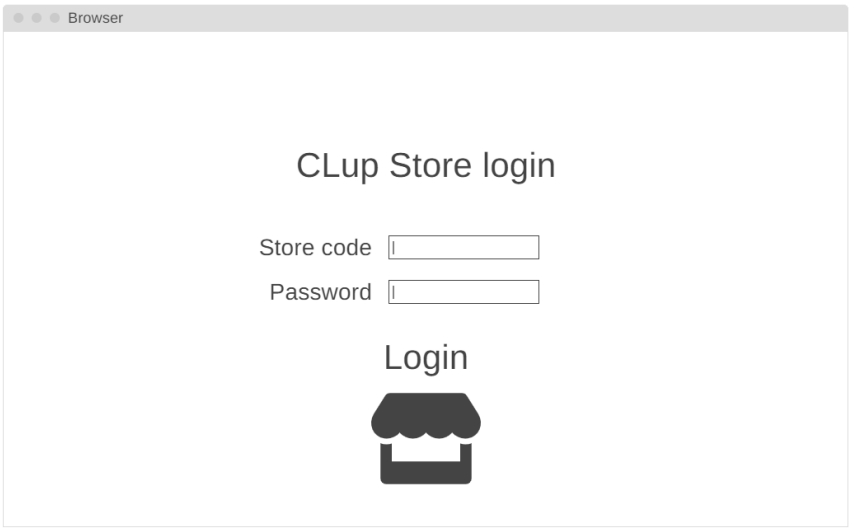
(a) Totem welcome screen (b) Totem shopping time input (c) Totem print screen

Figure 20: Totem ticket procedure

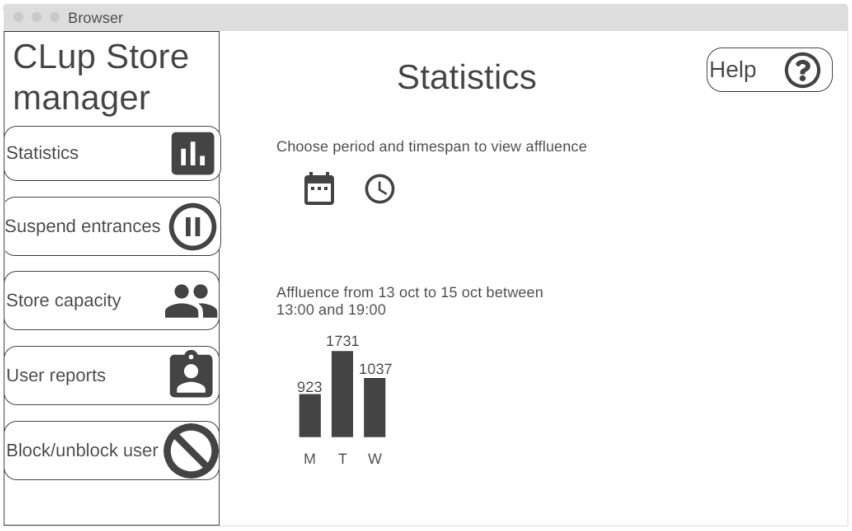
Internal use functionalities Another component of CLup targets store managers: when the store decides to join the CLup network, ad-hoc credentials to access the web app will be given. Special staff-only functions will then be achieved by the use of a web application, accessible via internal-use terminals.



(a) Manager login



(b) Manager main screen



(c) Manager statistics screen

Figure 21: Store manager web app

4 Requirements Traceability

G1	**Anybody is guaranteed possibility to make shopping at any supermarket in at most 1 hour** Requirements: R1, R6, R7, R10, R15 Components: • People Management • Data Processing • Store Connector • Queues • Notifications • Physical Totem

Table 1: G1 Mapping

G2	**Users can get to know the least crowded time slots** Requirements: R2, R11, R12, R18, R21 Components: • Customers • Store Connector • Notification • Data Processing

Table 2: G2 Mapping

G3	**Fair users can make a reservation to enter in a supermarket** Requirements: R9, R13, R15, R21, R23 Components: • Customers • Store Connector • Data Processing • Bookings • User Reporting

Table 3: G3 Mapping

G4	**Stores can easily monitor fluxes** Requirements: R3, R11, R22 Components: • Bookings • Queues • Store Management

Table 4: G4 Mapping

G5	**Only authorized users can access** Requirements: R4, R6, R13, R15, R19 Components: • People Management • Physical Totem • Store Connector • Queues • User Reporting

Table 5: G5 Mapping

G6	**Crowds are dramatically reduced outside supermarket stores** Requirements: R1, R2, R5, R6 Components: • Store Connector • Queues • Bookings • Notifications • Store Management

Table 6: G6 Mapping

G7	**CLup should not decrease customer affluence beyond 40% w.r.t. to normal** Requirements: R1, R5, R6, R7, R9, R10, R20, R21, R23 Components: • Data Processing • Notifications • Queues • Bookings • Store Management

Table 7: G7 Mapping

G8	**Same shopping capabilities guaranteed to offline users** Requirements: R1, R7, R10 Components: • Physical Totem • Data Processing • Queues

Table 8: G8 Mapping

G9	**Find the best (less crowded, soonest available) alternative among local supermarket stores** Requirements: R3, R11, R12, R20 Components: • Store Connector • Notifications • Data Processing

Table 9: G9 Mapping

G10	**Supermarkets do not overcrowd**
	Requirements: R2, R3, R4, R11, R12, R20
	Components: • People Management • Notifications • Data Processing • Store Availability Management

Table 10: G10 Mapping

R1	Every user can generate a quick ticket for any store
R2	Whenever user initiates a booking procedure, CLup must be able to compute a suggested least crowded time slot based on historical data
R3	CLup must elaborate and upload data about current global customer affluence to the store during use
R4	CLup must admit only valid QR codes for entrance
R5	CLup must allow users to know current queue status
R6	CLup must update user on tickets' validity change
R7	CLup must inform offline users about new tickets (un)availability
R8	CLup must allow users to indicate which product category they are going to purchase while booking
R9	CLup must suggest alternative stores when the combination of selected store|time gives no results
R10	CLup must reserve a non null number of paper tickets at any time for offline customers use
R11	CLup must gather all stores' data about entrance fluxes
R12	CLup is able to cross affluence data of any supermarket
R13	CLup keeps track of people who book an entrance and don't come
R14	CLup allows store managers to stop quick tickets availability
R15	CLup is able to generate QR codes
R16	CLup is able to authenticate users
R17	CLup is able to store users' data
R18	CLup is able to process users' data
R19	CLup makes quick ticket invalid after 15 minutes delay
R20	CLup can use stores' data to sort every store by crowdedness
R21	Users can see available day/time slots of a supermarket through CLup
R22	CLup shows to store managers flux data about their supermarket
R23	CLup must be able to process reservations

Table 11: Requirements list

5 Implementation, Integration and Test Plan

5.1 Overview

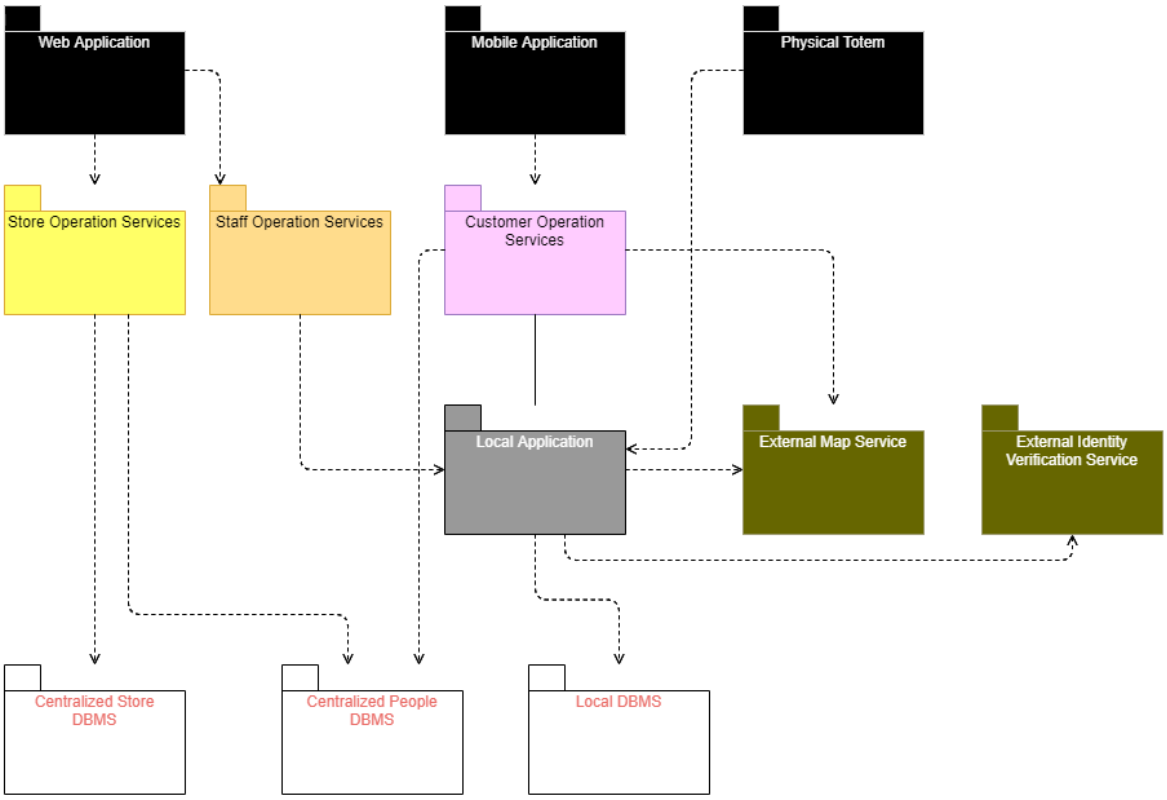


Figure 22: Implementation

In this part there will be an explanation on how and in which order the development, integration and testing of this project should be done. The idea is to complete a end user testable version with the bare minimum functionalities as soon as possible so that then it is possible to test this base extensively while adding the rest of the features.

5.2 Implementation Plan

The chosen approach for implementing, testing and integrating the system is bottom-up but by implementing in a first moment just the most basic functionalities of every component. The aim of this approach is to implement, integrate and test an initial version of the system as soon as possible so that then the missing features can be developed on a functioning base with the minimum need of drivers and stubs and by keeping them simple when they are strictly needed. This way it is in fact possible to firstly develop completely and properly test the core feature, which is the virtual queuing on the user part, and then add the other features i.e. booking the entrance on a certain time and date (which can partially be seen as an extension of the virtual queuing), ordered stores suggestions given filters and current queues and lastly the store management functions on the store manager's part. Thus the development of each component will be incremental once an initial version of the system with basic features from start to end is finished.

The development has to start with configuring the **Local DBMS** and creating the **Local Application** component (an instance of it will be hosted by every registered store), this component is in fact is the core of the management of information about queues, bookings, user reports and user notifications, and practically feeds the CLup system and clients with operational information. For now only the queues and notifications subcomponents will be developed. Also the interface with the physical totems (part of the queuing process) will be developed in order to allow people to queue in the stores even if they don't use the client application, which is part of the basic functionality. At this stage the **external components** will be connected to the system, they will not require any testing on their own since they come from sources that are assumed trustable and reliable.

After this the development of three more components can start: on the store side **Staff Operation Services**, which is needed for the functioning of the web app's control panel available to the store managers, for now only its functionality to allow and stop store entrances which is part of the *StoreAvailabilityManagement* subcoponent will be developed; on CLup's server side the components **Store Operation Services** together with **Centralized Store and People DBMSs** and **Customer Operation Services** can be developed/configured, the former with just the *StoreManagement* subcomponent while the latter with only the *StoreAvailabilityManagement*, according to what was developed in Local Application (e.g. no booking management).

Now it is possible to develop the user interfaces for each module: the **Mobile Application** for the smartphone users, the **Web Application** for the store managers and the **Physical Totem** software for those who don't use the application. For each of these interfaces (for now) it's enough to implement interactions just for those functions of the system that have already been developed.

After these steps there should be a functioning system composed of a mobile application, a program to be installed on store servers, a program to be deployed on physical totems and a system to be used on the main CLup servers to allow interactions between the previous: with these you should be able to accomplish the basic tasks objective of the purpose of this project such as register users, stores, do logins, get tickets to virtually queue in stores and enter and exit them with such tickets (it has to be noted that for the actually complete testing of the system there needs to be turnstiles and also proper machines for the totems in order to print tickets deployed at the store). The testing can then proceed on this basic version of CLup while the development keeps going on by implementing the missing features described in the RASD: the possibility for users to book entrances, algorithms to properly sort and suggest stores and times to go shopping and the other functions for the store manager's web app. Components wise the order in which to implement the various parts of these features over the existing parts of the system is the same as the one previously described. Summing up the implementation order goes in 3 phases with respective components:

1. The stores' local application: Local DBMS and Local Application

2. Central CLup system and local store interaction with this system: Staff Operation Services, Store Operation Services, Customer Operation Services, Centralized People and Centralized Store DBMSs

3. The users and store staff interfaces: Web Application, Mobile Application and Physical Totem

5.3 Integration Strategy

The integration of the components follows their implementation, so that no components will remain isolated at any time during development in order to avoid the risk of inconsistencies between them.

1. In a first moment the **Local DBMS** and **Local Application** are integrated, also the two **external components** are integrated in the system. A **stub** and a **driver** are used to simulate the components supposed to interact with the local application, anyway they don't need to be complex at all since all that is being implemented and integrated for now is just the basic functionalities and subcomponents, so Bookings and Reports will be left out for now.

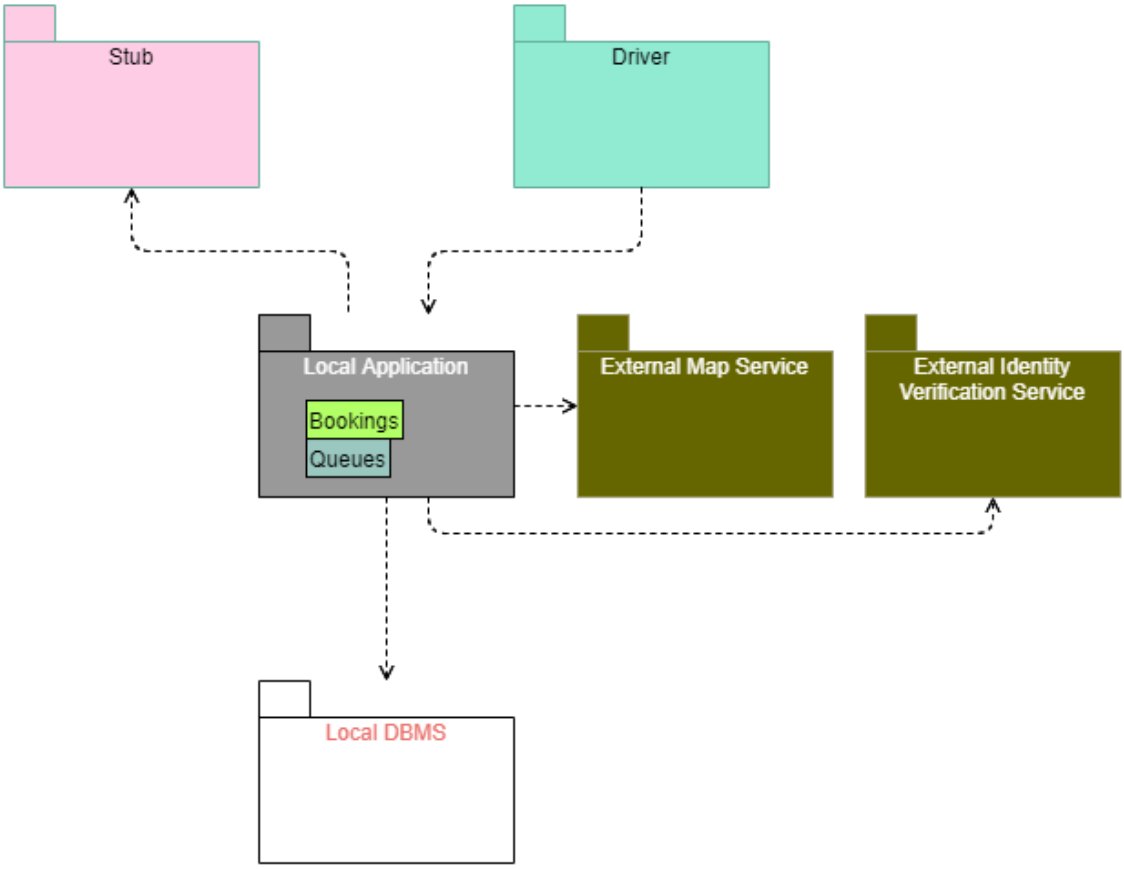


Figure 23: First step of the integration

2. Then **Staff Operation Services** is integrated with local application, also **Store Operation Services** with its **DBMSs** joins the system since they both share a **driver** for the Web Application. The other part of the central CLup's system can also be integrated since Local Application is ready: **Customer Operation Services**, there will be a **driver** for this part representing the mobile app and also a **driver** for the totem will remain from the previous step.

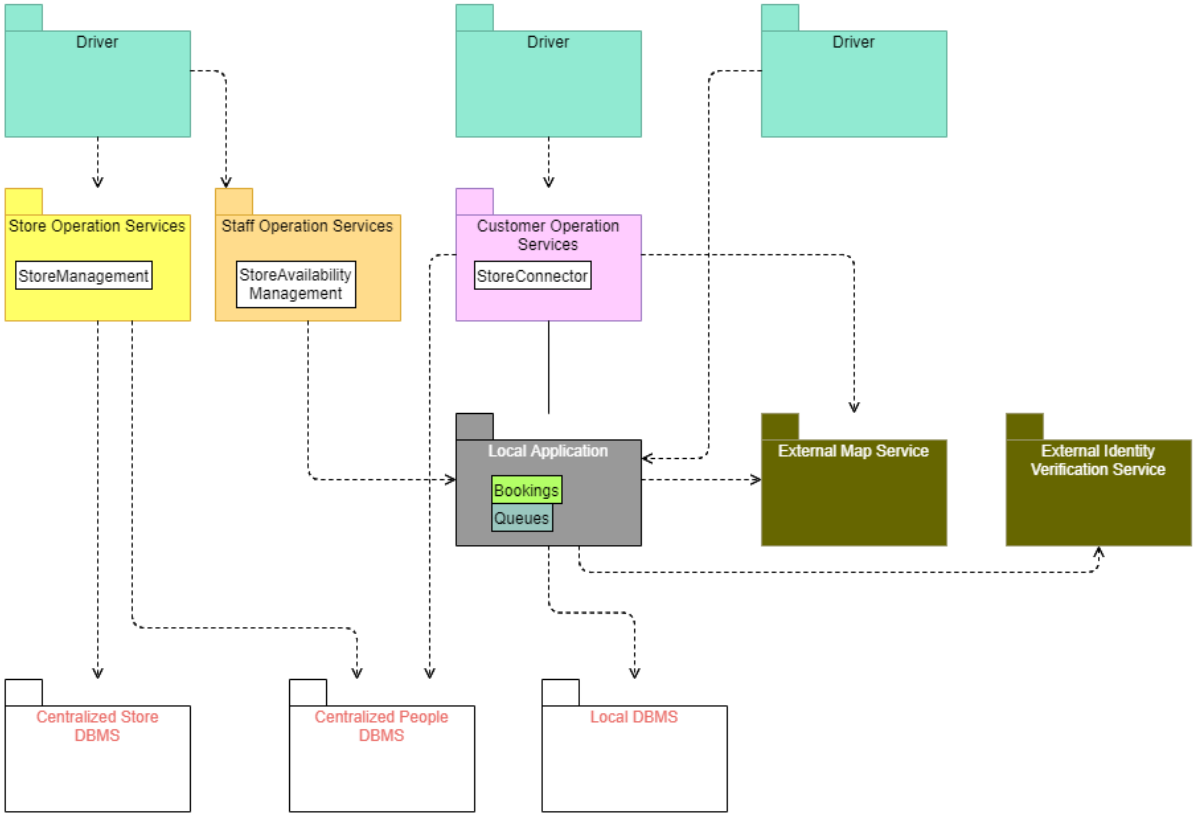


Figure 24: Second step of the integration

3. Lastly the user **interfaces** are integrated in order to complete the initial version of the project.

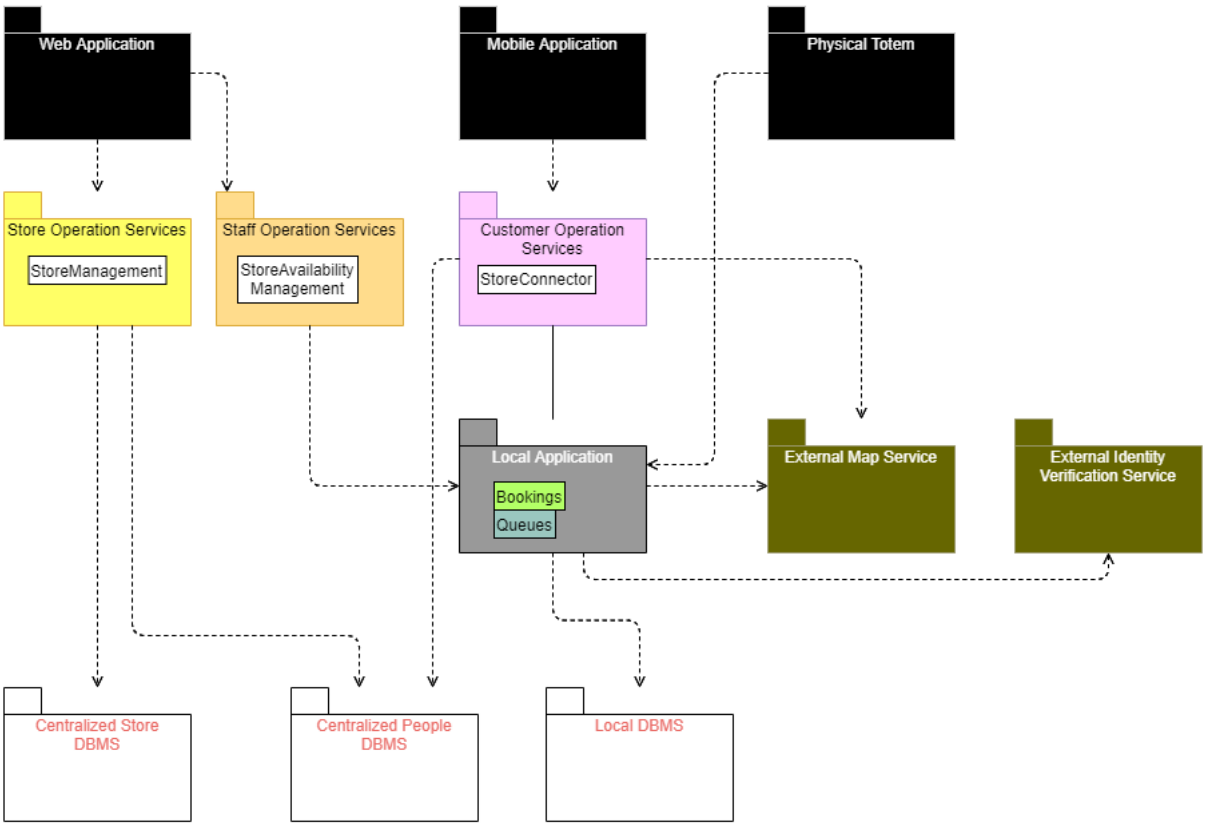


Figure 25: Third step of the integration showing the first version of the system capable of offering the basic functionalities

4. Now the remaining sub components will integrated following the order in which their respective components were integrated explained in this section.

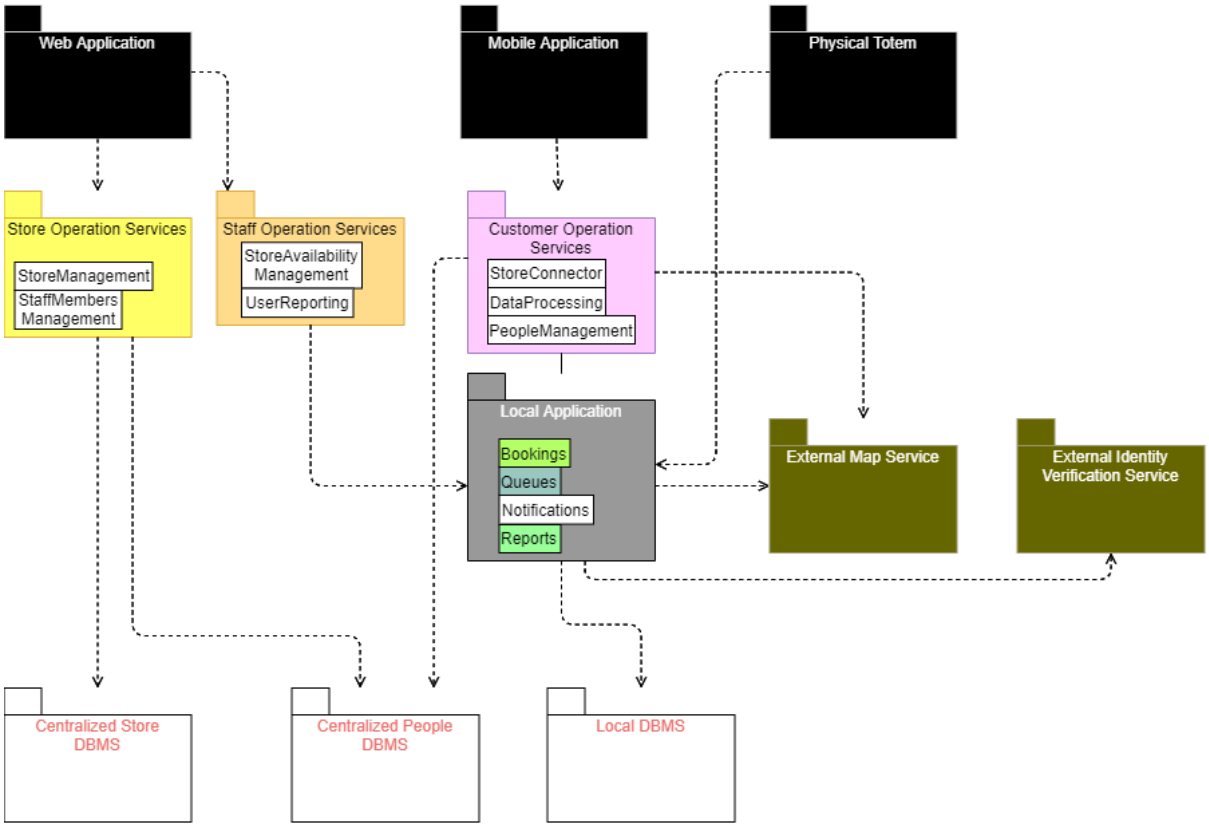


Figure 26: End of the integration showing the complete system

5.4 System Testing

Upon the completion of the basic version of the system, again at the completion of each system feature, and most importantly after the implementation and integration of every feature, the system must be tested in its entirety. The tests are meant to verify the satisfaction of both functional and nonfunctional requirements.

Common practices guide us on the types of tests we have to perform:

1. Functional testing: verify the satisfaction of the RASD functional requirements
2. Performance testing: find bottlenecks, response times affected, utilization and throughput, also run benchmarks
3. Load testing: identify upper limits for workload and expose possible bugs in memory management and buffer overflows
4. Stress testing: reveal how the system recovers from failures such as unexpected shutdowns and disconnections

5.5 Additional Specification on Testing

Here we will present some guidelines on how to analyse, test, verify and validate the project.

First of all an initial analysis over the RASD and DD documents should be conducted by a quality team. The analysis is divided in walkthrough and inspection, the first one should be performed in meetings with experts on the field of this project in order to do an initial check of

its correctness for its purpose. The second one also has to be performed, it focuses more on the formal aspects and is done by professional inspectors who will follow a checklist of things to be examined. The inspection activity must continue during the entire development since it will also check the generated code and it will also consist in meetings where the members of the developer team have to partecipate. The job of the quality team is to just report problems in the code while the developers have to be the ones correcting such problems, the fixes then will be checked again by the quality team. It is very important that the code is properly commented in a semi-formal way by the developers, so that the quality team can expect standardized work to analyse. Apart from the inspection work, in order to find problems with the code, effort must also be done by the developers in the generation of test cases which should cover at least 90% of the code, since this will help find problems earlier than the inspection. As previously mentioned the aim is to have an usable application with basic functionalities as soon as possible, so when it will be ready an additional team of testers will go through it and its next versions to test and evaluate the integrated system. The code testing has to be white-box systematic since this is incentivated by the bottom up approach and almost all the components are going to be built from scratch.

6 Effort Spent

In this section we provide an overview about how much time each section required us to be produced.

Section 1.1	1 hour
Section 1.2	1 hour
Sections 1.3, 1.4, 1.5, 1.6	0.5 hours
Section 2.1	2 hours
Section 2.2	4 hours
Section 2.4	1 hour
Section 2.5	6 hours
Section 2.6	3 hours
Section 3	0.5 hours
Section 4	8 hours
Section 5	0.5 hours
Section 6	0.5 hours

Table 12: Neroni's work overview

Section 1.1	0.2 hour
Section 1.2	0.2 hours
Sections 1.3, 1.4, 1.5	0.5 hours
Section 2.1	1 hour
Section 2.2	15 hours
Section 2.3	2 hours
Section 2.4	12 hours
Section 3	0.5 hour
Section 4	0.5 hours
Section 5	0.5 hours
Section 6	0.5 hours

Table 13: Pozzi's work overview

Section 1.1	0.5 hours
Section 2.2	2 hours
Section 2.3	0.5 hours
Section 2.5	2 hours
Section 3.1	12 hours
Section 5.1	1 hour
Section 5.2	6 hours
Section 5.3	2 hour
Section 5.4	2 hours
Section 5.5	3 hours

Table 14: Vetere's work overview

References

[1] Kirill Fakhroutdinov. **UML-Diagrams | Reference**. URL: https://www.uml-diagrams.org/.

[2] Software & Systems Engineering Standards Committee of the IEEE Computer Society. **IEEE Standard for Information Technology—Systems Design — Software Design Descriptions**. 2009-07-20. IEEE Std 1016TM-2009, revision of IEEE Std 1016-1998.

