FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

Flight Project

—

Badr El Haouni
Jena University

Supervised by
Friederike Klan

Overview

The goal of this project is to create a platform where we collect flight observations with relation to the weather status.

The platform allows users in case of a clear environment to access a map of flights within a 1000km radius to our location, where the user will be able to select collect information about the flight and save the observation in the database,

The platform is built in an ergonomic way to guide the user through this process and inform in case of bad weather that maybe it is not the best time to collect observations.

Goals

- Develop web application Evaluate flights and observations
- Detect flights in the nearby region
- Detect weather information
- Be able to detect the user location

Installation guide

Here are some requirement to have in order to run the machine:

- Download the Node.js version 12 or higher
- Make sure to have Git installed in your machine
- Have a local postgres database running in your machine or in docker container

Front End

In order to run the front application, follow these steps :

- Navigate to the client folder

- Make sure to install node_modules libraries with the command : **npm install**
- In order to run the application run in development: **npm run dev**

This will run the front End under the port **3000**, and in order to connect the application go to the browser with link : http://localhost:3000/

Back End

In order to run this backend application, follow these steps :

- Navigate to the server folder
- Make sure to install *node_modules* libraries with the command : **npm install**
- In order to create database tables make sure to run: **npm run migrate:up**
- In order to run the application run in development : **npm run dev**

This will run the back End under the port 5000

Postgres

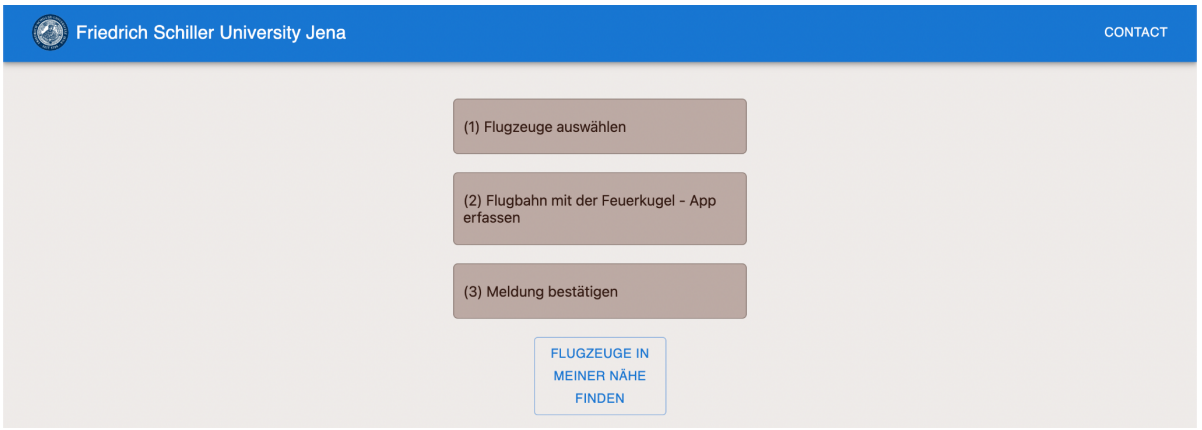
- Download postgres database **https://www.postgresql.org/**
- Download pgadmin tool: **https://www.pgadmin.org/download/**
- Create the database with the name: **GI**
- Set up the connection to the database by setting up the configuration in the **knexfile.js,** this file is at the root folder under the server application
- In production you should use the connect string in the environment variable DATABASE_URL

```
module.exports = {
 development: {
   client: 'pg',
   connection: 'postgres://postgres:postgres@localhost/GI',
 },
 production: {
   client: 'postgresql',
   connection: process.env.DATABASE_URL + '?ssl-true',
 },
};
```

Application overview

Home Page

When the user access the application at http://localhost:3000/, the user will be greeted with following home page:



In this page we have some explanation about the workflow of the website.

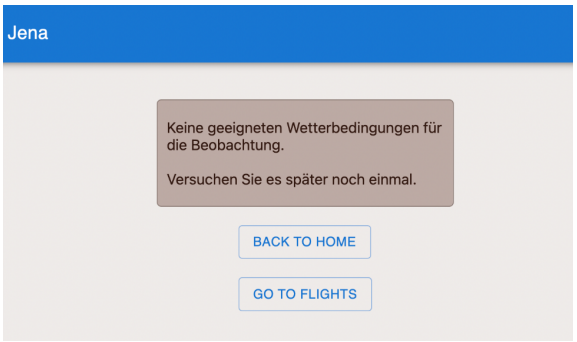
It explains that we need to select a flight, then capture its trajectory, and save and confirm the trajectory.

So in order to start the user should click the button “Find the flights near me”

Cloudy Page

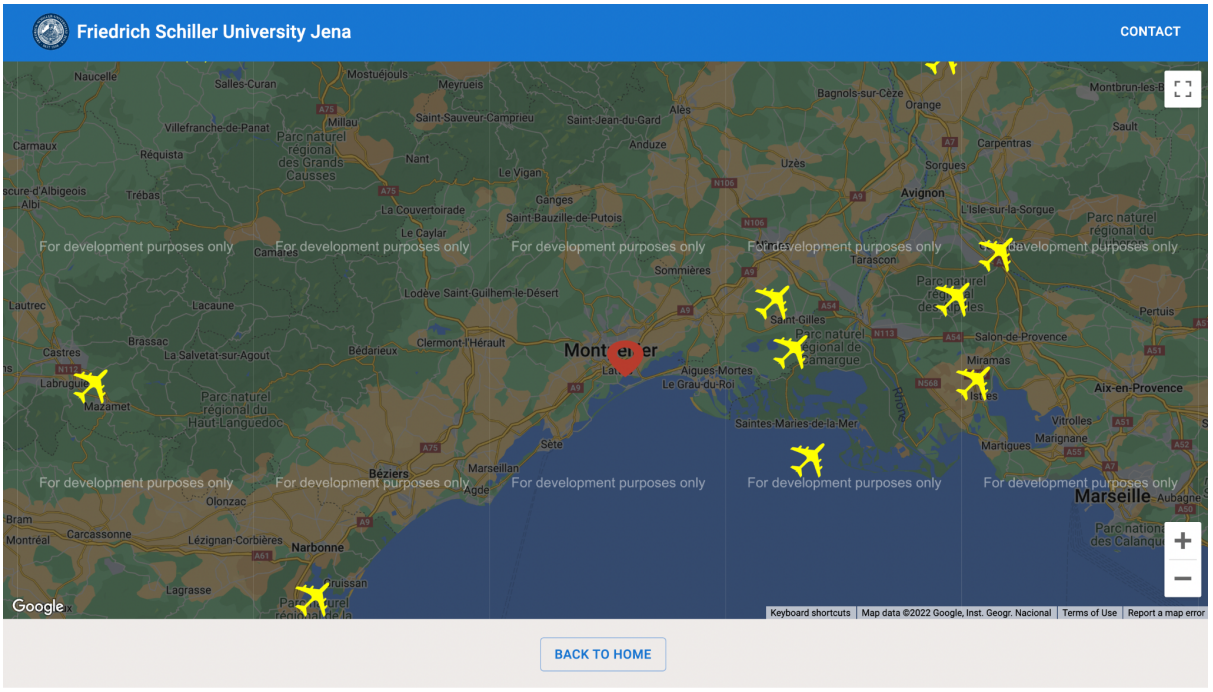
If the weather is cloudy then we will be redirected to the following page, which explains that it is not the perfect time to make observations.

The user still has the right to continue or to return back to the main page.



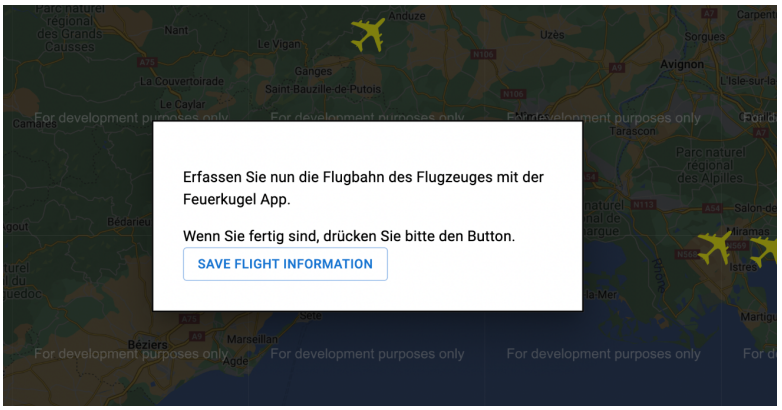
Flight Page

In the flights page, the user is shown a map with a pointer to the user location and the flights with a radius of 1000km.



The user should select a flight in order to save its information and make an observation.

When the user selected a flight a modal shows up as the following:



As we can see the modal explains that in order to capture the flight information the user should click on the button “Save flight information”,

This will trigger an action in the front end and send the data to the backend that proceeds to save in the database.

When saving the flight data, the backend takes the flight information received from the backend and fetches an observation from an api call and save them in the database.

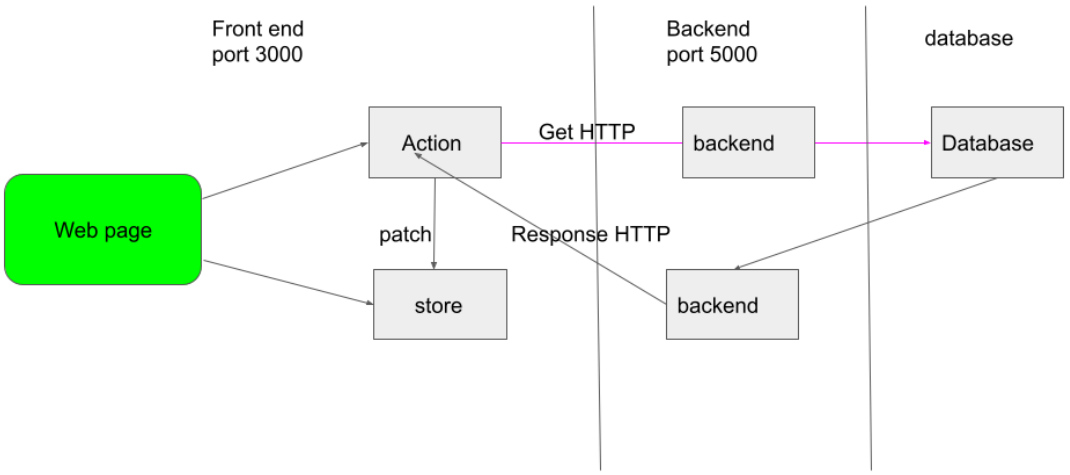
Technical overview

The application is a RESTFUL architecture; the frontend is a single page application developed with React/Redux and the backend is an API supporting multiple endpoints to be consumed by the front end.

We are also doing an **integration** to external apis that help us get the flight information/observation :

- https://airlabs.co/
- https://meteor.nachtlicht-buehne.de

This is a global overview of the application architecture:



FrontEnd

The front end is developed using the following technologies:

- Reactjs
- Redux
- Material ui

The application is organised as the following:

- Components:
    - Each part of the application is a component and all components are developed individually

- Actions : The actions are the interaction of the user with the application

- Store
    - The store is a global immutable object accessible to the all application where we store all the data necessary
    - The store represents the state of the application and if changed the components get refreshed

BackEnd

The backend is developed with nodejs and express framework, this allows us to create an API that exposes end-points to the front-end.
In order to communicate with the backend we use http calls to consume the different available end-point.

Postgres

The backend is connected to postgres database and we are using the ***knex*** package to communicate with the database, the configuration of the connection to postgres is located is ***knexfile.js***

```
module.exports = {
 development: {
   client: 'pg',
   connection: 'postgres://postgres:postgres@localhost/GI,
 },
```

```
 production: {
   client: 'postgresql',
   connection: process.env.DATABASE_URL + '?ssl-true',
 },
};
```

Conclusion

This experience was a learning experience for me. I was able to gain knowledge about APIs, frontend development, and how integration to external apis are done. It was also interesting to use figma as a tool to have a setup back before the start of development, that allowed us to better architect the solution and all its different components.

References/Documentation

https://reactjs.org/

https://expressjs.com/

https://redux.js.org/

https://material-ui.com/

https://axios-http.com/docs/