Taxi Service
Design Description

Version 1.1

Revision History

Date	Version	Description	Author
2012-11-06	0.1	Initial Draft	DSD staff
2012-11-08	0.2	Added component diagram	Leon Dragić
2012-11-08	0.3	Chapter 1 Draft	Jelena Jerat
2012-11-08	0.4	Chapter 2 Draft	Leon Dragić
2012-11-09	0.5	Chapter 3 Draft	Marko Coha
2012-11-09	0.6	Chapter 4 Draft	Igor Piljić
2012-11-09	0.7	Chapter 4 Draft	Karlo Zanki
2012-11-09	1.0	First Version finalized	Luca Zangari
2012-11-14	1.1	Updated domain model and protocols	Marko Coha

Table of Contents

1. Introduction

1.1 Purpose of this document

This document describes the system architecture and design of Taxi service system. It is initially defined at the beginning of the project, after collecting all necessary requirements and discussing the system architecture in general. However, it is intended to be constantly updated and revised, as new features are defined and implemented. The development process of this project is iterative and therefore the system architecture changes and develops continually. This document will always reflect the current state of the system architecture, and every delivered item (prototype or final product) will be based on this document.

1.2 Intended Audience

This document is intended to be used by all team members during the system development. It will also be used by the project supervisor to get the insight into the project work. This is a technical document and it is not initially intended for the customers. However, technically educated customers may also get the general overview of the project using this document. Finally, this document is intended for the developers who will be able to get the detailed description of the system architecture in order to continue working on this project in future.

1.3 Scope

The Taxi Service system is divided into three main components: Taxi Mobile Application, Customer Mobile Application and Server. This document will describe interfaces between components, as well as the interfaces to external components and users. It will also describe every component separately.

1.4 Definitions and acronyms

1.4.1 Definitions

Keyword	Definitions

1.4.2 Acronyms and abbreviations

Acronym or abbreviation	Definitions
MVC	Model-View-Controller design pattern
HTTP	HyperText Transfer Protocol
ORM	Object Relational Mapping
API	Application Programming Interface
JSON	JavaScript Object Notation

1.5 References

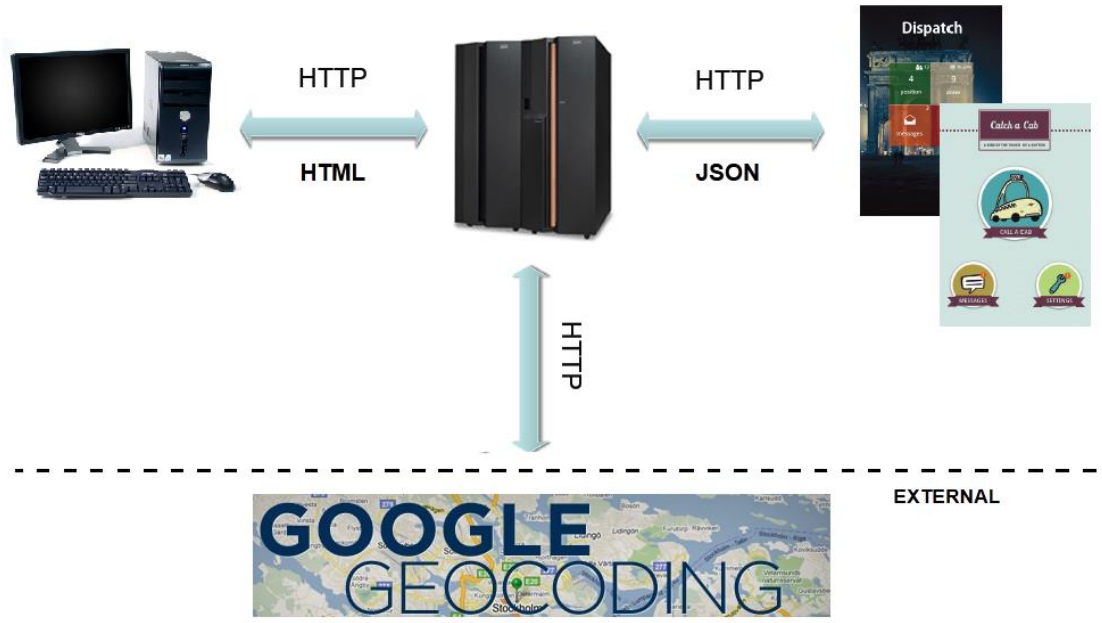
Project homepage: http://www.fer.unizg.hr/rasip/dsd/projects/taxi_service

2. Software architecture

2.1 Conceptual design

The system consists of 3 main components: Taxi client application, Main server and Customer client application. Main server provides the core system functionality. Because of the specific system requirements (Android clients and web users), the main server is built using MVC design pattern. MVC enables us to easily transform the output of methods, meaning it is possible to return either HTML, or JSON string to the users, depending on the request they made.

System Design



The main server and the database are deployed on the cloud which has many real benefits. It is easily configurable, no need for maintenance, it increases the availability of the service and gives greater freedom for the users cause data can be accessed from almost everywhere.

The application for taxis will be deployed on any Android device with Android 4.0 OS or greater and the application for customers will be deployed on any Android device with Android 2.3.3 OS or greater.

2.1.1 Main Server

Main server is the central part of the system. It provides the core functionality of the system. Its main purpose is receiving requests from clients, processing them, and sending results of processes to the client. The main resource that server needs is connection with Google Geocoding API which is an external component of the system and the connection with the database. Users of the server are taxi client, customer client and web client.

2.1.2 Taxi Client

Taxi client is making specific request to the main server in order to establish functionality of the client that is needed for taxi driver to perform his work. The taxi client gets all the information from the server. The main resource that taxi client needs is the Android device with GPS chip which provides the current location of the taxi. Users of taxi client are taxi drivers.

2.1.3 Customer Client

Customer client provides the functionality for ordering a taxi at the current position of the device. The client sends the request to the server. The main resource that customer client needs is the Android device with GPS chip which provides the current location the customer. The users are people who install the Android application on their Android device.

2.2 System specification

Server is hosted on the Windows Azure cloud. It is developed in .NET framework, using MVC design pattern.
Taxi client application is hosted on Android device with OS 4.0 or greater. It is developed in Java.
Customer client application is hosted on Android device with OS 2.3.3 or greater. It is developed in Java.
The MSSQL database is used. Entity framework is used as ORM.

2.3 External Components

The system is using Google maps for showing the specific positions on the map.

3. External interfaces

3.1 Hardware Interfaces

One of the main functions of this product is taxi tracking, i.e. using GPS to locate a taxi in any given time. Each of the taxi client applications must, therefore, be installed on an Android device with a built-in GPS. The other client application, the one used by customers, will also have to access GPS data in order to provide an accurate location of the customer ordering a taxi. If it would be required by other features, a GPS could be used for calculating the velocity and similar parameters of a taxi vehicle. The administrator application will be developed as a web application and will not require any specific hardware.

3.2 Software Interfaces

The server part of the system will provide a REST web service for client applications to use. The service will provide an interface for the core functionalities of the product, such as registering a taxi in the database, tracking a location of a taxi, taking orders from clients, etc. The interface is described in detail in the Protocols section below. It will not be open to public, but such an API could be developed in the future.

3.3 Communication Interfaces

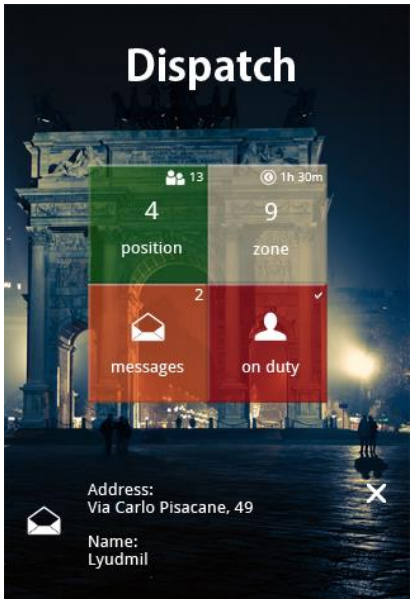
Each of the client applications will have to be connected to a 3G network at all times, in order to be able to send request to the REST service and receive results. All of the data will be sent over a 3G network. The customer client app will place orders and receive confirmations from the service, and the taxi client app will receive orders and respond to them via a 3G network.

3.4 User Interfaces

3.4.1 Customer client Android application

The Android application for customers only has the main dashboard designed so far. The dashboard consists of three buttons. The center button is used to call a taxi, the lower left button to view all messages received from the web service, and the lower right button to open the settings panel.

3.4.2 Taxi client Android application



The dashboard for the taxi client application consists of four main buttons. The first button displays the number of taxis in the queue and the position of the taxi in the queue. The second button is used to give the user information about the zone he is currently in. The messages button is used to give user access to messages from the web service, and the last button is a status button - it shows the user's current status, which can, for example, be on duty or off duty, and a checkmark showing the status of the connection between the user's Android device and the server. The lower part of the screen also shows some personal information about the user.

4. Detailed software design

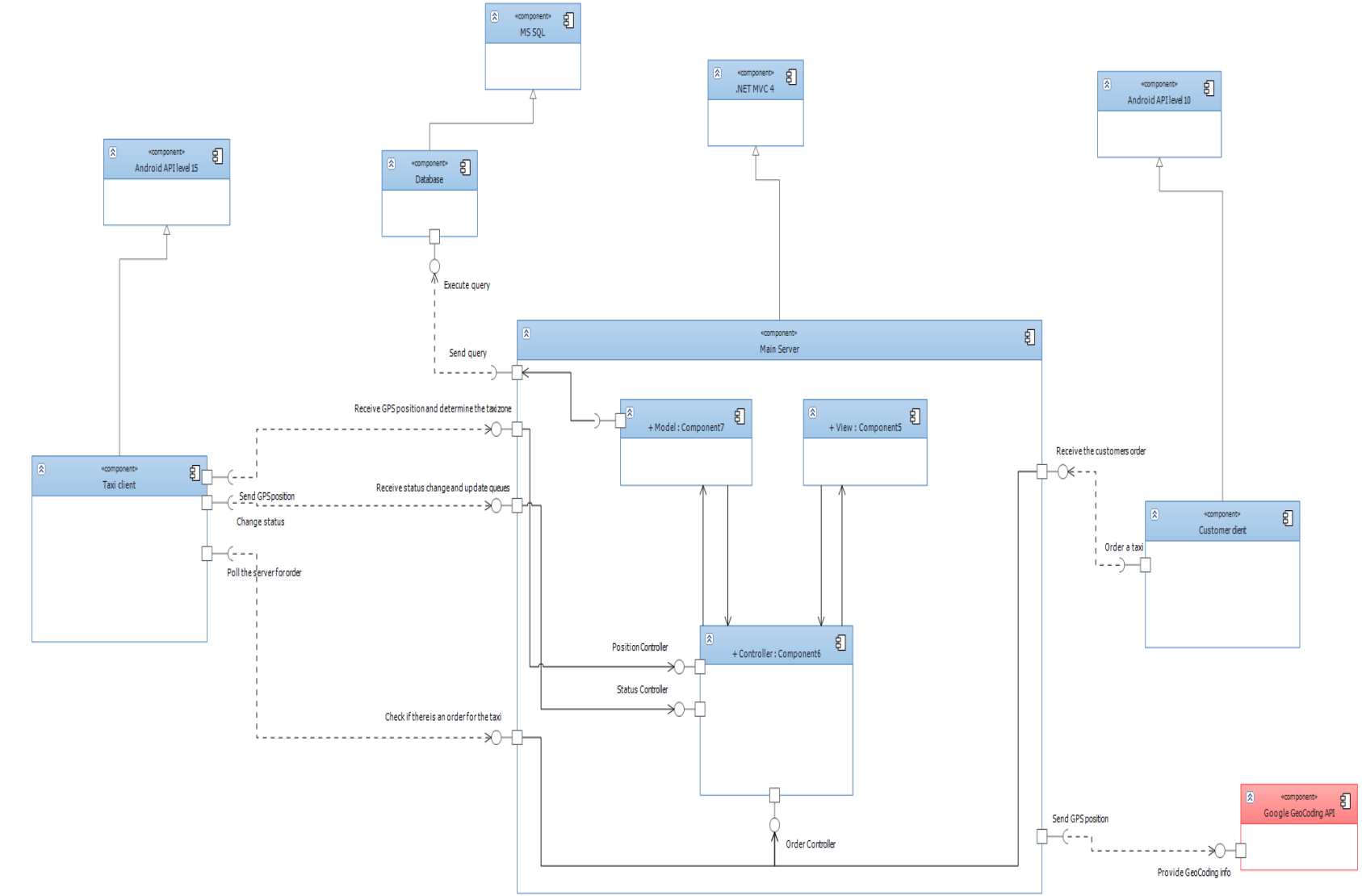
4.1 Implementation modules / components

Product contains of three main modules, Customer client application, Taxi driver's client application and a web service which coordinates the communication between these two components and merges them into a complete and unique system. Taxi driver's application tracks taxi movement, updates taxi's location and enables taxi driver to get information about potential pickup requests. Customer's application allows customers to make requests for taxi drives. Web service collects and handles the data received from Taxi driver's application and Customers application. It manages virtual queues and forwards drive requests received from customers to the taxi drivers.
This kind of deployment is natural since the product contains of two parts, one that is going to be used by the customers and the other that is going to be used by the taxi drivers. Third part, the Web service provides a way to integrate these two modules.

4.1.1 Deployment

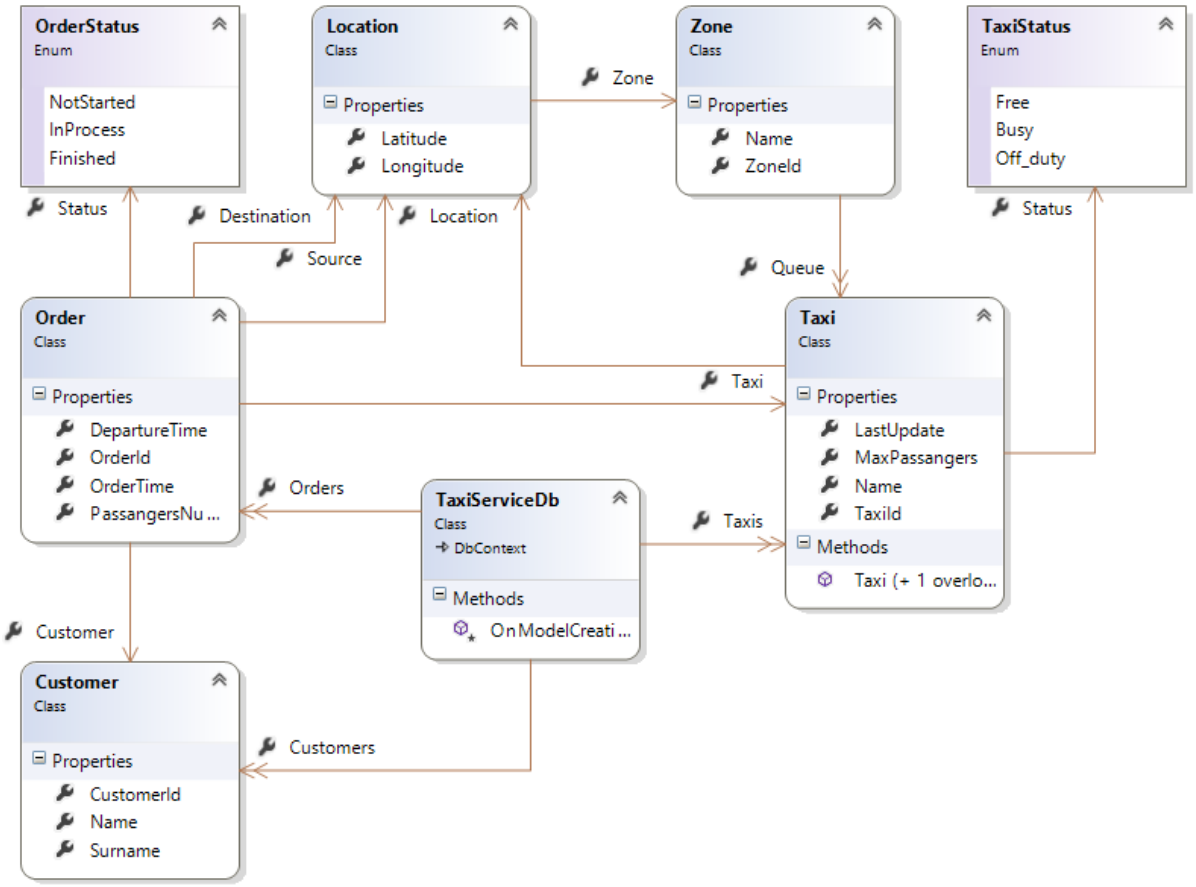
Taxi driver's application is deployed on an android device placed inside of the taxi. Customer application is deployed on customer's android devices and the Web service is hosted on the main server. Database is placed in the cloud, to be more precise, on the Microsoft's Azure platform.

cmp TaxiService



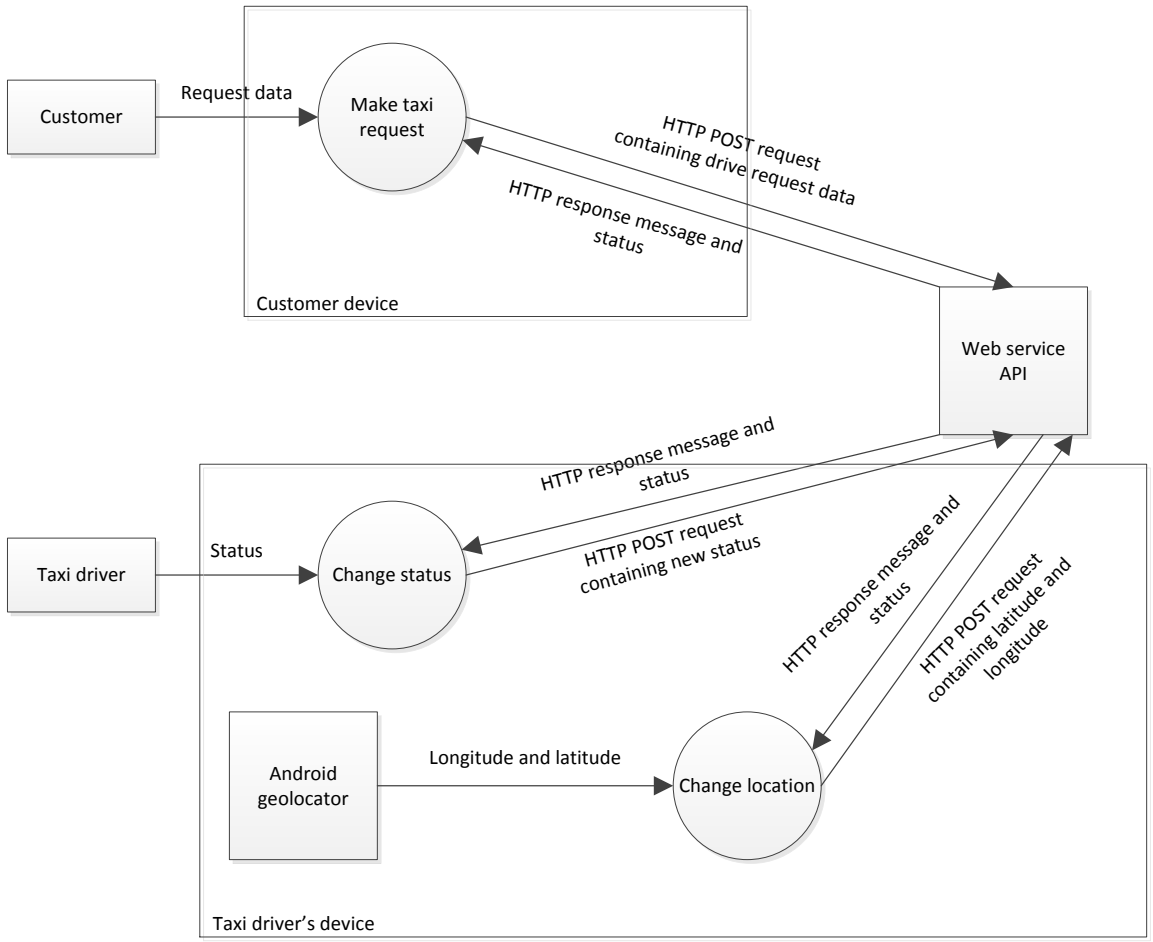
4.1.2 Domain model

Domain model describes core concepts of the product. Current domain model represents concepts needed for the first two iterations and it is supposed to grow during the project, and the concept description will become richer with the implementation of the new features.



4.2 Data flow / Interactions / Dependencies

Data inside of the system is being exchanged between the main three components, Customer application, Web service and Taxi driver's application. Taxi driver's application and Customer application communicate with the web service in the same way, by making HTTP REST requests. Each HTTP POST request contains a JSON string in its body, which is used to pass data associated with the request to the server. The communication is always one-way, from client applications to the web service which then sends back a HTTP response with a JSON string in its body containing success status and a short description.



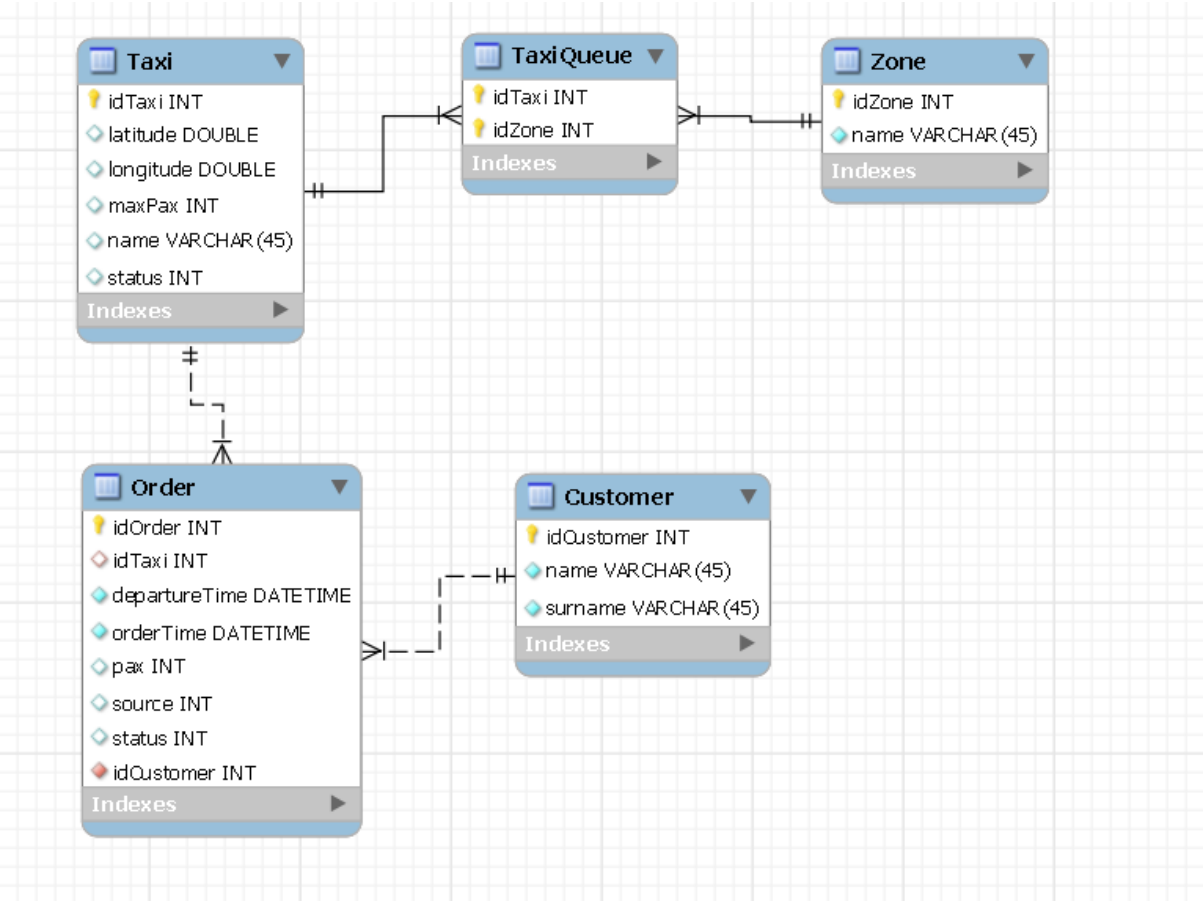
Dependency between different parts of the system is reduced by simple extraction of the interfaces. All interaction between the components is pointed towards the interface instead to the concrete implementation. This way the coupling is reduced, because it doesn't matter how the component is implemented as long as it implements the interface. This also enables separate testing of different system modules because data received through interaction can be mocked.

4.3 Data Types / Formats

Whole documentation will be written in both, .doc and .pdf formats.
For communication between server and clients team has decided to use JSON format that will be part of HTTP requests and responses. Examples of some JSON templates that were made for this communication are given in section 4.5.

4.4 Database Model

Current version of the database is designed for the first two iterations and it is not the final version, it will grow as the project moves to next iterations. This chapter will be update with new versions of database through the time.
Database stores information about taxis, their locations, customers and orders. It also stores information about city zones and virtual queues associated to them. Database entities and relations are described on the database diagram.



4.5 Protocols

HTTP is used for communication between server and clients. Every HTTP POST request has JSON string in its body and response from server is JSON string as well. Here are examples of JSON messages that are used in this communication:

- HTTP POST request for sending position from taxi client
  Url: /api/Taxis/id/Location
  {
  "Latitude":"1.2",
  "Longitude":"3.6",
  }

- Server response for HTTP POST request from taxi client in which taxi sends his position
  {
  "status":"OK",
  "zoneId":"20124" //zone in which taxi is currently in
  "zoneName":"Bovisa" //zone in which taxi is currently in
  "position":"5" //position in queue
  }
  Or:
  {
  "status":"ERROR",
  "description":"Out of Milan" //description not mandatory
  }

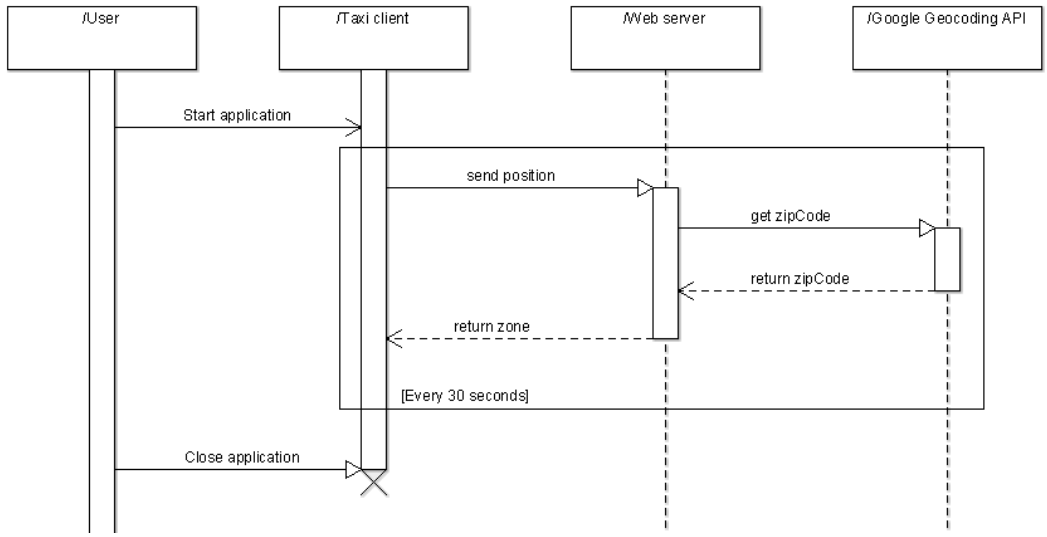
- HTTP POST request for changing taxi status
  Url: /api/Taxis/id/Status
  {
  "status":"free" //status can be “free”, “busy” or “off duty”
  }

- Server response for HTTP POST request form taxi client in which taxi changes its status
  {
  "status":"ok/error",
  "description":"Some description" // description of possible error in communication
  }

- HTTP POST request taxi client performs to ask server if there are any messages for him
  {
  "TaxiId":"17000"
  }

- Server response for previous GET request
  {"call":[{ // an array of calls info
  "From":{
  "Latitude":"12.1",
  "Longitude":"25.1",
  "StreetNumber":"15"
  "Street":"Via Pisacane"
  "City":"Milano"
  },
  "To":{
  "Latitude":"12.2",
  "Longitude":"25.15",
  "StreetNumber":"1"
  "Street":"Piazza Leonardo"
  "City":"Milano"
  },
  "People":"4" //How many people need a ride
  }]
  }

Sequential diagram that describes implementation of first two features (taxi sends its position, server determines zone) is given in the picture below.



4.6 Interfaces to External Systems

There are two external systems that our system will be interfacing to. These are:

- Google Maps
- Global Positioning System (GPS)

Maps will be embedded into client applications by using Google Maps API and JavaScript.
The classes of the Maps library offer built-in downloading, rendering, and caching of Maps tiles, as well as a variety of display options and controls.
GPS will be used to retrieve the current user location (in the mobile application). Using this location and the Google Maps API, we can localize the taxi position and calculate in which zone it is. Most mobile phones nowadays are equipped to use GPS. We can access the GPS features using the *android.location* package in the Android API.