



TAXI SERVICE

DD

Design Document

version 1.1



Rota Diego, 841344, 15 hours

Montalto Simone, 841359, 15 hours

Politecnico di Milano, A.A. 2015-2016

Software Engineering 2 – Prof.ssa Mirandola Raffaella

Table of contents

1. INTRODUCTION	3
1.1 PURPOSE AND SCOPE	3
1.2 VOCABULARY	3
1.3 ACRONYMS	3
1.4 REFERENCE DOCUMENTS.....	4
1.5 DOCUMENT STRUCTURE	4
2. ARCHITECTURAL DESIGN	5
2.1 OVERVIEW	5
2.2 HIGH LEVEL COMPONENTS AND THEIR INTERACTIONS	5
2.3 COMPONENT VIEW.....	6
2.4 DEPLOYMENT VIEW	7
2.5 RUNTIME VIEW	8
2.6 ARCHITECTURAL STYLES AND PATTERNS.....	9
3. ALGORITHM DESIGN	10
3.1 REQUEST GPS DATA.....	10
3.2 NEW VEHICLE IN QUEUE	10
3.3 ASSIGN DRIVER TO A REQUEST	10
4. USER INTERFACE DESIGN	12
4.1 LOGIN FROM WEBAPP	12
4.2 LOGIN FROM MOBILE APP	13
4.3 CUSTOMER BOOKED REQUEST FROM WEBAPP	14
4.4 CUSTOMER BOOKED REQUEST FROM MOBILE APP	15
4.5 DRIVER SEE ASSIGNED ZONE FROM MOBILE APP.....	16
4.5 DRIVER NOTIFICATION FROM MOBILE APP.....	17
4.6 CUSTOMER IMMEDIATE REQUEST FROM MOBILE APP.....	18
5. REQUIREMENTS TRACEABILITY.....	19
5.1 ALLOW A GUEST TO REGISTER AND LOG IN TO THE SYSTEM.....	19
5.2 ALLOW A DRIVER TO ACCEPT OR DECLINE A REQUEST	19
5.3 ALLOW DRIVER TO REPORT ITS AVAILABILITY	19
5.4 ALLOW A CUSTOMER TO PERFORM A REQUEST	20
5.5 ALLOW AN EXTERNAL DEVELOPER TO REQUEST API	20
5.6 SYSTEM SEND A NOTIFICATION.....	21

1. Introduction

1.1 Purpose and scope

This document has the purpose to explain our hardware decisions in order to guarantee the requirements specified in the RASD document. Will be shown the architectural design, with an overview of the high level components. To better understand how the component works, some diagrams show their static and dynamic interaction, focusing also on the runtime execution of the service. The most relevant algorithm are explained in the algorithm design section and are illustrated using pseudocode. In the last part of the Design Document, some mockup are available to better understand how the users can interact with the mobile application and the web application client side. In the requirements traceability sections, the requirements specified in the RASD document are mapped with the component introduced in this document.

1.2 Vocabulary

- **Guest:** it is an user that is not yet registered;
- **Customer:** it is an user that is registered and correspond to the passenger;
- **Driver:** it is an user that is registered and correspond to the taxi driver;
- **External Developer:** it is an user that is registered and can only require API of the system;
- **User:** it could be a Guest, Customer, Driver or External Developer.
- **Vehicle:** correspond to the taxi car that is driven by a Driver;
- **Request:** it is a generic reservation made by a Customer;
- **Immediate Request:** it is a specific Request, and it is made by the Customer to require a taxi as soon as possible;
- **Booked Request:** it is a specific Request, and it is made by the Customer to book a taxi for a specific hour, origin and destination address;
- **Zone:** it indicates a specific area of the city that include only one queue.

1.3 Acronyms

- **RASD:** Requirement Analysis and Specification Document;
- **DD:** Design Document.

1.4 Reference Documents

1. Taxi Service RASD Document – Rota Diego, Montalto Simone

1.5 Document Structure

- **Section 1 – Introduction:** it includes the purpose and the scope of the Design Documents, including the Vocabulary of the terms used in the document, acronyms and reference documents.
- **Section 2 – Architectural Design:** it includes component diagram, deployment diagram and runtime diagram.
- **Section 3 – Algorithm Design:** it includes some pseudocode to better explain how the major algorithms works.
- **Section 4 – User Interface Design:** it includes some mockup of the web app and the mobile app used by the Users.
- **Section 5 – Requirements Traceability:** it includes a mapping of the RASD requirements with the components introduced in the Design Document.

2. Architectural Design

2.1 Overview

In this section are listed the hardware components used in Taxi Server System and it is explained how they interact each other. Some diagrams shown static and dynamic behavior of the components and the logical path at runtime when a Customer insert a Request, with the sequence interaction of the components. At the end of the section is shown the architectural design pattern used, with a schema to better understand how it works.

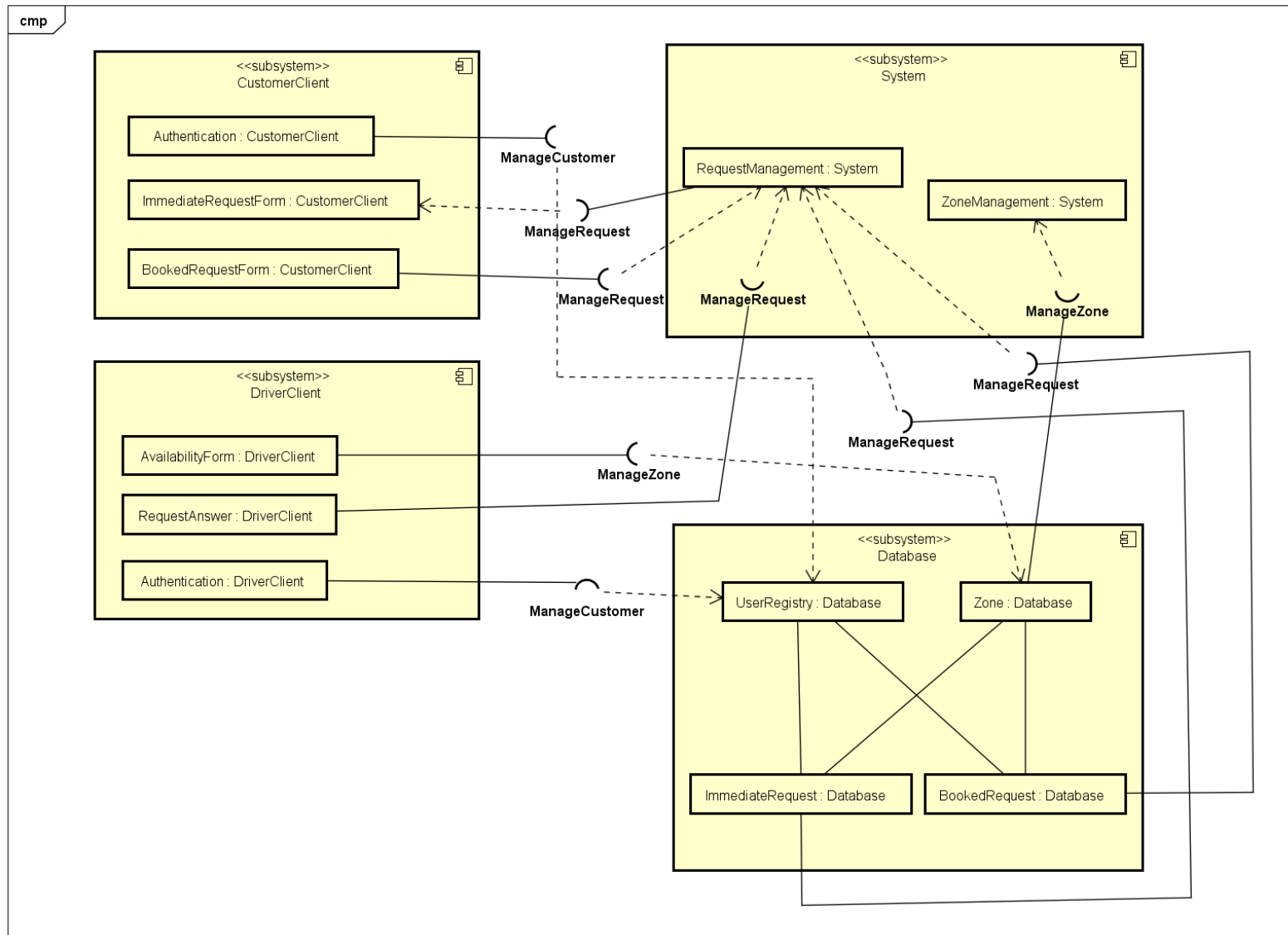
2.2 High level components and their interactions

In Taxi Service System we have identified different hardware components in order to obtain the goal:

- **Web Server:** this component manage the communication between the client device and the server and it generates all the view that will be displayed on Customer and Driver devices;
- **Application Server:** this component elaborate the information sent by the User and communicate with the database in order to apply the logical part of the system;
- **Database Server:** this component store all the data used by the system. Here there are all the request (Booked and Immediate), Users data, Zone data and so on;
- **Client (devices for Web Application or Mobile Application provided by Customer and Driver):** this component is used to compile the form necessary to perform a request by the users and response to a request and send availability by the drivers. Client devices receive the view from the Web Server.

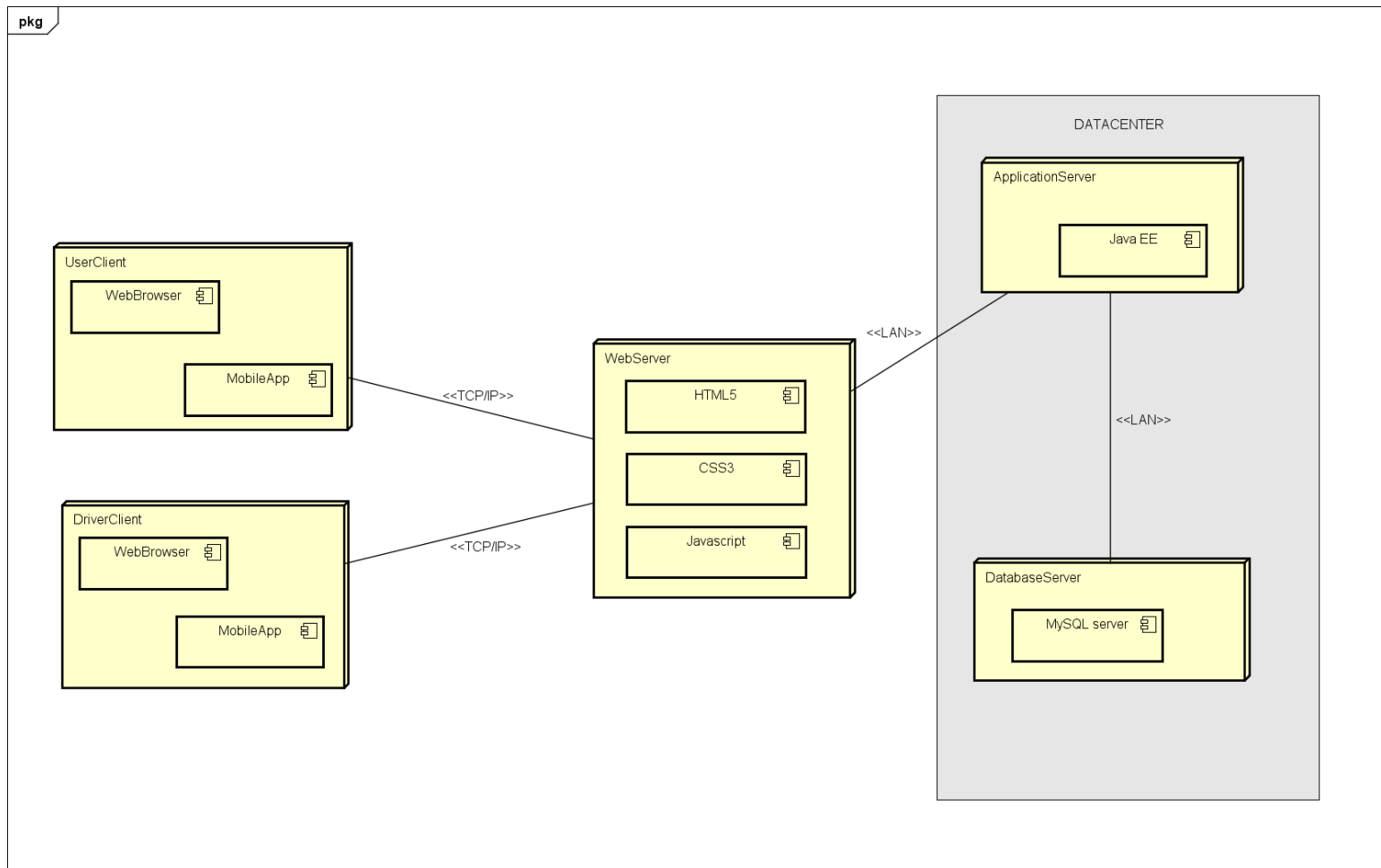
2.3 Component View

We use the Component View Diagram in order to explain how the components of our system are wired together, to form a larger system.



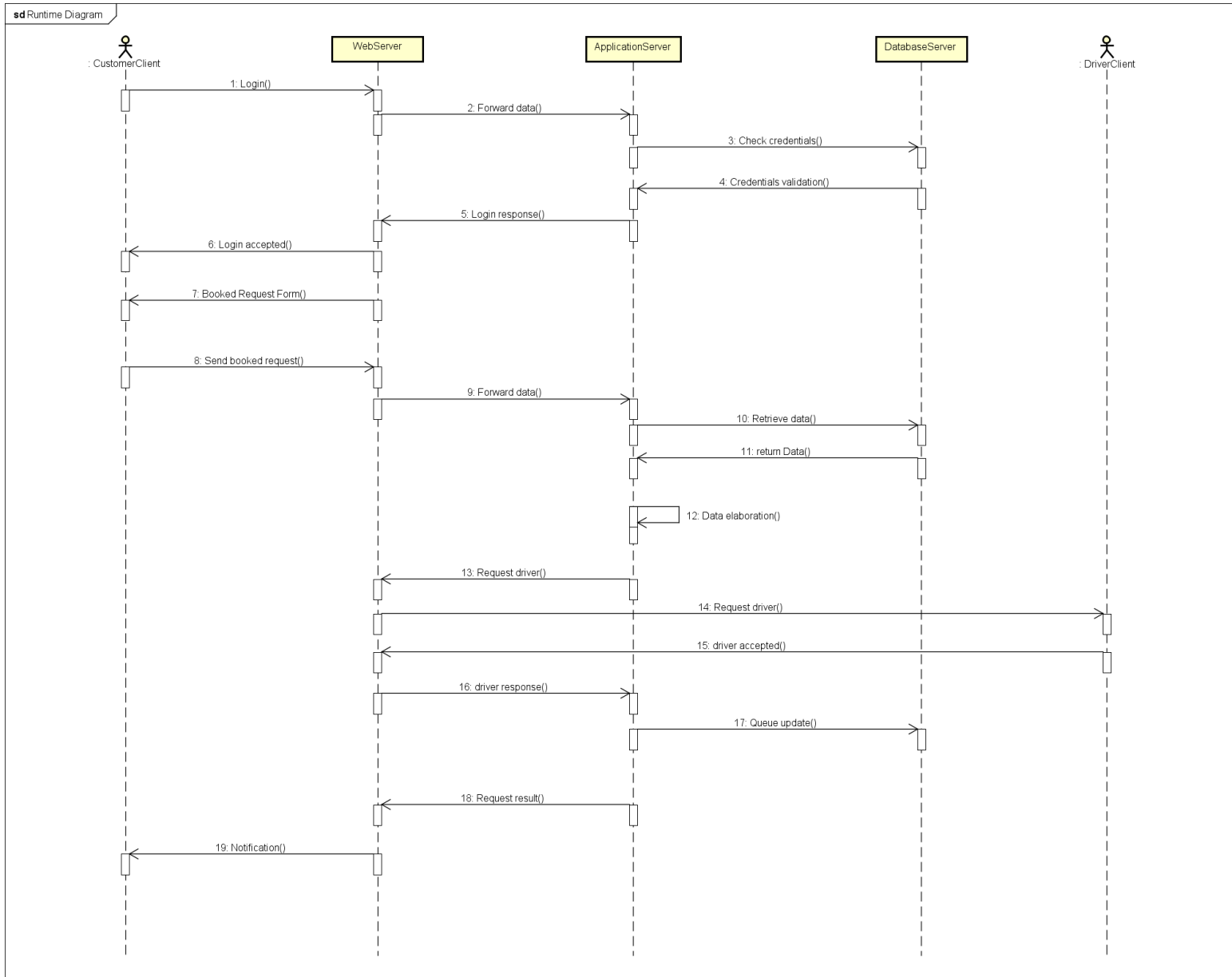
2.4 Deployment View

This Deployment View Diagram is a static diagram used to describe the hardware resources and the relation between them.



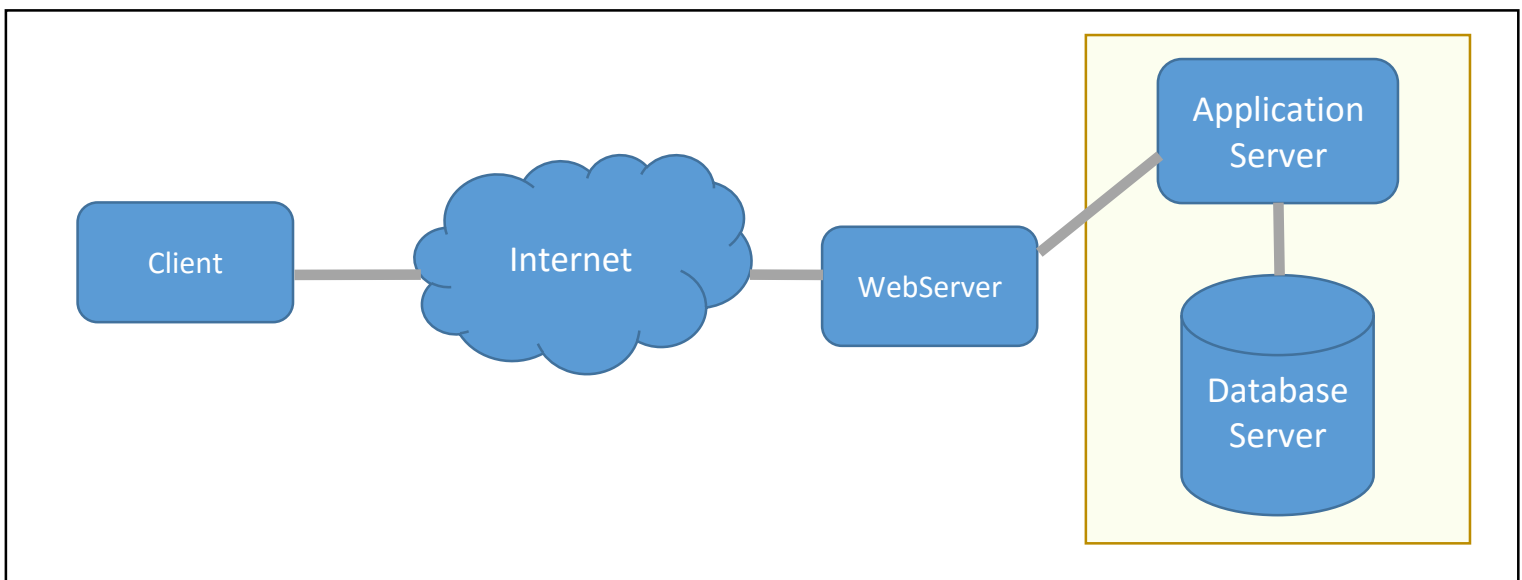
2.5 Runtime View

We have used the sequence diagram and expanded all the components in order to show and describe how the components interact together during the elaboration of a work.



2.6 Architectural styles and patterns

The Taxi Service System is based on a three tiers architectures Client/Server. In this kind of architectures we have a Client, a Web Server, an Application Server and a Database Server. In Client side there is not application logic. For this reason, the Client have to sent all the information to the Application Server. To accomplish this, the Web Server receiver all the information from the Client and forward it to the Application Server. The Application Server communicates with the Database Server to perform the operations and store data. Web Server doesn't communicate with the Database Server, that can be accessed only by the Application Server. When the Application Server have to sent information to the Client, it sends the data to the Web Server that creates an HTML page to show to the user devices.



3. Algorithm design

In this section will be explained the most relevant algorithm used in the Taxi Service System.

3.1 Request GPS Data

This algorithm periodically obtains (every 1 minute) the GPS position of each Vehicle and sends this information to the Application Server that provides to store this data in the Database Server.

PSEUDOCODE CLIENT SIDE

```
while(true) {  
    if (timer == 1 minute) {  
        sendGPStoServer();  
        resetTimer();  
    }  
}
```

3.2 New Vehicle in queue

When a Driver gave its availability, this algorithm takes the last saved position of the Vehicle, search the nearest zone and assign the Vehicle to the queue of this zone in the last position.

PSEUDOCODE APPLICATION SERVER SIDE

```
if (driver.isAvailable) {  
    zone = nearestZone(driver.position);  
    zone.assignVehicle( driver.getVehicle() );  
}
```

3.3 Assign Driver to a Request

When there is a new Request (Immediate or Booked), this algorithm asks to each Driver in the queue of the requested zone, starting from the first in the queue, if it accept the request or not. If a Driver accept, then the algorithm return the driver that

has accepted the request, else, if no Drivers gives the availability, then the algorithm return NULL.

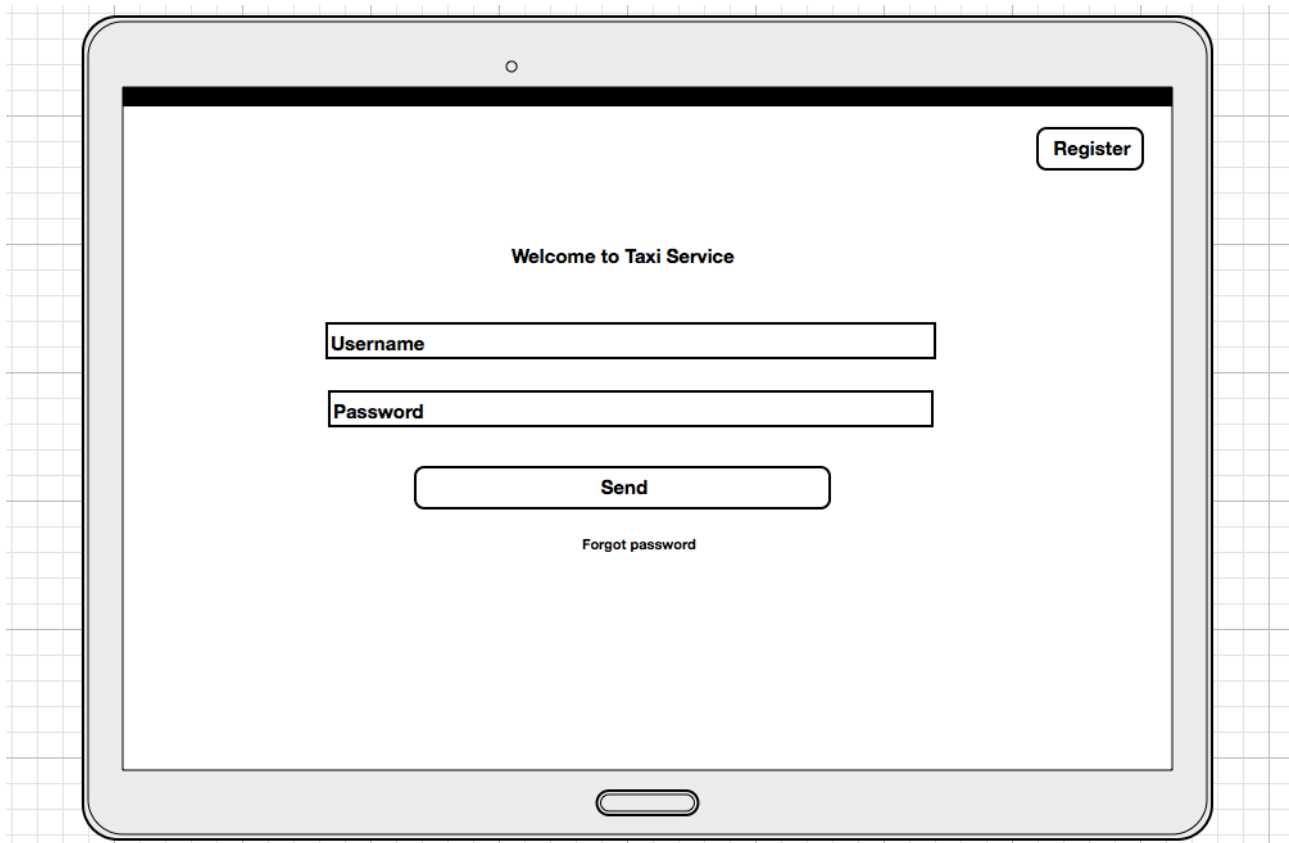
PSEUDOCODE APPLICATION SERVER SIDE

```
for ( driver in zone.queue ) {  
    driverAnswer = driver.getAvailabilityRequest(request);  
    if (driverAnswer)  
        return driver;  
}  
return NULL;
```

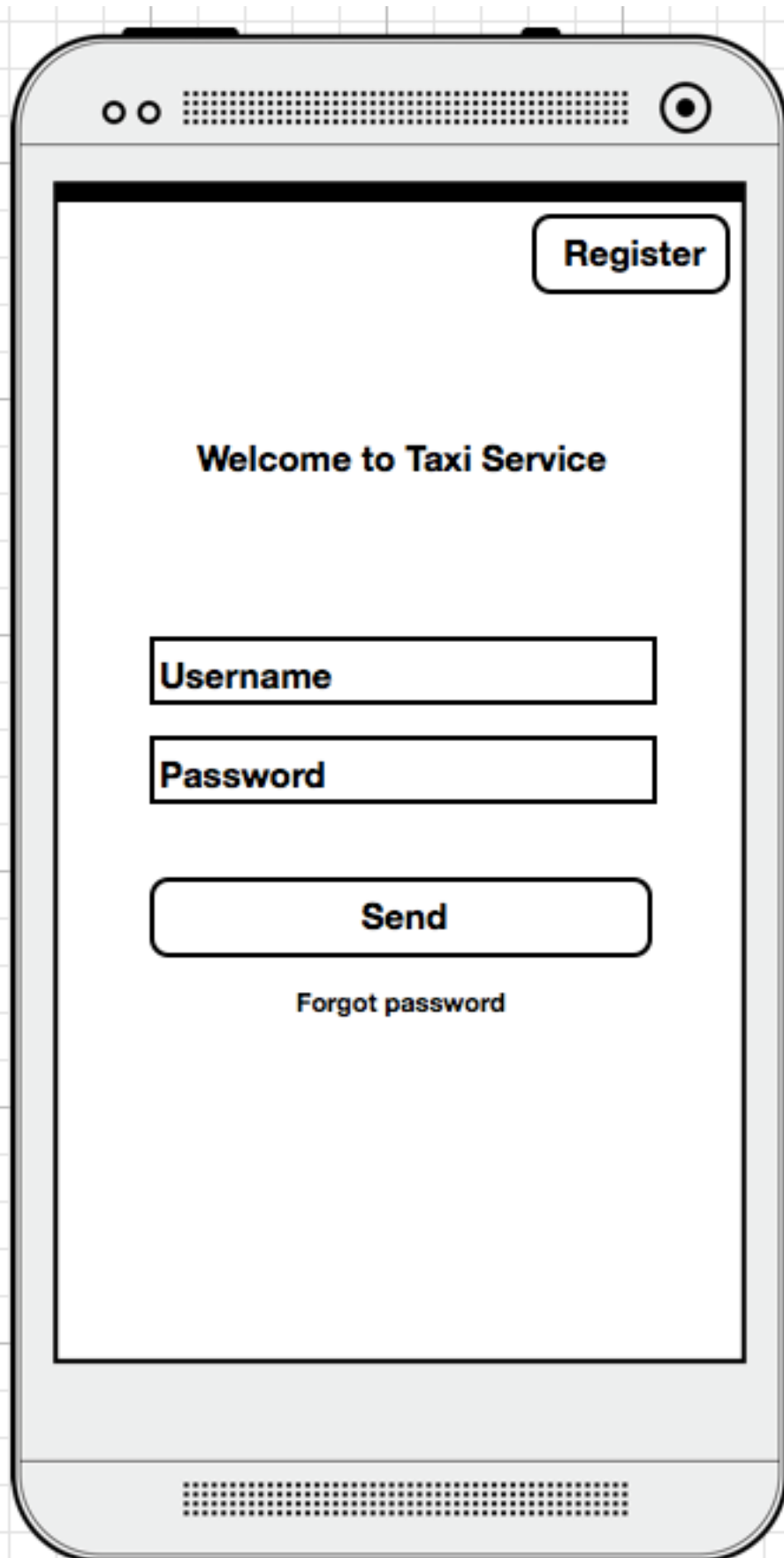
4. User Interface Design

In this section are shown the most relevant mockup of the Taxi Service System.

4.1 Login from webapp



4.2 Login from mobile app



A wireframe illustration of a mobile application login screen. The screen is set against a light gray grid background. At the top, there is a header bar with two small circles on the left, a speaker grille in the center, and a circular camera lens on the right. The main content area is white and contains the following elements: a 'Register' button in the top right corner; the text 'Welcome to Taxi Service' in the center; two input fields labeled 'Username' and 'Password' stacked vertically; a 'Send' button below the input fields; and a 'Forgot password' link centered at the bottom of the form area. The bottom of the screen features a wide speaker grille.

Register

Welcome to Taxi Service

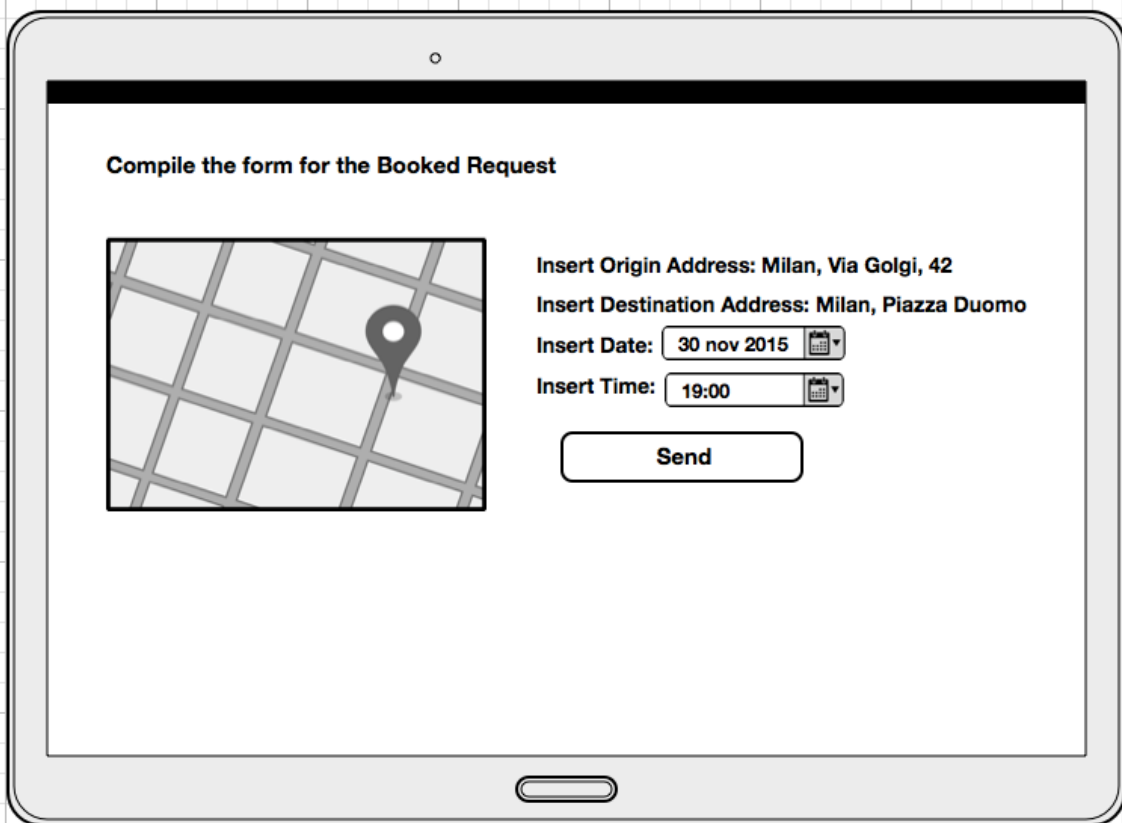
Username

Password


Send



Forgot password

4.3 Customer Booked Request from webapp



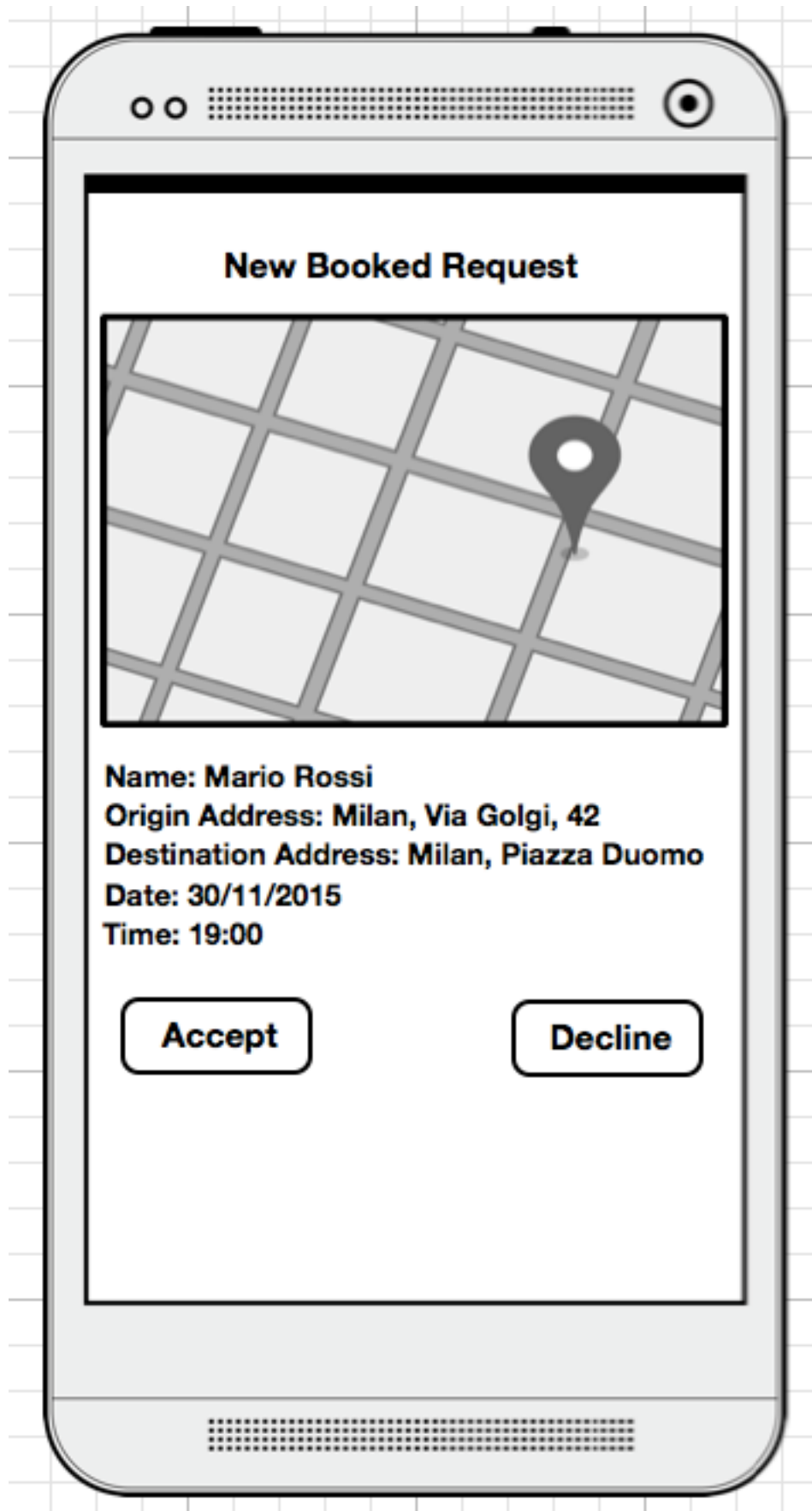
Compile the form for the Booked Request



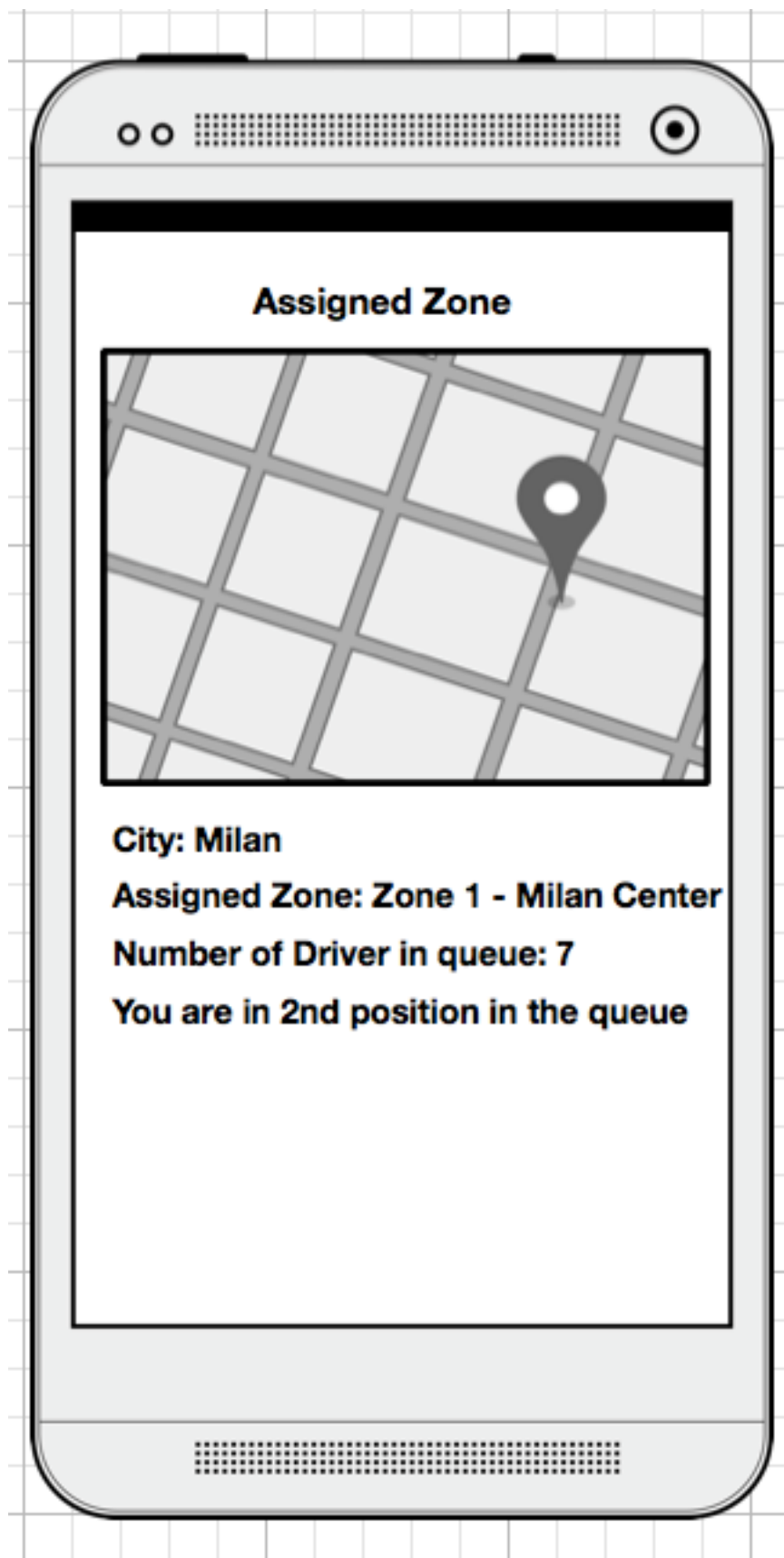
Insert Origin Address: Milan, Via Golgi, 42
Insert Destination Address: Milan, Piazza Duomo
Insert Date: 30 nov 2015 
Insert Time: 19:00 

Send

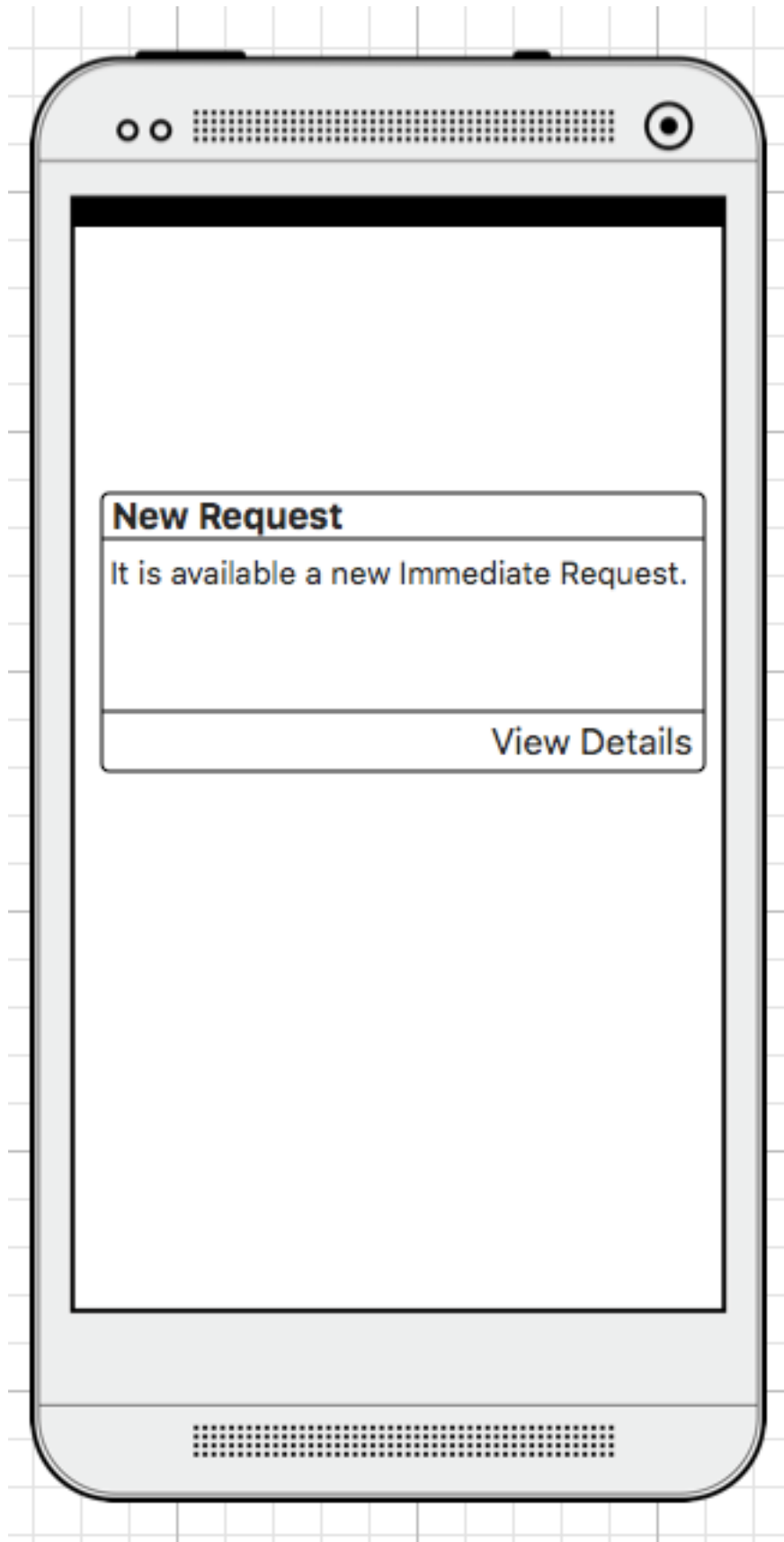
4.4 Customer Booked Request from mobile app



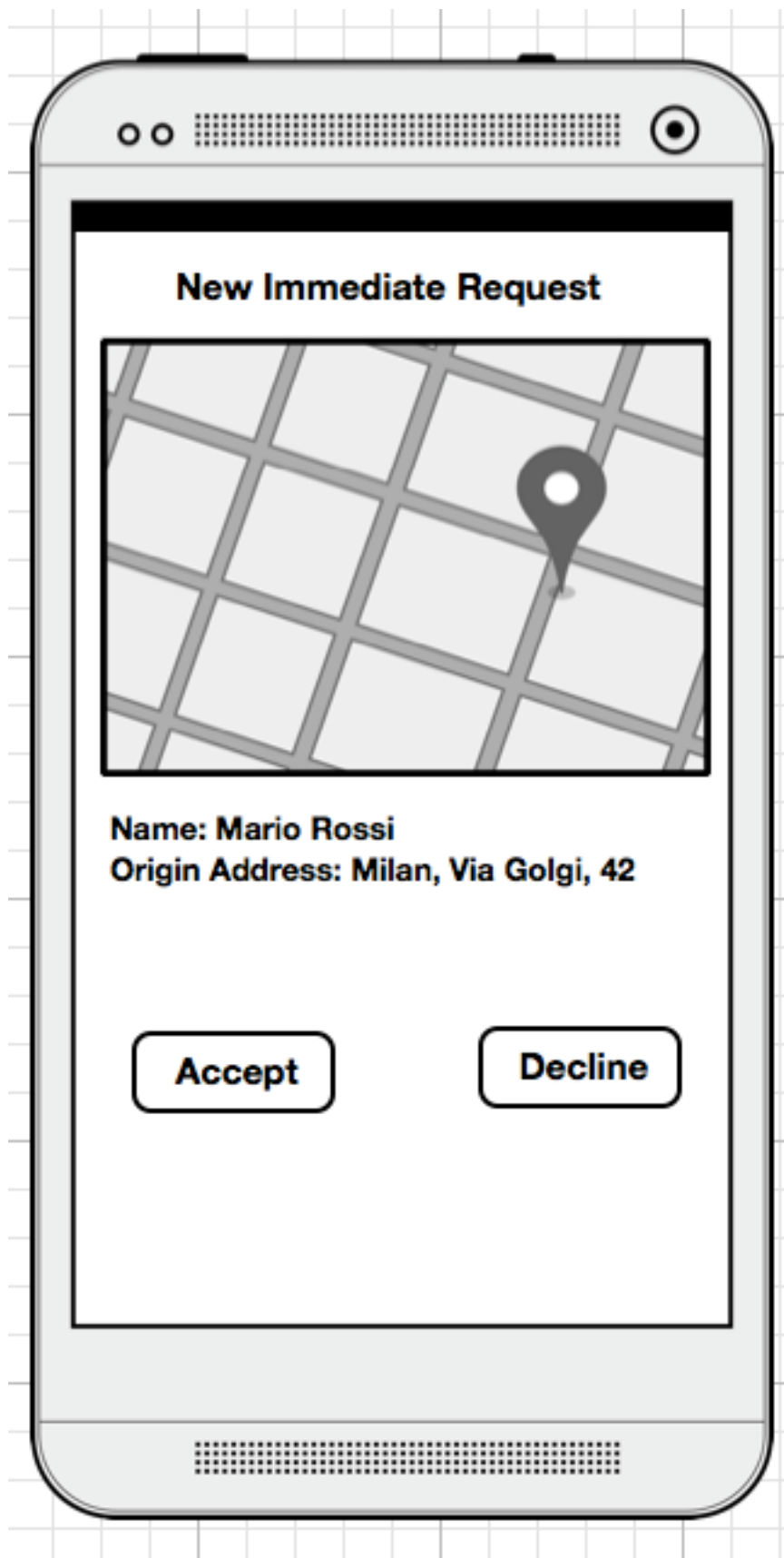
4.5 Driver see assigned zone from mobile app



4.5 Driver notification from mobile app



4.6 Customer Immediate Request from mobile app



5. Requirements traceability

In this section we'll map the design elements introduced in this document with the requirements defined in the RASD.

5.1 Allow a Guest to register and log in to the System

Requirement	Component	Description
Fill the login/registration form	UserClient	Guest can fill the registration and login form on its device
Data validation	ApplicationServer	The System check the correctness of data
Retrieve data	DatabaseServer	The System take data from the database
Data presentation	WebServer	The webserver generate the HTML response pages

5.2 Allow a Driver to accept or decline a Request

Requirement	Component	Description
Driver accept/decline the Request	DriverClient	Driver can decide to accept or not a Request on its device
Data update	ApplicationServer	The System check the response
Update data	DatabaseServer	The System update data
Data presentation	WebServer	The webserver generate the HTML response pages

5.3 Allow Driver to report its availability

Requirement	Component	Description
Driver push availability button	DriverClient	Driver send its availability to the System
Data update	ApplicationServer	The System check the response

Update data	DatabaseServer	The System update data
Data presentation	WebServer	The webserver generate the HTML response pages

5.4 Allow a Customer to perform a Request

Requirement	Component	Description
Customer make a Request	CustomerClient	Customer choose between Immediate or BookedRequest and fill the corresponding form
Request Elaboration	ApplicationServer	The System elaborate the request
Update data	DatabaseServer	The System update data
Data presentation	WebServer	The webserver generate the HTML response pages

5.5 Allow an External Developer to request API

Requirement	Component	Description
Data request through API	CustomerClient	External Developer requires data
Data Request Elaboration	ApplicationServer	The System elaborate the structured request
Retrieve data	DatabaseServer	The System retrieve data
Data response	WebServer	The webserver provide the required data

5.6 System send a notification

<i>Requirement</i>	<i>Component</i>	<i>Description</i>
Generate notification	ApplicationServer	The Application Server generate a notification for the user
Forward Notification	WebServer	The webserver receive data from Application Server and delivery it
Receive notification	DriverClient/ CustomerClient	The user receive on its device the notification