# SE2 Design Document

Edoardo Giacomello Mattia Fontana

December 4, 2015

# Contents

# 1 Introduction

## 1.1 Purpose

This document explains the general architecture of the MyTaxiService application and is intended to describe the architectural decisions taken in the design process, and justify them. It also describes the algorithms used in the application and show the interface design that describe the application.

Note that this document does not cover all the details about implementation and technologies needed for developing this system; this is made on purpose for taking advantage of the system developers' specific capabilities.

## 1.2 Scope

A design document is a representation used for recording design information and communicating those information to the key design stakeholders. A software design description is a written description of a software product that a software designer writes in order to give a software development team overall guidance to the architecture of the software project. A design document contains an architecture diagram with pointers to detailed feature specifications of smaller pieces of the design. It is useful to connect all parts of the software, and illustrate how they will work.

## 1.3 Definitions, Acronyms, Abbreviations

**GUI** : Graphical User Interface. It is the set of graphical elements such text, buttons and images the user can interact with.

**Thin Client** : It is a design style for the client in which the application running on the client contains the least business logic possible. In our case the client application will only serve as a presentation interface.

**Waiting time** Time in minutes between the submission of a request and the arrival of a taxi.

**Request** ***(when made by the passenger to the taxi service)*** In this document we refer to a *request* when the passenger's intent is to meet the taxi as soon as possible.

**Reservation** In this document we refer to a *reservation* when the passenger's intent is to meet the taxi at a precise moment in the future

**Backend** Shorthand for the Data-Access layer. In this document is a reference to the application server, in between from the presentation layer and the data layer.

### 1.4 Reference Documents

- Structure of the Design Document

### 1.5 Document Structure

The first part of this document will focus on a description of the architecture of the system, it shows the components of the system, firstly from a higher level and then in a more detailed perspective. Then some BCE diagrams will allow to understand the operation of the system. The middle part of the document will show the algorithms that allow the system to manage the taxi drivers, the zones, and the passenger requests and reservations. The last part of the document will describe the relation between the requirements described in the RASD document and the component, the algorithms and the diagram described in the design document.

The document is thus organized in:

- **Introduction**: general description of design document.
- **Architectural Design**: description of the structure of the application, with class diagram, Bce diagram and description of the interaction between components.
- **Algorithm Design**: Shows the algorithms that are relevant to the core business.
- **User Interface Design**: shows the interface that allows user to interact with the system.
- **Requirement Traceability**: explain the relation between requirements describe in the Rasd document and the elements of the design document.
- **Reference**: reference material.

## 2 Architectural Design

### 2.1 Overview

The MyTaxiService application has been designed to run on a client-server architecture.
In particular, the general system has been divided in some sub-systems and each of those has been designed to run on a different physical machine. Since the modularity of the system has been taken into account, eventual further design decisions that could increase or decrease the tier size of the architecture will be supported.
The sub-systems that have been identified are the following:

**Client Application (1..N)** : This is the user front-end of the system. It comes in form of a web-interface or a mobile web application and it has two possible realizations, which are the Passenger Interface or the Taxi Driver interface. There's included also a realization that is currently designed only in a web interface form, which is the system administrator interface.

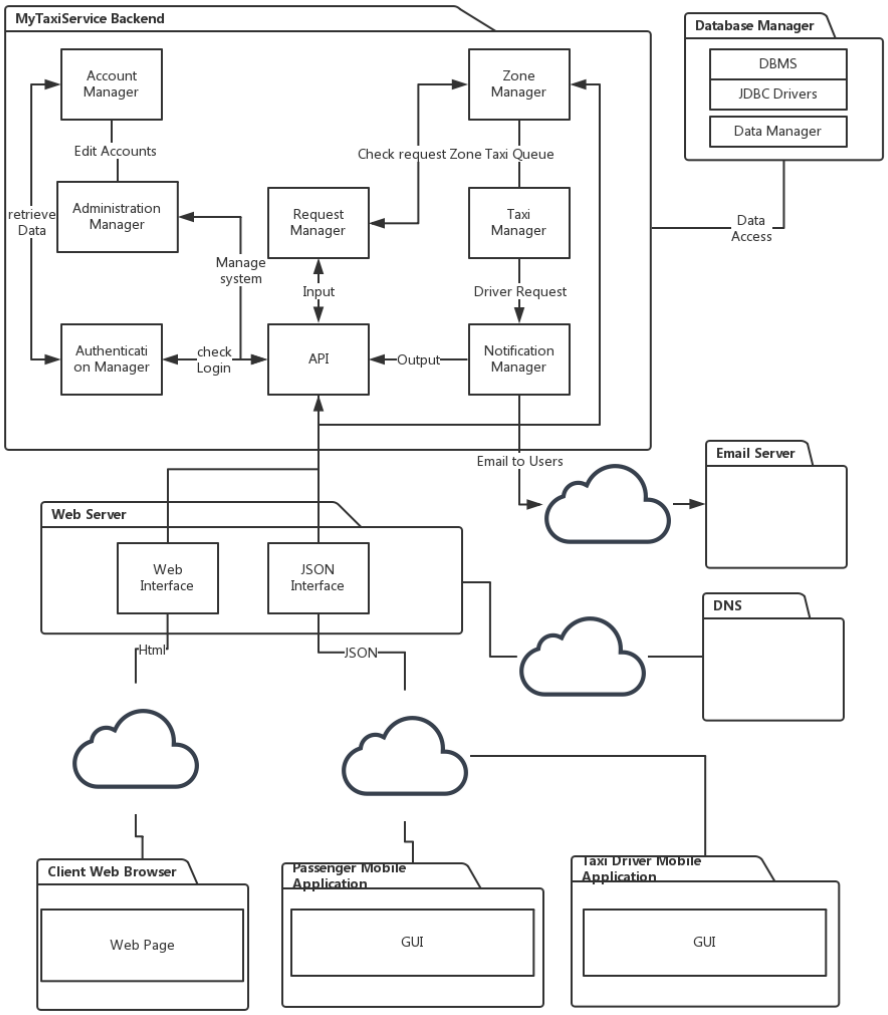
**Web Server (1..N)** : This is the main interface between the Backend and the Client application. The web-server could be distributed on several machines that are managed by a load-balancer, and should provide a firewall as well. The main purpose of this sub-system is that to generate the web-pages for the web-interface and forwarding/translating requests from the client and the backend.

**Backend (1)** : This is the sub-system in which the business logic resides.

**Database (1..N)** This is the sub-system in which the business data resides. It can be replicated or distributed for avoiding data corruption and undesired behaviours.

**Third Party Services (N)** Those sub-systems are auxiliary third-party services that the system can avail itself of for providing core services that will not be directly implemented on the system, such that email notification, DNS lookups, Map Services, etc.

## 2.2 High level components and their interaction



Here follows the high-level description of each component or module of the application, which is devided in a particular subsystem.

1. **Client Application**
   (a) **Website** This is the web interface the passengers and administrators can access from a web browser. It will make use of AJAX for fetching data from the server.
   (b) **GUI** These are the user interfaces for passenger or taxi drivers.
   (c) **Mobile Client** This is a client for making requests to the Server and retreiving results
2. **Web Server**
   (a) **Browsing Request Interface** This component hosts the web interface and generates web pages for displaying results to the user
   (b) **Resource Request Interface** This component collects all the requests from user web and mobile interfaces and redirect them to the server APIs.
3. **DataBase Manager**
   (a) **DBMS** This component is the actual DBMS which manages the business data.
   (b) **JDBC Drivers** Drivers that provide an interface between the DBMS and the application.
   (c) **Data Manager** This component will host all the functions useful for managing data on the server and it will offer an interface of the database for the backend. In particular, the SQL and DDL queries are stored here.
4. **Backend**
   (a) **API** This is the main interface for the backend and will provide all the system functionalities that can be accessed by other subsystems. It will parse all incoming JSON requests and activate the other components accordingly. It will also support an access for system developers, that will be regulated by an authorization token mechanism.
   (b) **Authentication Manager** This component will be in charge of managing all the authentications for the system. In particular it will manage the login functionality and the authorization token for accessing the APIs.

(c) **Account Manager** This component will manage the user accounts and the password reset service.

(d) **Request Manager** This component will host the core algorithm which manages the taxi queues and will process the requests and reservations.

(e) **Taxi Manager** This component will manage the distribution of the active taxis and the notification of incoming requests by interacting with the Notification Manager.

(f) **Notification Manager** This component will send notifications to the users, such as email for the passengers and notifications for the taxi drivers.

(g) **Zone Manager** This component will manage the zones in which the area is divided. It will also make the look-up for a location into a zone and it will offer map utility functions.

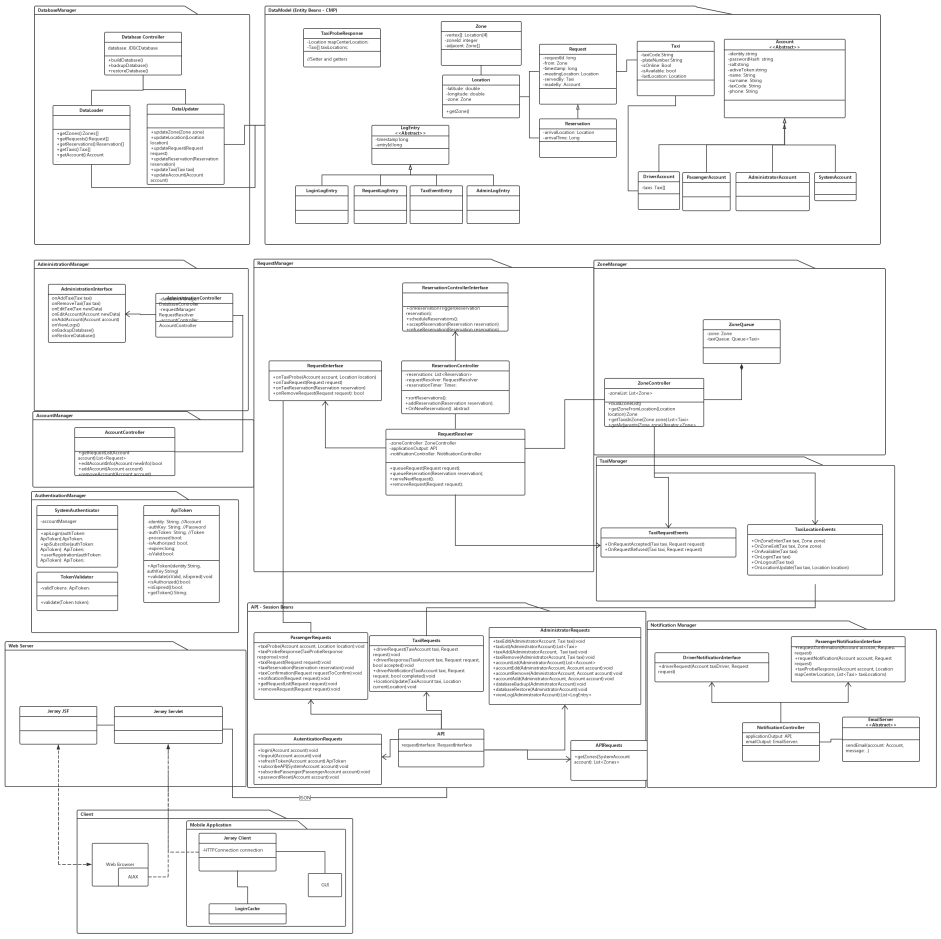
(h) **Log Manager** This component will manage all the system logs, both debugging and business inspection.

5. **Third-Party Services**

(a) **DNS** This service will offer Domain Name Lookup for accessing the various subsystems.

(b) **Email Server** This service will allow the unidirectional communication between MyTaxiService and the users.

A more in-depth view of the system is presented in this diagram:

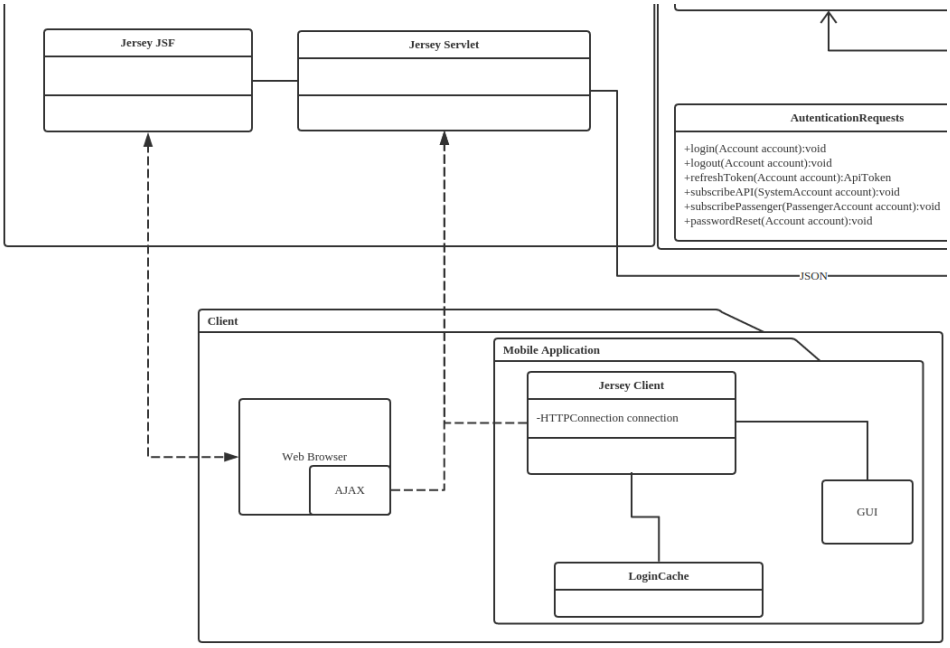


## 2.3 Component View

In this section a more detailed description of each component will be presented

### 2.3.1 Client Application

This is the client application for the passenger or the taxi manager. It consists on the following main components:



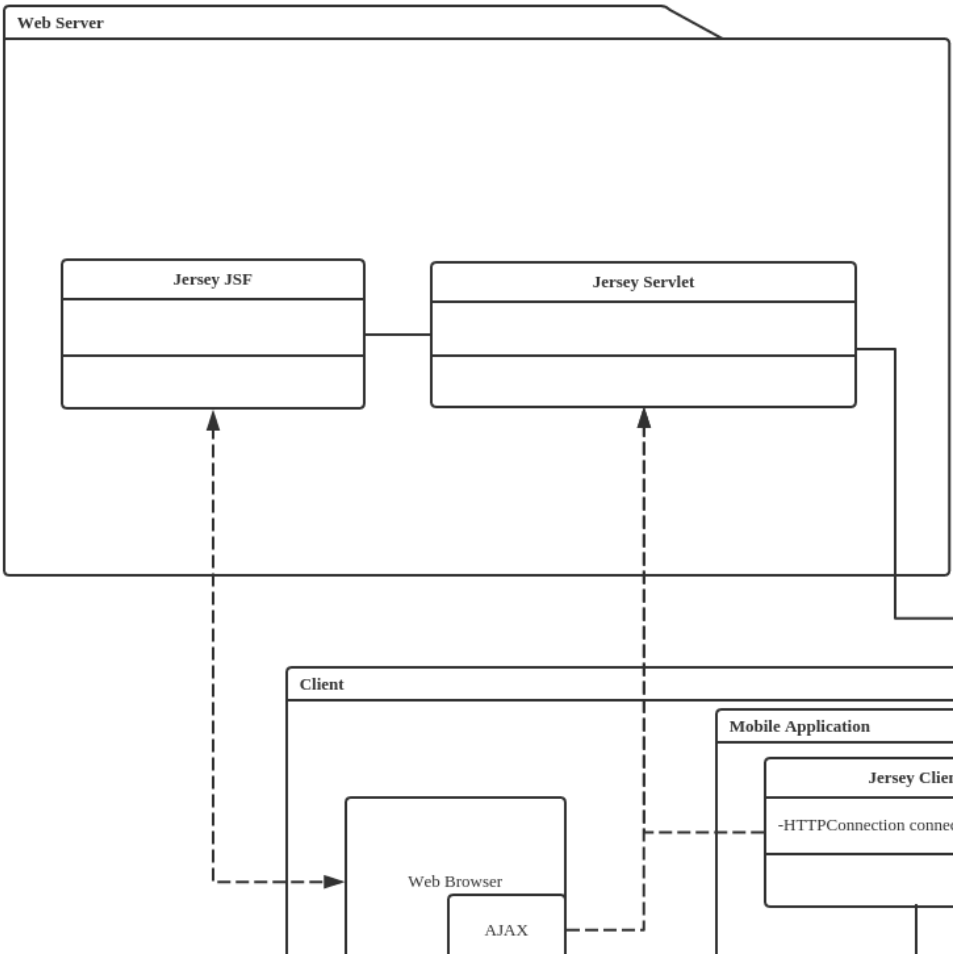

**GUI** It is the graphical user interface as described in the "User Interface Design" section.

**RequestBuilder** : It is the class that takes inputs from the GUI and builds requests for the server or, vice-versa, receives the messages from the server and display them on the GUI

**LoginCache** : This class will temporary store the user login information and the authorization token, for avoiding to request login data to the user as much as possible. For the implementation it will possible to rely on specific OS functions (as those offered by Android).

**Client Browser** This is the component for accessing the web interface.

### 2.3.2 Web Server



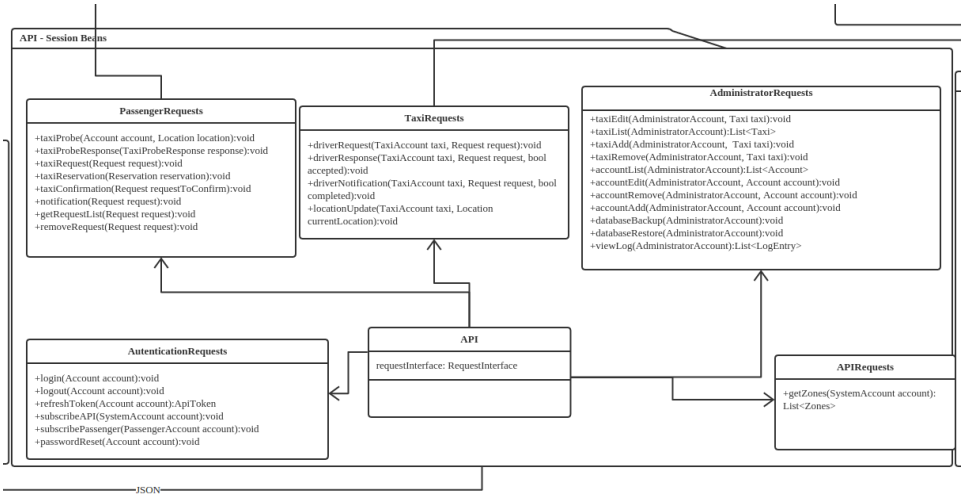

This component hosts the website along with some mechanisms for managing sessions and secure communications to the server APIs. Other than hosting the website, the purpose of this component is to avoid unauthorized requests to the server, and requesting the users a new login in the case their auth token is invalid or expired.
The authorization token is a key that is generated by the application server after a successful login and it has an expiration date, after which the client will have to login again in order to make new requests. The client must provide an active auth token along with every request it makes in order to be correctly identified by the system. This will apply both for users and systems that accesses the APIs, for further information please check the sequence diagrams section. The web browser will make use of Jersey, which is a reference implementation of JAX-RS for RESTful systems. The communications will occour using JSON, for which a jersey plugin is available.

**Jersey JSF** This component generates the web pages and provides a web

interface for the users.

**Jersey Servlet** This component manages all the incoming and outcoming requests from and to the application APIs. It both generate contents for the AJAX calls from the web browser and the mobile application.

### 2.3.3 API




This component is the main input/output interface of the application server and its main purpose is that to route request and responses to the right component of the application. Another important aspect that the implementation must cover is the masking of responses, that is the removal of sensible business data from the responses: i.e. when a user requests for the available taxi, he is supposed to recevive only the taxi code and the taxi location and not all the data which belongs to the Taxi class. Another example can be the masking of authentication data for a system administrator who requests the list of the accounts.
This class will also offer a programmatic access for system developers.

**API** This is the main API class and will implement serveral interfaces explained below. In the case a request is incoming it will translate it into Java messages and call a proper function of the application server. Otherwise, if a request is outcoming of the application server it will build the message for the client.

**AuthenticationRequests** This interface contains all the methods useful for authenticating a system or a user.

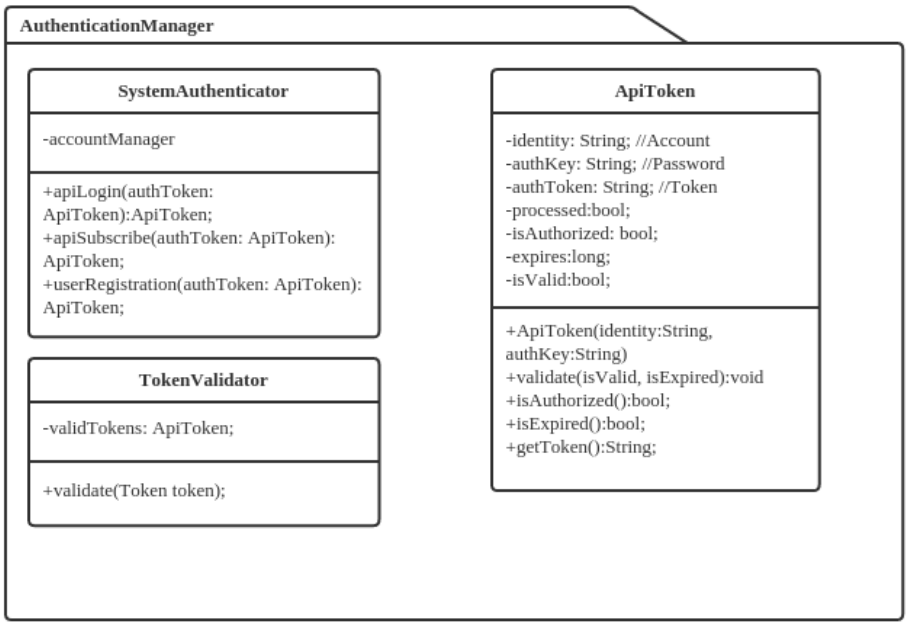
**PassengerRequests** This interface contains all the methods useful for managing the requests from/to the passengers.

**TaxiRequests** This interface contains all the methods useful for communicating with the taxi driver application

**AdministrationRequests** This interface contains all the methods useful for system administrators

**APIRequests** This interface will contain all the methods useful specifically for the system developers.

### 2.3.4 Authentication Manager



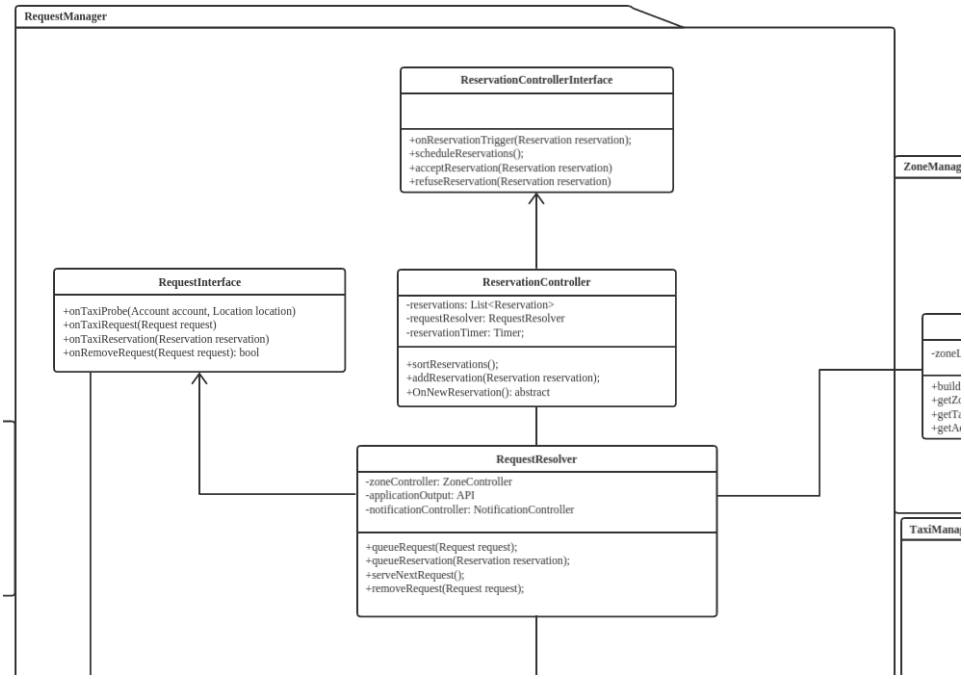

This component will check if a system or user is making a legitimate login request, and eventually generate and assign a new AuthToken.

**SystemAuthenticator** This class offers the main methods for logging in or registering a new user or a new dependant system.

**ApiToken** This class represents ad authorization token which is generated or fetched from the database. It will also offer methods for checking the token validity and permissions to a user.

**TokenValidator** This class offers functionalities for checking if a request is legitimate given an Auth token.

### 2.3.5 Request Manager




This component hosts the core of the application, that is the management of taxi, zones and requests.

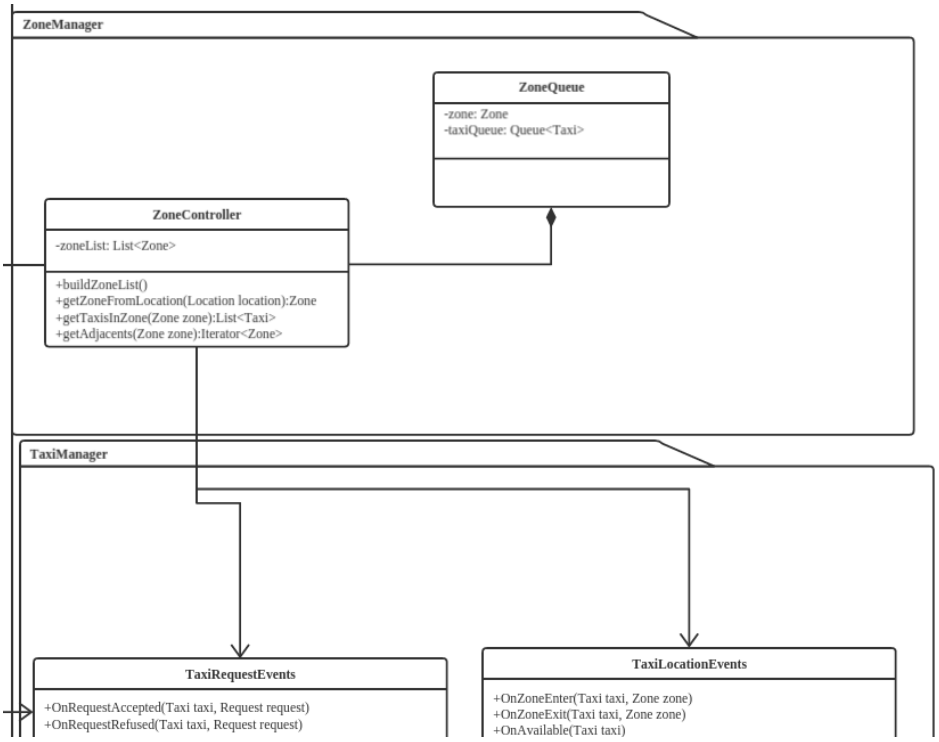
**RequestResolver** This class implements the core algorithm for managing incoming requests. It will run the core algorithm for managing an assigning taxi to requests

**ReservationController** This class will trigger pending reservations that are stored in the database at the right time. When a reservation has to be triggered, it will be pushed in the request queue in order to be processed.

**RequestInterface** This interface contains methods for managing the incoming requests from the users, such as creation and deletion of requests and taxi availability requests.

**ReservationControllerInterface** This interface contains all the methods that the reservation controller has to implement in order to manage reservations.

### 2.3.6 Zone Manger

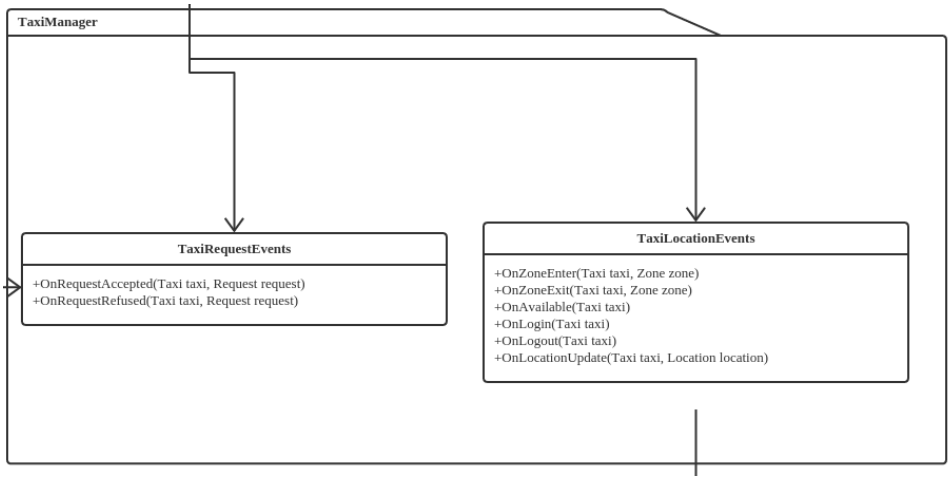



This component is a container for the definition of the zones in which the city is subdivided.

**ZoneController** This class will contain all the zones specified for the city and will implement functionalities for managing the tracking of taxis and accessing their state and location

**ZoneQueue** This is the actual taxi queue associated to a zone

### 2.3.7 Taxi Manager

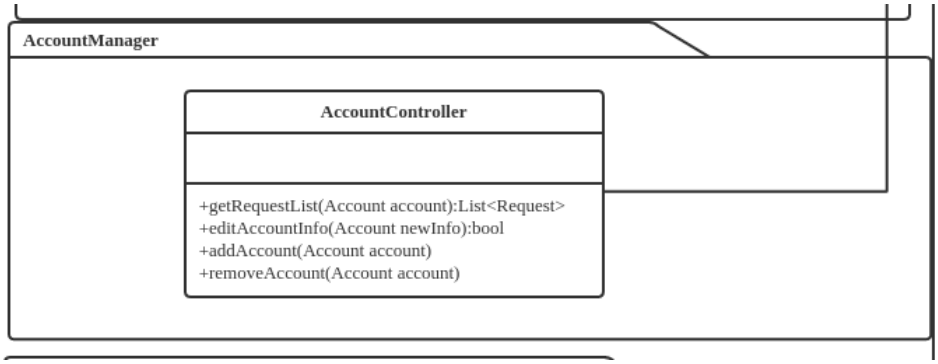



This component is a set of interfaces that will have to be implemented in order to manage the taxis

**TaxiRequestEvents** This interface contains methods for managing the acceptance or refusal of requests made by the taxi drivers.

**TaxiLocationEvents** This interface contains all the events that belongs to a taxi object, such as login/logout, moving from a zone to another, location updates and availability state changes.

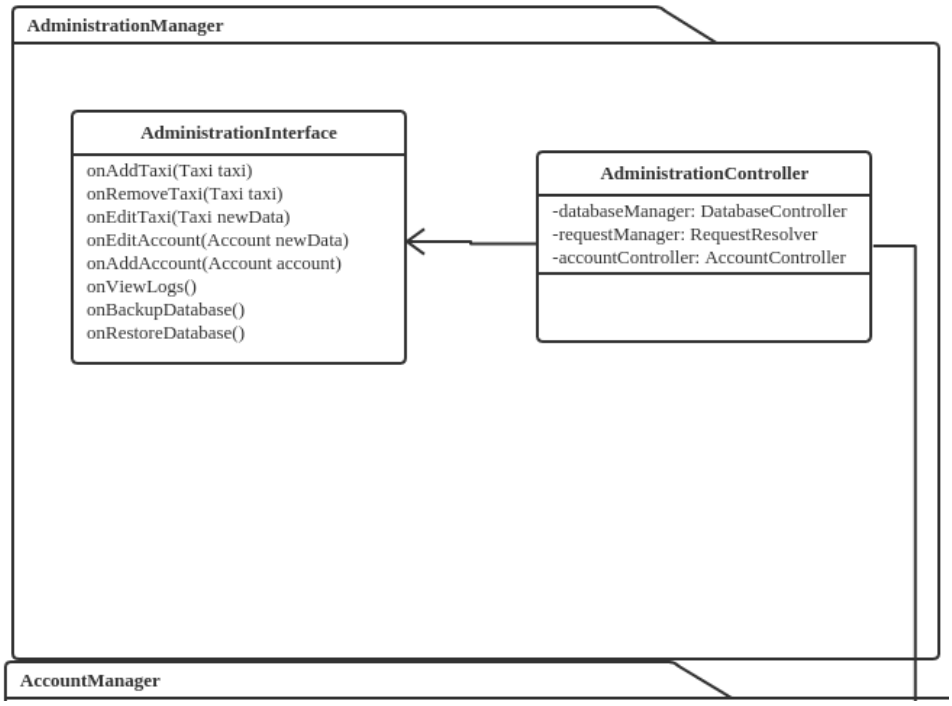
### 2.3.8 Account Manager




This component provides functionalities for managing and accessing stored account data

**AccountController** This class will provide functionalities for creating, accessing, deleting or modifying a user or system account

### 2.3.9 Administration Manager

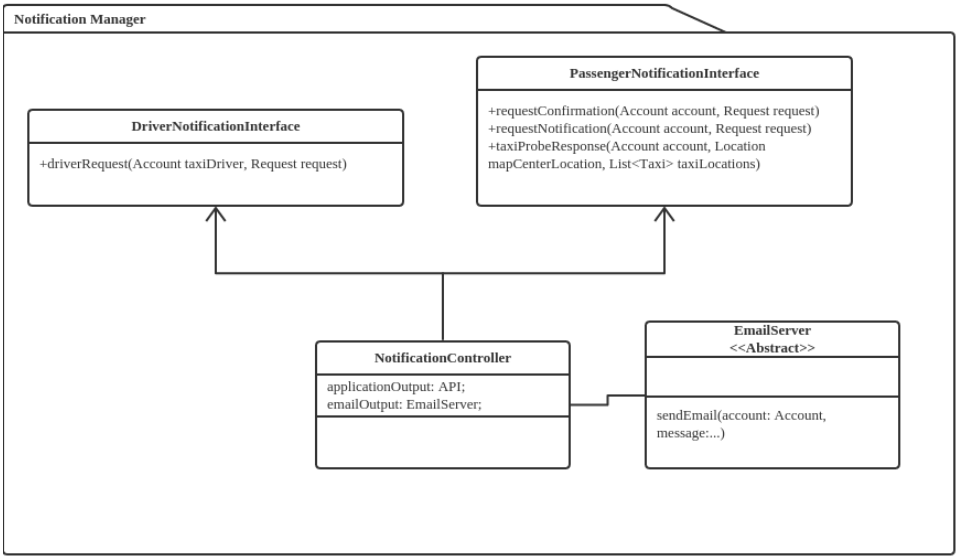



This component will offer an interface for the system administrators

**AdministrationController** This class will offer access to all the components that the system administrator can interact to

**AdministrationInterface** This interface contains all the methods for the system administrator, such taxi management, account management, log access, database manitenance

### 2.3.10 Notification Manager




This component will manage all outgoing notifications for the users

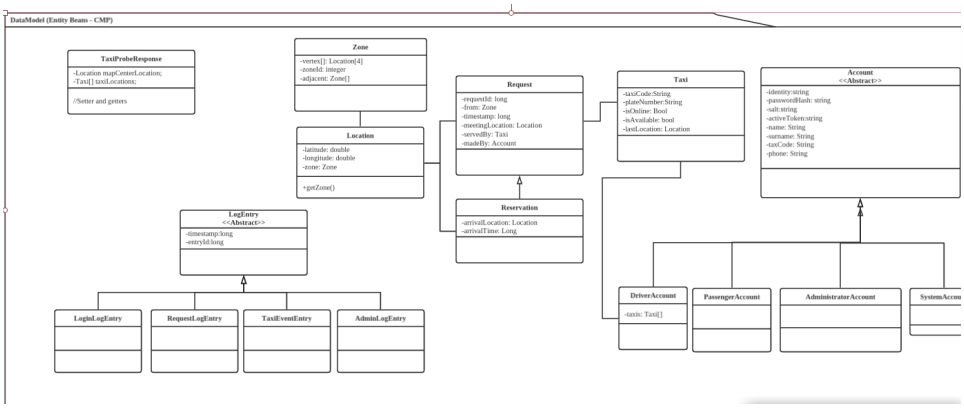
**NotificationController** This is the controller for managing all the application output.

**DriverNotificationInterface** This interface contains methods for issuing notifications to the driver application

**PassengerNotificationInterface** This interface contains methods for issuing notifications to the passenger application

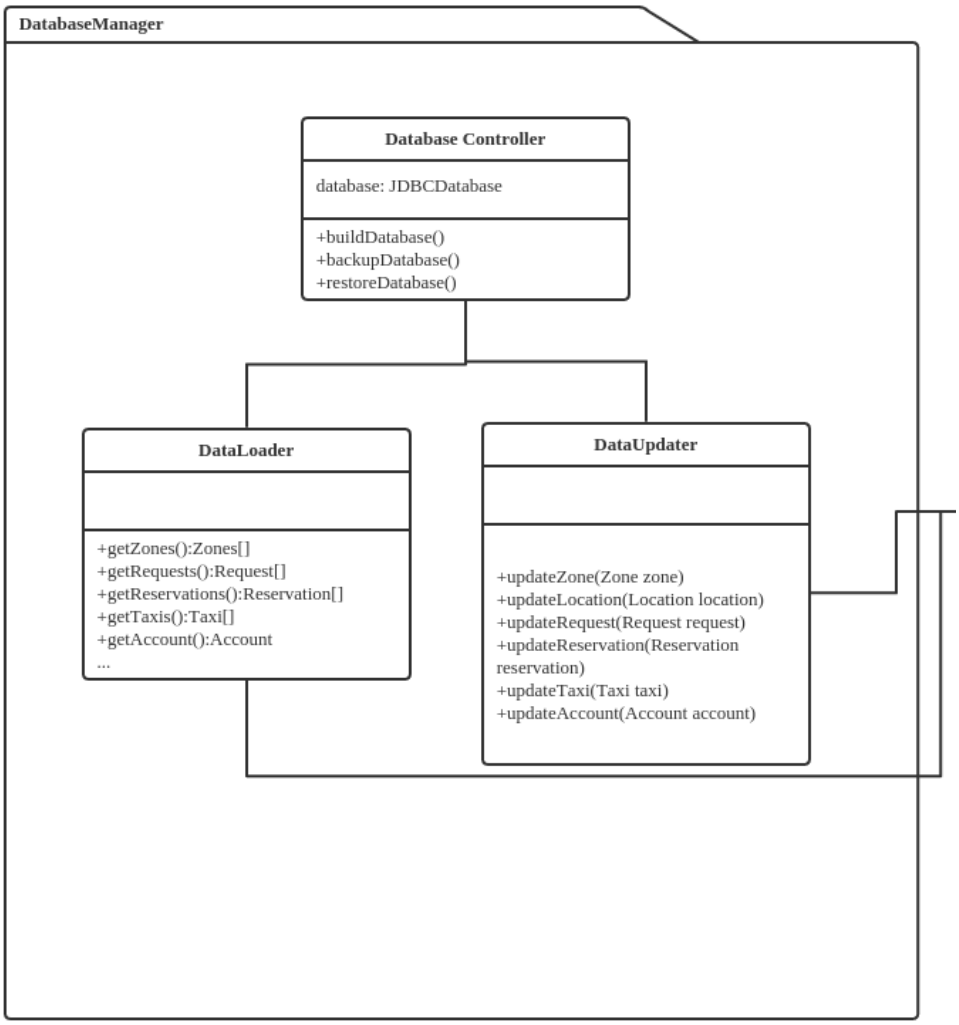
**EmailServer** This is an abstract object representing the email service in use with this application. The implementation will depend on further design choices.

### 2.3.11 Data Model




This component is the actual data model of this application. The classes specified in this component will reflect the database data schema. It will also contain some support data models such as the TaxiProbeResponse, which are not meant to be stored on the server but only generated and issued to the client application.

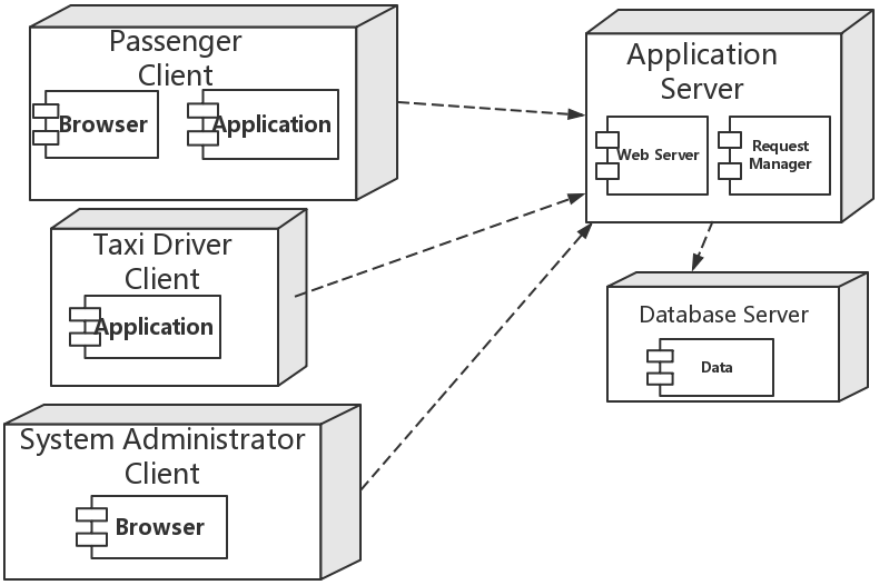
### 2.3.12 Database Manager




This component will manage the data access and data persistance. The implementation will be compliant to the J2EE specifications for data persistance.

## 2.4 Deployment View

The system is meant to be deployed in a 3-tier architecture, which consist in serveral clients (both web and mobile applications), a web-server hosted on the same machine of the application server and a database hosted on a dedicated machine.




## 2.5 Runtime View

### 2.5.1 Overview

In this section, a different point of view of the system is given. In particular, while the class diagram in the previous section will give a more implementation-oriented perspective, we now present a BCE for describing the system in a more abstract way, highlighting the user inputs and outputs and the system operations. Every concept described in the BCE can be mapped to one or more classes explained above, even if this relation is kept implicit. We lastly present some sequence diagrams relative to the BCE section.

### 2.5.2 BCE

For describing the BCE we used a Class Diagram with appropriate stereotypes <<boundary>>, <<entity>>and <<control>>. This is the function of the stereotypes:

1. **boundary** : represents the interface between users and system;
2. **entity** : represents a class that allows to access the data in a database;
3. **control** : controls the choice of the user.

### 2.5.3 BCE Passenger

1. **Login**
   (a) **CheckRegistration** This controller manages the insertion of a new user and it controls if the mandatory fields are correctly filled. If the fields are filled in in a proper way, it stores the data in the database and creates a new user.
   (b) **CheckLoginInformation** This controller manages the authentication: it examines if the fields are filled correctly, it examines the password and the email. If the fields are correctly, the controller can load the correspondent home page.
   (c) **ResetPassword** this controller manages the reset of the password in the case the user forgot his one. It modifies the user information in the database with a new password that is generated by the system and sent to the user via email.
2. **ControlTaxi**
   (a) **getPassengerGPSLocation** This controller manages the position of the passenger and it shows the taxis that are close to the passenger position;
   (b) **getAddressAndZone** This controller manages the address and the zone where the passenger is.
   (c) **checkAvailabilityTaxi** This controller manages the taxi availability, it checks the taxi position and the passenger position for research the best solution. It shows the user the available taxis.
   (d) **updateAvailabilityTaxi** This controller updates the taxi position and availability, for a example when a taxi becomes free.
   (e) **updateMap** This controller updates the position of the taxi on the map.
3. **ControlChoice**
   (a) **taxiReservation** This controller manages the passenger reservation, it controls if the information that the passenger inserts are correctly, it controls if the hours is possible, if the street exist; after it controls that the reservation is done 2 hours before the meeting time.
   (b) **taxiRequest** This controller manages the passenger request, it stores the request and send it to the first free taxi in the queue;
   (c) **updateAccount** This controller updates the passenger information and checks if the information that the passenger inserts are correctly.

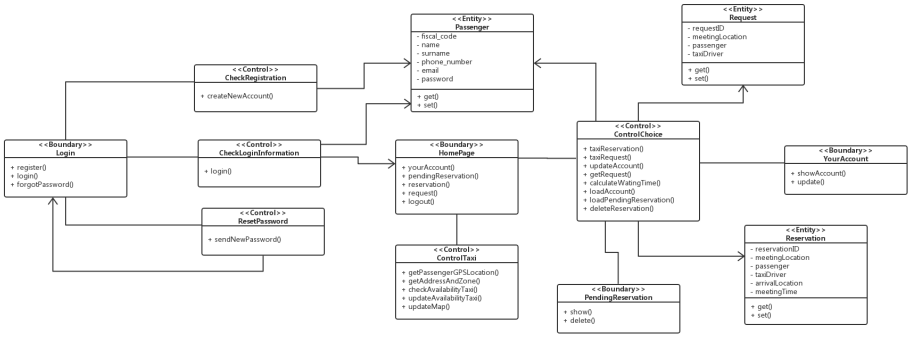
(d) **getRequest** This controller informs the passenger about his request.

(e) **calculateWaitingTime** This controller manages the waiting time that the passenger must wait for a taxi.

(f) **loadAccount** This controller manages the passenger account; it shows to the passenger the information that he required.

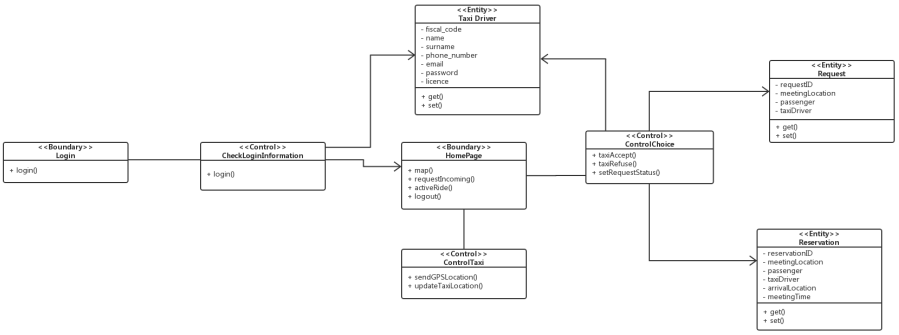
(g) **loadPendingReservation** This controller manages the passenger reservation and request; it shows the reservation or the request that the passenger will see.

(h) **deleteReservation** This controller deletes the reservation that the passenger decides to remove; it deletes the reservation from the queue of reservations.



### 2.5.4 BCE Taxi Driver

1. **Login**
2. **Login**
   (a) **CheckLoginInformation** This controller manages the authentication: it examines if the fields are filled correctly, it examines the password and the email. If the fields are correctly, the controller can load the correspondent home page.
3. **ControlTaxi**
   (a) **sendGPSLocation** This controller manages the GPS position from the taxi and it uses this position for defining if the taxi can reply to a request.
   (b) **updateTaxiLocation** This controller updates the position of the taxi on the map.
4. **ControlChoice**
   (a) **taxiAccept**This controller manages the reply of the taxi driver when he accepts a request; it sends a reply to the passenger about the taxi code and the waiting time.
   (b) **taxiRefuse**This controller manages the reply of the taxi driver when he refuses a request; it sends a request to another taxi driver;
   (c) **setRequestStatus**This controller manages the request; if the request is accepted, it delete the request from the queue of request and updates its status from free to accepted and deletes taxi from the taxi queue; if the request isnt accepted, it reinsert taxi in the queue.



### 2.5.5 BCE System Administrator

The description is divided into several sections:

1. **Login**

   (a) **CheckLoginInformation**This controller manages the authentication: it examines if the fields are filled correctly, it examines the password and the email. If the fields are correctly, the controller can load the correspondent home page.

   2)ControlTaxi :

   (a) **taxiManagement**This controller manages the taxi; it can delete, add and update the information about the taxi in the database. If the insertion or update fails, this controller must show what went wrong.

   (b) **accountManagement**This controller manages the account of the user, it can add, delete or update the information. If the insertion or update fails, this controller must show what went wrong.

   (c) **logs**This controller manages all information history about login, request, reservation and the taxi ride.

   (d) **zone**This controller manages the different zone in the city; it can add, delete or update a zone. This controller must report if a zone is not correct or already exist.

   (e) **backup**This controller stores all information about the system; This controller must report if a backup doesnt come to a successful conclusion.

   (f) **restore**This controller restores the information stored with a previous backup. This controller must report if a restore doesnt come to a successful conclusion.

2. **ControlChoice**

   (a) **updateUserAccount**This controller manages the account of the user, it can update the information about the user.

   (b) **deleteUserAccount**This controller manages the account of the user, it can delete the information about the user or delete the user account if the user does this request.

   (c) **addUserAccount**This controller manages the account of the user, it can add the information about the user or add new user account.

(d) **resetPasswordUser**This controller manages the send of new password with email to a user if he requires a new password. It modifies the user information in the database with a new password that it has generated.

(e) **viewLogs**This controller manages all information about login, request, reservation and taxi ride and it shows the requested information.

(f) **downloadLogs**This controller manages the download of the logs.

(g) **saveData**This controller saves all data for a backup.

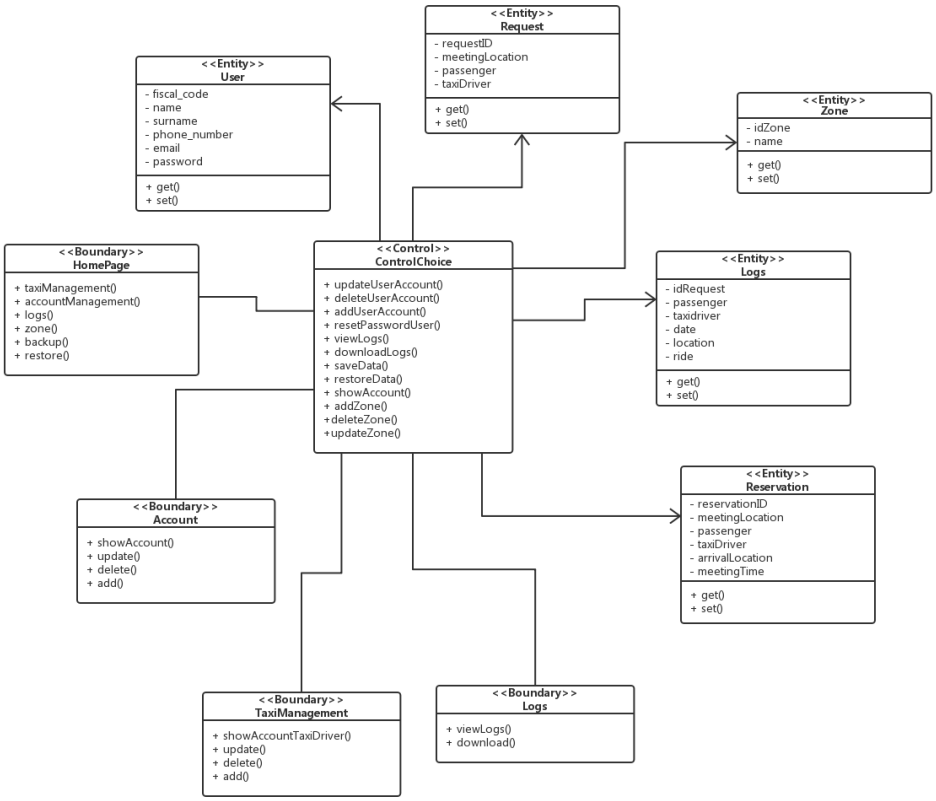
(h) **restoreData**This controller extract data from a previous backup and reload it.

(i) **showAccount**This controller manages all information about user and it shows the requested information.

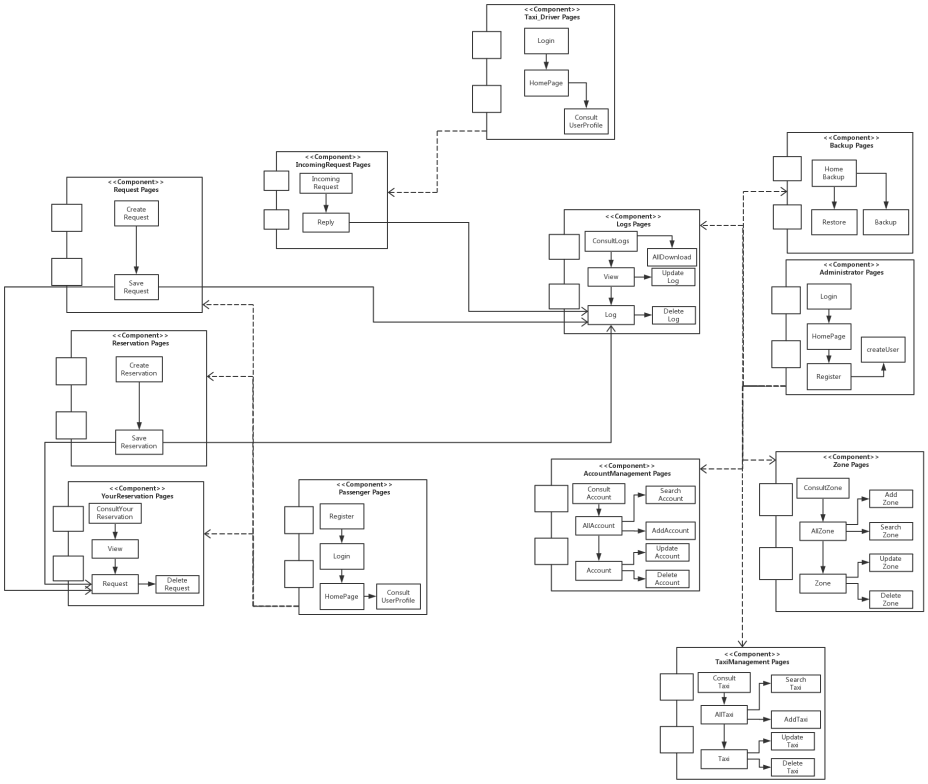
(j) **addZone**This controller manages the different zones of the city; it allows to add new zones in the city.

(k) **deleteZone**This controller manages the different zones of the city; it allows to delete a zone from a list of zone of the city.

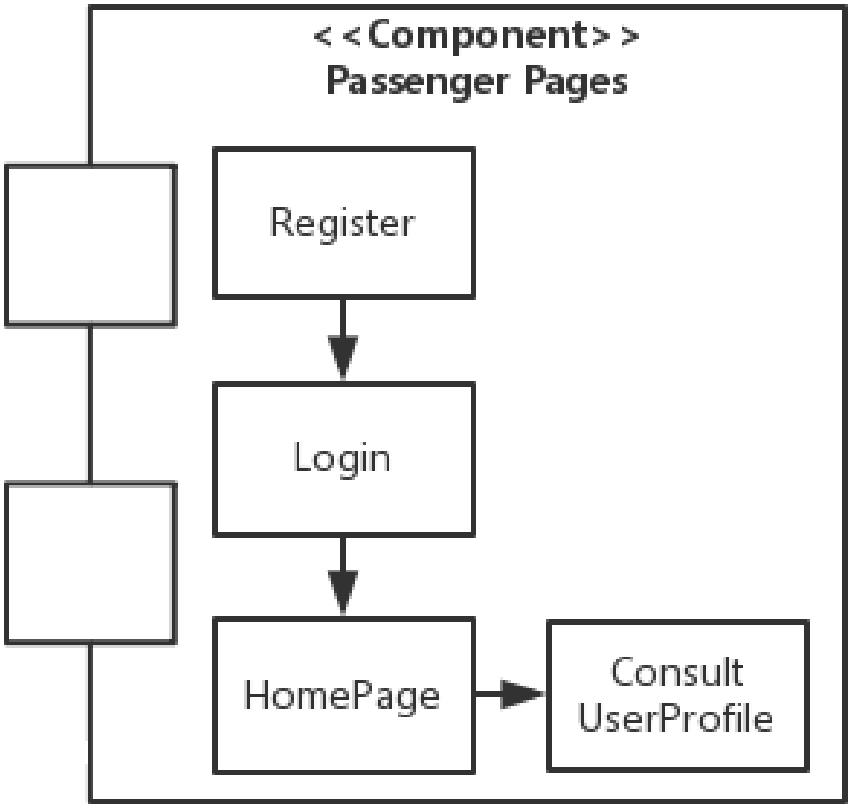
(l) **updateZone**This controller manages the different zones of the city; it allows to update the information of a zone of the city.



### 2.5.6 Web Component



1. **Passenger Pages**



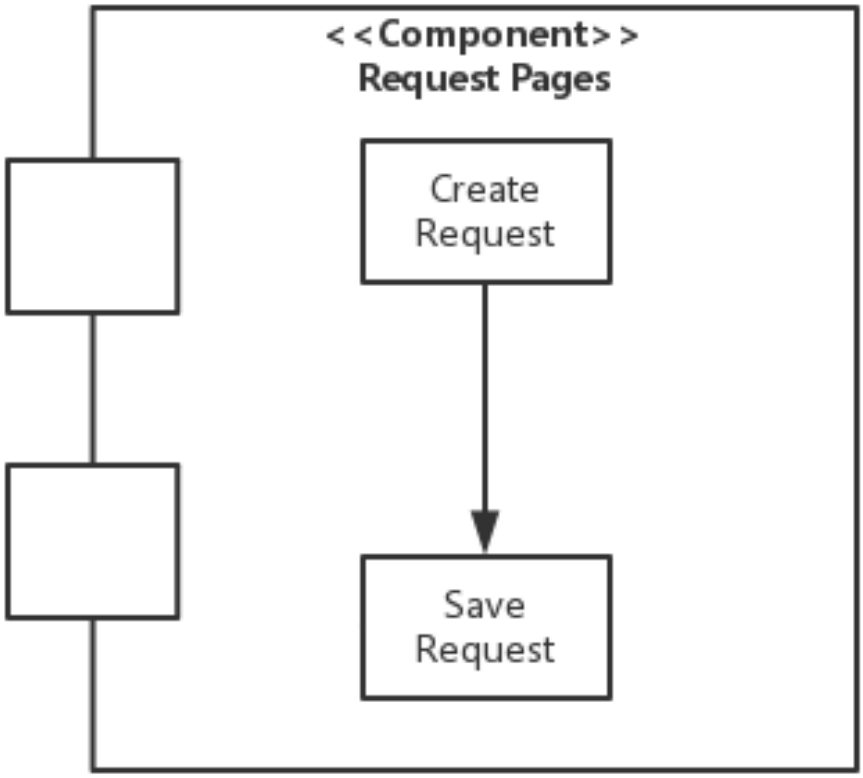
| **Name** | Register |
|---|---|
| **Classification** | Web Page |
| **Definition** | Passenger interface for registering a passenger to the system. |
| **Responsibilities** | • Display the Passenger a dialog showing the field to complete for registering a new passenger.<br>• Capture the data insert by passenger.<br>• Send data.<br>• Confirm the success or not of the operation to the passenger. |
| **Constraints** | • The data in the fields are valid for sending and storing.<br>• The web page must be loaded completely. |
| **Composition** | Login |
| **User/Interactions** | When the registration is finished, the login page is displayed. |

| **Name** | Login |
|---|---|
| **Classification** | Web Page |
| **Definition** | Passenger interface for login |
| **Responsibilities** | • Display to the Passenger a dialog showing the field to complete in order to login(Email,Password)<br>• Capture the data inserted by the passenger.<br>• Check data.<br>• Confirm the success (show the HomePage) or failure (show LoginPage)of the operation to the passenger. |
| **Constraints** | • The data in the fields are valid for sending.<br>• The web page must be loaded completely. |
| **Composition** | HomePage |
| **User/Interactions** | When the login is finished, the HomePage is displayed. |

| **Name** | HomePage |
|---|---|
| **Classification** | Web Page |
| **Definition** | Passenger interface for selecting the choice. |
| **Responsibilities** | • Display to the Passenger a list of possible functions that he can do:<br>– ReserveATaxi<br>– RequestATaxi<br>– PendingReservation<br>– YourAccount<br>– Logout<br>• Redirect Passenger to the page of the choice. |
| **Constraints** | • The web page must be loaded completely.<br>• One button must be pressed in order to continue with the application. |
| **Composition** | • RequestPage<br>• ReservationPage<br>• LoginPage<br>• YourReservationPage<br>• AccountPage |
| **User/Interactions** | • When Passenger clicks to ReserveATaxi, the ReservationPage is loaded.<br>• When Passenger clicks to RequestATaxi, the RequestPage is loaded.<br>• When Passenger clicks to PendingReservation, the YourReservationPage is loaded.<br>• When Passenger clicks to YourAccount the PassengerPage is loaded.<br>• When Passenger clicks to Logout, the LoginPage is loaded. |

| **Name** | ConsultUserProfile |
|---|---|
| **Classification** | Web Page |
| **Definition** | Passenger interface for seeing the information that Passenger inserts when he registered to the system. |
| **Responsibilities** | • Display to the Passenger a list of data that he inserts.<br>• Check data, if Passenger modify something information.<br>• Confirm or not if the action is go to a successful conclusion. |
| **Constraints** | • The web page must be loaded completely.<br>• Parameters must be valid.<br>• Button Confirm must be pressed. |
| **Composition** | HomePage |
| **User/Interactions** | When Passenger clicks on Home, the HomePage is loaded. |

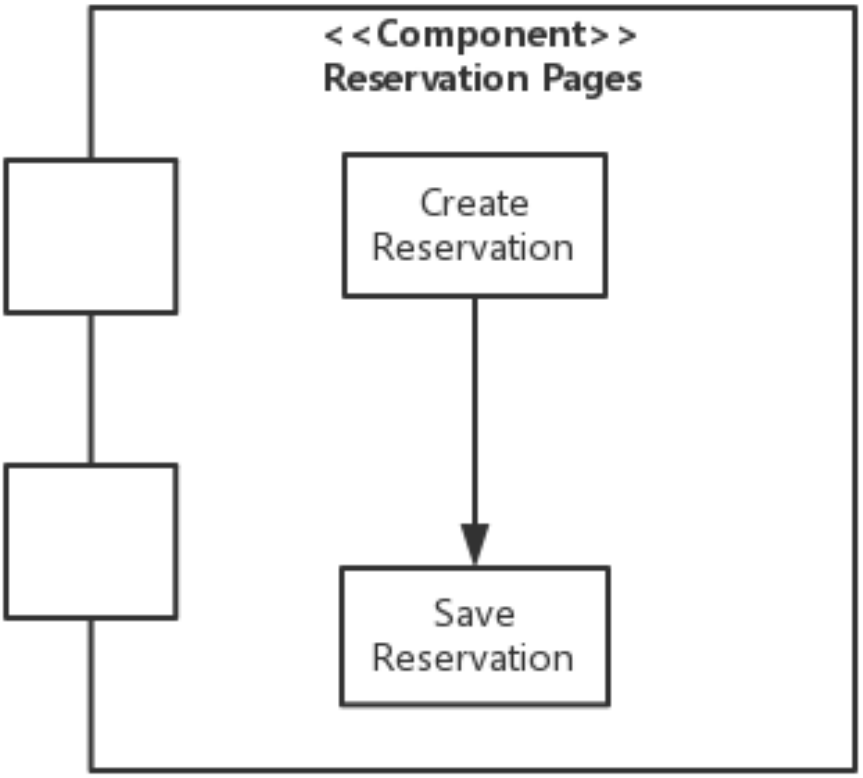
2. **Request Pages**



| Name | CreateRequest |
|---|---|
| **Classification** | Web Page |
| **Definition** | Passenger interface for insert a taxi request into the system. |
| **Responsibilities** | • Display to the Passenger a dialog showing the field to complete for requesting a taxi.<br>• Capture the data insert by the passenger.<br>• Send data.<br>• Insert request in the queue of request.<br>• Confirm the success or not of the operation to the passenger. |
| **Constraints** | • The data in the fields are valid for sending and store.<br>• The web page must be loaded completely. |
| **Composition** | SaveRequest |
| **User/Interactions** | When the request is finished, the SaveRequest is displayed. |

| Name | SaveRequest |
|---|---|
| **Classification** | Web Page |
| **Definition** | Passenger interface for confirming that the request is confirmed. |
| **Responsibilities** | Display to the Passenger the success or not of the operation. |
| **Constraints** | The web page must be loaded completely. |
| **Composition** | HomePage |
| **User/Interactions** | When the request is finished, the HomePage is displayed |

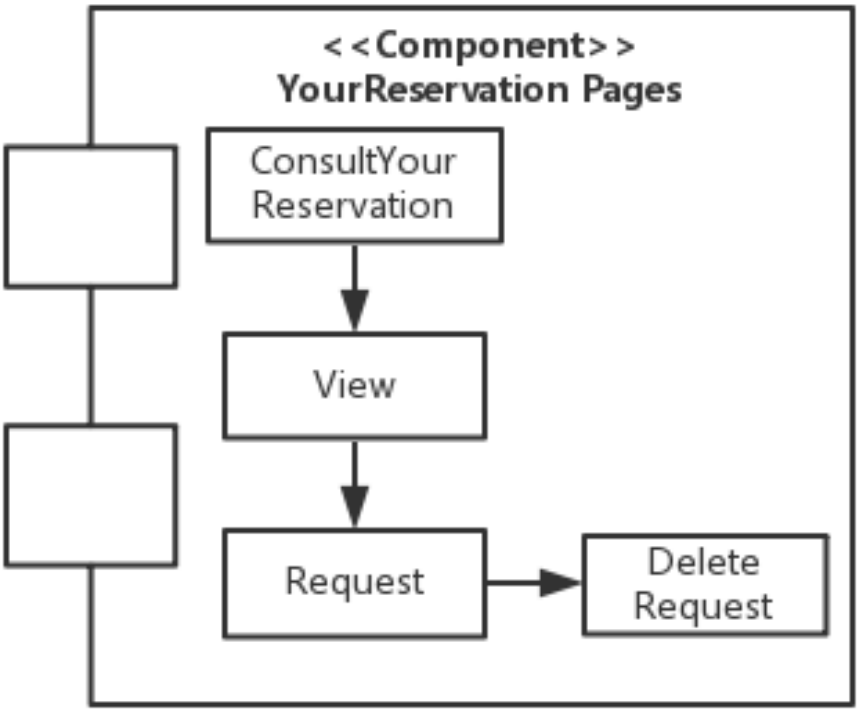
3. **Reservation Pages**



| **Name** | CreateReservation |
|---|---|
| **Classification** | Web Page |
| **Definition** | Passenger interface for inserting a taxi reservation into the system. |
| **Responsibilities** | • Display to the Passenger a dialog showing the field to complete in order to reserve a taxi.<br>• Capture the data inserted by passenger.<br>• Send data.<br>• Insert reservation in the list of reservation.<br>• Confirm the success or not of the operation to the passenger. |
| **Constraints** | • The data in the fields are valid for sending and storing.<br>• The web page must be loaded completely. |
| **Composition** | SaveReservation |
| **User/Interactions** | When the request is finished, the SaveReservation is displayed. |

| **Name** | SaveReservation |
|---|---|
| **Classification** | Web Page |
| **Definition** | Passenger interface for confirming that the reservation is store and for show the information inserted by Passenger. |
| **Responsibilities** | Display to the Passenger the success or not of the operation with a resume. |
| **Constraints** | The web page must be loaded completely. |
| **Composition** | HomePage |
| **User/Interactions** | When the operation of reservation is finished, the HomePage is displayed. |

4. **YourReservation Pages**



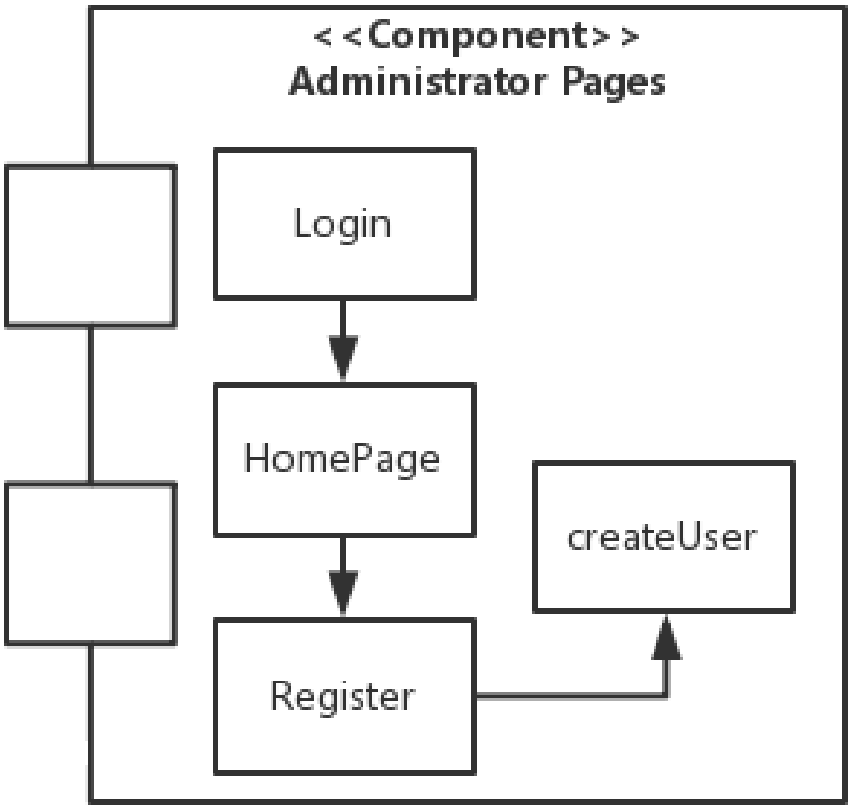

| **Name** | ConsultYourReservation |
|---|---|
| **Classification** | Web Page |
| **Definition** | Passenger interface for showing what Passenger can do in this section. |
| **Responsibilities** | • Display to the Passenger a desciption of the service. |
| **Constraints** | The web page must be loaded completely. |
| **Composition** | View |
| **User/Interactions** | When Passenger clicks on button, the View is displayed. |

| **Name** | View |
|---|---|
| **Classification** | Web Page |
| **Definition** | Passenger interface for displaying the list of request or reservation that he did. |
| **Responsibilities** | • Display to the Passenger a list of request or reservation with a button respectively<br>• Capture the button clicks by passenger.<br>• Send choice. |
| **Constraints** | • The button must be clicked<br>• The web page must be loaded completely. |
| **Composition** | Request |
| **User/Interactions** | When Passenger clicks on button, the Request is displayed |

| **Name** | Request |
|---|---|
| **Classification** | Web Page |
| **Definition** | Passenger interface for showing the information about the selected request. |
| **Responsibilities** | • Display to the Passenger a description of the request:<br>– hours<br>– destination<br>– taxi driver<br>– taxi<br>– if it is a reservation<br>– location.<br>• Display three buttons:<br>– continue with see another request;<br>– delete the request;<br>– go to HomePage.<br>• Capture the button clicks made by the passenger.<br>• Send choice. |
| **Constraints** | The web page must be loaded completely. |
| **Composition** | • HomePage<br>• Delete<br>• View |
| **User/Interactions** | • When Passenger clicks on Home, the HomePage is load.<br>• When Passenger clicks on Delete the DeletePage is load.<br>• When Passenger clicks on Request, the ViewPage is load. |

| **Name** | DeleteRequest |
|---|---|
| **Classification** | Web Page |
| **Definition** | Passenger interface for deleting a request. |
| **Responsibilities** | • Display to the Passenger what happen if he deletes a request and if it is possible and a button for deleting a request.<br>• Capture the button clicks by passenger.<br>• Send choice.<br>• Confirm the success or not of the operation to the passenger. |
| **Constraints** | • The button must be pressed.<br>• The web page must be loaded completely. |
| **Composition** | HomePage |
| **User/Interactions** | When the delete is finished, the HomePage is displayed. |

5. **Administrator Pages**



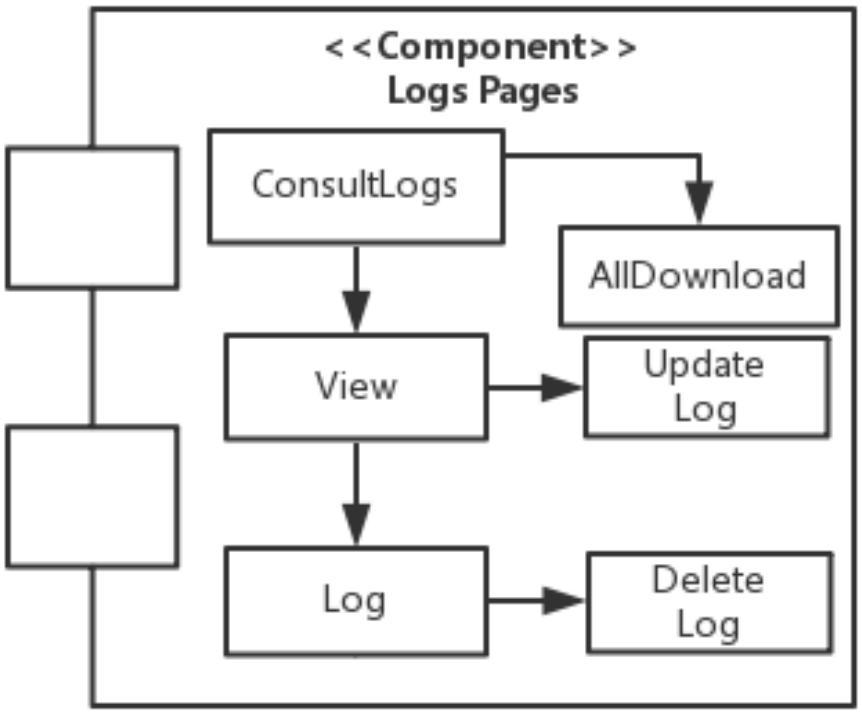
| **Name** | Register |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for registering a particular User to the system. |
| **Responsibilities** | • Display to the Administrator a dialog showing the field to complete for registering a new User.<br>• Capture the data insert.<br>• Send data.<br>• Confirm the success or not of the operation. |
| **Constraints** | • The data in the fields are valid for sending and store.<br>• The web page must be loaded completely. |
| **Composition** | CreateUser |
| **User/Interactions** | When the registration is finished, the CreateUser page is displayed. |

| **Name** | Login |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for login |
| **Responsibilities** | • Display to the Administrator a dialog showing the field to complete for login(Email,Password).<br>• Capture the data insert.<br>• Check data.<br>• Confirm the success (show the HomePage) or not (show LoginPage)of the operation. |
| **Constraints** | • The data in the fields are valid for sending.<br>• The web page must be loaded completely. |
| **Composition** | HomePage |
| **User/Interactions** | When the login is finished, the HomePage is displayed. |

| **Name** | HomePage |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for selecting the choice. |
| **Responsibilities** | • Display to the Administrator a list of possible functions that he can do.<br>• Redirect Administrator to the page of the choice |
| **Constraints** | • The web page must be loaded completely.<br>• One button must be pressed for continuing with the application. |
| **Composition** | • LogsPage<br>• AccountManagementPage<br>• LoginPage<br>• TaxiManagementPage<br>• AdministratorPage<br>• ZonePage |
| **User/Interactions** | • When Administrator clicks to TaxiManagement, the TaxiManagementPage is load.<br>• When Administrator clicks to AccountManagement, the AccountManagementPageis load.<br>• When Administrator clicks to Logs, the LogsPage is load.<br>• When Administrator clicks to account the AdministratorPage is load.<br>• When Administrator clicks to Logout, the LoginPage is load.<br>• When Administrator clicks to Zone, the ZonePage is load. |

| **Name** | CreateUser |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for registering an Administrator or a particular User to the system. |
| **Responsibilities** | • Display to the Administrator a dialog showing the field to complete for registering a new User.<br>• Capture the data insert.<br>• Check data.<br>• Confirm the success (show the HomePage) or not (show LoginPage)of the operation. |
| **Constraints** | • The web page must be loaded completely.<br>• One button must be pressed for continuing with the application. |
| **Composition** | |
| **User/Interactions** | |

6. **Logs Pages**




| | |
|---|---|
| **Name** | ConsultLogs |
| **Classification** | Web Page |
| **Definition** | Administrator interface for showing what Administrator can do in this section. |
| **Responsibilities** | • Display to the Administrator a description of the service.<br>• Display a button for downloading the logs. |
| **Constraints** | The web page must be loaded completely. |
| **Composition** | View |
| **User/Interactions** | When Administrator clicks on button, the View is displayed. |

| **Name** | View |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for see the list of logs |
| **Responsibilities** | • Display to the Administrator a list of logs with a button respectively organized by Passenger.<br>• Capture the button clicks by Administrator.<br>• Send choice. |
| **Constraints** | • The web page must be loaded completely.<br>• One button must be pressed for continuing with the application. |
| **Composition** | Log |
| **User/Interactions** | When Administrator clicks on button, the LogPage is displayed. |

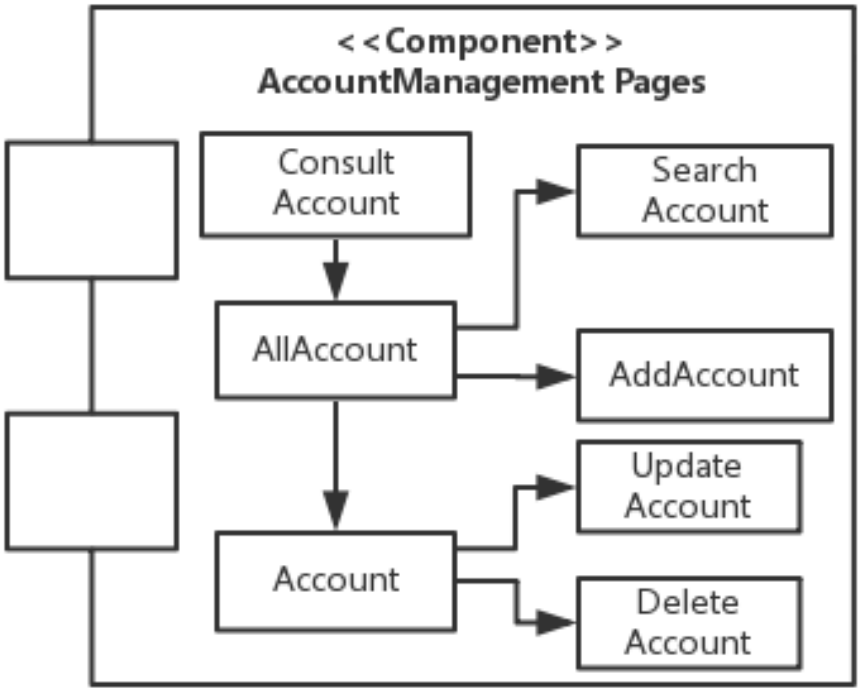
| **Name** | Log |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for showing the information about the log select. |
| **Responsibilities** | • Display to the Administrator a description of the log : hours,destination,taxi driver, taxi, passenger of the request and if it is a reservation the location, the reply of the taxi that receive the request.<br>• Display four buttons :<br>– continue with see another Log;<br>– delete Log;<br>– update Log<br>– go to HomePage.<br>• Capture the button clicks by Administrator.<br>• Send choice. |
| **Constraints** | • The web page must be loaded completely.<br>• One button must be pressed to continue with the application. |
| **Composition** | • HomePage<br>• Delete<br>• View<br>• Update |
| **User/Interactions** | • When Administrator clicks to Home, the HomePage is load.<br>• When Administrator clicks to Delete the DeletePage is load.<br>• When Administrator clicks to Log, the ViewPage is load.<br>• When Administrator clicks to Update, the UpdatePage is load. |

| **Name** | DeleteLog |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface to delete a log. |
| **Responsibilities** | • Display to the Administrator what happen if he deletes a log and if it is possible and a button for delete a log.<br>• Confirm the success or not of the operation.<br>• Capture the button clicks by Administrator.<br>• Send choice. |
| **Constraints** | • The web page must be loaded completely.<br>• The button must be pressed. |
| **Composition** | HomePage |
| **User/Interactions** | When the delete is finished, the HomePage is displayed |

| **Name** | UpdateLog |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for updating a log. |
| **Responsibilities** | • Display to the Administrator what happen if he update a log and if it is possible.<br>• Confirm the success or not of the operation.<br>• Send data.<br>• Check the informations update. |
| **Constraints** | • The web page must be loaded completely.<br>• The button must be pressed. |
| **Composition** | HomePage |
| **User/Interactions** | When the update is finished, the HomePage is displayed. |

| **Name** | AllDownload |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for downloading all logs. |
| **Responsibilities** | • Display to the Administrator a button to press for a download.<br>• Show the finish of the download. |
| **Constraints** | • The web page must be loaded completely.<br>• The button must be pressed. |
| **Composition** | HomePage |
| **User/Interactions** | When the download is finished, the HomePage is displayed. |

7. **Account Managements Pages**




| **Name** | ConsultAccount |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for showing what Administrator can do in this section. |
| **Responsibilities** | • Display to the Administrator a desciption of the service. |
| **Constraints** | The web page must be loaded completely. |
| **Composition** | AllAccount |
| **User/Interactions** | When Administrator clicks on button, the AllAccountPage is displayed. |

| **Name** | AllAccount |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for see the list of account. |
| **Responsibilities** | • Display to the Administrator a list of account with a button respectively.<br>• Display two buttons for searching an account and for adding a new account.<br>• Capture the button clicks by Administrator.<br>• Send choice. |
| **Constraints** | • The web page must be loaded completely.<br>• One button must be pressed for continue with the application. |
| **Composition** | • Acccount<br>• AddAccount<br>• SearchAccount<br>• HomePage |
| **User/Interactions** | • When Administrator clicks on button Account, the AccountPage is displayed.<br>• When Administrator clicks on button AddAccount, the AddAccountPage is displayed.<br>• When Administrator clicks on button SearchAccount, the SearchAccountPage is displayed.<br>• When Administrator clicks to Home, the HomePage is load. |

| **Name** | Account |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for showing the information about the account select. |
| **Responsibilities** | • Display to the Administrator a description of the account : name, surname, email, address.<br>• Display four buttons :<br>– continue with see another Account;<br>– delete Account;<br>– update Account<br>– go to HomePage.<br>• Capture the button clicks by Administrator.<br>• Send choice. |
| **Constraints** | • The web page must be loaded completely.<br>• One button must be pressed to continue with the application. |
| **Composition** | • HomePage<br>• Delete<br>• Update |
| **User/Interactions** | • When Administrator clicks to Home, the HomePage is load.<br>• When Administrator clicks to Delete the DeletePage is load.<br>• When Administrator clicks to Update, the UpdatePage is load. |

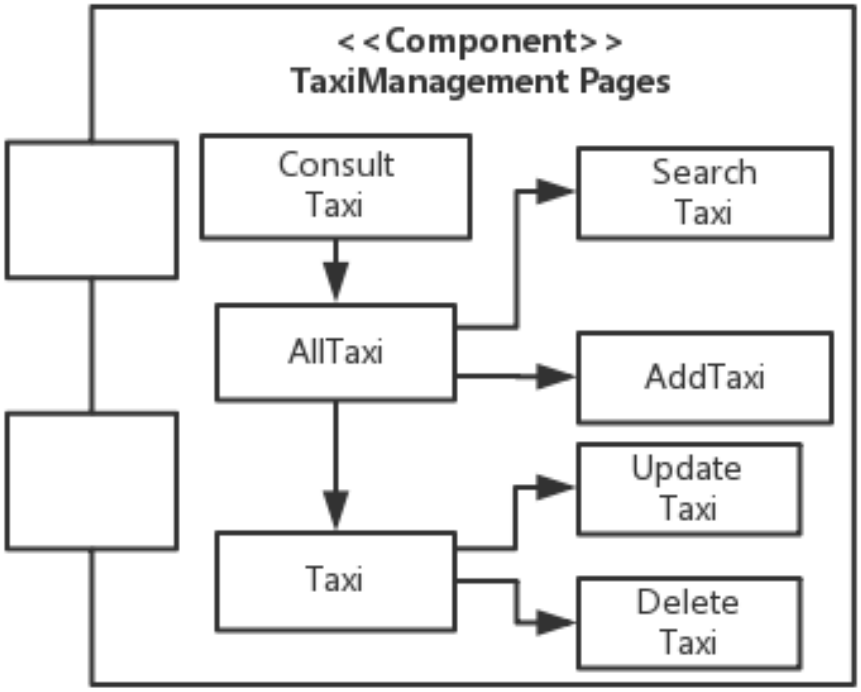
| **Name** | DeleteAccount |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for deleting a account. |
| **Responsibilities** | • Display to the Administrator what happen if he deletes a account with a button for deleting an account.<br>• Confirm the success or not of the operation.<br>• Capture the button clicks by Administrator.<br>• Send choice. |
| **Constraints** | • The web page must be loaded completely.<br>• The button must be pressed. |
| **Composition** | HomePage |
| **User/Interactions** | When the delete is finished, the HomePage is displayed. |

| **Name** | UpdateAccount |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for updating one account. |
| **Responsibilities** | • Display to the Administrator what happen if he update account.<br>• Confirm the success or not of the operation.<br>• Send data.<br>• Check the informations update. |
| **Constraints** | • The web page must be loaded completely.<br>• The button must be pressed. |
| **Composition** | HomePage |
| **User/Interactions** | When the update is finished, the HomePage is displayed. |

| **Name** | SearchAccount |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for searching one specific account. |
| **Responsibilities** | • Display to the Administrator fields to complete for searching a specific account.<br>• Show the result of the research and a button to see all information about the account. |
| **Constraints** | • The web page must be loaded completely.<br>• The button must be pressed.<br>• The account research must be exist. |
| **Composition** | • HomePage<br>• Account |
| **User/Interactions** | • When Administrator clicks to Home, the HomePage is load.<br>• When Administrator clicks on button Account, the AccountPage is displayed |

| **Name** | AddAccount |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for add one specific account. |
| **Responsibilities** | • Display to the Administrator fields to complete for adding a specific account.<br>• Capture the button pressed by Administrator .<br>• Check the informations insert.<br>• Send information.<br>• Confirm the success or not of the operation. |
| **Constraints** | • The web page must be loaded completely.<br>• The button must be pressed for saving the account. |
| **Composition** | HomePage |
| **User/Interactions** | When Administrator clicks to Home, the HomePage is load. |

8. **Taxi Management Pages**




| **Name** | ConsultTaxi |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for showing what Administrator can do in this section. |
| **Responsibilities** | • Display to the Administrator a desciption of the service.<br>• Display a button for continuing with see the taxis. |
| **Constraints** | The web page must be loaded completely. |
| **Composition** | Alltaxi |
| **User/Interactions** | When Administrator clicks on button, the AlltaxiPage is displayed. |

| **Name** | Alltaxi |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for seeing the list of taxi. |
| **Responsibilities** | • Display to the Administrator a list of taxi with a button respectively.<br>• Display two button for researching a taxi or a button for adding a new taxi.<br>• Capture the button clicks by Administrator.<br>• Send choice. |
| **Constraints** | • The web page must be loaded completely.<br>• One button must be pressed for continuing with the application. |
| **Composition** | • Taxi<br>• AddTaxi<br>• SearchTaxi<br>• HomePage |
| **User/Interactions** | • When Administrator clicks on button Taxi, the TaxiPage is displayed.<br>• When Administrator clicks on button AddTaxi, the AddTaxiPage is displayed.<br>• When Administrator clicks on button SearchTaxi, the SearchTaxiPage is displayed.<br>• When Administrator clicks to Home, the HomePage is load. |

| **Name** | Taxi |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for showing the information about the Taxi select. |
| **Responsibilities** | • Display to the Administrator a desciption of the Taxi :taxiCode, plateNumber.<br>• Display four buttons :<br>– continue with see another Taxi;<br>– delete Taxi;<br>– update Taxi<br>– go to HomePage.<br>• Capture the button clicks by Administrator.<br>• Send choice. |
| **Constraints** | • The web page must be loaded completely.<br>• One button must be pressed to continue with the application. |
| **Composition** | • HomePage<br>• Delete<br>• Update |
| **User/Interactions** | • When Administrator clicks to Home, the HomePage is load.<br>• When Administrator clicks to Delete the DeletePage is load.<br>• When Administrator clicks to Update, the UpdatePage is load. |

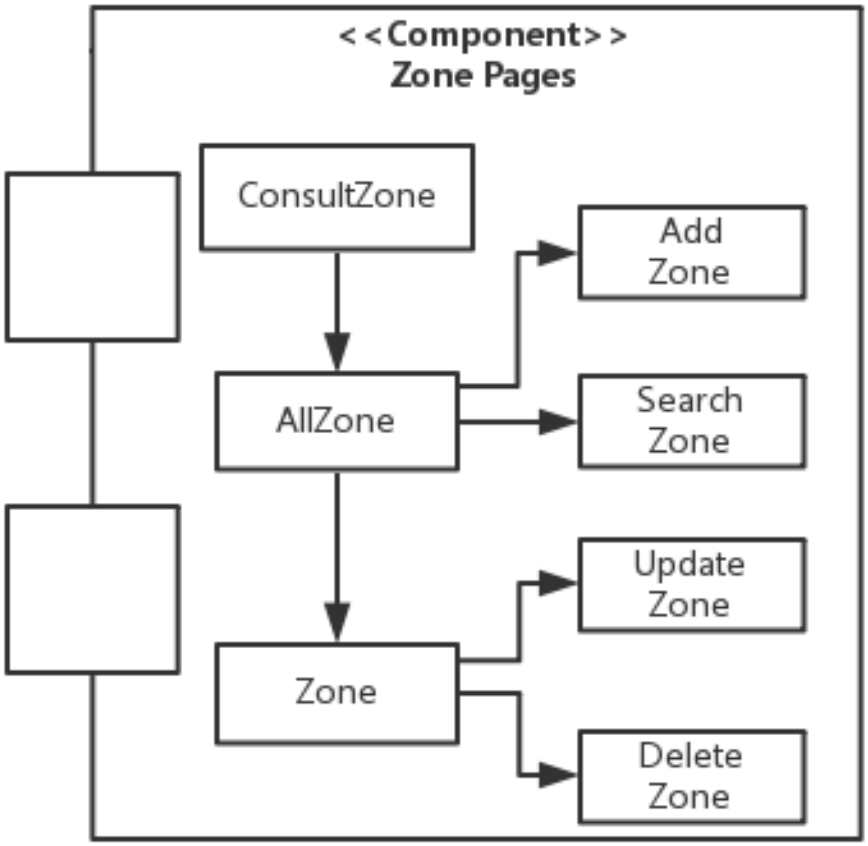
| Name | DeleteTaxi |
|---|---|
| Classification | Web Page |
| Definition | Administrator interface for deleting a Taxi. |
| Responsibilities | • Display to the Administrator what happen if he deletes a Taxi with a button for deleting Taxi.<br>• Confirm the success or not of the operation.<br>• Capture the button clicks by Administrator.<br>• Send choice. |
| Constraints | • The web page must be loaded completely.<br>• The button must be pressed. |
| Composition | HomePage |
| User/Interactions | When the delete is finished, the HomePage is displayed. |

| Name | UpdateTaxi |
|---|---|
| Classification | Web Page |
| Definition | Administrator interface for updating a Taxi. |
| Responsibilities | • Display to the Administrator what happen if he update Taxi.<br>• Confirm the success or not of the operation.<br>• Send data.<br>• Check the informations update. |
| Constraints | • The web page must be loaded completely.<br>• The button must be pressed. |
| Composition | HomePage |
| User/Interactions | When the update is finished, the HomePage is displayed. |

| **Name** | SearchTaxi |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for searching one specific Taxi. |
| **Responsibilities** | • Display to the Administrator fields to complete for searching a specific Taxi(plateNumber).<br>• Show the result of the research and a button for seeing all information about the taxi selected. |
| **Constraints** | • The web page must be loaded completely.<br>• The button must be pressed.<br>• The Taxi research must be exist. |
| **Composition** | • HomePage<br>• Taxi |
| **User/Interactions** | • When Administrator clicks to Home, the HomePage is load.<br>• When Administrator clicks on button Taxi, the TaxiPage is displayed |

| **Name** | AddTaxi |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for adding one specific Taxi. |
| **Responsibilities** | • Display to the Administrator fields to complete for adding a specific Taxi.<br>• Capture the button pressed by Administrator .<br>• Check the informations insert.<br>• Send information.<br>• Confirm the success or not of the operation. |
| **Constraints** | • The web page must be loaded completely.<br>• The button must be pressed for saving the account. |
| **Composition** | HomePage |
| **User/Interactions** | When Administrator clicks to Home, the HomePage is load. |

9. **Zone Pages**




| **Name** | ConsultZone |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for showing what Administrator can do in this section. |
| **Responsibilities** | • Display to the Administrator a desciption of the service.<br>• Display a button for continuing with see the Zones. |
| **Constraints** | The web page must be loaded completely. |
| **Composition** | AllZone |
| **User/Interactions** | When Administrator clicks on button, the AllZonePage is displayed. |

| **Name** | AllZone |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for seeing the list of taxi. |
| **Responsibilities** | • Display to the Administrator a list of Zone with a button respectively.<br>• Display two button for researching a Zone or a button for adding a new Zone.<br>• Capture the button clicks by Administrator.<br>• Send choice. |
| **Constraints** | • The web page must be loaded completely.<br>• One button must be pressed for continuing with the application. |
| **Composition** | • Zone<br>• AddZone<br>• SearchZone<br>• HomePage |
| **User/Interactions** | • When Administrator clicks on button Zone, the ZonePage is displayed.<br>• When Administrator clicks on button AddZone, the AddZonePage is displayed.<br>• When Administrator clicks on button SearchZone, the SearchZonePage is displayed.<br>• When Administrator clicks to Home, the HomePage is load. |

| **Name** | Zone |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for showing the information about the Zone select. |
| **Responsibilities** | • Display to the Administrator a desciption of the Zone :name, latitude, longitude.<br>• Display four buttons :<br>– continue with see another Zone;<br>– delete Zone;<br>– update Zone;<br>– go to HomePage.<br>• Capture the button clicks by Administrator.<br>• Send choice. |
| **Constraints** | • The web page must be loaded completely.<br>• One button must be pressed to continue with the application. |
| **Composition** | • HomePage<br>• Delete<br>• Update |
| **User/Interactions** | • When Administrator clicks to Home, the HomePage is load.<br>• When Administrator clicks to Delete the DeletePage is load.<br>• When Administrator clicks to Update, the UpdatePage is load. |

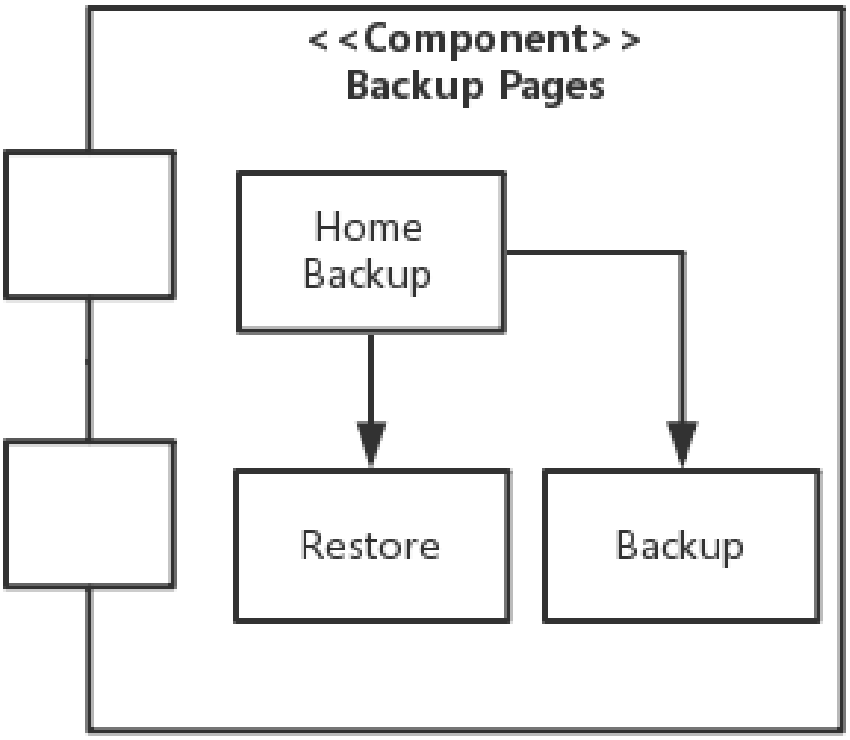
| **Name** | DeleteZone |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for deleting a Zone. |
| **Responsibilities** | • Display to the Administrator what happen if he deletes a Zone with a button for deleting Zone.<br>• Confirm the success or not of the operation.<br>• Capture the button clicks by Administrator.<br>• Send choice. |
| **Constraints** | • The web page must be loaded completely.<br>• The button must be pressed. |
| **Composition** | HomePage |
| **User/Interactions** | When the delete is finished, the HomePage is displayed. |

| **Name** | UpdateZone |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for updating a Zone. |
| **Responsibilities** | • Display to the Administrator what happen if he update Zone.<br>• Confirm the success or not of the operation.<br>• Send data.<br>• Check the informations update. |
| **Constraints** | • The web page must be loaded completely.<br>• The button must be pressed. |
| **Composition** | HomePage |
| **User/Interactions** | When the update is finished, the HomePage is displayed. |

| **Name** | SearchZone |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for searching one specific Zone. |
| **Responsibilities** | • Display to the Administrator fields to complete for searching a specific Zone(name).<br>• Show the result of the research and a button for seeing all information about the Zone selected. |
| **Constraints** | • The web page must be loaded completely.<br>• The button must be pressed.<br>• The Taxi research must be exist. |
| **Composition** | • HomePage<br>• Zone |
| **User/Interactions** | • When Administrator clicks to Home, the HomePage is load.<br>• When Administrator clicks on button Zone, the ZonePage is displayed |

| **Name** | AddZone |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for adding one specific Zone. |
| **Responsibilities** | • Display to the Administrator fields to complete for adding a specific Zone.<br>• Capture the button pressed by Administrator .<br>• Check the informations insert.<br>• Send information.<br>• Confirm the success or not of the operation. |
| **Constraints** | • The web page must be loaded completely.<br>• The button must be pressed for saving the account. |
| **Composition** | HomePage |
| **User/Interactions** | When Administrator clicks to Home, the HomePage is load. |

10. **Backup Pages**

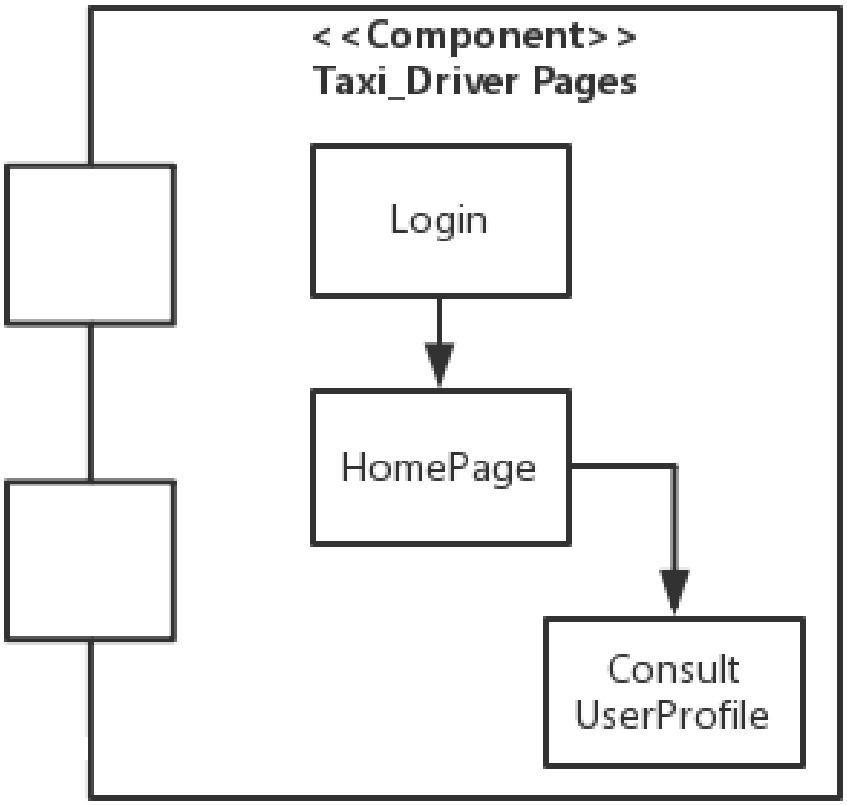


| **Name** | HomeBackup |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for showing what Administrator can do in this section. |
| **Responsibilities** | • Display to the Administrator a desciption of the service.<br>• Display a button for backup.<br>• Display a button for restore.<br>• Capture the button pressed. |
| **Constraints** | The web page must be loaded completely. |
| **Composition** | • Restore<br>• Backup<br>• HomePage |
| **User/Interactions** | • When Administrator clicks on button Home, the HomePage is displayed.<br>• When Administrator clicks on button Backup, the BackupPage is displayed.<br>• When Administrator clicks on button Restore, the RestorePage is displayed. |

| **Name** | Backup |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for showing as do a backup. |
| **Responsibilities** | • Display to the Administrator a desciption of the service.<br>• Display a button for starting a backup.<br>• Capture the button pressed. |
| **Constraints** | The web page must be loaded completely. |
| **Composition** | • Backup<br>• HomePage |
| **User/Interactions** | • When Administrator clicks on button Home, the HomePage is displayed.<br>• When Administrator clicks on button Backup, the operation of backup start. |

| **Name** | Restore |
|---|---|
| **Classification** | Web Page |
| **Definition** | Administrator interface for showing as do a restore. |
| **Responsibilities** | • Display to the Administrator a desciption of the service.<br>• Display a button for starting a restore<br>• Capture the button clicks by Administrator.<br>• Send choice. |
| **Constraints** | • The web page must be loaded completely.<br>• One button must be pressed to continue with the application. |
| **Composition** | • HomePage<br>• Restore |
| **User/Interactions** | • When Administrator clicks on button Home, the HomePage is displayed.<br>• When Administrator clicks on button Restore, the operation of restore start. |

11. **Taxi Driver Pages**

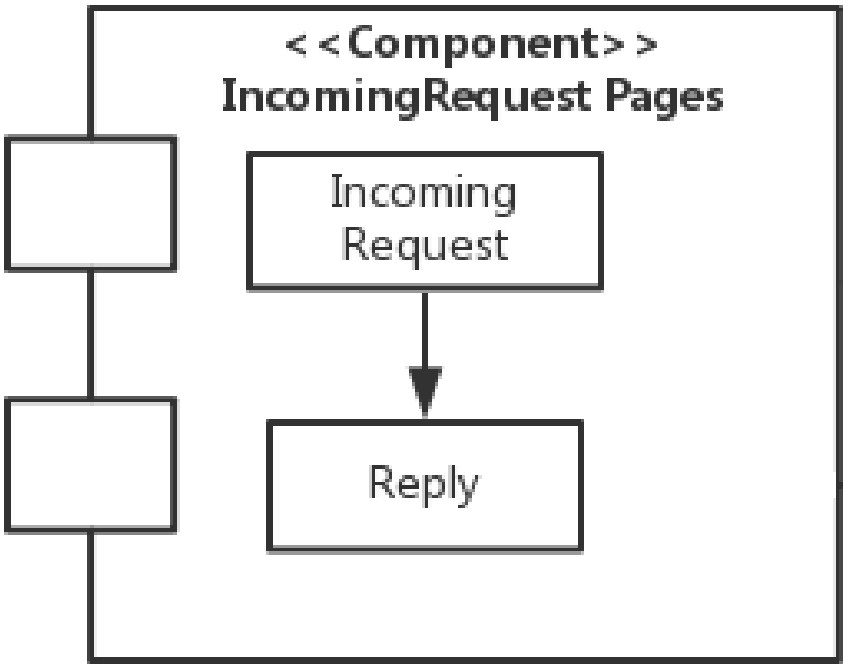


| **Name** | Login |
|---|---|
| **Classification** | Web Page |
| **Definition** | Taxi Driver interface for login. |
| **Responsibilities** | • Display to the Taxi Driver a dialog showing the field to complete for login(Email,Password).<br>• Capture the data insert by Taxi Driver.<br>• Check data.<br>• Confirm the success (show the HomePage) or not (show LoginPage). |
| **Constraints** | • The web page must be loaded completely.<br>• The data in the fields are valid for sending. |
| **Composition** | HomePage |
| **User/Interactions** | When the login is finished, the HomePage is displayed. |

| **Name** | HomePage |
|---|---|
| **Classification** | Web Page |
| **Definition** | Taxi Driver interface for select the choice. |
| **Responsibilities** | • Display to the Taxi Driver a list of possible functions that he can do:<br>– IncomingRequest<br>– Account<br>• Redirect Taxi Driver to the page of the choice. |
| **Constraints** | The web page must be loaded completely. |
| **Composition** | • LoginPage<br>• IncomingRequest<br>• Account |
| **User/Interactions** | • When Taxi Driver clicks to Account, the Taxi-Driver Page is load.<br>• When Taxi Driver clicks to Logout, the LoginPage is load.<br>• When Taxi Driver clicks to IncomingRequest, the IncomingRequestPage is load. |

| **Name** | ConsultUserProfile |
|---|---|
| **Classification** | Web Page |
| **Definition** | Taxi Driver interface for seeing the information that Taxi Driver inserts when he registered to the system. |
| **Responsibilities** | • Display to the Taxi Driver a list of data that he inserts.<br>• Check data if Taxi Driver modify something. |
| **Constraints** | • The web page must be loaded completely.<br>• Parameters must be valid. |
| **Composition** | HomePage |
| **User/Interactions** | When Taxi Driver clicks to Home, the HomePage is load. |

12. **Incoming Request Pages**



| **Name** | IncomingRequest |
|---|---|
| **Classification** | Web Page |
| **Definition** | Taxi Driver interface for accepting or refusing the request that arrived. |
| **Responsibilities** | • Display to the Taxi Driver the request that has only just arrived.<br>• Display two button:<br>– accept<br>– decline<br>• Capture the button pressed.<br>• Send reply with the choice. |
| **Constraints** | The web page must be loaded completely. |
| **Composition** | • HomePage<br>• IncomingRequest<br>• Reply |
| **User/Interactions** | • When Taxi Driver clicks to Home, the HomePage is loaded.<br>• When Taxi Driver clicks to Decline, the IncomingRequestPage is preserved.<br>• When Taxi Driver clicks to Accept, the ReplyPage is loaded. |

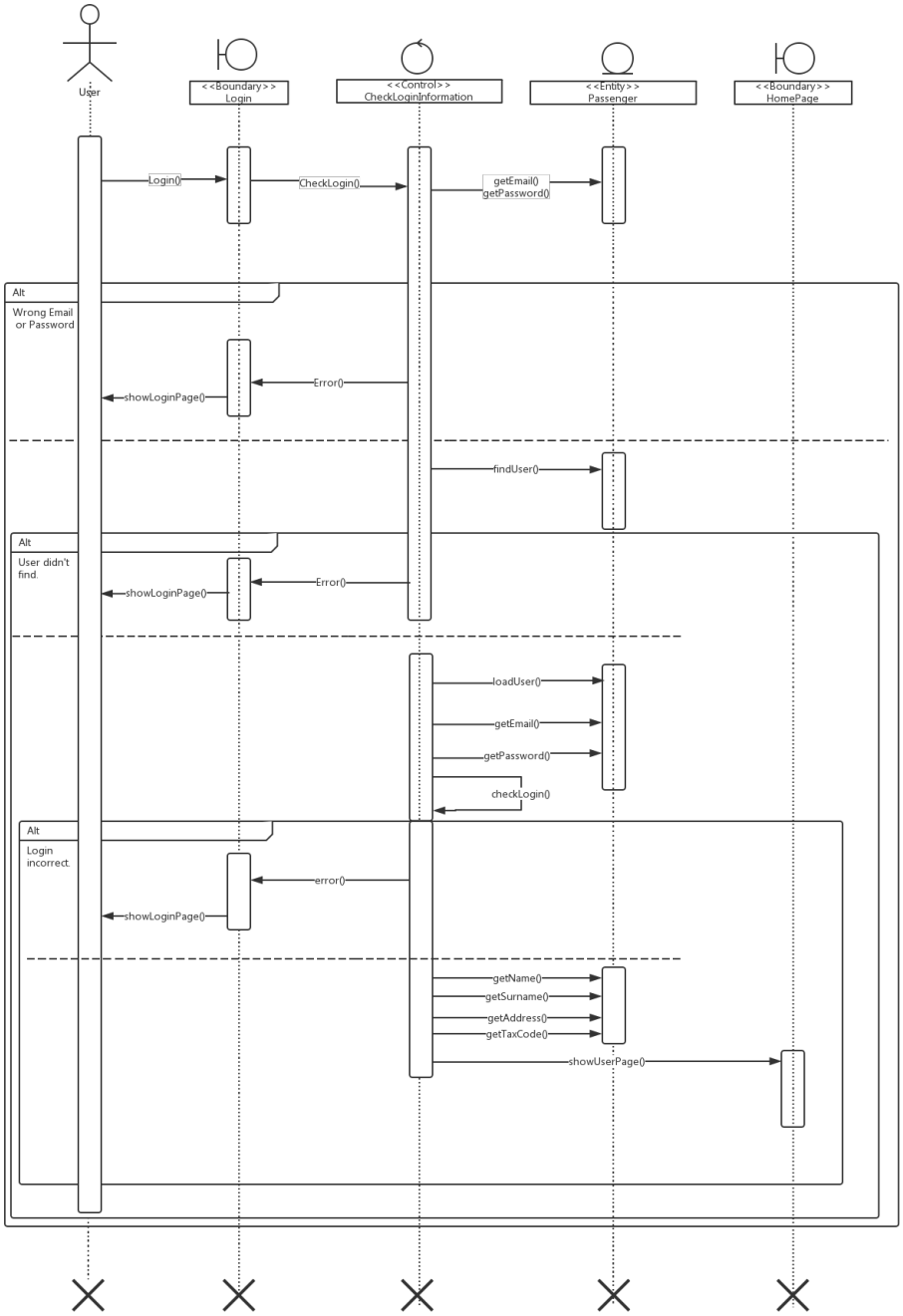
| Name | Reply |
|---|---|
| **Classification** | Web Page |
| **Definition** | Taxi Driver interface that see when accept a request. |
| **Responsibilities** | • Display to the Taxi Driver the request that has accepted.<br>• Display two buttons:<br>– RideCompleted<br>– Release<br>• Capture the button pressed.<br>• Send reply with the choice. |
| **Constraints** | The web page must be loaded completely. |
| **Composition** | • HomePage<br>• IncomingRequest |
| **User/Interactions** | • When Taxi Driver clicks to Home, the HomePage is load.<br>• When Taxi Driver clicks to RideComplete, the IncomingRequestPage is load.<br>• When Taxi Driver clicks to Release, the IncomingRequestPage is load. |

### 2.5.7 Sequence Diagram

These diagram describe in detail that is show in the BCE diagram.

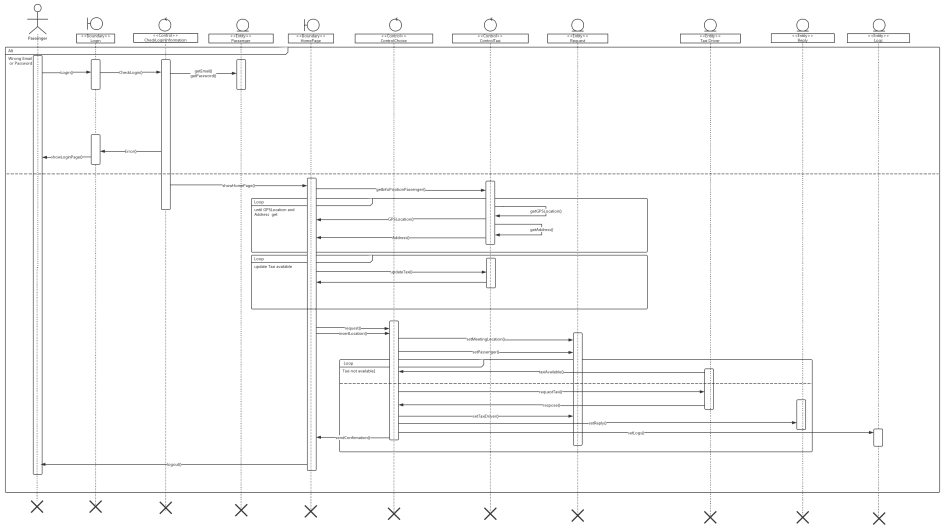
### 2.5.8 Sequence Diagram Login

Login can be done by all authorized User: * Passenger * Taxi Driver * System Administrator


User
<<Boundary>>
Login
<<Control>>
CheckLoginInformation
<<Entity>>
Passenger
<<Boundary>>
HomePage
Login()
CheckLogin()
getEmail()
getPassword()
Alt
Wrong Email
or Password
Error()
showLoginPage()
findUser()
Alt
User didn't
find.
Error()
showLoginPage()
loadUser()
getEmail()
getPassword()
checkLogin()
Alt
Login
incorrect.
error()
showLoginPage()
getName()
getSurname()
getAddress()
getTaxCode()
showUserPage()

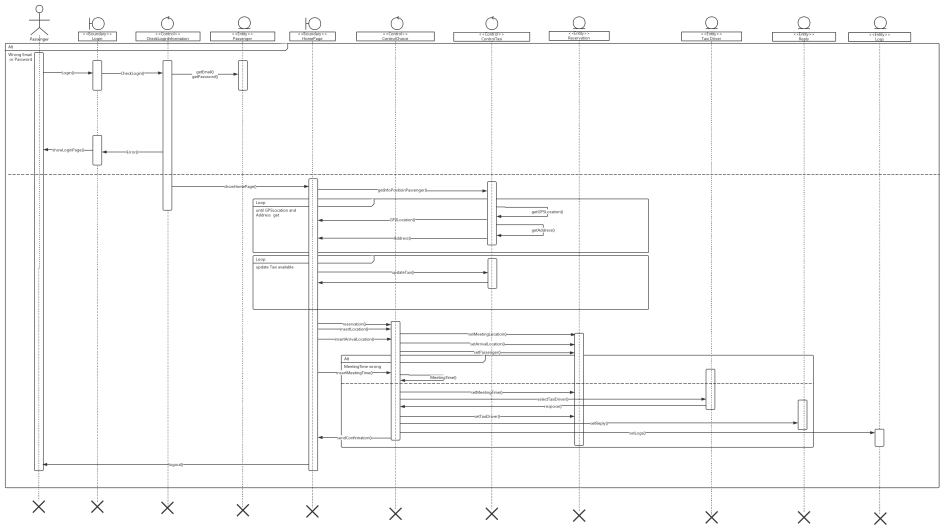
### 2.5.9 Sequence Diagram Request

Request can be done by all authorized Passenger.



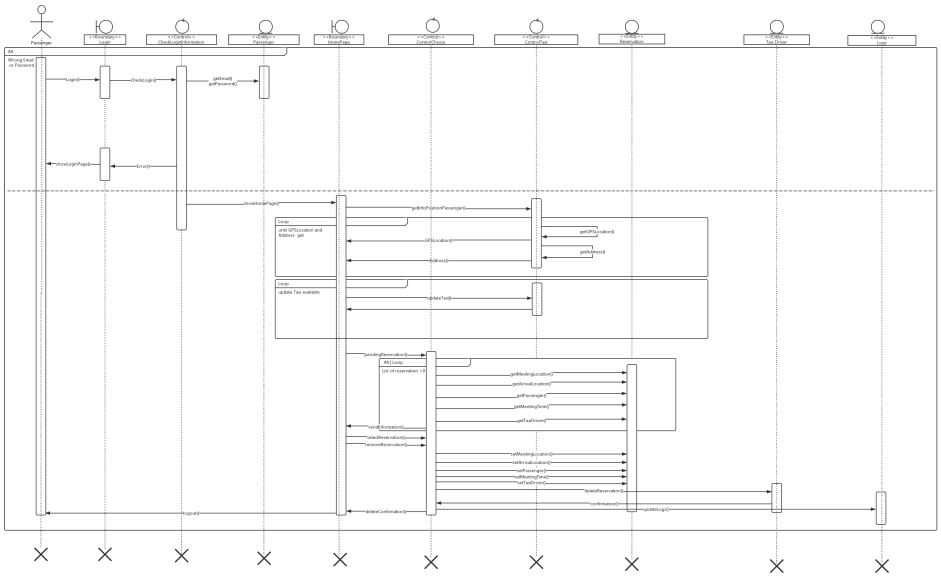
### 2.5.10 Sequence Diagram Reservation

Reservation can be done by all authorized Passenger.



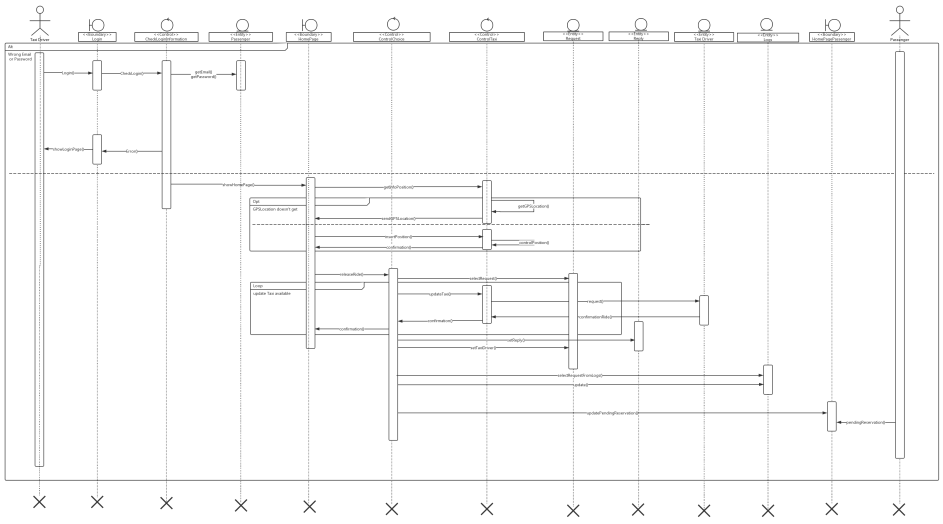
### 2.5.11 Sequence Diagram Reservation Delete

Delete a reservation can be done by all authorized Passenger.

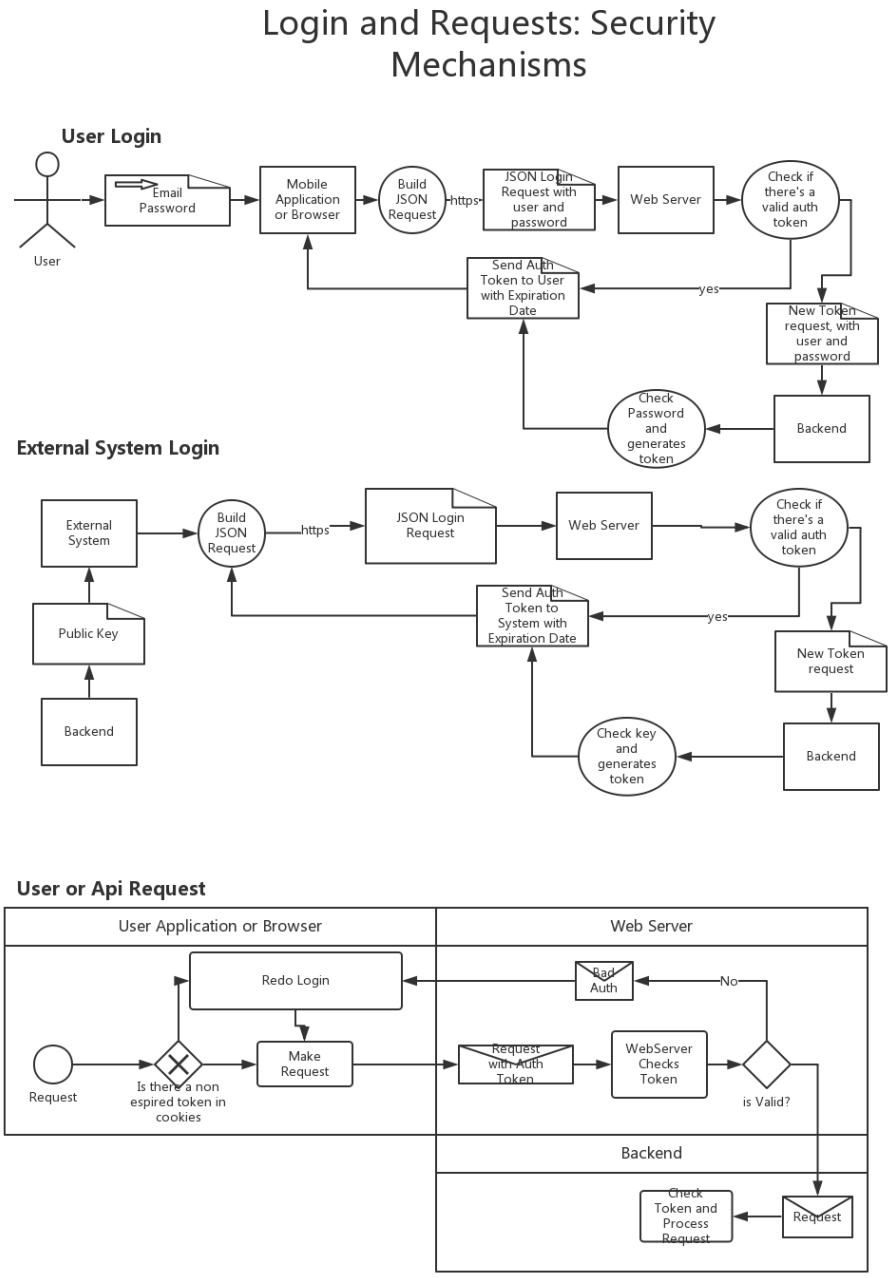


### 2.5.12 Sequence Diagram Taxi Request Release

Release a request can be done by a Taxi Driver that can go to the location of the request that he received.



### 2.5.13 Security Data Flow



## 2.6 Component Interfaces

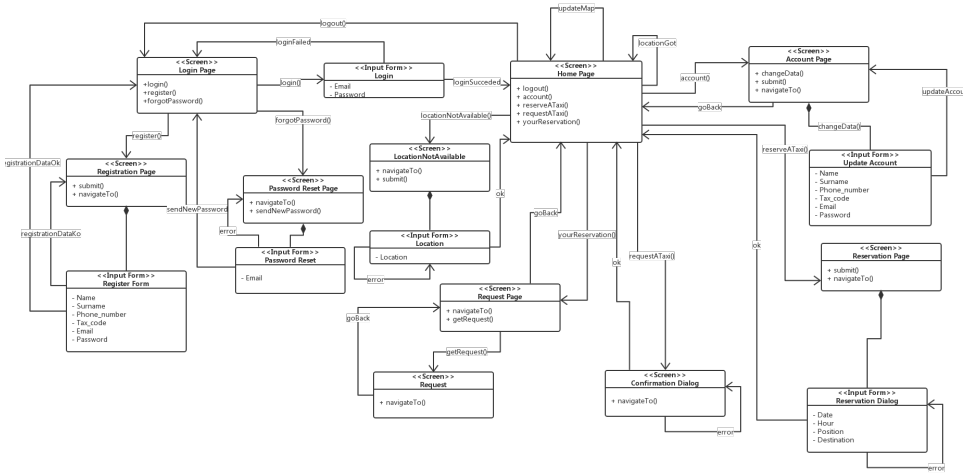
### 2.6.1 User Experience

For describing the User Experience (UX) we used a Class Diagram with appropriate stereotypes <<screen>>and <<input form>>While <<screen>>represents pages, <<input form>>represents input fields that can be complete with by user with the information that the form require.

### 2.6.2 UXPassenger

We can see in the Diagram the possible action that the passenger can do when he uses the application. This the most important pages:

1. **LoginPage** It is the first page that the user see when start the application; in this page the user can do:
   (a) **register** User can insert his date and so can register to the system and can use the application;
   (b) **login** User can insert password and email to enter in his private page;
   (c) **forgotPassword** User can request a new password because he forgot his.
2. **HomePage** It is the private page where the user can do request and reservation; in this page the user can do :
   (a) **account** The user can see his private information and can modify them;
   (b) **reserveATaxi** The user can compile a form with the information about time, location and destination of the ride request;
   (c) **requestATaxi** The user can see the time of wait for a taxi and if his request is accept or no;
   (d) **yourReservation** The user can see all his request and reservation that he do with the application;
   (e) **logout** The user can exit from the application.




### 2.6.3 UXTaxiDriver

We can see in the Diagram the possible action that the taxi driver can do when he uses the application. This the most important pages:

1. **LoginPage** It is the first page that the taxi driver see when start the application; in this page the user can do:

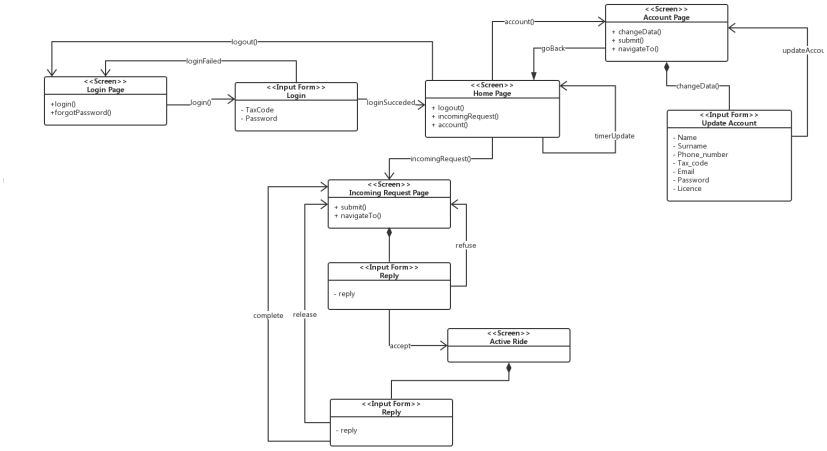
   (a) **login** Taxi driver can insert password and taxCode to enter in the application.

2. **HomePage** It is the private page where the taxi driver can accept or refuse the request that arrive from the passenger; in this page the taxi driver can do :

   (a) **logout** The taxi driver can exit from the application;

   (b) **incomingRequest** The taxi driver see the request and he decides to accept or refuse it, if he accepts the request he will see the screen with the information or the ride.

   (c) **account** The taxi driver can see or modify your private information.




### 2.6.4 UXSystemAdministrator

We can see in the Diagram the possible action that the system administrator can do when he uses the application.This the most important pages:

1. **LoginPage** It is the first page that the system administrator see when start the application; in this page the user can do:
   (a) **login** System administrator can insert password and email to enter in the application.
2. **HomePage** It is the private page where the system administrator can manage all information about taxi, account, logs, backup, restore and zone; in this page the system administrator can do :
   (a) **taxiManagement** System administrator can visualized, add, delete and update the information about a taxi;
   (b) **accountManagement** System administrator can visualized, add, delete and update the information about a user: passenger, taxi driver.
   (c) **logs** System administrator can view, delete or update and download the informations about a ride: passenger, ride, taxi driver, request, date, location.
   (d) **zone** System administrator can visualized, add, delete and update the information about a zone.
   (e) **backup** System administrator can do a backup of the system.
   (f) **restore** System administrator can restore a previous backup of the system.
   (g) **logout** System administrator can exit from the application.

3. **TaxiManagementPage** In this page system administrator can visualized, add, delete and update the information about a taxi; in this page the system administrator can do :

   (a) **allTaxi** System administrator can visualizzaed all taxi and then can select a taxi ; then he can delete or update the information about the taxi.

   (b) **taxi** System administrator can visualizzed a taxi that he requests with a form in what he inserts taxiCode and plateNumber.

4. **AccountManagementPage** In this page system administrator can visualized, add, delete and update the information about a user; in this page the system administrator can do :

   (a) **allAccount** System administrator can visualizzaed all account and then can select a account or add a new account; then he can delete or update the information about the account.

   (b) **account** System administrator can visualizzed a account that he requests with a form in what he inserts name, surname and email.

5. **ZonePage** In this page system administrator can visualized, add, delete and update the information about a zone; in this page the system administrator can do :

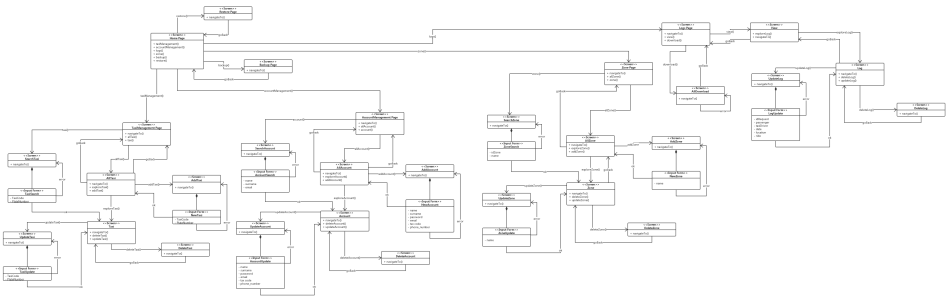
   (a) **allZone** System administrator can visualizzaed all zone and then can select a zone or add a new zone; then he can delete or update the information about the zone.

   (b) **zone** System administrator can visualizzed a zonethat he requests with a form in what he inserts name.

6. **LogsPage** In this page system administrator can visualized, download, delete and update the information about log (ride, request, passenger, taxi driver, date, location); in this page the system administrator can do :

   (a) **view** System administrator can visualizzed all log and then can select a log ; then he can delete or update the information about the log.

   (b) **download** System administrator can download the all log.



## 2.7 Architectural Styles and Patterns

The following design patterns have driven the design process of this project:

- MVC: Model-View-Controller design pattern. This pattern separates the business data, the user interface (or the interface between systems) and the core modules that runs the business logic.
- Thin-Client: The application client is used only for the data presentation and input, therefore it will contain the least business logic possible.
- 3-Tier Architecture: The system is divided in client application, Server application (where resides the web server and the back-end) and the data management system, that run on different machines.

## 2.8 Other Design Decisions

## 2.9 Implementation Technologies

This service will be implemented using the Galssfish framework which contains all the modules needed to satisfy all the requirements. In particular it will make use of Jersey as an implementation of JAX-RS for RESTful systems. For the communication interface between the app, the webserver and the backend, a JSON plugin for Jersey will be used instead of XML, for its portability and it compatibility with mobile OS such android and because it's more lightweight than XML for this special case. The backend will be developed in JEE.

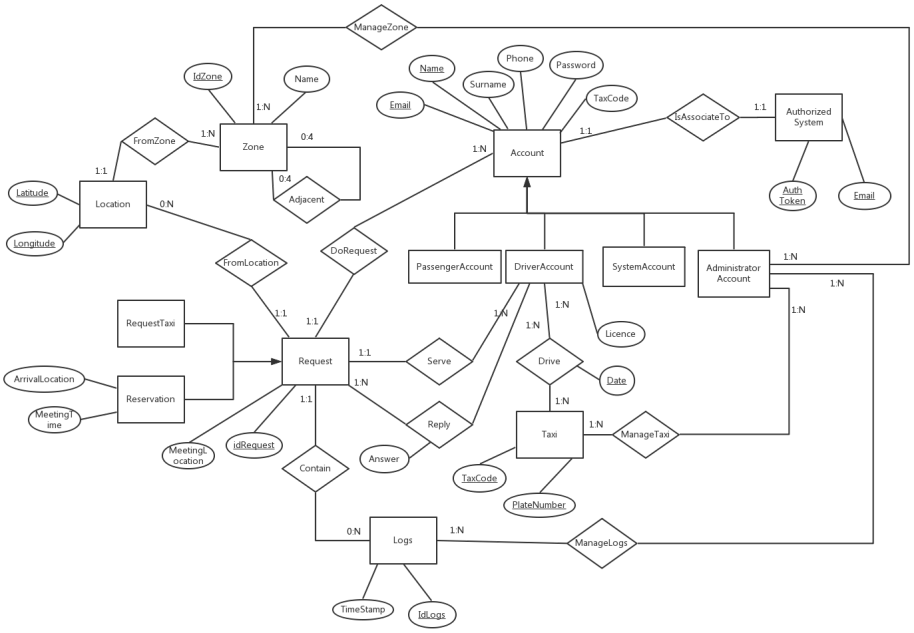
### 2.9.1 DataBase Description

In our system we have four types of Account:

1. PassengerAccount
2. DriverAccount
3. AdministratorAccount

4. SystemAccount

PassengerAccount and AdministratorAccount have the same information, instead DriverAccount has the same information with a another characteristic : Licenze. So we decided to use a one table called Account with a boolean field : if it is 0 the user is a passenger or administrator; if it is 1 the user is a taxi driver. Licence must be completed if the Account is a DriverAccount. In our system we have two types of Request : RequestTaxi and Reservation. They have the same common information and the Reservation has ArrivalLocation and MeetingTime. So we decided to use one table called Request in that there is a boolean field called RequestReservation: if it is 1 the Request is a RequestTaxi and so the field ArrivalLocation and MeetingTime can be free; if it is 0 the Request is a Reservation and so the field ArrivalLocation and MeetingTime must be completed. System Administrator is connected to Zone, Logs and User with ManageZone, ManageLogs and ManageTaxi. ManageZone contains the list of Administrator and the zone that he manages. ManageLogs contains the list of Administrator and the logs that he manages. ManageTaxi contains the list of Administrator and the taxi that he manages. Taxi Driver is associate to a taxi with the table Drive, it contains the list of a taxi driver and the taxi that he used and the data of use. All Account are associate to a AuthorizedSystem that it contains a AuthToken. Logs defines a list of Request and it is one. Request can be contain from only one list, so in request there is a connection with a Logs. Location is associate to a only one Zone. Zone can have a lot of Location. Request is distinguishable only with the passenger information, the taxi driver and the location with the basic information of the request.




### 2.9.2 Cardinalities

1. **Request - Contain - Logs** Request can be contain from only one Logs and Logs can contains a lot of Request.

2. **Request - Serve - DriverAccount** Request can be satisfied from only one Taxi Driver and DriverAccount can serve different Request.

3. **Request - Reply - DriverAccount** Request can receive reply from different Taxi Driver if the request is send to different taxi driver because same of them refuse the request and DriverAccount can reply to different request.

4. **Request - DoRequest - Account** Request can be done by one Account and Account can do different Request.

5. **Request - FromLocation - Location** Request can be done from one Location and from a Location can be done different Request.

6. **Location - FromZone - Zone** Location can be associate to only one Zone and a Zone can have different location.

7. **Zone - Adjacent - Zone** Zone can border with max 4 Zone and Zone can border with max 4 Zone.

8. **Zone - ManageZone - AdministratorAccount** Zone can be managed by different System Administrator and AdministratorAccount can manage a lot of Zone.

9. **Logs - ManageLogs - AdministratorAccount** Logs can be managed by different System Administrator and AdministratorAccount can manage a lot of Logs.

10. **Taxi - ManageTaxi - AdministratorAccount** Taxi can be managed by different System Administrator and AdministratorAccount can manage a lot of Taxi.

11. **Taxi - Drive - DriverAccount** Taxi can be drove by different Taxi Driver and DriverAccount can drive different Taxi.

12. **Account - IsAssociateTo - AuthorizedSystem** Account has a only one AuthToken stored in a AuthorizedSystem and a AuthorizedSystem is associate to only one Account.

### 2.9.3 Translation to Logical Model

1. Relation Contain between Logs and Request is translated inserting a foreign key into the table Request called Logs-idLog.

2. Relation FromLocation between Request and Location is translated inserting two foreign keys into table Request called Location-Latitude and Location-Longitude.

3. Relation DoRequest between Request and Account is translated inserting two foreign keys into table Request called Account-NamePassenger and Account-EmailPassenger.

4. Relation Serve between Request and DriverAccount is translated inserting two foreign keys into table Request called Account-NameTaxiDriver and Account-EmailTaxiDriver.

5. Relation Reply between Request and DriverAccount is translated inserting a table called Reply that contain two foreign keys called Account-NameTaxiDriver and Account-EmailTaxiDriver from table DriverAccount and one foreign key called Request-idRequest from table Request and a information called Date.

6. Relation Drive between Taxi and DriverAccount is translated inserting a table called Drive that contain two foreign keys called Account-NameTaxiDriver and Account-EmailTaxiDriver from table DriverAccount and two foreign keys called Taxi-TaxiCode and Taxi-PlateNumber from table Taxi and a information called Date.

7. Relation Adjacent between Zone and Zone is translated inserting a table called Adjacent that contain one foreign key called Zone-idZone from table Zone and one foreign key called Zone-idZoneAdj from table Zone.

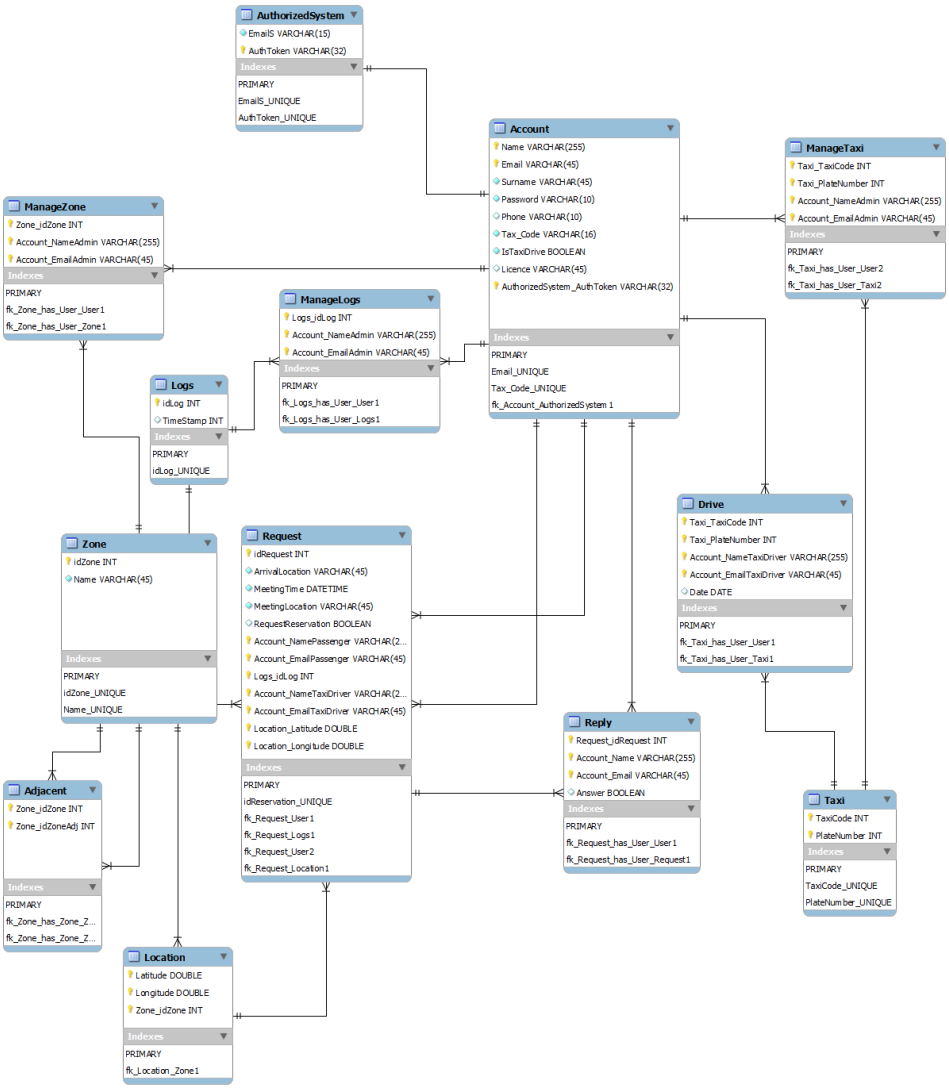
8. Relation ManageZone between Zone and AdministratorAccount is translated inserting a table called ManageZone that contain one foreign key called Zone-idZone from table Zone and two foreign keys called Account-NameAdmin and Account-EmailAdmin from table AdministratorAccount .

9. Relation ManageLogs between Logs and AdministratorAccount is translated inserting a table called ManageLogs that contain one foreign key called Logs-idLog from table Logs and two foreign keys called Account-NameAdmin and Account-EmailAdmin from table AdministratorAccount .

10. Relation ManageTaxi between Taxi and AdministratorAccount is translated inserting a table called ManageTaxi that contain two foreign keys called Taxi-TaxiCode and Taxi-PlateNumber from table Taxi and two foreign keys called Account-NameAdmin and Account-EmailAdmin from table AdministratorAccount .

11. Relation IsAssociateTo between Account and AuthorizedSystem is translated inserting one foreign key into table Account called AuthorizedSystem-AuthToken .

12. Relation FromZone between Location and Zone is translated inserting a foreign key into table Location called Zone-idZone.




### 2.9.4 Logical Scheme

1. **Zone**( IdZone, Name)
2. **ManageZone**(idZone, NameAdmin, EmailAdmin)
3. **Adjacent**(idZone, idZone)
4. **Request**(idRequest, MeetingLocation,RequestReservation, ArrivalLocation, MeetingTime, NamePassenger,EmailPassenger, NameTaxiDriver, EmailTaxiDriver, idZone, IdLog)
5. **Logs**(IdLog, TimeStamp)
6. **ManageLogs**(IdLog, NameAdmin, EmailAdmin)

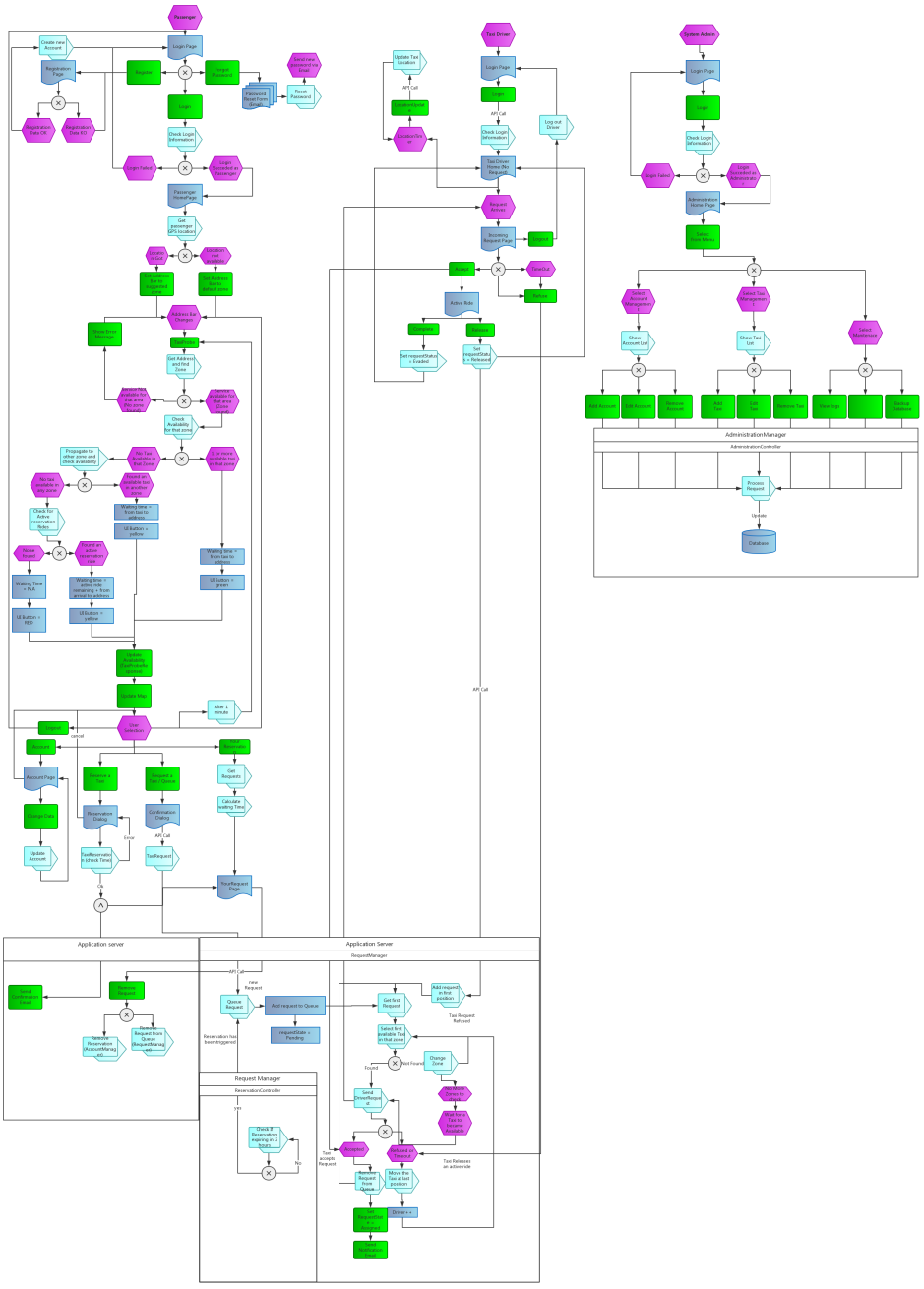
7. **Drive**(NameTaxiDriver, EmailTaxiDriver, TaxiCode, PlateNumber, Date)
8. **Account** (Name, Email, Surname, Phone, Password, TaxCode,IsTaxiDriver,Licence,EmailS, AuthToken)
9. **AuthorizedSystem**(EmailS, AuthToken)
10. **Taxi**(TaxiCode, PlateNumber)
11. **ManageTaxi**(TaxiCode, PlateNumber, NameAdmin, EmailAdmin)
12. **Reply**(idRequest, NameTaxiDriver, EmailTaxiDriver, Answer)
13. **Location**(Latitude, Longitude, IdZone)

# 3 Algorithm Design

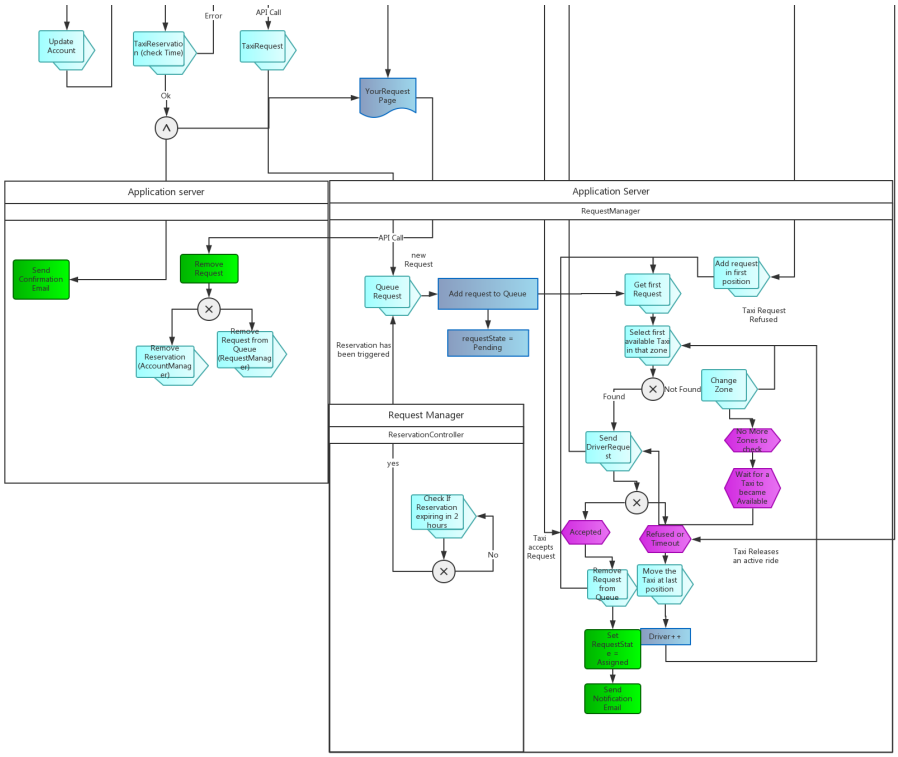
## 3.1 Overview

In this section we provide a general view of the main algorithms that must be implemented, for getting a better comprehension of how the system operates. We will then focus on the core business algorithm which is hosted in the RequestManager. We have chosen to represent the algorithms using an Event Driven Diagram, because it's easier to understand what is the context of a certain call and what is the flow of operations. Every implementation detail has been skipped for maintaining the description on a high level of abstraction.

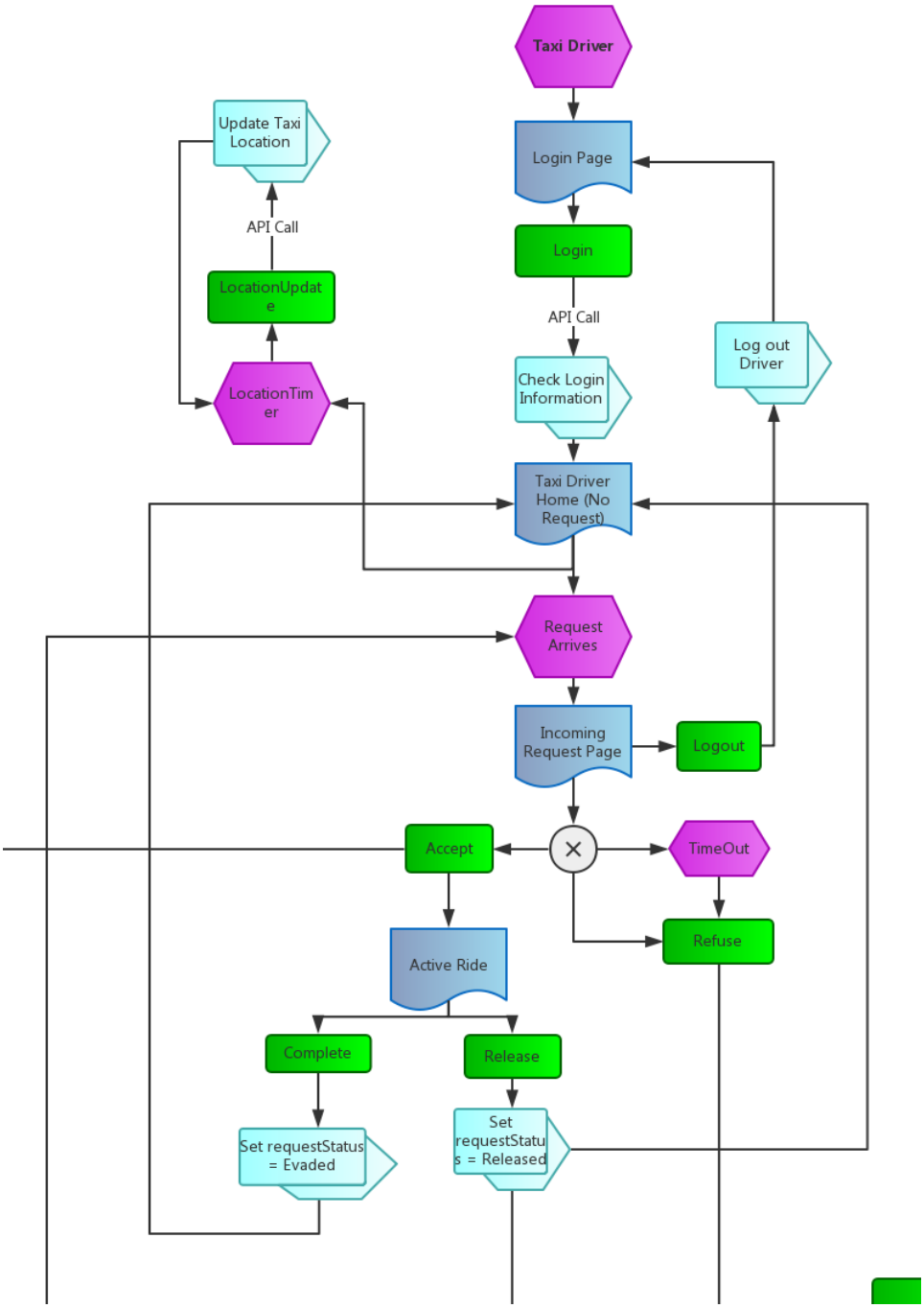
## 3.2 General process



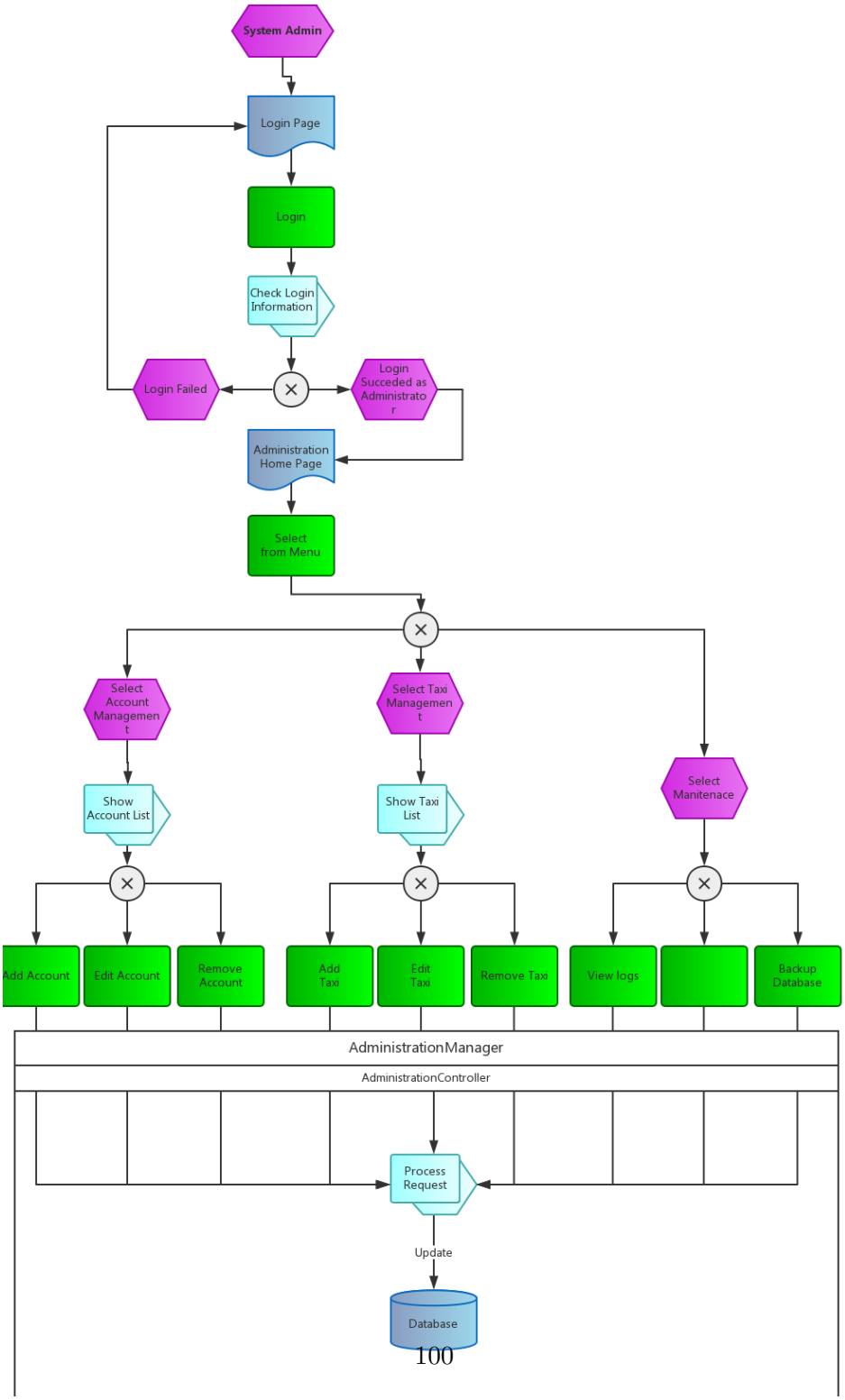
## 3.3 Request and Reservation Handling process



## 3.4 Taxi Interaction process



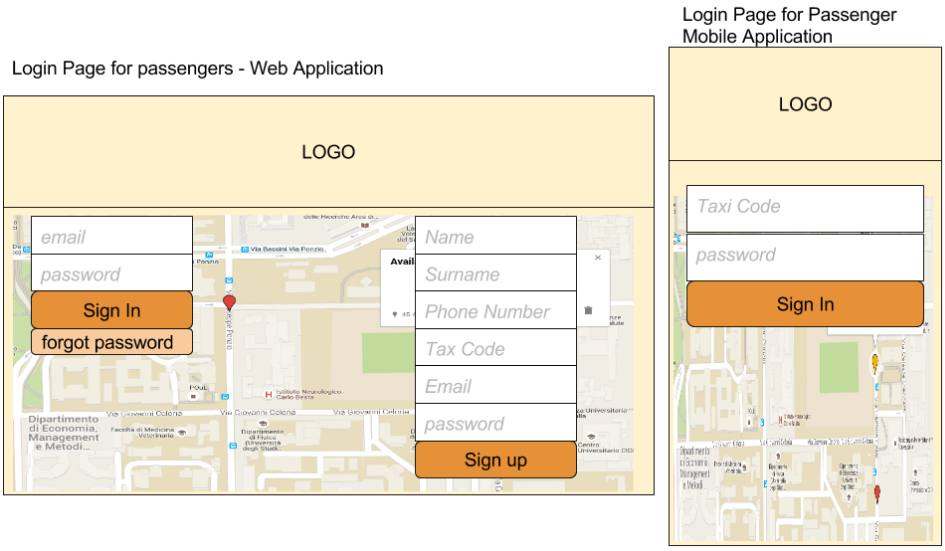
## 3.5 System Administrator process



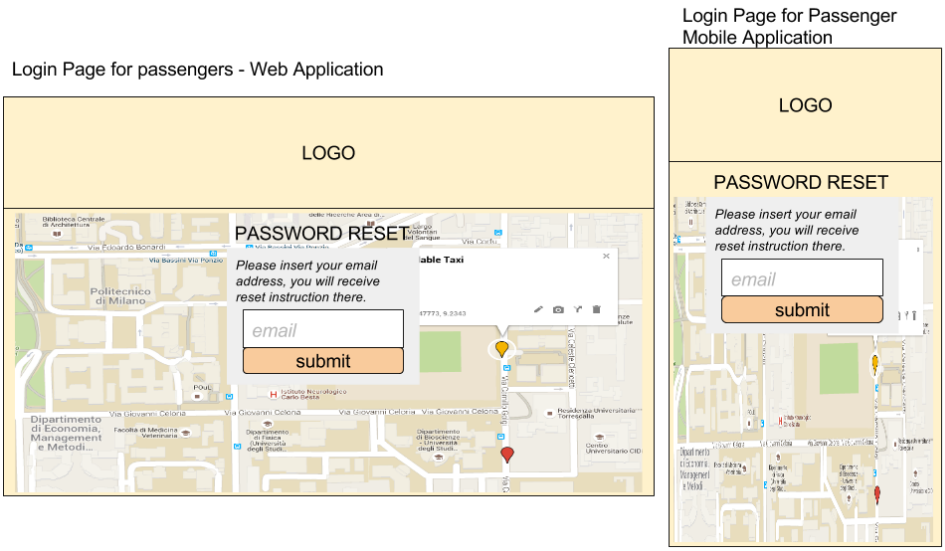
# 4 User Interface Design

This interface is a mock-up for providing the developers a starting point for the actual system look-like. The real implementation may differ from this but it should offer at least the same functionalities.

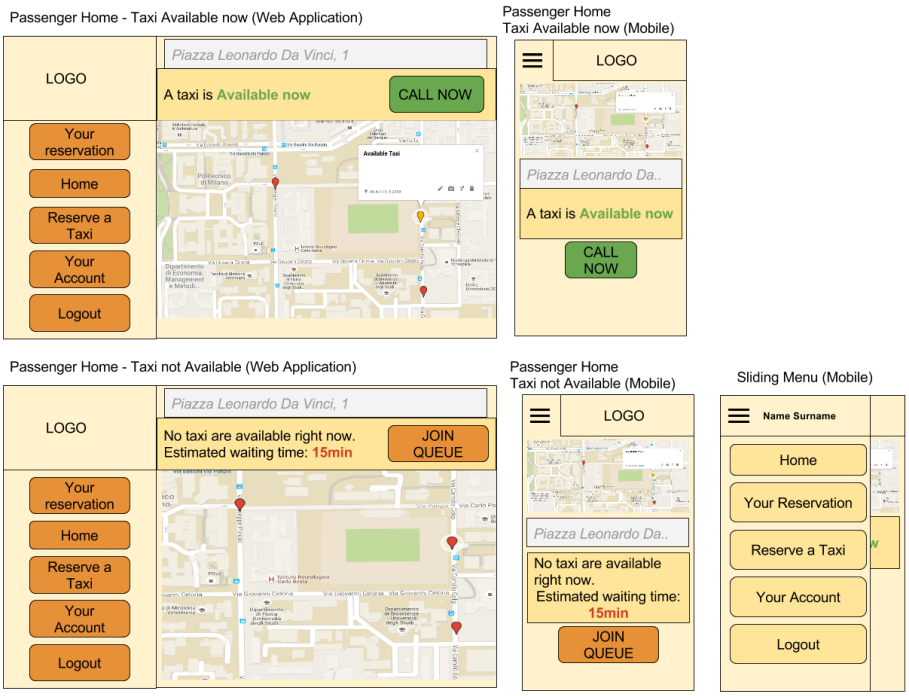
## 4.1 GUI: Login



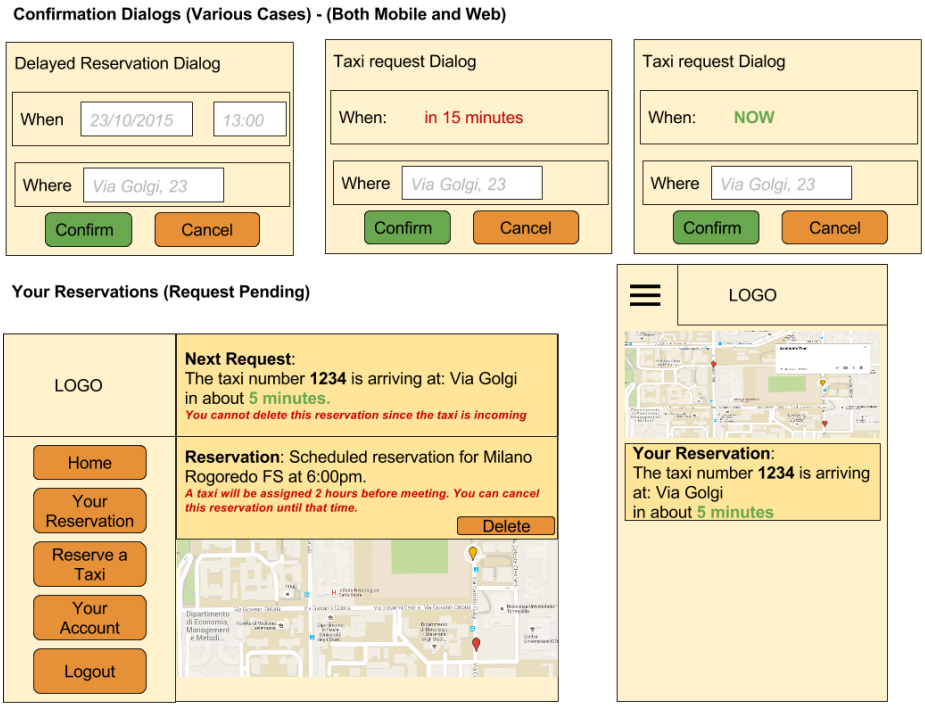

## 4.2 GUI: PasswordReset



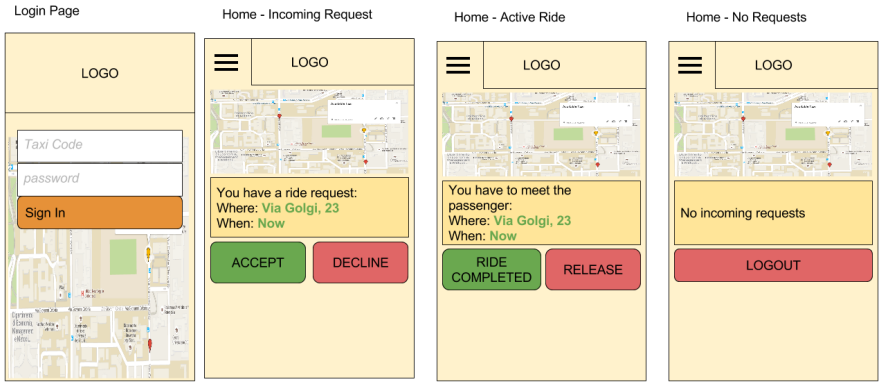
## 4.3 GUI: Passenger




## 4.4 GUI: Requests

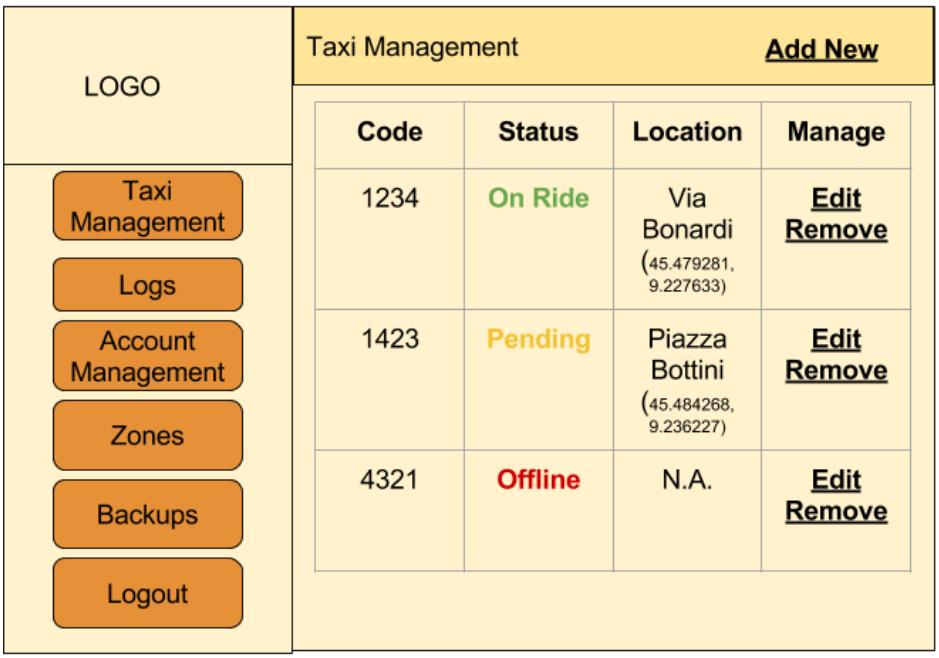


## 4.5 GUI: Taxi

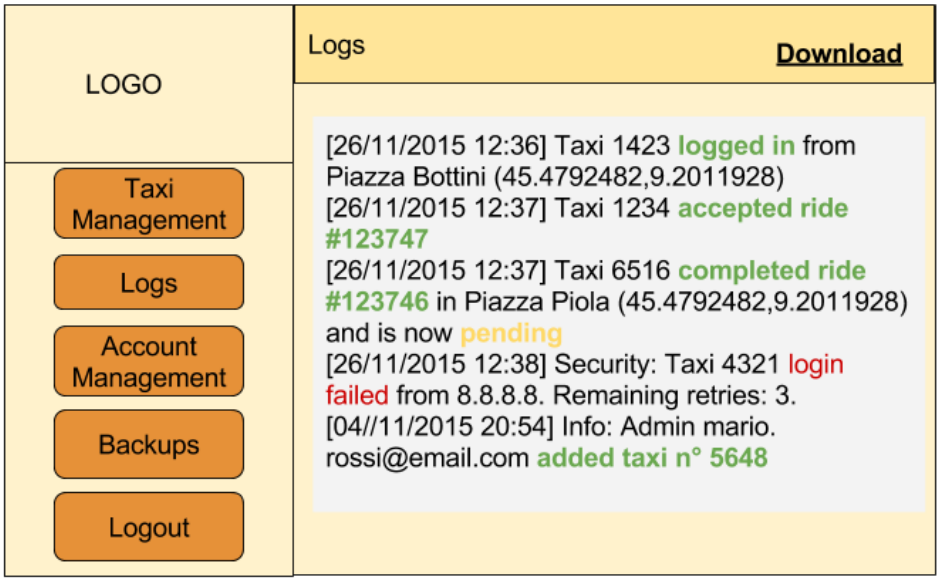

Login Page
LOGO
Taxi Code
password
Sign In
Home - Incoming Request
LOGO
You have a ride request:
Where: Via Golgi, 23
When: Now
ACCEPT
DECLINE
Home - Active Ride
LOGO
You have to meet the passenger:
Where: Via Golgi, 23
When: Now
RIDE COMPLETED
RELEASE
Home - No Requests
LOGO
No incoming requests
LOGOUT

## 4.6 GUI: Administration

**Administration - Taxi Management (Web Application)**




**Administration - Logs**



# 5 Requirements Traceability

This section will highlight the design choices that realizes the specific requirements stated in the RASD document. For each requirement, a small recap is presented and a description of the components involved will follow.

## 5.1 Product Functions Requirements

1. The system shall provide the passengers both a web and a mobile application with equivalent functionalities for requesting a ride

   This requirement is satisfied by the web interface and the mobile client interfaces.

2. The system shall provide a mobile application for taxi drivers, in which they can inform the system about their availability and confirm or refuse the proposed request.

   This is satisfied by the Taxi Client Application which interacts with the ZoneManager as shown in the component and algorithm section.

3. The system shall associate a queue of taxis to each zone according to the GPS location of the taxis

   This is satisfied by the ZoneManager component.

4. The system shall provide a response to the passenger who requested a ride, the notification will contain the code of the taxi that accepted the request and the estimated waiting time.

   This is satisfied by the NotificationManager component.

5. The system shall comprehend a programmatic interface that offer the access for the main functionality of the system itself.

   This is satisfied by the API component: After suscribing as a dependent system, a developer could build his application on top of the current system using the same system call that the clients and administrator currently use.

6. The system interface for passengers shall provide the functionality of taxi reservation for rides that will take place after 2 hours. The information required by the system include the origin and destination and the departure date.

   This is satisfied by the ReservationController in the RequestManager component.

7. The system shall keep track of any active ride, along with the current taxi position and provide an history.

   This is satisfied by the log functionalities.

8. The system should provide a web interface for system administrators, in which they can manage the taxi drivers and the passenger profiles

   This is satisfied by the administrator interface and the administration controller.

9. The system sends a remainder email or push notification to the user who reserved a taxi 2 hour before the meeting time, just after that it has assigned a taxi to that request.

   This is satisfied by the request controller and the notification manager.

10. The system should manage exceptional situations such the availability of taxis or accidents/vehicle failures.

    This is managed by the taxi driver interface, which provide the possibility to release an active drive, the TaxiController which offer the system interface and the RequestController that manages the exception as shown in the algorithm section.

## 5.2 Specific Requirements

1. The mobile application for Taxi Drivers shall transmit the taxi GPS location to the system every 30 seconds, in order to keep the taxi queues updated.

   This is satisfied by the Taxi Driver Application an the TaxiManager interface. The taxi queues will be kept updated by the zone controller

2. When a taxi becomes available, the system shall store the taxi identifier in the queue of the corresponding zone

   This is an implementation detal that must be implemented in the Zone Controller.

3. When a request arrives from a certain zone and there is at least one available taxi in that zone, the system shall forward it to the first taxi queuing in that zone.

   This is the default behaviour that is meant to obtain in the RequestController.

4. Upon an incoming request, If the Taxi Drivers confirms the system shall inform the passenger, if he doesnt confirm then the system will forward the request to the next driver

in queue and move the taxi driver which refused in the last position in the queue.

This behaviour is explained in the algorith section of this document.

5. The passenger will be informed of any unexpected event by email if he made the request via the web interface and via email and push notification if hes using the mobile application.

   This is satisfied by the NotificationManager.

6. In any case, the passenger can check the request details and information by accessing the web or mobile interface

   This is satisfied by the "YourReservation" page and the accountManager component.

7. If the taxi driver who accepted a request is unable to reach the passenger in a reasonable amount of time (i.e. due to an accident or a vehicle failure) then the taxi driver should be able to release its request. The request is forwarded to the first available taxi and the passenger will receive an update about the new taxi code and the new estimated waiting time.

   This is satisfied by the "Release" functionality of the taxi client, and managed into the core algorithm section.

8. The reservation request is accepted only if it is made at least two hours before the ride. An earlier reservation should be made impossible to select from the GUI and double checked in the backend.

   This is satisfied in the WebServer and the ReservationManager

9. For guaranteeing the precedence of reservation over real-time requests, a reservation is forwarded to a taxi 2 hours before the meeting time: in this way the taxi will result not available for eventual real-time requests (which has a maximum time length, as stated in the assumption **??**) that occur in the meanwhile and the request will be forwarded to the next taxi.

   This is managed both into the webserver and the API component: if the requirement on the time is not met, an exception will be issued and the webserver will show an error to the user.

10. Along with the taxi queue, there will be a Request Queue, with a FIFO policy. In this way, the first passenger who

makes a real-time request is the first that is served by the first available taxi.

This is satisfied by the RequestController and ZoneManager components.

11. The system should provide passengers a "password reset" service.

This is satisfied by the WebServer and the AccountManager.

12. The system should let the user cancel a previous reservation he made until 2 hours before the meeting time.

This is managed by the AccountManager, the RequestController and the YourReservations page.

13. The system will send notification to a user as soon as the request has been accepted by a taxi driver

This is satisfied by the TaxiManager interface and the NotificationManager

14. The GPS localization of the passenger is used merely for providing a better GUI experience (i.e. showing the right portion of the map) and it is an optional requirement for the passenger. The actual localization occurs by the information that the user provides by inserting the meeting location while making a request.

This is satisfied by the client interface and the OnTaxiProbe mechanism.

15. If the Taxi Driver doesn't reply to a request within a minute, the request is considered refused.

This is satisfied by the requestManager and the TaxiClient application.

# 6 References

- MyTaxiService Requirement Analysis and Specification Document

# 7 Tools and Document Information

## 7.1 Tools

**Pencil** for the GUI

**Texstudio** for the Design Document

**ProcessOn.com** for the diagrams

**MySQLWorkbench** For the ER-Schema

## 7.2 Work Hours

- Edoardo Giacomello: 65h
- Mattia Fontana: 63h