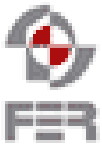


PedalaMi
Design description

Version 1.2

Revision History

Date	Version	Description	Author
2021-11-12	1.0	First version	Entire Team
2021-12-10	1.1	Minor changes to the first version. High level system description revisited and link to the interfaces documentation added	Vincenzo Riccio
2022-01-13	1.2	Fixing a drawing error	Vincenzo Riccio

1. Introduction

1.1 Purpose of this document

The purpose of this document is to illustrate the system design and architecture. It will be continuously updated during the project development process by delivering more details for each architectural unit and its desired functionalities.

1.2 Document organization

The document is organized as follows:

- Section 1, *Introduction,* describes the contents of this document and its scope, including the references used during the development process.
- Section 2, *High-level system description*, describes the system and its components at a high-level
- Section 3, *Software architecture and design*, describes the lower level architecture of the project
- Section 4, *Graphical User Interface*, shows mockups of the initial project's design

1.3 Intended Audience

The intended audience is:

- ***Team members***, the development team should have full access to this document, to develop the required solution using intended design approach and chosen technologies.
- ***Supervisors***, the professors from POLIMI and MDH should have full access to this document, to evaluate and grade the progress of the development as well as to advise the team where it is needed.
- ***Customer***, Deloitte Digital, to whom deliver the final product. They should have full access to this document, to revise the development progress with the compliance of the Product Owner.

1.4 Scope

This document covers the design and architecture of the project software, describing its components and interactions between them, as well as the software development tools and technologies that are going to be used to develop the desired application.

1.5 Definitions and acronyms

1.5.1 Definitions and synonyms

Keyword	Definitions
Mockup	An high-level model to give an idea of the design the application will have
NoSQL	Non-Relational (database)
App	(Mobile) Application
Web App	Web application for the dashboard
Hybrid App	Is a software application that combines elements of both native apps and web applications
Competition	Synonym of event

1.5.2 Acronyms and abbreviations

Acronym or abbreviation	Definitions
UML	Unified Modeling Language
GUI	Graphical User Interface
API	Application Programming Interface
REST / RESTful API	API that adheres to the REST architectural style
JSON	JavaScript Object Notation

1.6 References

- *Project Plan*: https://www.fer.unizg.hr/_download/repository/Project_vision_and__plan_document.pdf
- *Requirements definition document*: https://www.fer.unizg.hr/_download/repository/Requirements_Definition_Document%5B8%5D.pdf
- *Firebase Database*: https://firebase.google.com/docs/database
- *MongoDB Database*: https://www.mongodb.com/
- *Flutter:* https://flutter.dev/
- *Draw.io:* https://app.diagrams.net/
- *Figma software*: https://www.figma.com/

2. High-level system description

2.1 Project background

PedalaMi project aims for an easy and user-friendly design as well as to offer the best possible user experience.

This project promotes the use of bikes as a green and sustainable alternative to the non-zero-emissions means of transport, which still represent the most preferred choice in the field. This objective is pursued by providing the users with an interactive system that encourages cycling thanks to a fun reward and challenge-based system, in order to enrich the standard riding experience to make it more appealing to casual bikers.

2.2 Functionalities overview

As can be evicted from the background, the main actor of the system is the user. He mainly uses the system in order to get **points** while riding the bike in order to redeem **rewards** offered by the application. The main way to reach this goal is to record a ride using the mobile application. At the end of the ride the user is rewarded with points, which are calculated based on the data retrieved from the ride recording. After each ride, the system takes care of updating the statistics of the user and of possibly assigning them badges.

Badges are specific rewards added to the user profile which attest that the user has completed a specific challenge, like “ride for 10km in a day”. Besides being trophies to show on the personal profile, badges can also be required to access exclusive events, which are later on described in detail.

Users can also create and join **teams**, which are meant to allow them to compete as a team and to get more points by participating in team events.
In fact, users can join events to earn more points. Events are of two main types:

- **individual events**, which each user can join individually. They are public and they are created by PedalaMi admins. The final prize is won by the first player in the event scoreboard. How the scoreboard is maintained will be later specified.
  A special type of individual events are the exclusive events. They are individual events which are restricted to users who have obtained a certain badge after completing specific challenges.
- **team events**, which are dedicated to the users who joined a team. First the team admin subscribes to a team event, then each user can decide whether or not to join the event as a member of the team. How the scoreboard is maintained and how the prizes are assigned will be later specified. There are two types of team events: **public team events** and **private team events**.
  - *Private team events* are restricted to two teams: the hosting team, that is the team whose admin created the event, can invite another team, the guest team, to join the event in a 1vs1 competition.
  - *Public team events* are proposed by a team admin and must be approved by a PedalaMi admin in order to be public and to be joined by other teams as an all vs all competition.

After collecting enough points, users can spend part of them to redeem rewards, which are provided by the system admins.

2.3 Domain Model Diagram

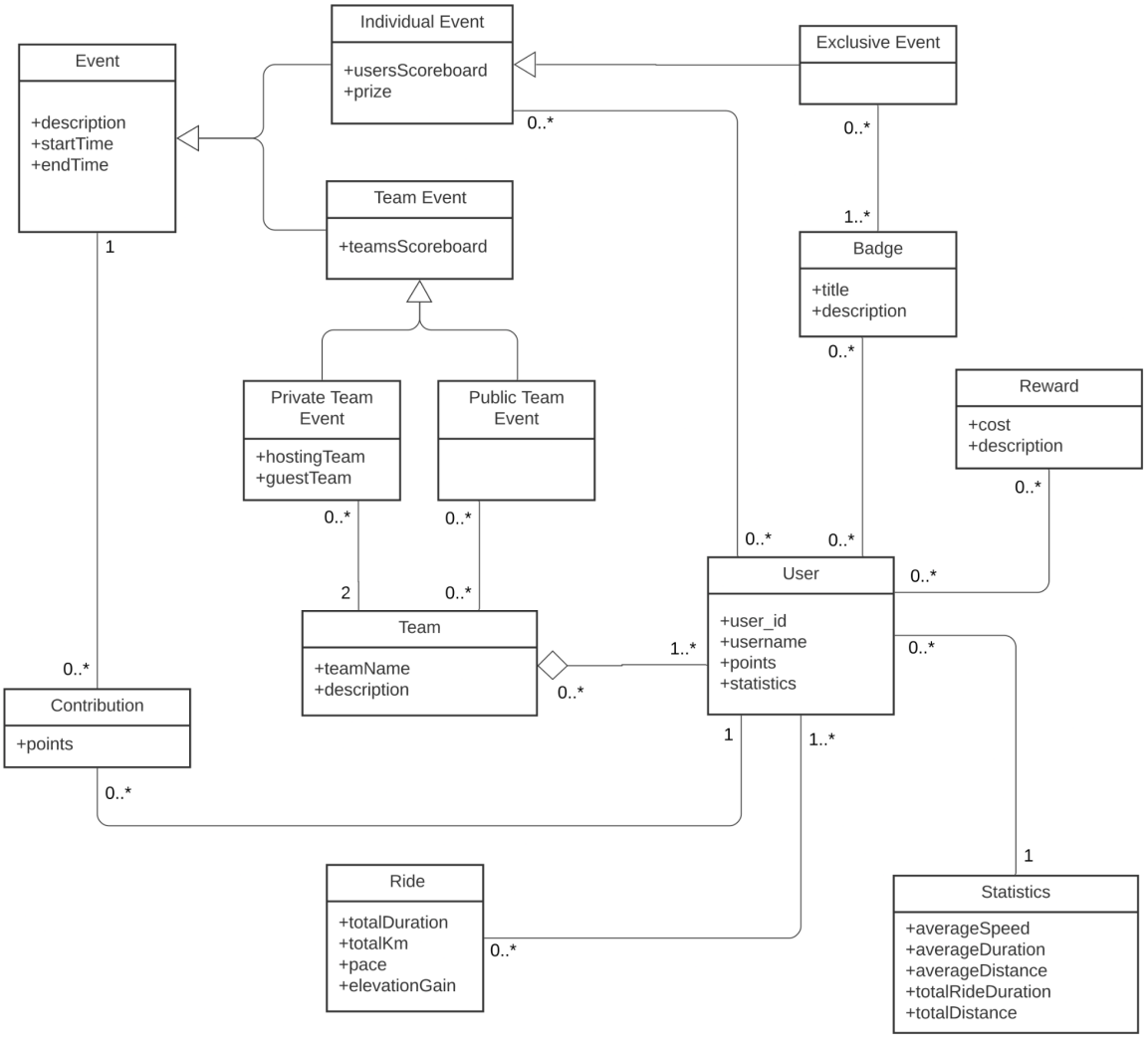


Figure 1: Domain Model Diagram

The above Domain Model diagram is a conceptual model of the system defining the main entities of PedalaMi and the relationships among them. It is meant to be a guideline during the development process, helping developers in visualizing how the domain entities are represented in the system.

Generalizations are used to distinguish the different types of events already described before.

In addition to what has been described in the previous section, the *Contribution* entity has been introduced to model the system. It is used to represent the contribution of users to each event: each user has to contribute to an event in terms of points, in order to keep track of the effort the user is putting in winning the competition.

2.4 High-level overview of the system

PedalaMi application is modelled according to a multi-tier client-server architecture. The software back-end will be implemented using NodeJs and Heroku as a cloud-based application platform, while the data storage will be split into two non-relational document-based databases, MongoDB and Firebase.

The Flutter framework will be used to develop a hybrid mobile application and a web page that visualizes a dashboard containing ride statistics. This is because Flutter gives the opportunity to have a single code for hybrid development. Moreover, it gives efficient and effortless maintenance to all platforms simultaneously. Front-end parts will interact with the back-end system using RESTful API.

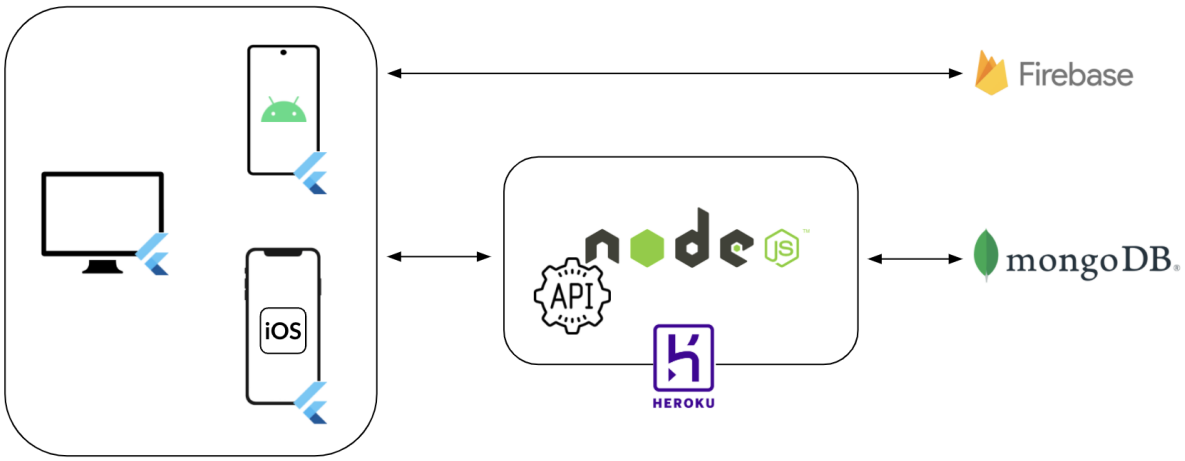


Figure 2: High-level overview of the entire system

A more detailed description of the architecture is provided in section 3.

3. Software architecture and design

The system has 3 main components:

- the Hybrid mobile application and a Web page
- a Cloud-based platform hosting the server
- Realtime and NoSQL Databases

3.1 Front-End

The main components of the front-end part are the hybrid mobile application and the webapp. The main objective of the **hybrid application** is to collect data during users' bicycle rides and to provide a system that allows users to join teams and compete with others. In order to calculate statistics and to track users' rides, the system relies on the Maps Service provider and the devices' sensors.

The **web page** will be created to visualize the statistics on a dashboard from each ride that the user has completed or the overview of all of the rides and accomplishments displayed for that particular user.

3.2 Back-End

The back-end is based on Node.js, an open-source, cross-platform runtime environment and is hosted on Heroku, a container-based cloud Platform as a Service (PaaS).
It is composed of a central server logically organized into different components, each with a different purpose.

The back-end task is to manage and respond to requests sent by the application and the web app and it contains the application logic. In particular, it includes the algorithms which deal with the computation of statistics and with the assignment of points based on the statistics of the bike rides. To do this, it communicates with the MongoDB non-relational database.

3.3 Databases

Regarding the management of system data and users' personal information, it was decided to use two distinct databases: MongoDB and Firebase, the first to store all the information necessary for the functioning of the system; the latter is instead used for easier management of authentication and for the storage of relevant personal information about users.

Firebase is a platform developed by Google for managing applications. In our system, Firebase is used only to effectively and securely manage authentication and to manage the personal account information of user profiles.

MongoDB is a document-based non-relational (NoSQL) database program. In PedalaMi, it is used as the main database for saving bike rides, teams and points system data. The database is hosted on the Cloud through the MongoDB Atlas platform, in line with the design choice referring to the use of Heroku to host the Node.js server.

3.4 Component Diagram

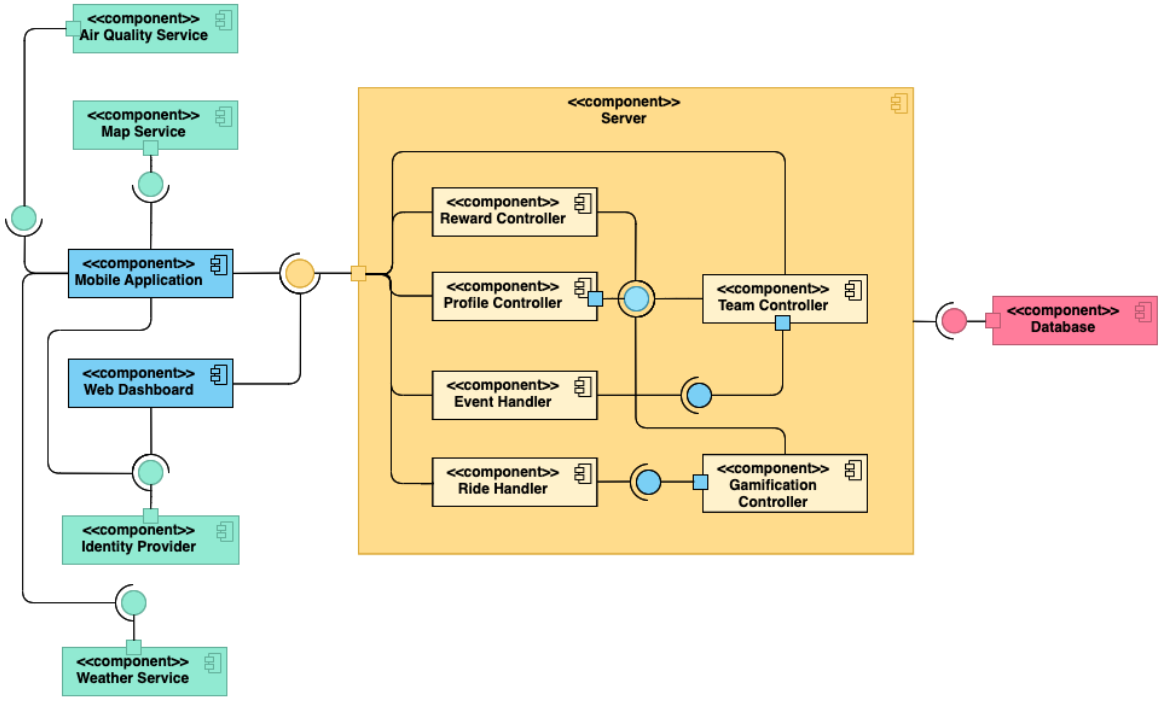


Figure 3: Component Diagram

The system architecture is further represented by the Component Diagram shown above, in Figure 3. The presentation tier, made by the mobile and the web application, is marked in blue; the application tier, represented by the server, is marked in orange; the data tier is marked in red. The external components are marked in green.

It also depicts the application server's internal structure in terms of components and their interfaces. Since the diagram focuses on the application layer, the remaining layers (presentation and data) are represented as black boxes.

The database component represents the combination of two separate databases. In fact, as already said, Firebase is used to save users' personal data such as the user's profile image and the username and to authenticate them, while MongoDB is used to store all the data linked to rides and events.

The mobile and web application communicate with the backend using REST API provided by the server, while the server interacts with MongoDB using its proprietary interface.

Further, there will be explained the following roles of the components in more detail.

Reward controller
It is the component in charge of providing the user with the list of rewards and of allowing him to redeem the desired reward.

Profile controller
It is the component in charge of letting the user login into the system and edit his profile. It also provides the user with his statistics and general profile information.

Ride handler
It is the component in charge of handling the elaboration of a new ride recording. It computes the ride data, updates the user's statistics and then informs the gamification controller of the new ride.

Gamification controller
It is the component in charge of dealing with the points assignment process given the data of a ride, taking also into account what was the weather like during the ride and what events the user is enrolled in, if any, to compute his contribution to each event.
It possibly assigns badges to the user.

Team controller
It is the component in charge of creating and retrieving teams and of allowing users to join them. It also checks if a user is the admin of a certain team.

Event handler
It is the component in charge of letting team admins and system admins create new events and manage the already existing ones. It is also in charge of allowing users to discover the available events they are allowed to enrol for.

External services
The **Air Quality Service** will be used to check how good the air quality is thereby, informing the user about the health risks of starting a new ride in those conditions.
The **Map Service** will be used to track the bike ride and to retrieve raw data of the ride.
The **Identity Provider** will be used for handling authentication for the login of the user and during its registration.
The **Weather Service** will be used to check how the weather is, then rewarding the user more points if biking in adverse weather conditions.
The providers of the mentioned external services will be selected during the implementation.

3.5 Documentation of the interfaces

In order to facilitate the maintainability and the readability, the interfaces will be documented using Swagger, an open-source software to design, build and document RESTful web services. The documentation is integrated in the server and is available at:
https://pedalami.herokuapp.com/api-docs/

3.6 Mapping between requirements and components

ID	Description	Involved Components
R1	The application must have authentication, allowing users to login	Identity provider, Profile Controller
R2	The application must allow users to record bike rides	Ride Handler
R2.1	The app should allow users to see the rides the made	Ride Handler
R3	The application must collect users' rides statistics and display them	Ride Handler
R4	The application must allow users to edit their personal profile	Profile Controller
R5	The application must allow users to earn points from their rides	Ride Handler, Gamification Controller, Profile Controller
R6	The application must assign badges to the users as a reward, based on their rides' statistics	Ride Handler, Gamification Controller
R7	The app should allow users to redeem rewards	Reward Controller, Profile Controller
R8	The app should display information about the redeemed rewards	Reward Controller
R9	The application should allow users to create teams and become team admins	Team Controller, Profile Controller
R10	The application should allow users to search for teams and join them	Team Controller, Profile Controller
R11	The application should allow team admins to create private team events	Event Handler, Team Controller
R12	The application should allow system admins to create individual public events	Event Handler
R13	The application should allow system admins to create exclusive public events	Event Handler
R14	The application should allow a Team admin to propose a public team event that has to be approved by a System Admin	Event Handler, Team Controller
R15	The application should display to the user the available	Event Handler

	events and allow users to enrol for them	
R16	The application could allow team admins to invite other teams to join private events	Event Handler, Team Controller
R17	The app should give extra points to the user if they bike in adverse weather	Ride Handler, Gamification Controller, Weather Service, Profile Controller
R18	The web-app should display all the user's statistics	Profile Handler
R19	The app could alert the user when the air quality is low	Air Quality Service

3.7 Sequence Diagrams

The following sequence diagrams are high-level diagrams created according to the component diagram presented earlier in section 3.4, and their purpose is to show the intended flow of actions originating from the user interaction with the system. However, these diagrams do not address any kind of exceptions thrown during the interaction.

3.7.1 Record a Ride

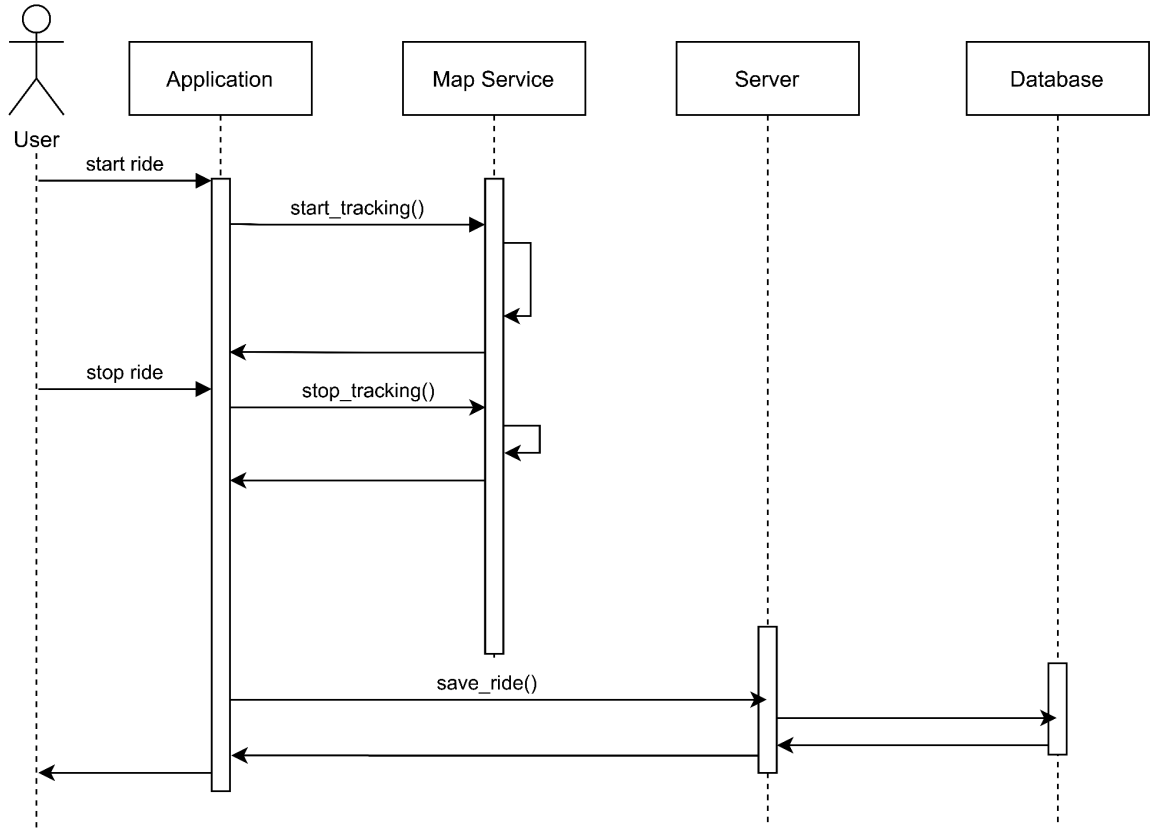


Figure 4: Sequence Diagram - Record A Ride

When the user wants to record a ride, the application keeps track of him with the Map Service and, when he wants to stop that ride, the application sends all the data to the server, which elaborates and saves them on the database.

3.7.2 Check User Statistics

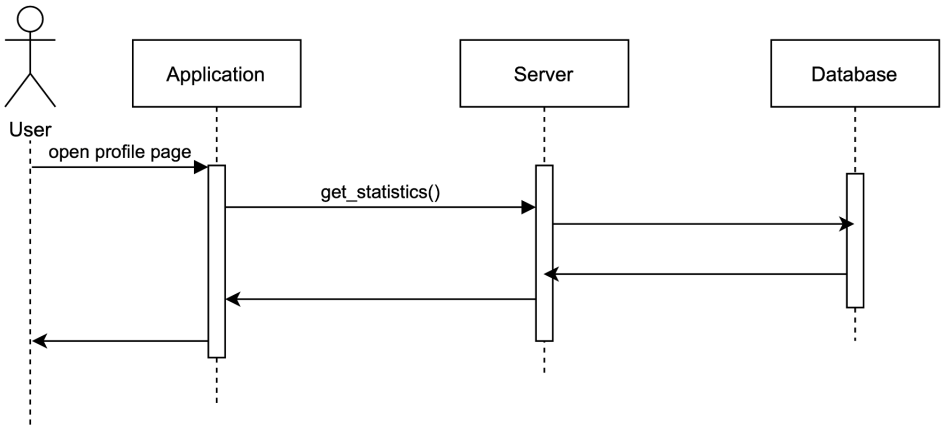


Figure 5: Sequence Diagram - Check User Statistics

When the user wants to retrieve information about his profile and his statistics, the application contacts the server which collects all the necessary data, elaborates it and sends it back to the application.

3.7.3 Join Events

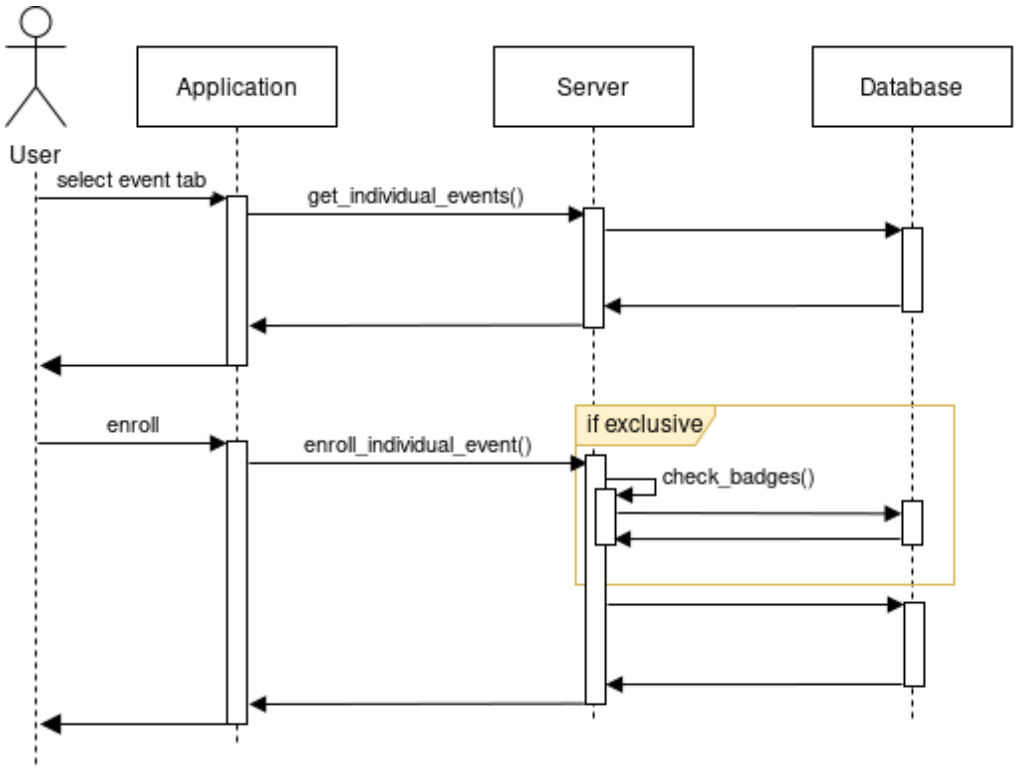


Figure 6: Sequence Diagram - Join Individual Events

When the user wants to join an individual event, the application contacts the server which retrieves all the individual events that are available from the database, and sends those back to the application. The user can then choose an event and subscribe to it. Doing that, the application sends the request to the server which checks with the database if the user meets all the requirements needed for that event and, if there are no problems, the server saves the necessary data to the database and sends back the needed data to the application.

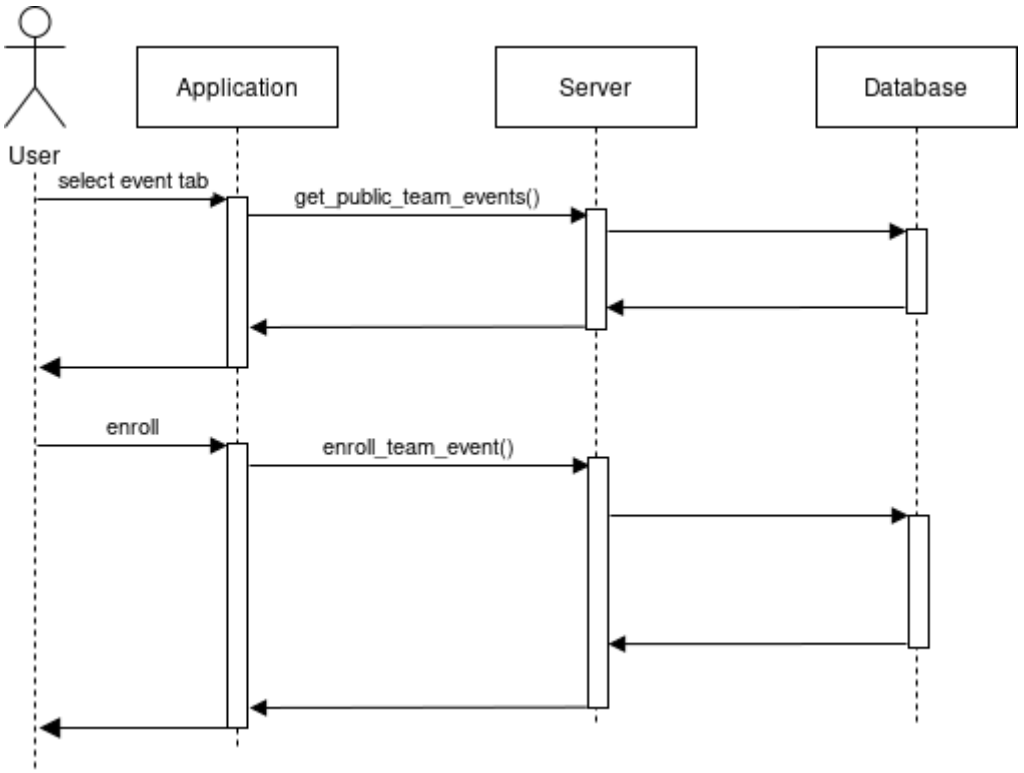


Figure 7: Sequence Diagram - Join Public Team Events

When the user wants to join a team event, the application contacts the server which retrieves from the database all the team events that are available and sends those back to the application. The user can choose an event and subscribe to it. Doing that, the application sends the request to the server which checks with the database if the user meets all the requirements needed for that event and, if there are no problems, the server saves the necessary data to the database and sends back the needed data to the application.

3.7.4 Create Events

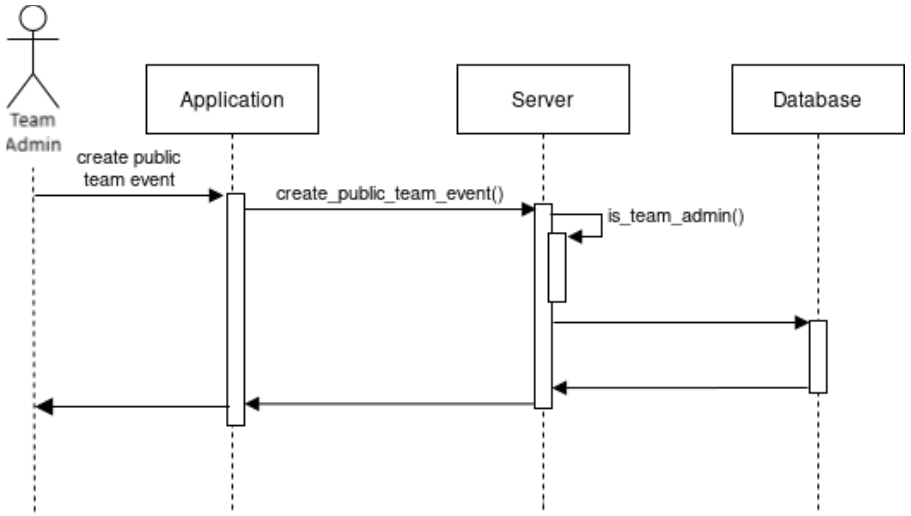


Figure 8: Sequence Diagram - Create Public Event

When a team admin wants to create a public team event, the application contacts the server which checks if the user is an admin for the team and, in the affirmative case, the server saves the data on the database and sends the necessary data to the application.

3.7.5 Redeem Rewards

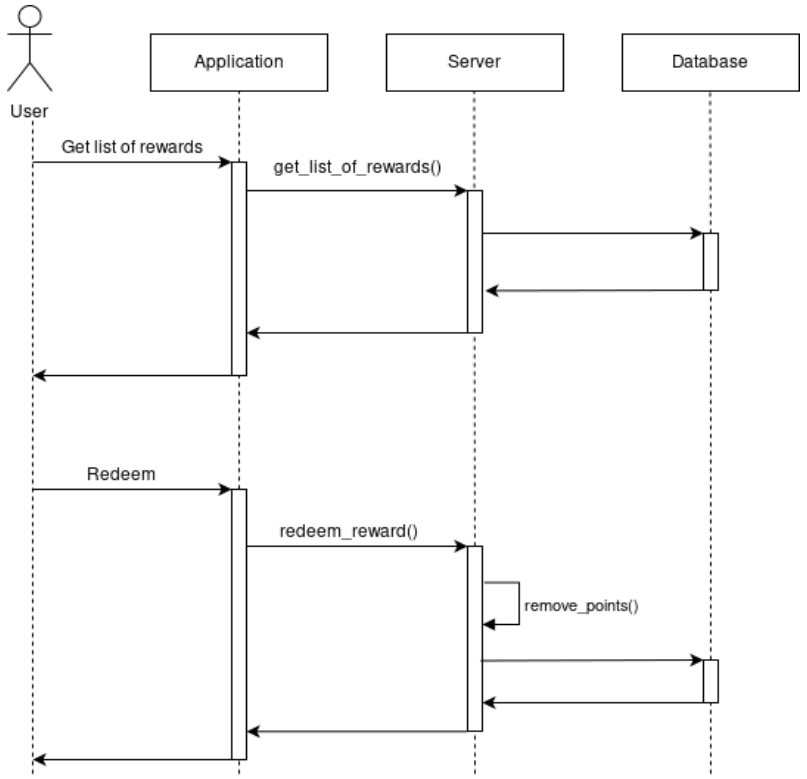


Figure 9: Sequence Diagram - Redeem Rewards

When the user wants to check the list of rewards, the application contacts the server which retrieves those from the database and gives back the necessary data to the application which displays those to the user.
If the user wants to redeem a reward, the application contacts the server which decreases the needed points from the user's balance and updates the database. After that, the server sends back the necessary data to the application, which displays the information to the user.

3.7.6 View Rewards

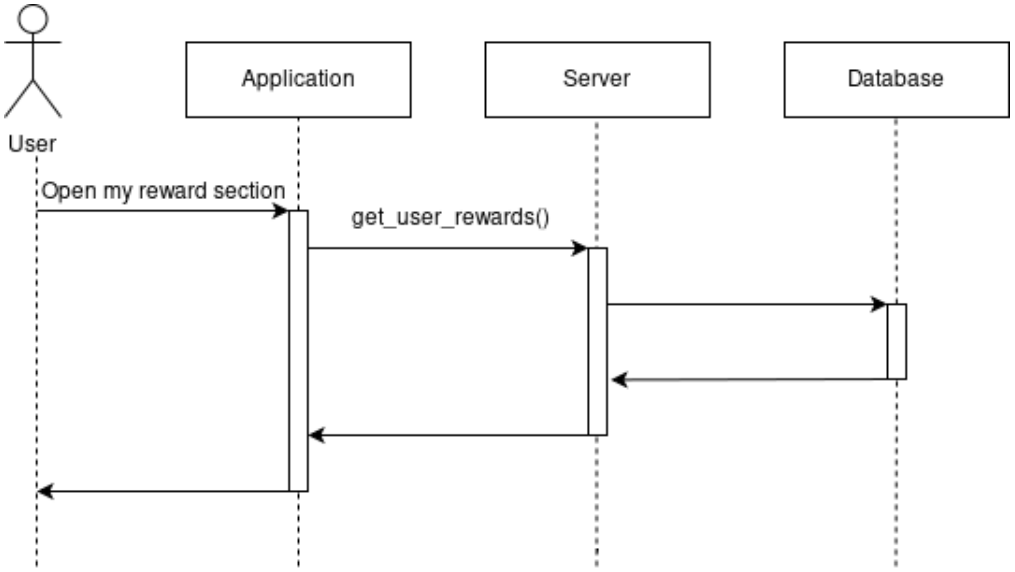


Figure 10: Sequence Diagram - View Rewards

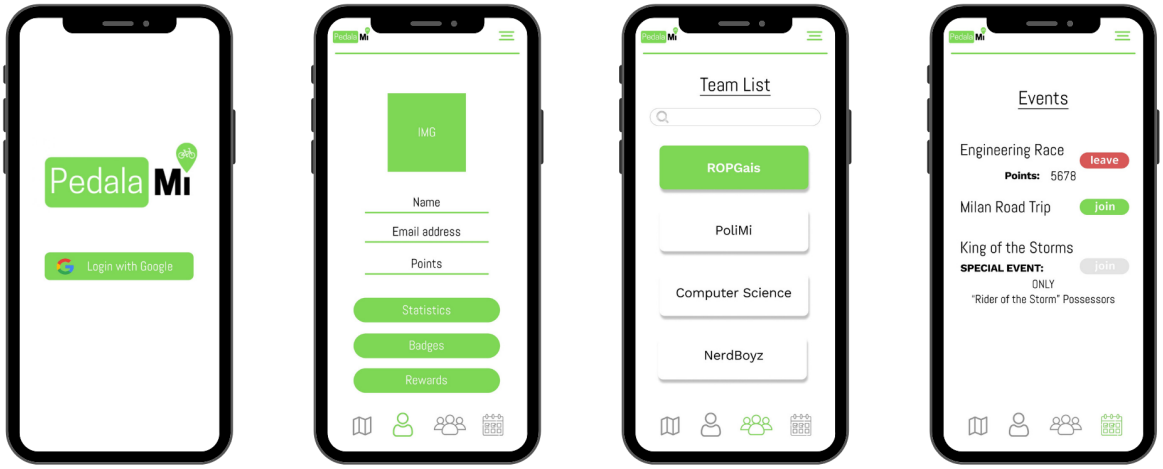
When the user wants to open the rewards section in order to retrieve his rewards, the application contacts the server, which returns the necessary data from the database and sends those back to the application.

4. Graphical User Interface

4.1 Mockups

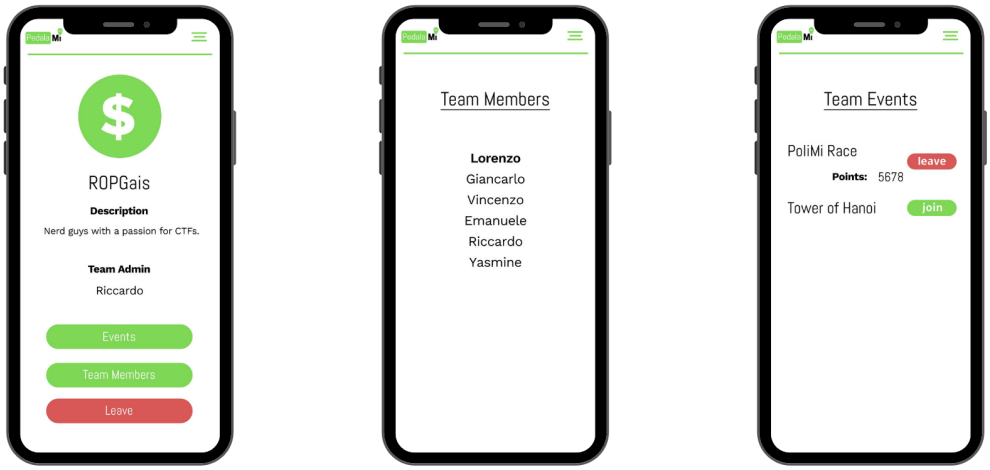
All the mockups for the mobile application and the web page are created using a software called Figma. The general structure and layout of the application is based on similar activity recording applications on the market and follows the most basic user interaction flow.

4.1.1 Application - Main Tabs

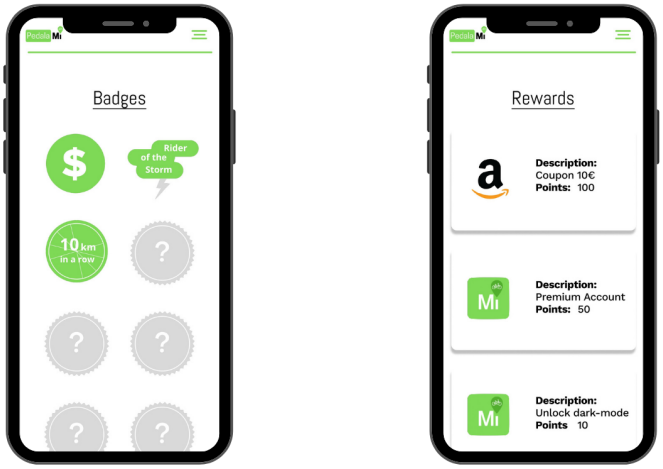


4.1.2 Application - Record a Ride

4.1.3 Application - Team



4.1.4 Application - Badges and Rewards



4.1.5 Web App - Dashboard