Project 4 2021 – Group 1 – Vestigo

General Idea

The idea of this app is to motivate people to explore their region by bike and discover new bike routes on and off road. The app serves as a cycling fitness app but also as a social media of sorts, where users can track their created routes and share them with other bikers in their region. They can also add descriptions and pictures to them and connect their routes to points of interest in their region.

Whenever a user is biking – either on an already created route or tracking a new route – the app is registering this, and the user gets awarded points. Writing reviews and rating other user's routes also awards the user points.

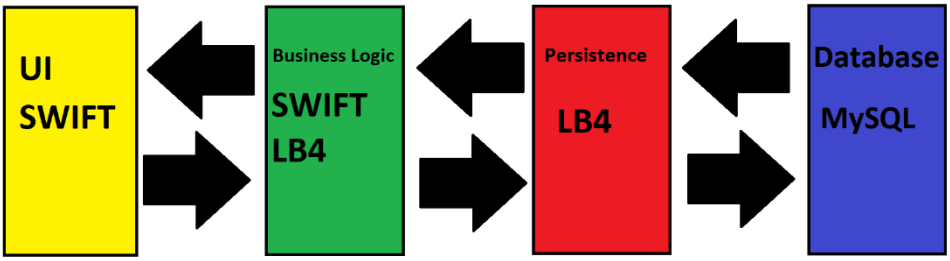
All information about the user is presented on the Profile page. Contents include the profile information, route history, reviews, rankings and points gained from interacting and cycling within the app.

The ranking page gives an overview of the most active users and their points. It includes a leader board and the total accumulated points from all users.

Architecture & Technologies

The project uses mainly a layered architecture which is also split over different machines. For our database layer we use a SQL – server which is hosted by remotemysql.com. Our persistence layer is represented by our Loopback 4 installation in the IBM Cloud. To a certain degree our business layer is also included in the Loopback installation. These are services which shall also be accessible with the usage of the REST API which Loopback provides. An example for this is the login functionality.

The majority of the business layer is inside of the app itself which is built in Swift. Inside the app we separated the views from the logical functionalities. This allows us to easily switch out components and thus makes maintaining the app simpler.



The main technologies we used were Swift, Loopback4 (TypeScript/NodeJS) and MySQL. Swift is used for the app itself. In Swift we implemented the ui and most of the business logics. Some of the business logics are also included in the

We used Loopback 4, which is a Node.js and TypeScript framework, to fastly create a running and easily usable REST API.

To store our data in a database we used the service of www.remotemysql.com which provides a free remote SQL database.

Individual Kęstutis

In this project I took a role of a back-end developer. I was mainly responsible for implementing the business logic of our application. From getting data from loopback to displaying information on the screen.

First week I was given the task to make a base for our application, so everyone was able to develop tasks on his own, not interfering with others.

Afterwards, I was given task to implement login functionalities. Started off from creating basic login page with text fields. Afterwards, I stared working on business logic. It required quite a bit of time to familiarise with async working way. Problems occurred with saving promised data. JSON web token was used for authentication of the user and was implemented in my implementation of login. Also, additional environment object was used to store decoded data that was received from loopback.

After Login had been successfully implemented me and Marek focused on Profile page and did pair programming. We implemented editing profile information functionality, so user can edit and add additional information to his profile. This meant we needed to implement everything from loopback to business logic and finishing at front end where information is displayed.

Individual Mark

My role during this project was of a Project Owner and mainly a front-end developer responsible in designing, implementing and hooking up data to and from the HTTP Handler.

First, we set out the pages that needed to implemented, followed by a visual simulation in Visual paradigm, where I designed the page layouts and the style for the app. We had numerous revisions of the design which we discussed withing the group during our daily meetings.

I implemented the Profile page from the ground up. All the content of the pages and the visualisation was agreed upon with the team during meetings.

In the Points section, I also implemented animations, which animate the shapes and point number counter upon user clicking on the Points section.

After the design was agreed upon, I set out to implement the final look of the Login, Registration and Points page. Afterwards with the help on Niklas and Kęstutis we implemented the HTTP Handler class which allows to accept certain inputs from the front-end fields and handles the requests to the loopback REST API. It also requests data from loopback (which pulls the data from the database) to be shows in certain front-end fields.

Individual Niklas

In this project my role was the role of a mediator. I took over the role of the Scrum Master to lead through the meetings of the week. In this context I also did a lot of the

communication with the lecturers. In the beginning of the project I was strongly involved in the discussions which functionalities are feasible for the initial release. Programming wise I was more focused on the business logics. I also created the HTTP Handler, which is as the name suggest responsible for the communication between the iOS app and the loopback 4 REST API.

Together with Mino I created the first version of the loopback structure and the database structure. Generally, I was heavily involved in the database design.

The priority of the database design was that we can reach a good level of maintainability and extendibility.

In the App I build the first registration form and the underlying functionalities. This was later restyled by Mark.

Individual Julia

At the start of the project Mark and I developed some wireframes for the app and developed a basic idea of the apps "look and feel". We decided on a navigation bar on the bottom of the App to switch between the four main views of the app, namely Home, Rankings, Navigation and Profile.

During the implementation phase my main task was to set up the Apple MapKit environment. For this I created the view controller "Navi.swift" in which a map is shown.

At the beginning I had planned to include a custom search with the HandleMapSearch protocol. This however led to many difficulties and our team could not solve the problems which is why we decided that for our cases an implementation of the MKLocalSearchCompleter would also suffice, especially because the app's main purpose is to motivate the user to explore his or her immediate surroundings.