MyRide

Project Idea

MyRide is a platform that allows users to book a bike and use it for a certain period. At the end of this period, the user can leave the bike and declare its condition or extend the booking period. If the condition of the bike is drastic the user can contact through a chat a maintainer that will check the bike.

Functional Requirements

The users of the application will be divided into three categories: **unregistered users**, **registered users** and **maintainers**. Registered users will be allowed to use the main functionalities of the application. Will be provided a login system using username and password, through which the user will be correctly identified. A registration form will allow new users to register within the application as registered users.

Unregistered user:

- Unregistered user can register to the platform.

Registered user:

- A registered user can login to the platform with his username and password.
- A registered user can logout from the platform.
- A registered user can book a bike.
- A registered user can delete a ride.
- A registered user can extend the period of the ride.
- A registered user can create a chat for a specific bike with a maintainer.

Maintainer:

- A maintainer can login to the platform with his username and password.
- A maintainer can logout from the platform.
- A maintainer can add a new station.
- A maintainer can add a new bike to a specific station.
- A maintainer user can check for bikes that need maintenance.
- A maintainer user can repair or remove a bike from a specific station.
- A maintainer can chat with a registered user.

Non-Functional Requirements

• Every ride has a duration time.

• Every bike is identified by an id.

• Every bike has its own type.

• Every bike has a price per hour.

• Every bike can be left at any station.

• Every station has a number of available bikes.

• Every station has a position.

• Registered users are identified by their username.

• The client side and server side must use **HTTP** protocol to communicate.

• **Usability**: The application needs to be user friendly, providing a GUI.

• **Maintainability**: The code shall be readable and easy to maintain.

• **Concurrency**: The application must handle multiple users at the same time.

• **Persistency**: The application must achieve data persistence.

• The chat service needs to provide low latency, high availability and tolerance to single points of failures (SPOF) and network partitions, for which the chat functionality will be designed in order to prefer the Availability (**A**) and Partition Protection (**P**).

Erlang Usage

The chat functionality will be developed using **Erlang**. In order to take advantage of Erlang's features and to meet out requirements we will deploy the chat in a distributed manner, on multiple nodes. Distributed Erlang nodes will share chat and user status information through **Mnesia**, a distributed database. A **load balancer** will be also deployed to distribute the incoming requests to the available Erlang server nodes.