UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Web Applications A.Y. 2022-2023

Homework 1 – Server-side Design and Development

Master Degree in Computer Engineering
Master Degree in Cybersecurity
Master Degree in ICT for Internet and Multimedia

Deadline: 28 April, 2023

Group Acronym	WASCOOT	
Last Name	**First Name**	**Badge Number**
Crivellari	Alberto	2061934
Gharehzad	Shiva	123456
Huang	Borwoei	2044019
Kuijpers	Nick	2096202
Mada	Sreeshalini	2049543
Niknamhesar	Sara	123456
Tahan	Paria	2043889

1 Objectives

The objective of this project is to develop a web application that assists managers in the scooter business with their daily operations. The scope of the project was defined for the Master 1 computer engineering program and involved 10 students at the beginning, with 7 students ultimately completing the project.

The web application will allow managers to analyze customer, product, and staff data to make informed business decisions. The application will enable basic modifications such as adding or deleting data, and it is critical that the owner can perform these changes when necessary. The system will provide a comprehensive overview of all necessary information, allowing for long-term decision-making based on data analysis.

The main goal of the project is to create a platform that streamlines business processes, ultimately increasing efficiency and profitability. This application will be designed with user-friendly interfaces and interactive features, making it accessible to a wide range of users. The project will leverage modern web technologies to ensure the application is scalable and flexible enough to accommodate future business growth.

In summary, this project aims to create a web application that provides managers with the necessary tools to analyze and manage their business, ultimately improving their overall performance.

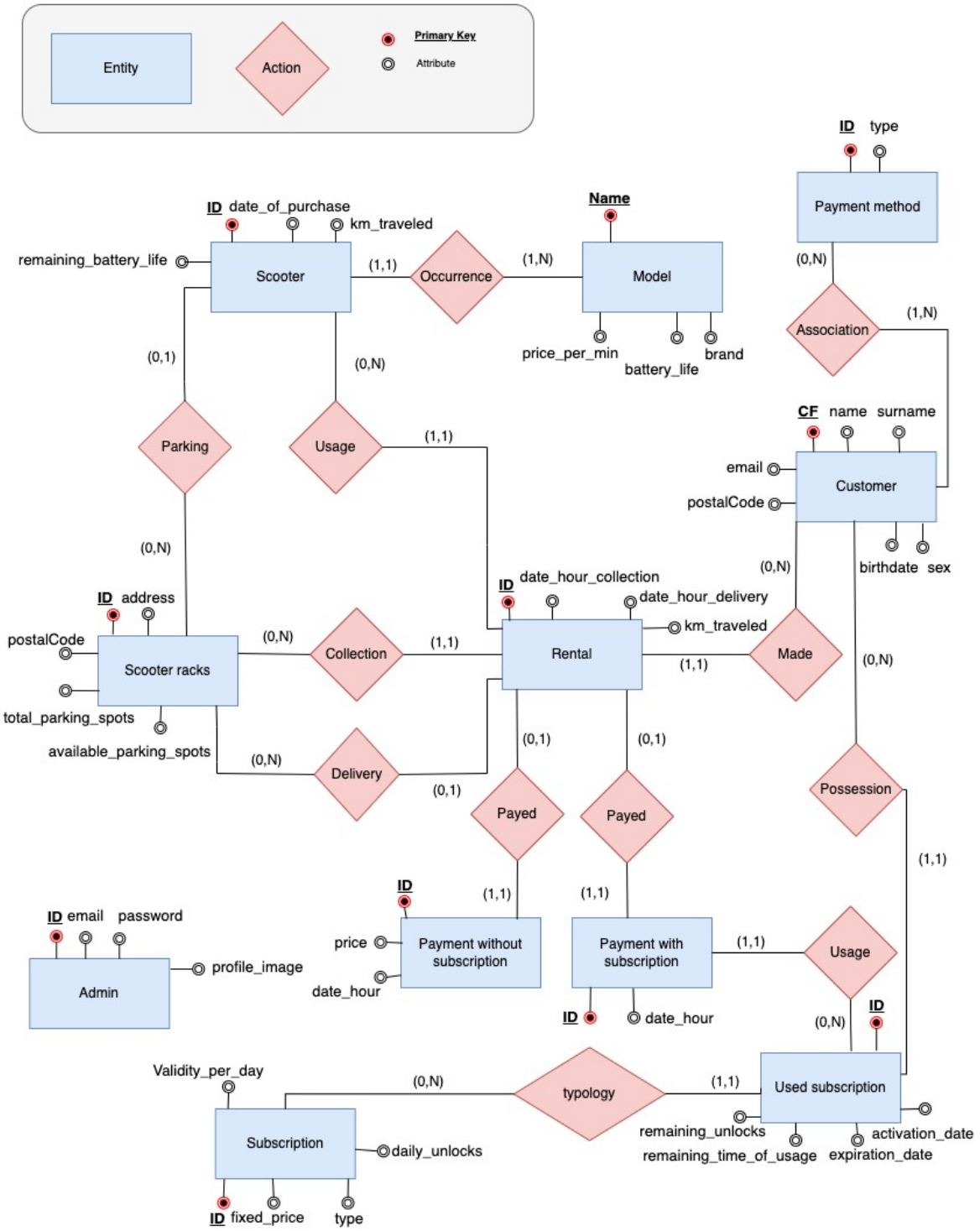
2 Main Functionalities

The web application is used to allow administrator to interact with the database of WASCOOT, a web application regarding electric scooters to rent. Also another important functionality of the web application is to show the administrators some analytics regarding the rented scooters and customers, with the objective to give some hint to improve the service. The website is divided in three main areas :

- Dashboard area: here we have our analytics and the most useful information displayed, such as the scooter racks positions;
- Login area: here administrators can login and access the web application services;
- Manage pages area: this is the core of the web application, here the administrators can interact with the database, we have one of this pages for every core table in the database (for example scooters' manage page) and from there the administrator can do a few operations :
  - LIST, list all the rows of the table, e.g.: list all the scooters in the database;
  - CREATE, create a new instance/row of that entity, e.g.: create a new scooter;
  - EDIT, edit an existant row of the table, e.g.: change something of a scooter.

3 Data Logic Layer

3.1 Entity-Relationship Schema



3.2 Data Dictionary: Entity Table

Entity	Description	Attributes	P.Key
Model	Entity representing the various models of a scooter.	• **Name**; specific type or version of the scooter; • **Price per min**; The amount of money for renting scooter for a minute; • **Brand**: model's manufacturer/company; • **Battery life**: model's battery life.	Name
Scooter	This entity represents the specific scooter. A number of scooters are provided for renters.	• **ID**: scooter identifier; • **Date of purchase**: when scooter has been bought; • **Km traveled**: distance covered by the scooter; • **Remaining battery life**: how much of the original battery life is left.	ID
Scooter racks	Structures designed to store and secure scooters when not in use.	• **ID**: scooter rack identifier; • **Address**: address of the parking spot; • **Postal code**: postal code of the rack's area; • **Total parking spots**: parking spot in this scooter rack; • **Available parking spots**: unoccupied parking spots in this scooter rack.	ID
Rental	This entity represents the process of renting a scooter.	• **ID**: id of a specific scooter rental transaction; • **Date and hour collection**: date and hour in which a scooter is rented; • **Date and hour delivery**: date and hour in which a scooter is returned and delivered; • **Km traveled**: km traveled by the customer with this rental.	ID

Customer	People registered in the application that can rent a scooter.	• **CF**: fiscal code of the customer; • **Email**; Email address of customer; • **Name**; The name of customer; • **Surname**; The surname of customer; • **Postal code**; The address of postal of customer; • **Sex**: customer's gender; • **Birthdate**: date when the customer was born on.	CF
Admin	Entity representing the information regarding application's admins.	• **ID**; Admin identifier; • **Profile image**: admin's profile picture; • **Email**; The email address of admin • **Password**. Secret access to admin;	ID
Payment Method	Entity that lists the various types of payment methods.	• **ID**; Payment methods identifier; • **Type**: name of the payment method.	ID
Payment without Subscription	Entity representing the one time payments.	• **ID**; The one-time payment identifier; • **Price**: price payed at the checkout after using the scooter; • **Date and hour**: when the payment has been received.	ID
Payment with Subscription	Entity representing payments	• **ID**; Payment identifier; • **Date and hour**: when the subscription's payment has been received.	ID

Used Subscription		• **ID**; Usage identifier; • **Remaining time of usage**: how much time until the subscription ends; • **Activation date**: when the subscription started; • **Expiration date**: when the subscription ends; • **Remaining unlocks**: how many other times can the customer unlock a scooter today.	ID
Subscription		• **ID**; Subscription identifier; • **Fixed Price**: price of this subscription; • **Type**: subscriptions' types differ based on how many days it covers (examples: 1day, 30days, 6months, ...) • **Daily Unlocks**: how many times can the customer unlock scooters in a day with this subscription. • **Validity per Day**: how many hours can the customer use scooters with this subscription.	ID

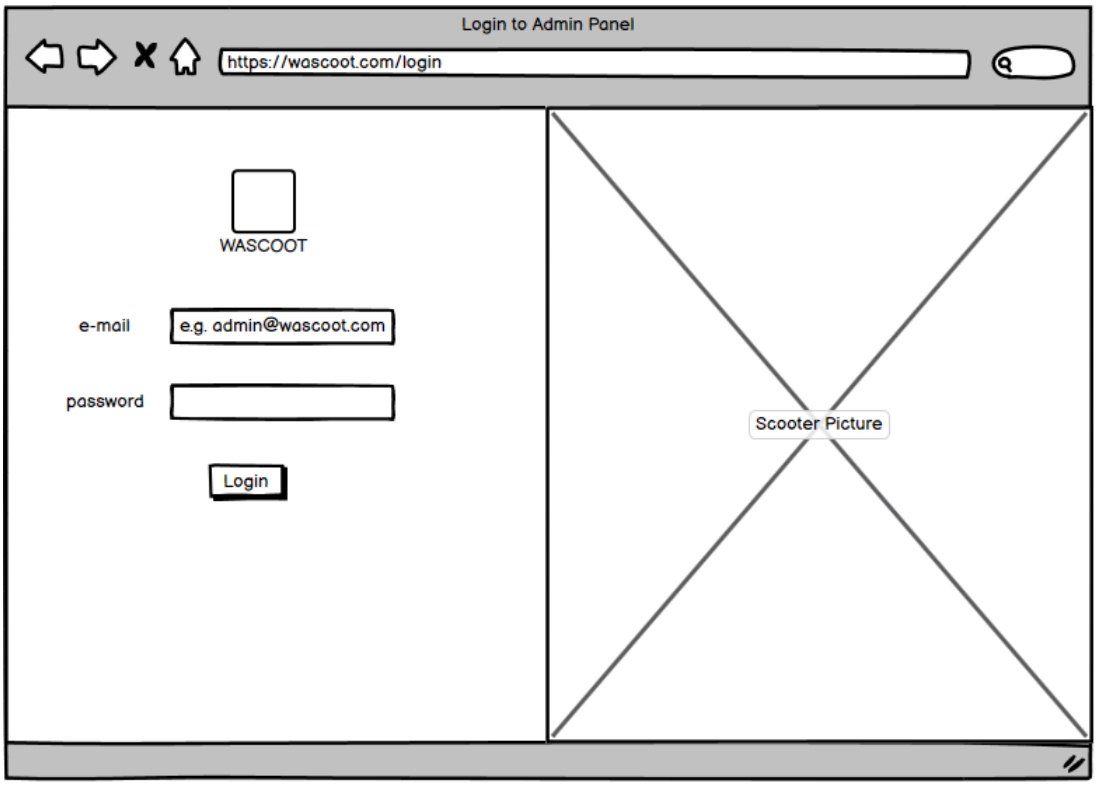
4 Presentation Logic Layer

The presentation logic layer of our web application consists of several pages that allow users to interact with the system. These pages are developed using JSP, servlets and REST technologies. We will describe the main pages.

4.1 Homepage

The homepage is the main entry point for the web application. It provides users with several options to interact with the system:

- **Login**: Users can login to the system by clicking on the "Login" button, which will direct them to the login page.
- **Create Administrator**: Users can create a new administrator account by clicking on the "Create Administrator" button. This will direct them to a form where they can enter the administrator's details, including an optional profile picture.
- **Forgot Password**: Users who have forgotten their password can click on the Forget Password "Enter ID" or "Enter Email" button, which will direct them to a page where they can enter their ID or email address to retrieve their password.

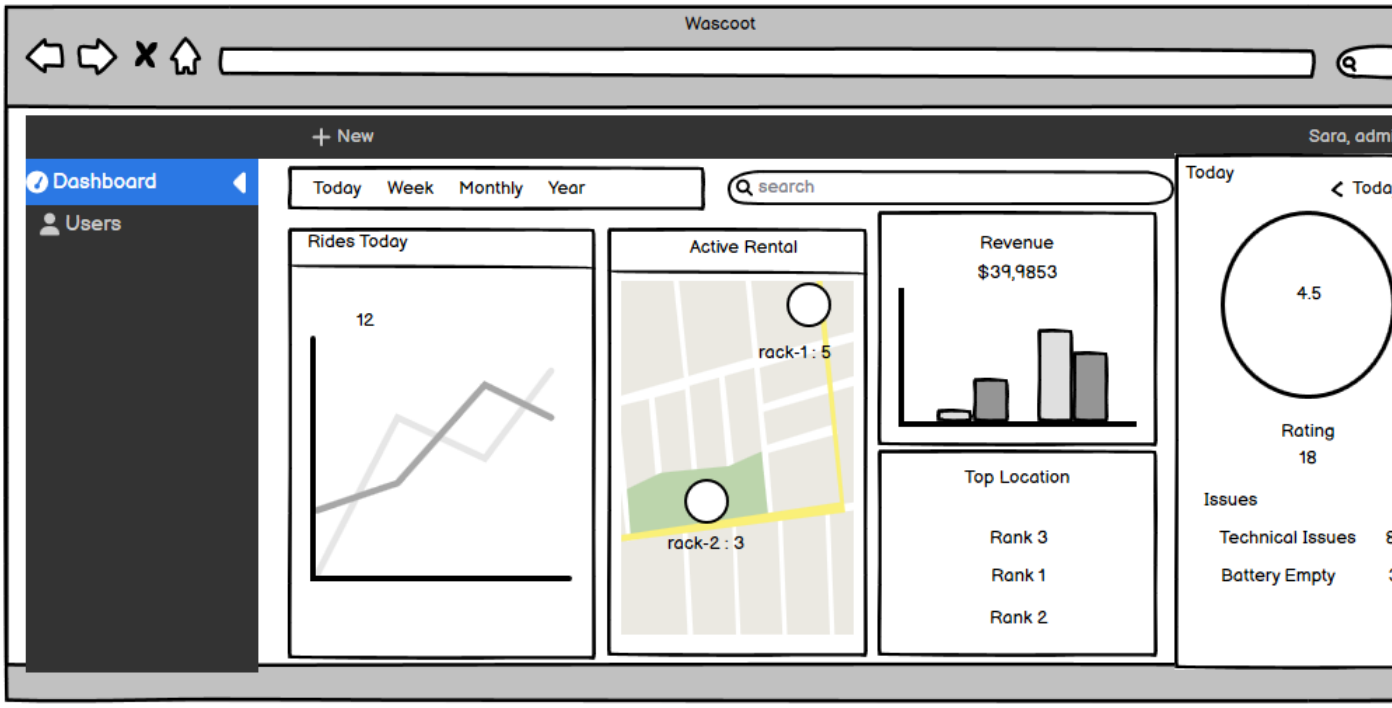


4.2 Dashboard

The dashboard is the main page of the web application. It provides users with an overview of their business data and allows them to visualize this data in various ways. The dashboard is developed using JSP and REST technologies.

On the dashboard, users can view the following information:

- **Scooterrack Locations**: Users can view the locations of their scooterracks on a map, along with information about each rack such as its capacity and availability.
- **Revenue**: Users can view their business revenue over a specified time period.
- **Top Locations**: Users can view a list of their top-performing locations based on their utilization.
- **Customer Information**: Users can view information about their customers, including age, rental history, and gender.



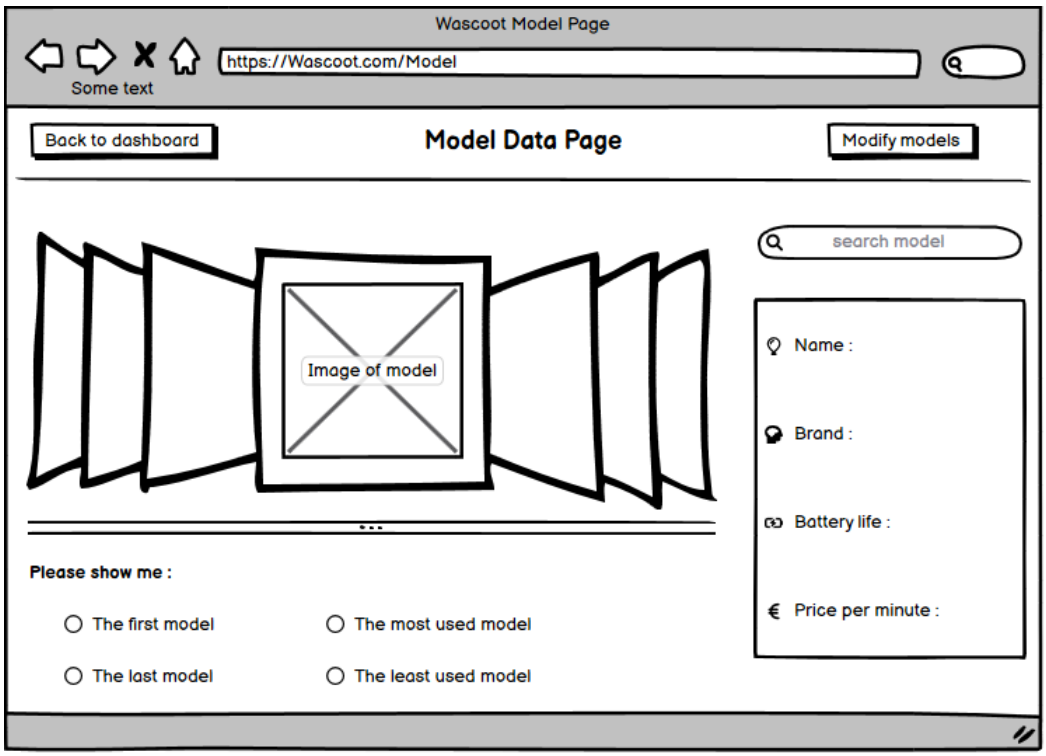
4.3 Manage Pages

In addition to the homepage and dashboard, our web application includes several manage pages that allow users to modify and create data. These pages are developed using JSP and REST technologies.

4.3.1 Model Page

The model page allows users to view information about scooter models and to modify or create new models through a form. The following pages are available:

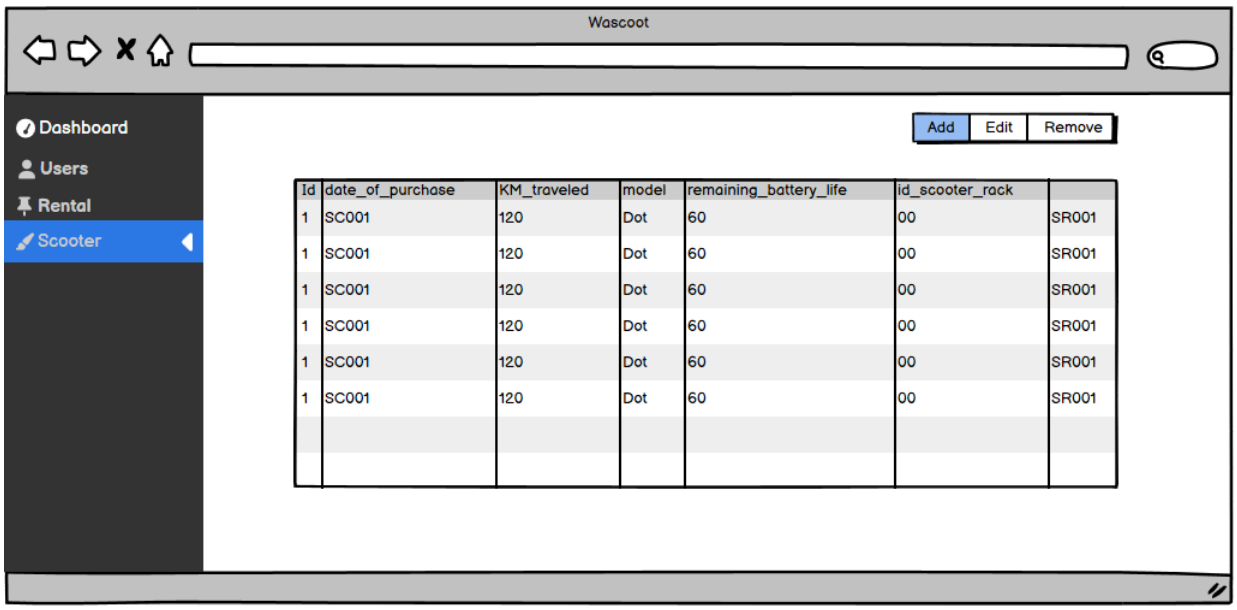
- **Show Models**: Users can view a list of all scooter models in the system, along with information such as the model name, brand, and specifications.
- **Modify Model**: Users can modify an existing scooter model by selecting it from the list and editing its information in a form.
- **Create Model**: Users can create a new scooter model by filling out a form with the model's details.



4.3.2 Scooter Page

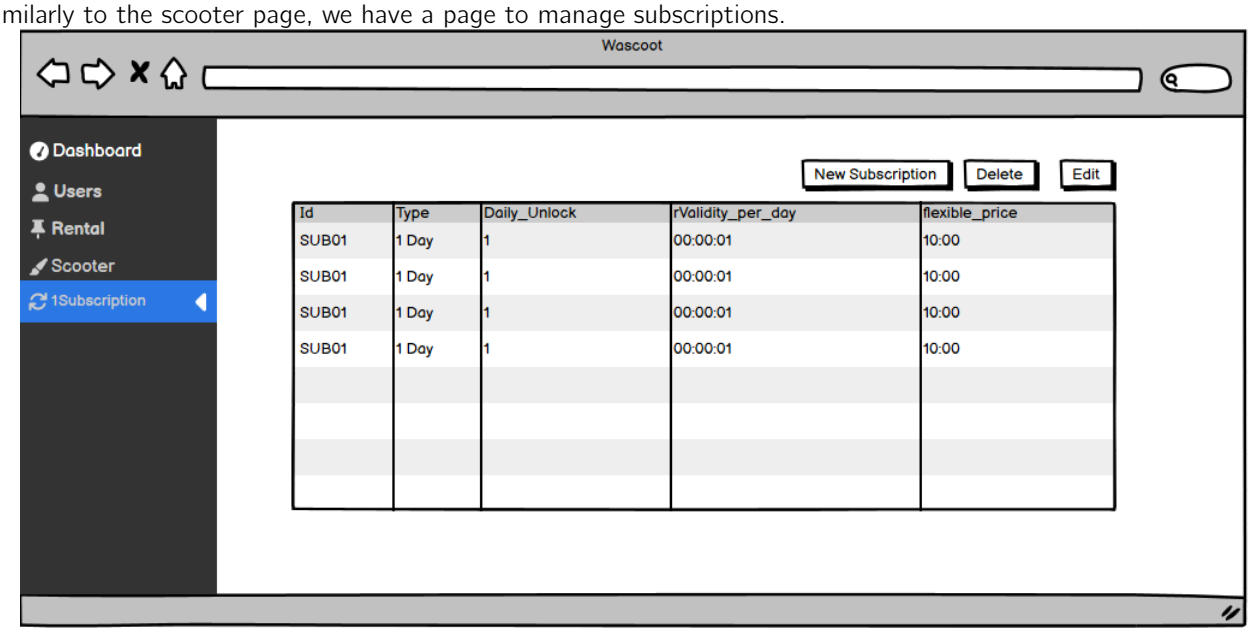
The scooter page allows users to view information about scooters and to modify or create new scooters through a form. The following pages are available:

- **Show Scooters**: Users can view a list of all scooters in the system, along with useful information.
- **Modify Scooter**: Users can modify an existing scooter by selecting it from the list and editing its information in a form.
- **Create Scooter**: Users can create a new scooter by filling out a form with the scooter's details.



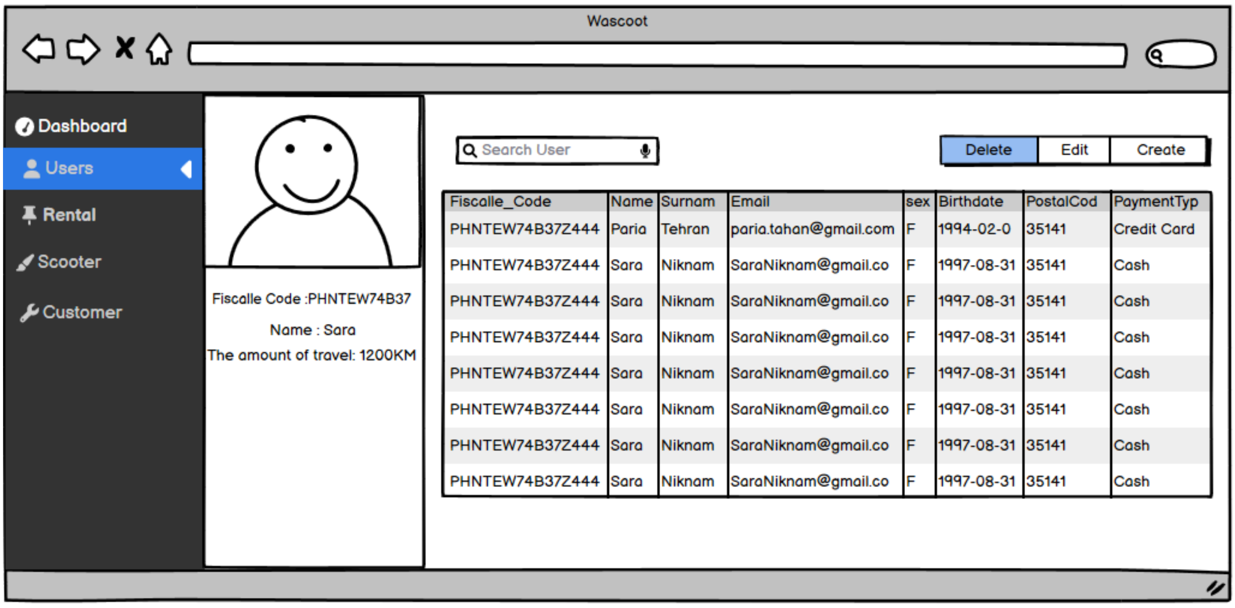
4.4 Subscription Page

Similarly to the scooter page, we have a page to manage subscriptions.



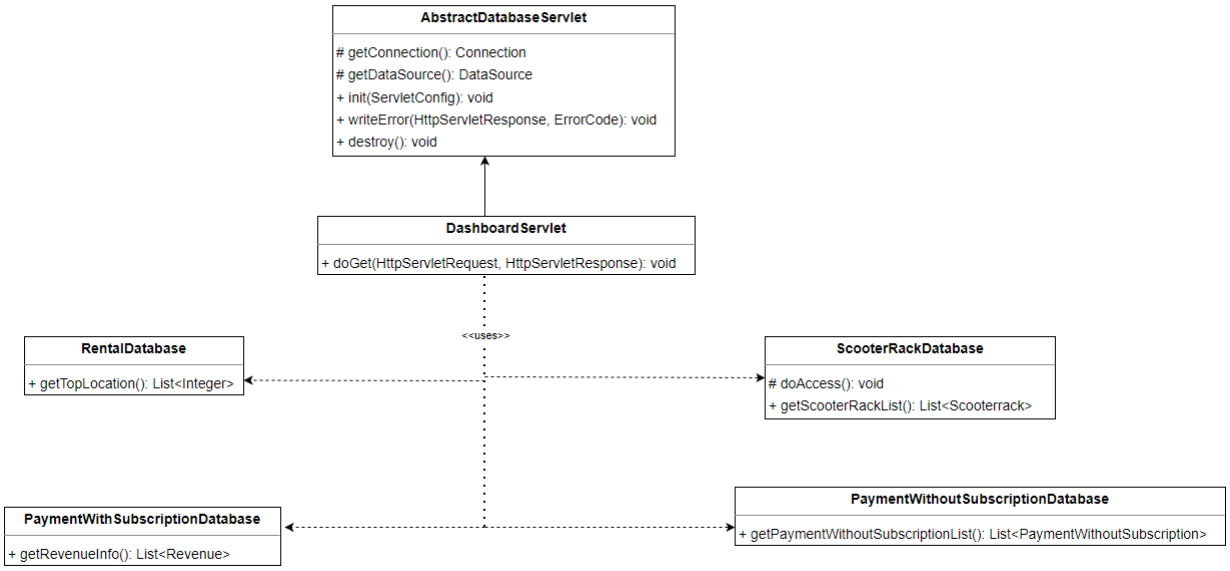
4.5 Customer Page

Similarly to scooter and subscription pages, we have a page to manage customers.

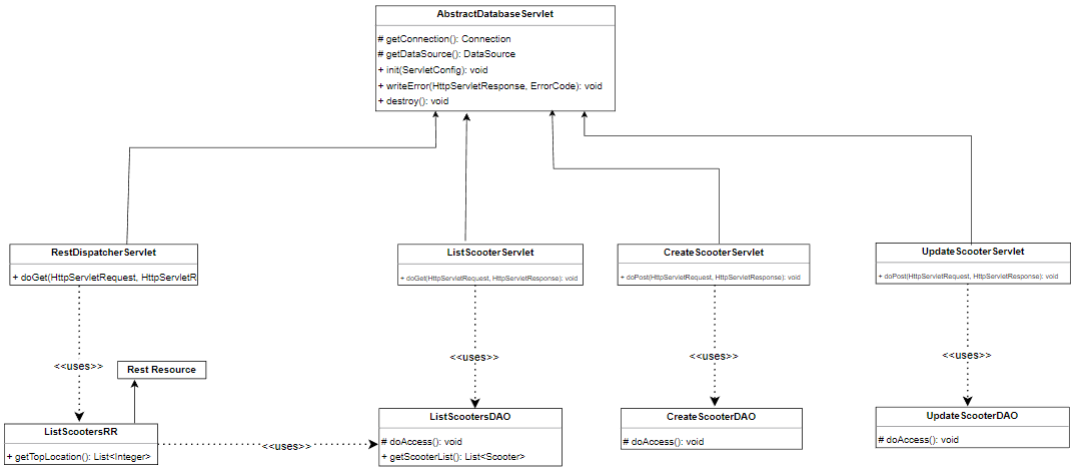

Wascoot
Dashboard
Users
Rental
Scooter
Customer
Fiscalle Code :PHNTEW74B37
Name : Sara
The amount of travel: 1200KM
Search User
Delete
Edit
Create
Fiscalle_Code Name Surnam Email sex Birthdate PostalCod PaymentTyp
PHNTEW74B37Z444 Paria Tehran paria.tahan@gmail.com F 1994-02-0 35141 Credit Card
PHNTEW74B37Z444 Sara Niknam SaraNiknam@gmail.co F 1997-08-31 35141 Cash
PHNTEW74B37Z444 Sara Niknam SaraNiknam@gmail.co F 1997-08-31 35141 Cash
PHNTEW74B37Z444 Sara Niknam SaraNiknam@gmail.co F 1997-08-31 35141 Cash
PHNTEW74B37Z444 Sara Niknam SaraNiknam@gmail.co F 1997-08-31 35141 Cash
PHNTEW74B37Z444 Sara Niknam SaraNiknam@gmail.co F 1997-08-31 35141 Cash
PHNTEW74B37Z444 Sara Niknam SaraNiknam@gmail.co F 1997-08-31 35141 Cash
PHNTEW74B37Z444 Sara Niknam SaraNiknam@gmail.co F 1997-08-31 35141 Cash

5 Business Logic Layer

5.1 Class Diagrams

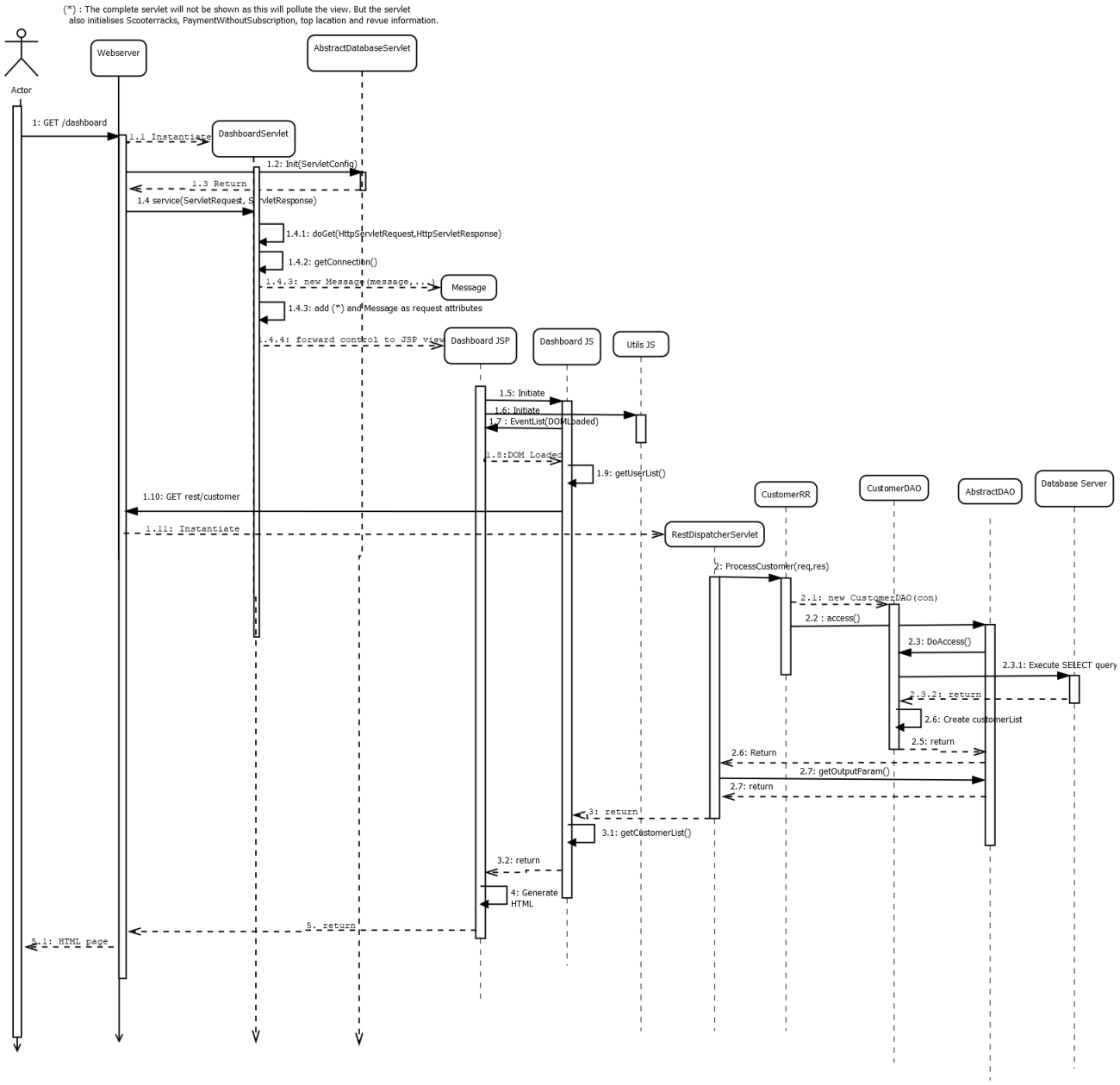


This class diagram represents the classes involved in the dashboard servlet, which manages the dashboard page. This servlet uses the classes shown to access the database.



This second class diagram represents one of the manage pages servlets and rest. The manage pages are model, scooter, scooterrack, rental, payment methods, customer and subscription. On this pages we can interact with the respective tables of the database, for example viewing the list, adding or editing that entity. In this case we shown the scooter entity, we have a servlet for viewing the list of scooters, one for adding a new scooter and one for editing an already existant scooter. Then we also provide the REST API for getting the list of scooters. Both the rest and the servlets use some classes to access the database, shown in the diagram.

5.2 Sequence Diagram



This sequence diagram depicts the communication flow for a REST API call from a JSP page (Dashboard) in our web application. The objective of this diagram is to illustrate the specific steps involved in the communication flow between the different components involved in this process.

At the start of the sequence, an actor sends a request to the WebServer for the Dashboard page. The WebServer forwards this request to the DashboardServlet, which in turn renders the DashBoardJSP page. The DashBoardJSP page loads the necessary DashBoardJS script, which makes a HTTP GET request for retrieving customer data from the REST API (/rest/customer).

The RestDispatcherServlet, which is responsible for handling all REST API requests in the web application, delegates the handling of the request to the CustomerRR class, which interacts with the CustomerDAO class to

retrieve the customer data from the database. The customer data return to the CustomerRR object, which returns it to the RestDispatcherServlet. The RestDispatcherServlet then sends the response back to the dashboard js call, which in turn passes it to the last function (getCustomerList) which updates the HTML page.

Overall, this sequence diagram shows the complex interaction between different components involved in a REST API call in our web application, and how each component plays a specific role in this process. It was challenging to comprehend this communication flow, and this diagram helps to provide a visual representation of how the different components work together to retrieve and display data from the REST API.

5.3 REST API Summary

The list of the rest API for this homweork are still "in progress" because we have to decide according to the frontend where is better to use a REST call or not. There are some resources we used for rest in the next table.

URI	Method	Description
rest/scooter	get	Returns the list of scooters available in the database
rest/scooterrack	get	Returns the list of scooterracks available in the database
rest/model	get	returns the list of models available in the database
rest/customer	get	returns the list of customers available in the database
administrator/id	GET	Returns the ID of the administrator from the database
administrator/id	POST	Allows to insert the ID of the new administrator into the database
administrator/id	DELETE	Allows to delete the ID of the administrator from the database
administrator/id	PUT	Allows to update the list of administrators in the database
administrator/email	GET	Returns the Email-ID of the administrator from the database
administrator/email	POST	Allows inserting the Email-ID of the new administrator into the database
administrator/email	DELETE	Allows deleting the Email-ID of the administrator from the database
administrator/email	PUT	Allows updating the list of administrators in the database
administrator/password	GET	Returns the password of the administrator from the database
administrator/password	POST	Allows inserting the password of the new administrator into the database
administrator/password	DELETE	Allows deleting the password of the administrator from the database
administrator/password	PUT	Allows updating the list of administrators in the database
login/email	GET	Returns....
login/password	GET	....

5.4 REST Error Codes

Here is the list of errors defined in the application.

Error Code	HTTP Status Code	Description
-103	BAD_REQUEST	Email missing
-104	BAD_REQUEST	Password missing
-105	BAD_REQUEST	Submitted credentials are wrong
-200	BAD_REQUEST	Operation unknown
-999	INTERNAL_SERVER _ERROR	Internal Error
-E5A1	INTERNAL_SERVER _ERROR	unable to serve the REST request
-E4A1	BAD_REQUEST	Accept request header missing
-E4A2	NOT_ACCEPTABLE	unsupported output media type. resources are represented only in application/json
-E4A3	BAD_REQUEST	Input media type not specified. content-type request header issing
-E4A4	BAD_REQUEST	unsupported input media type. resources are represented only in application/json.
-E4A5	METHOS_NOT _ALLOWED	unsupported operation
-E5A1	INTERNA_SERVER _ERROR	Cannot create the administrator: unexpected error.
-E4A8	BAD_REQUEST	cannot create the administrator: no administrator JSON object found in the request
-E5A2	CONFLICT	cannot create the administrator. it already exists
-E10A1	INTERNAL_SERVER _ERROR	cannot list customers: unexpected error
-E5A3	NOT_FOUND	administrator Not founD. cannot delete it.

Table 4: Describe in this table your REST API

6 Group Members Contribution

Alberto Crivellari: implemented rest of scooter, scooterrack and model, wrote the section regarding class diagrams, data dictionary:entity table and main functionalities, and helped with rest api section