Politecnico di Milano
A.A. 2015-2016
Software Engineering 2: "myTaxiService"
Design **D**ocument

Roberto Clapis (841859), Erica Stella (854443)

December 9, 2015

Contents

1 Introduction

1.1 Purpose

The purpose of the Design Document is to provide documentation in order to aid the development of myTaxiService's system by providing a description of how it should be built and how its components are expected to interact with each other.

1.2 Scope

This Design Document is intended to explain the design and architecture of myTaxiService, a new application that will provide an easy way to access the taxi service in a city. It describes the system both from a software and hardware point of view, in order to clarify the system's structure and how it accomplishes its functionalities.

1.3 Definitions, Acronyms, Abbreviations

1.3.0.1 Definitions

- *End users:* this category comprises all those who use the application[1]: administrators, taxi drivers, logged in users and guests.

1.3.0.2 Acronyms

- *UI:* User Interface through which the end users can interact with the application.
- *DB:* Database.
- *DBMS:* Database Management System.
- *SSL:* Secure Socket Layer, a protocol that ensures safe end-to-end transmissions.
- *API:* Application Programming Interface.
- *ETA:* estimated time of arrival: the time, estimated by the system, that the closest available taxi will take to get to the starting location of the ride.
- *CAT:* Closest Available Taxi, the taxi with the smallest ETA called if there are no taxi in the starting location's zone.

[1]For their definition refer to the RASD's section 1.6

1.4 Reference Documents

- Document with the assignment for the project
- RASD for myTaxiService
- Template for the Design Document

1.5 Document Structure

The following parts of this document are structured in 3 sections: architectural design, user interface design and requirements traceability. The architectural design section describes the software and hardware components of the system and their interactions. The user interface design section refers to the "User Interfaces" subsection of the RASD. The requirements traceability section explains how the proposed design meets the requirements that have been defined in the RASD.

2 Architectural Design

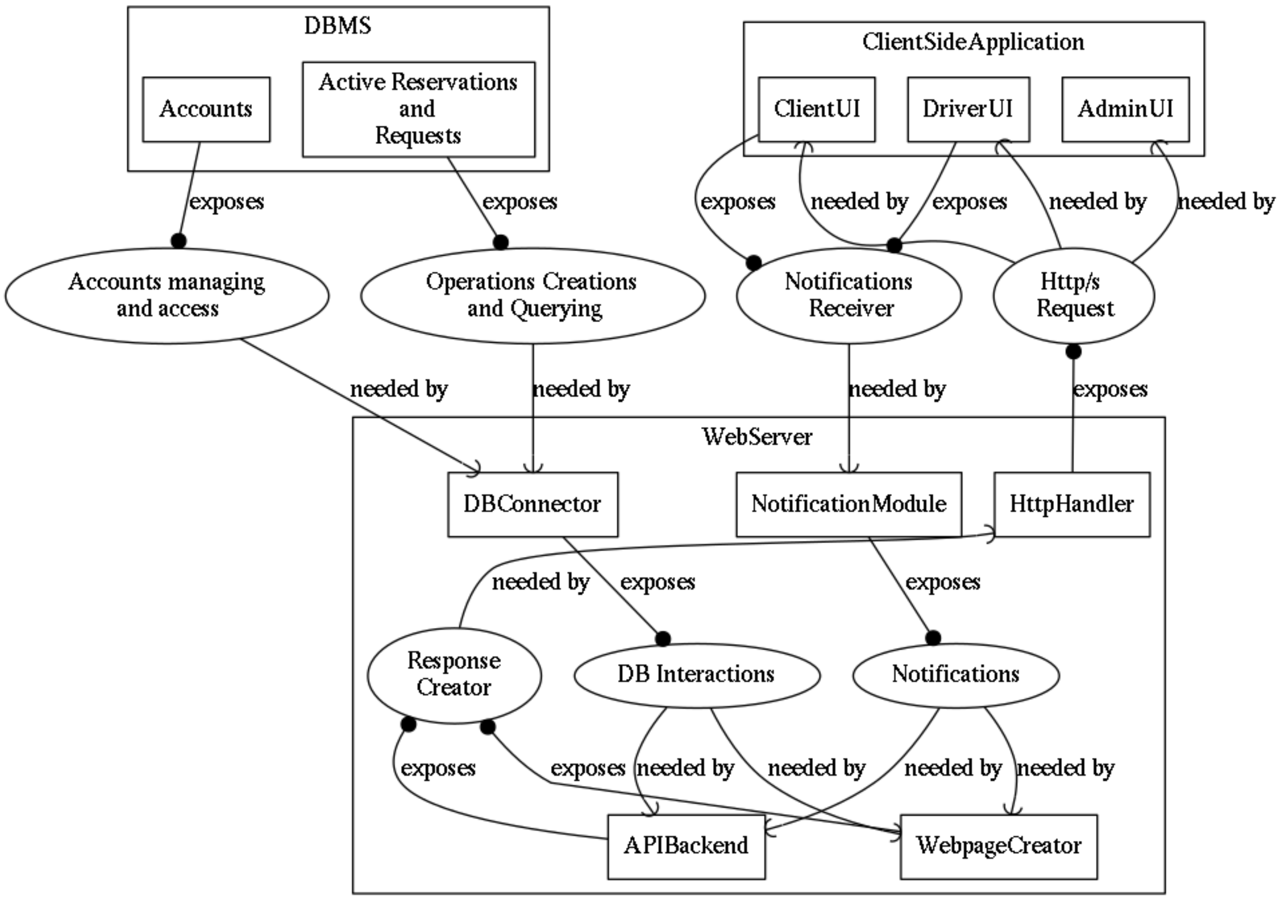
2.1 Overview

As mentioned before, the system to be developed will be used to provide an easy access to the taxi service of a city. Therefore, its main functionalities, that will have to be supported by the design and architecture, are: the storage of the taxi drivers' and clients' accounts, the computation of the taxi queue of each zone, the handling of requests and reservations and the exposure of publicly available API. Furthermore, the system will have to comply to quality of service attributes as specified in the RASD.

2.2 High Level Components and Their Interaction

myTaxiService's system is composed by three main components: DBMS, Web Server and client application. The client application provides the UI through which end users can access the its services. These requests are forwarded to the Web Server which is in charge of providing a response, eventually querying the Database in the DBMS. The Web Server is also responsible for answering to the API calls coming from external applications and notifying the end users for particular events, like when a request is accepted by a taxi driver. The DBMS stores all the information of the end users's accounts and the active requests and reservations.

2.3 Component View

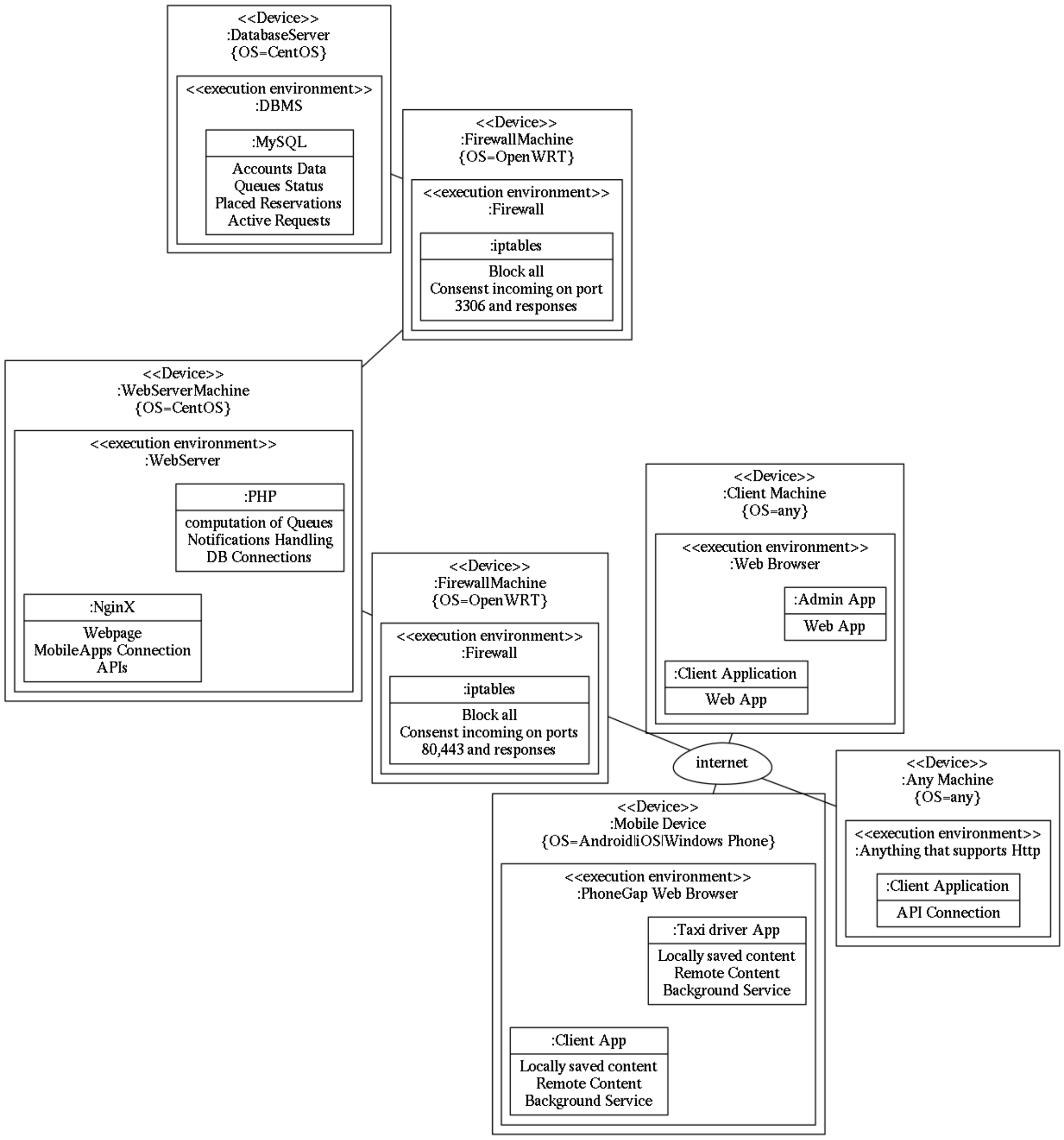


While the DBMS and the Client Side application are self explanatory the WebServer part requires some details:

- The Webpage Creator is the responsible for both the Mobile App and Web App responses. The HttpHandler will recognize the request by the parameters and ask the Webpage Creator to either create the full HTML page (in the case of the Web App) or just send a partially created page that only contains the dynamic data that the Mobile App will then include in its interface.
- The API Backend will be called by the HttpHandler and will respond only with the requested data in a JSON format.
- The HttpHandler is responsible to forward the requests either to the APIBackend or the WebpageCreator depending on their origin.

- The DBConnector provides triggers to communicate to the Notification-Module changes in the state of requests or reservations that need to be notified.

2.4 Deployment View



The system will have a standard configuration with DMZ, Internal network, double firewall and untrusted network, see more in section 2.8

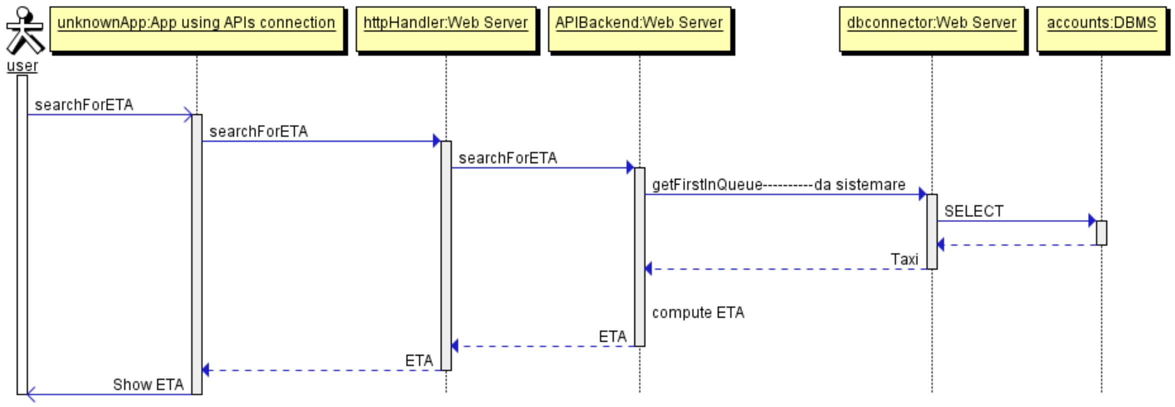
2.5 Runtime View

This section provides an insight on how the components interact during common use cases of the system.
Some details:

- HTTPS negotiations are omitted in order to preserve readability, but they would all be between the apps (Web or Mobile) and the httpHandler .
- Queues are not represented in the database, but they are all re-computed, whenever it is requested, with a query that selects the taxis that are active, are in the desired area and returns them sorted by the time they were marked as active.

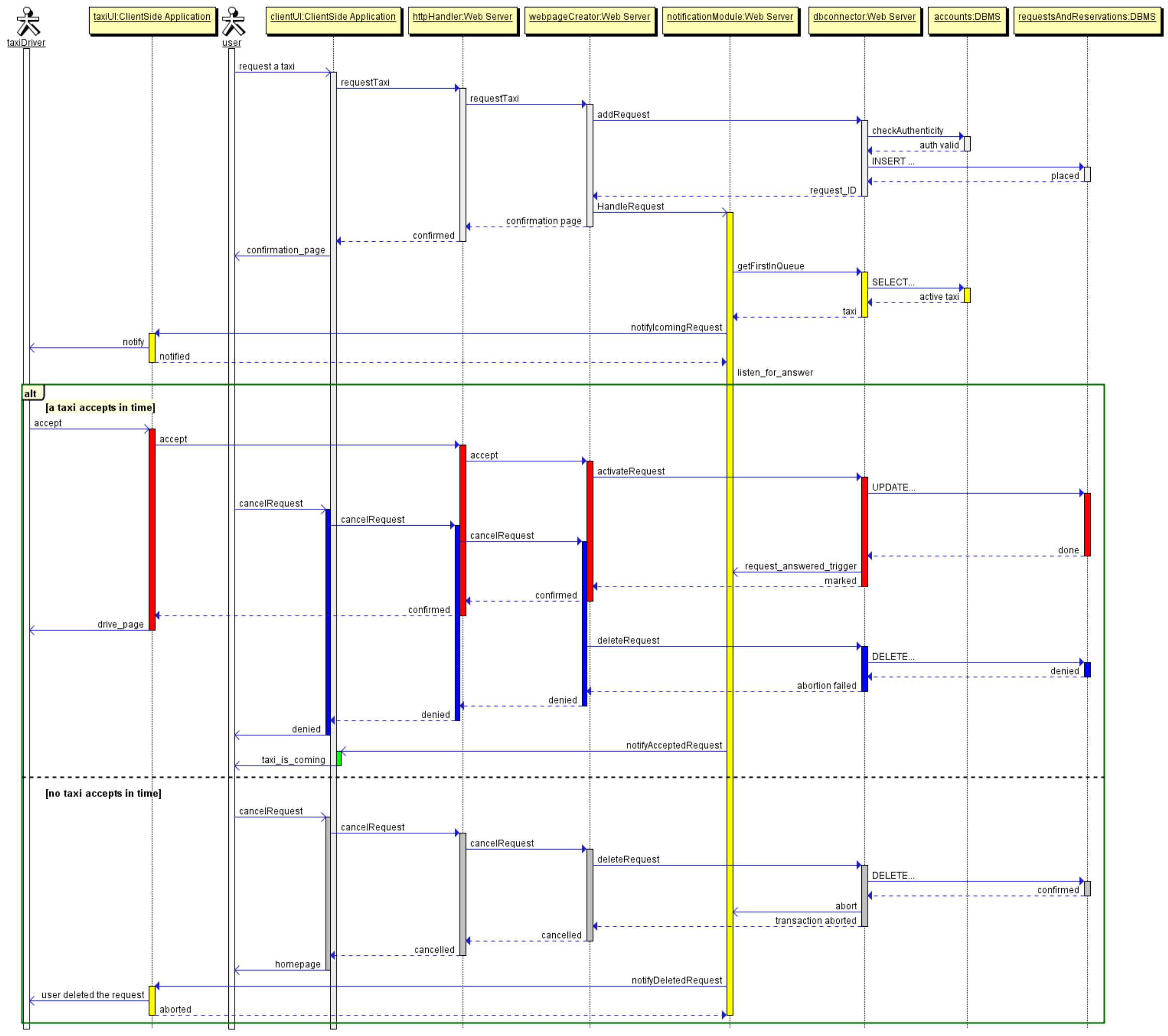
2.5.1 Search for ETA

The following Sequence Diagram is shown with a guest but it's equally applicable to a logged in user searching for the ETA.



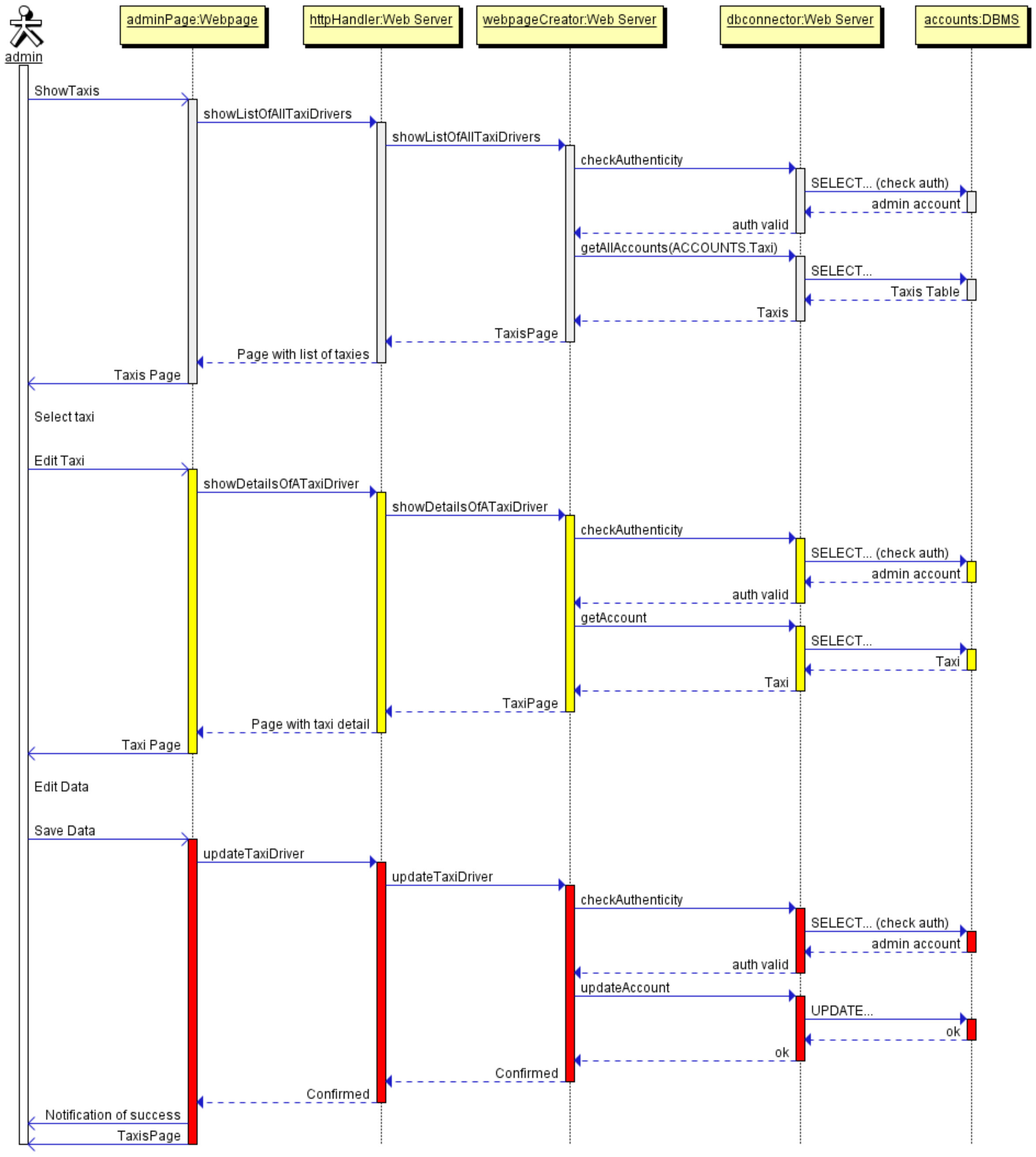
2.5.2 Cancel a request

The following Sequence Diagram shows how the components interact when a user tries to cancel a request. If the starting location's queue is empty, the CAT is chosen calculating the ETA of all the available taxis and choosing the smallest one. The last passage is omitted for readability.



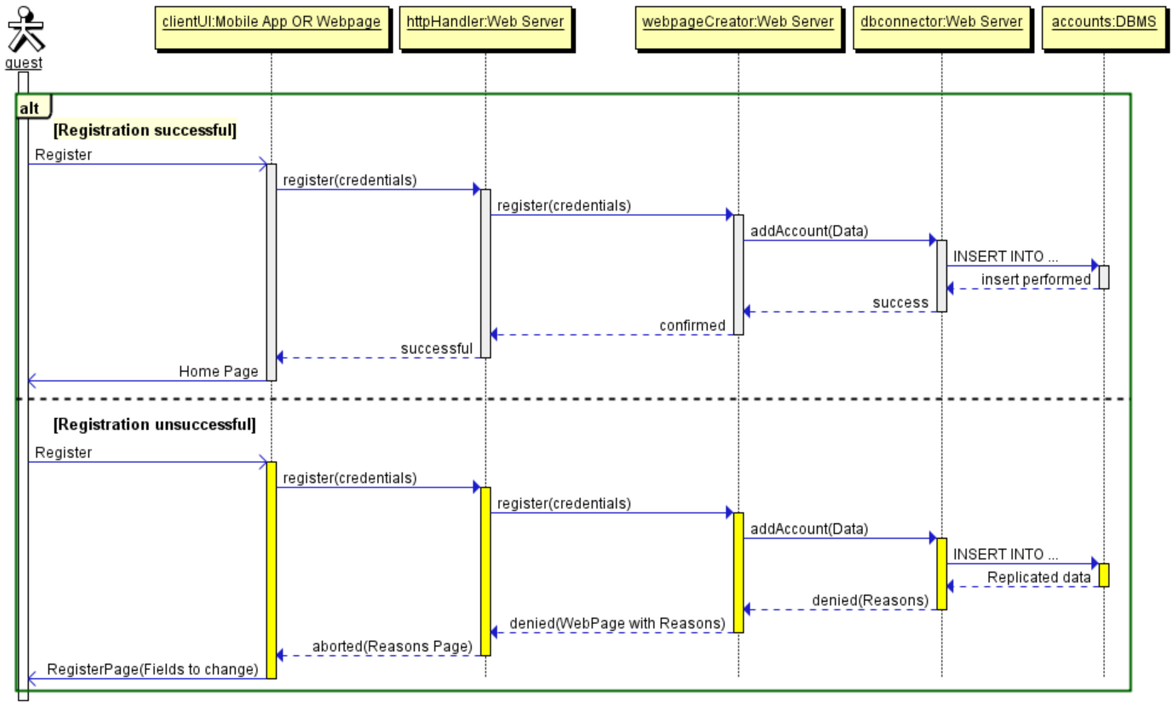
2.5.3 Administrator's usage

The following Sequence Diagram provides an example of how the administrator can update the account of a taxi driver.



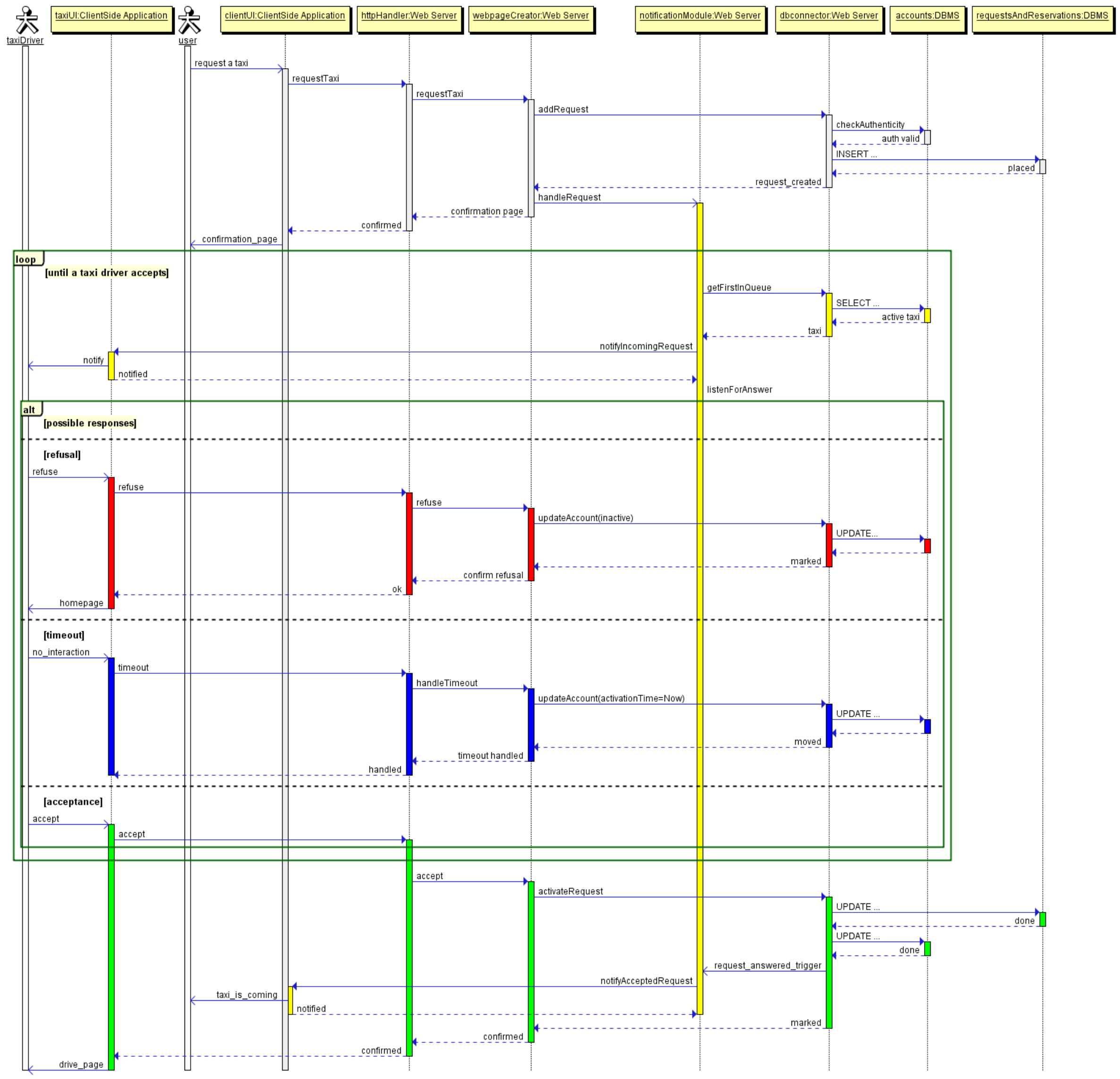
2.5.4 Registration

The following Sequence Diagram shows what happens when a guest tries to register to the system both with valid credentials and with credentials that are already in the DB.



2.5.5 Request a taxi

The following Sequence Diagram demonstrates how a user can request a taxi.



2.6 Component Interfaces

2.6.1 DB Interactions

This interface encapsulates and exposes all the operations the Web Server needs to interact with the DB. The operations are divided in 2 categories based on the DB's part they manage:

- *Accounts managing and access:* this interface exposes methods to manage the stored end users accounts.

Method	Parameters required	Notes
addAccount	account type and credentials	adds an account to the DB of the specified type if there's not another one with the same ID
updateAccount	user ID, attribute to update, new value	updates the attribute specified to the new value
deleteAccount	user ID	deletes the account specified by the ID
getAccount	user ID	returns the account corresponding to the ID with all its attributes
getAllAccounts	type of the accounts	returns all the accounts of the specified type
getFirstInQueue	location	returns the first taxi (by activation time) that is in the given zone.
checkAuthenticity	user ID	returns true if the specified account is in the DB, false otherwise

- *Operations creations and querying:* this interface exposes methods to manage and retrieve requests and reservations.

Method	Parameters required	Notes
addRequest	starting location, destination, user ID	adds a new "not accepted" request to the DB from the user that requested it and returns its ID to the caller
deleteRequest	request ID	deletes an existing request from the DB and returns true. If the request is marked as not removable it returns false
activateRequest	request ID, taxi driver ID	changes the state of the request from "not accepted" to "accepted" and the taxi's state to unavailable. It also marks the request as not removable
getRequest	request ID	returns the request specified by the ID and all its attributes
getAllUserCalls	user ID	returns all the active requests and reservations of the specified user
addReservation	starting locations, destination, meeting time	adds a new reservation to the DB from the specified user and return its ID
deleteReservation	reservation ID	deletes an existing reservation from the DB and returns true. If the reservation is marked as not removable it returns false
activateReservation	reservation ID, taxi ID	changes the state of the reservation from "not accepted" to "accepted" and the taxi to unavailable. It also marks the reservation as not removable
getReservation	reservation ID	returns the reservation specified by the ID with all its attributes

2.6.2 Response creator

This interface exposes all the functionalities end users require to access the services offered by myTaxiService. In the following list the methods are divided by type of end user that needs them.

- *Functionalities exploited by guests:*

Method	Parameters required	Notes
searchForETA	starting location	returns the ETA of the first taxi in the queue of the starting location or, if the queue is empty, the ETA of the CAT
register	credentials	if the credentials don't refer to an already existing account, a new one is created
login	credentials	the guest logs into the system

- In the following tables, user (or taxi driver) IDs are always passed as a parameter, as they are needed for the authentication, but they're omitted for readability.

- *Functionalities exploited by logged in user:* logged in user can access the searchForETA method exposed for guests along with the following methods:

Method	Parameters required	Notes
requestTaxi	starting location, destination	requests a taxi for the specified starting location
cancelRequest	request ID	cancels the request if it's not been accepted yet
showRequestDetails	request ID	shows the details of the specified request
reserveTaxi	starting location, destination, meeting time	reserve a taxi for the specified starting location and meeting time
cancelReservation	reservation ID	cancels the reservation if it's not been accepted yet
showReservationDetails	reservation ID	shows the details of the specified reservation
logout		the user is logged out of the system

- *Functionalities exploited by taxi drivers:*

Method	Parameters required	Notes
accept	request or reservation ID	the request or reservation for which the taxi driver has been notified, specified by its ID, is accepted
refuse	request or reservation ID	the request or reservation for which the taxi driver has been notified, specified by its ID, is refused
toggleState		the state of the taxi driver is changed from not available to available, or vice versa

- *Functionalities exploited by administrators:*

Method	Parameters required	Notes
showListOfAllTaxiDrivers		shows the list of all the taxi drivers in the DB
showDetailsOfATaxiDriver	taxi driver ID	shows the details of the specified taxi driver
addTaxiDriverAccount	credentials	a new account for a taxi driver is created with the specified credentials
updateTaxiDriverAccount	attributes to update, new values, taxi driver ID	the attributes of the account of the taxi driver, specified by its ID, are updated to the new values
deleteTaxiDriverAccount	taxi driver ID	the account of the taxi driver specified by its ID is deleted

2.6.3 Http/s Request

This interface exposes the standard GET and POST Http methods, that will be handled and forwarded either to the Web Page Creator or the API Backend or will be rejected if invalid.

2.6.4 Notification Receiver

This interface will expose only one method, that will receive the messages from the server and will handle them according to their destination:

- API: this will just be a listener that will put the message in a synchronized queue for the user to pull
- ClientUI/DriverUI: this will change the form shown with a new one (encoded in the message) that will show the notification.

2.6.5 Notifications

This interface exposes methods to notify end users (except for administrators) of particular events. A call to this interface will spawn asynchronous workers that will notify the user of the changes of a request/reservation. If a worker for that request already exists it will handle the new notification too. It also provides the methods handleRequest and handleReservation to handle the life cycle of requests and reservations after they've been placed by the user. The following list distinguishes notifications sent to taxi drivers from those sent to logged in users.

– *Notifications sent to logged in users:*

Method	Parameters required	Notes
notifyAcceptedRequest	request ID, taxi driver ID, ETA	the user is notified of the taxi driver that has accepted the specified request and the ETA
notifyAcceptedReservation	reservation ID, taxi driver ID	the user is notified of the taxi driver that has accepted the specified reservation

– *Notifications sent to taxi drivers:*

Method	Parameters required	Notes
notifyIncomingRequest	starting location, user ID, request ID	the taxi driver is notified of an incoming request from the specified user and starting location
notifyIncomingReservation	starting location, user ID, meeting time, reservation ID	the taxi driver is notified of an incoming reservation for the specified meeting time, from the specified user and starting location
notifyDeletedRequest		the taxi driver is notified that the user has deleted the request for which he was prompted to accept or refuse

2.7 Selected Architectural Styles and Patterns

The most important part of the application is based on a classical Client-Server pattern: the service is completely provided by the centralized core, to which clients connect in order to perform any operation.
More in details, the pattern involves a very thin client with almost no functionalities except for connecting to the server and displaying the received information.
For the Web Server it was decided to adopt a 3-tier subdivision: the untrusted internet, the DMZ and the internal network.

2.8 Other Design Decisions

In order to provide an easier maintenance and to allow the project to scale easily, two main points were stated:

- A cross-platform web-based framework should be chosen to develop the mobile version of the application
- A cloud based approach should be chosen instead of buying the hardware to host the service

3 User Interface Design

This section has been explored in RASD's section 2.1.1 "User Interfaces" so we refer to that one.

4 Requirements Traceability

Functional requirements:

- Data accessibility, mutability and creation for the end users is granted thanks to the HTTP interface exposed on the Internet.

- Notification delivery is granted through the interface exposed by the mobile applications and web applications.

Non functional requirements:

- Secure channels will be established using SSL
- Authentication will always be checked before accessing sensible data.
- Reliability, scalability and availability of resources will be granted by using a cloud-based system. The deployment will not be made on physical machines but on virtual ones rented from reliable providers.

5 References

- http://www.uml-diagrams.org/deployment-diagrams.html
- http://www.uml-diagrams.org/component-diagrams.html
- http://www.uml-diagrams.org/sequence-diagrams.html

6 Appendix

Appendix for Roberto Clapis
Work hours: 20

Software Used:

Task	Software
Edit LaTeX Source	Vim
Edit Graphs Sources	Vim
Edit sources for Sequence Diagrams	Vim
Convert Sequence Diagrams to images	Quick SequenceDiagramEditor
Generate and Raster directed graphs	Dot
Generate and Raster undirected graphs	Fdp
General images mangling and cropping	ImageMagick & Shotwell
Convert LaTeX source to PDF	LaTeXMK
Spell Check	Aspell
LaTeX Check	LaCheck

Appendix for Erica Stella
Work hours: 20

Software Used:

Task	Software
Edit LaTeX Source	TexStudio
Convert LaTeX source to PDF	LaTeXMK
Edit sources for Sequence Diagrams	Quick Sequence Diagram Editor
Convert Sequence Diagrams to images	Quick SequenceDiagramEditor