Team 5

NGRY HUB virtual testbed

16th January, 2020

1. Purpose

This design document is produced to capture all the important design decisions that have been decided for this project. The purpose of this document is that the reader should provide a good basis for understanding the implementation that will be done. The document is written in a way that a reader with no knowledge about the project, can easily understand in order to be able to join the team and start to maintain the system further.

2. Client and project

Our client is Future Energy Center, MDH. The European Regional Development Fund project, NRGY HUB, aims at creating a HUB for innovative energy solutions, and one important part of the project is a virtual testbed that will digitally describe a city with open data on its various flows with regards to energy and water. The city that will be digitally describe is Västerås, and in order to accomplish that a map will be used. Due to GDPR regulations the data on energy and water usage can only be displayed by clustering at least five close buildings together, and then calculate the average water and energy usage within each cluster. The displayed data will be averaged water and energy usage within each cluster.

However, the requirements to the project changed and the solution will need to show the information for each building at first, the clustering part could be implemented in the future.

3. System Design

3.1. System main parts and communication

Our system consists of one relational database made with 4 tables which contains information about the users, company, buildings and meters. The inserted information are not real and has been inserted just to show how the project works.

Our system also consists of a web page. The first page a user will see is an login/registration page. Here the user can register into the system or they can login if they already have an registered account. If a user logs in, and the login is successful, they are directed to the main page. On the main page the user will be able to see a map with different markers along the city of Västerås. Each marker shows data for the building that it is marking on the map. When one marker is clicked, the webpage will send a request to the NRGY database to find information about that building. The information will be sent back as a string to the web page that will display the result beside the marker as a popup on the map. The information displayed are dependent on what type of user is logged in and which company the user works for. This will be consistent as long as the user wants to use the webpage.

An admin panel was added to the project giving a specific user permission to add information about new buildings.

3.2. Communication with external systems

Our system does not have to interact with external system at the moment. However, the request of the client is to make this possible for the future. The data that the user will include in the database will be provided from an energy company located in Västerås (Sweden) called Mälarenergi. The data from their database will be sent to ours. As the client requested the project to be deployable on AWS we decided to use a database that was going to be compatible with this system. For this reason we chose Aurora database which is an Amazon service. However, our database got hacked three days before deployment so at today's date we are using a local version of a database. The database structure is made in order to welcome those

information that are needed for the project purpose and it will be easy to insert this information into the database considering the information received by the company.

The data collected will be displayed on our own webpage that will present a map and different markers with information of energy consumption and district heating. The map that will be imported in our project will be an online software of a “free to use”-component called Open Street Map. In order to use their map without compromising other maps we will restrict the access to the map with user access.

3.3. 3rd party frameworks

For the system we are using the following technologies, IDE and third party frameworks:

Technologies	IDE	Third party
React JS	VS Code	Open Street Map
Java	IntelliJ IDEA	Leaflet
MySQL	MySQL Benchmark	

ReactJS is the main technology that we are using for developing the Front-End part of our system. When editing Front-End we’re using Visual Studio Code. For developing the Back-End part of our system we are using Java programming language. For editing the written code and running the project, we use IDE IntelliJ IDEA. For displaying the markers with the buildings data we use Open Street Map. OpenStreetMap (OSM) is a collaborative project to create a free editable map of the world. The geodata underlying the map is considered the primary output of the project. Leaflet works as a JavaScript library for OSM to mad the map interactive, markers were displayed thanks to this technology and works efficiently across all major desktop and mobile platforms. For accessing database locally we’re using MySqlWorkbench and remotely on freemysqlhosting server we’re using phpMyAdmin.

3.4. Structured Implementation

The implementation of our application is structured in Front-End and Back-End. In the future implementation we are planning to connect these two modules.

The Front-End part is conformed by the following folders and files:

- **Public** - folder where all the static files reside.
    - **favicon.ico**
    - **Index.html** - is the page template with main div (id = “root”)
    - **Manifest.json**
- **Src** - Folder where all the dynamic files reside. All .css and .js code have to be in this folder.
    - **Components folder** - All components and their styles are divided into new subfolders in this folder.
    - **Pages** - Sign in and Sign up forms are in this folder.
    - **App.css** - The style of the main page is defined in this file.
    - **App.js** - The layout of main page is defined in this file. So everything that will be rendered in the div with id=”root”.
    - **App.test.js** - In this file tests for the app will be defined.
    - **Index.css** - The style of application should be in this file.
    - **Index.js** - This file represents the JavaScript entry point.
    - **serviceWorker.js** - This lets the app load faster on subsequent visits in production and gives it offline capabilities. However, it also means that developers will only see deployed updates on subsequent visits to a page.
- **.gitignore** - file that we use for defining module that we want to be ignored when pushing repository on Github.
- **README.md** - contents the information related to the authors and also the modules that are required in order to deploy the project and execute it.
- **package-lock.json** - package-lock.json is automatically generated for any operations where npm modifies either the node_modules tree.

- **package.json** - package-lock.json is automatically generated for any operations where npm modifies either the node_modules tree.
- **yarn.lock** - Yarn caches every package it downloads so it never needs to download it again. It has more features than npm so if somebody is used to yarn start for running project, they can do that.

Npm means node package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. Since our Front-End framework is ReactJS which is JavaScript library, we use npm for installing the dependencies needed for our project.

The Back-End part is conformed by the following main folders and files:

- **Src**
    - **Main**
        - **Java** - This folder contains the main class for running the application where is specified the way of how the app is going to work and things that are initialized at the beginning. All collections, repositories, controllers, services are written in this folder, separated into four different folders:
            - **Models** - In this folder structure of each model of the table we are going to have in the database is made in separate classes.
            - **Controllers** - In this folder controllers per collection are defined using services per collection.
            - **Repositories** - In this folder repositories per collection are defined as interfaces and extended with CRUDRepository because we are working with MySQL so it was easier to use this Repository since we need only basic crud operations.
            - **Swagger** - Swagger is added since it represents a tool that helps developers to design, build, document, and consume RESTful web services.
        - **Resources** - This folder contains one crucial file for defining properties of application as well as json files that are imported in the beginning to database as collections.

- **application.properties** - definition of properties of application and database connection string.
- **users.json** - json file with some users to fill collection in database.

  - **Test** - Folder where we can specify test and run them in order to test our services.

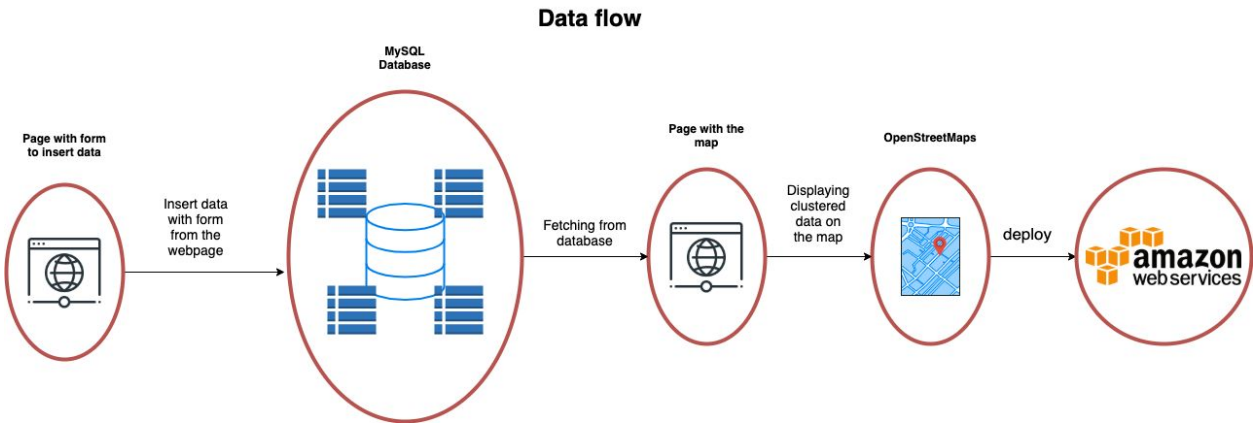
- **README.md** - contents the information related to the authors and also the modules that are required in order to deploy the project and execute it.
- **mvnw** - This file is imported from Maven Wrapper. It allows user to run the Maven project without having Maven installed and present on the path.
- **mvnw.cmd** - This file is imported from Maven Wrapper. It allows user to run the Maven project without having Maven installed and present on the path.
- **pom.xml** - This file contains all dependencies of our project.

Links for the repositories https://github.com/nejrabahtic/NRGY-HUB-Virtual-TestBed-FrontEnd and https://github.com/nejrabahtic/NRGY-HUB-Virtual-TestBed-BackEnd for the frontEnd and BackEnd folders respectively.

3.5. Data and control flow

The system is formed by several parts that collaborate together by transmitting data from one part to the next in order to provide the required functionalities.

In the beginning we will ask the user to register into the application. When the registration page is finished, the data that the user inserted will be stored in our database. A relational database (MySQL) will be used in order to store the information provided by the clients and to create relations between the different tables.. The web application will also contain a page where the user will be able to insert information about buildings and companies into the database. The web application will fetch from the database the informations related to the meters, buildings and users in order to know which data should be displayed on the map, and also what are the information that can be provided to that specific user. In the end, this application should be deployed in the Amazon Web Services (AWS) and running there.



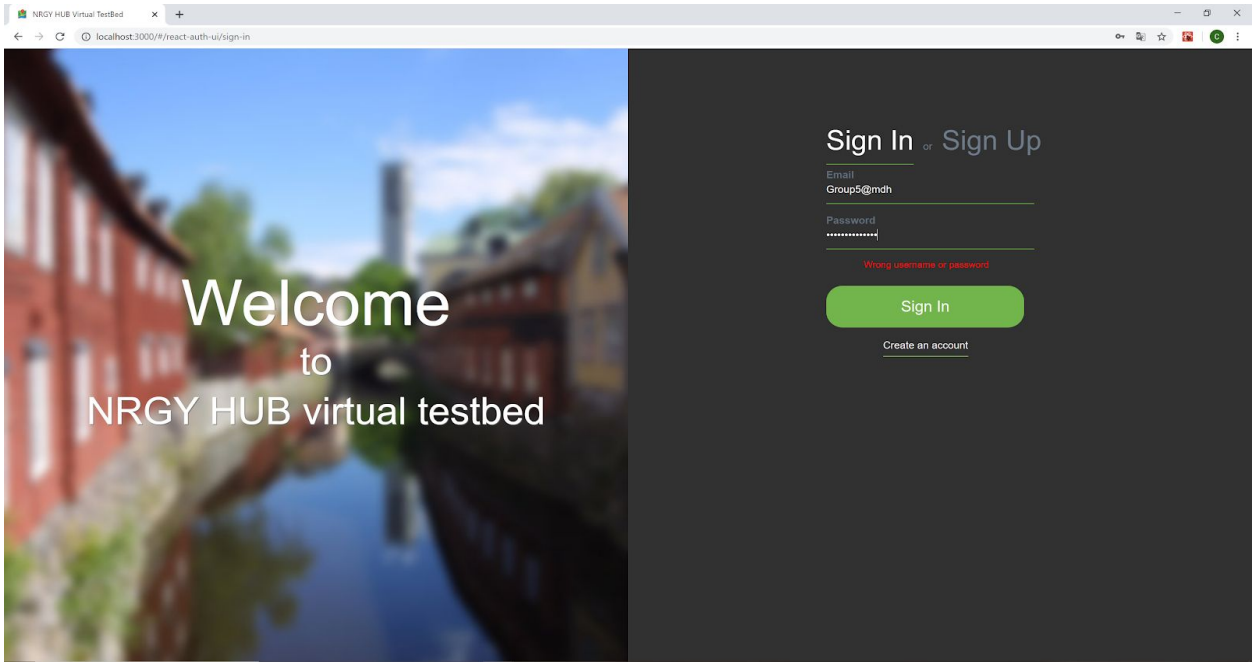
Picture 1. The Diagram shows the flow of the data between the parts of the application.

4. System work

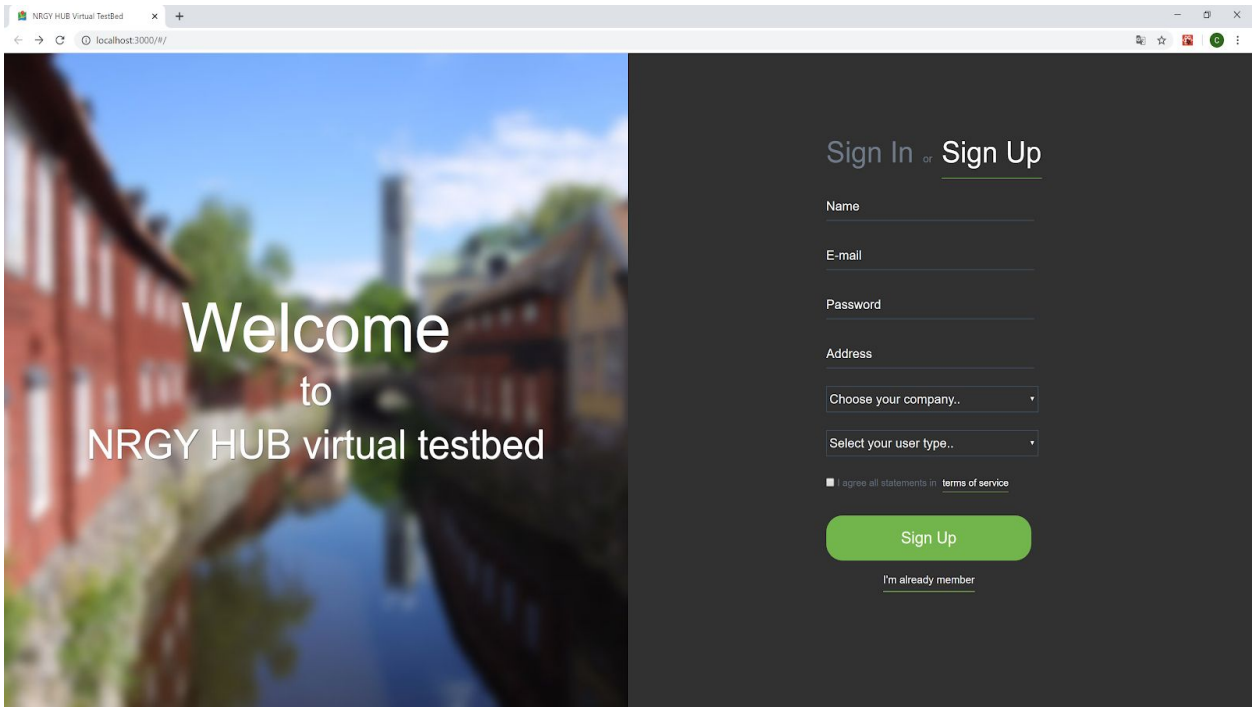
4.1. GUI structure

The website consists of a total of four different pages. When first enter the website, a login page will be displayed (Picture 2). There the user is required to fill in username and password. If the user do not already have an account there will be a label with a question about if the user is not registered. When pressing that label, the user will be navigated to the registration page instead (Picture 3). On the registration page there are some required forms to fill in in order to create an account. A label is displayed on this page in case the user already is in possession of an active account. By pressing this label, the user will be redirected to the login page. When logged in, the main page will be reached. The main page consists of a big map that covers most of the screen (Picture 4). The main page is the core of the project, it consist of a map centered on the city of Västerås. On the map we will display the markers that are linked to the information of each building. When the user click on a marker, a popup (Picture 5) will appear with the information about the building. The user can then click on the X of the popup to close it. Our fourth and last page is only available for the admin (Picture 6). The last page is for the admin to add a new building or company, by filling in the forms and press the save button. There is also a

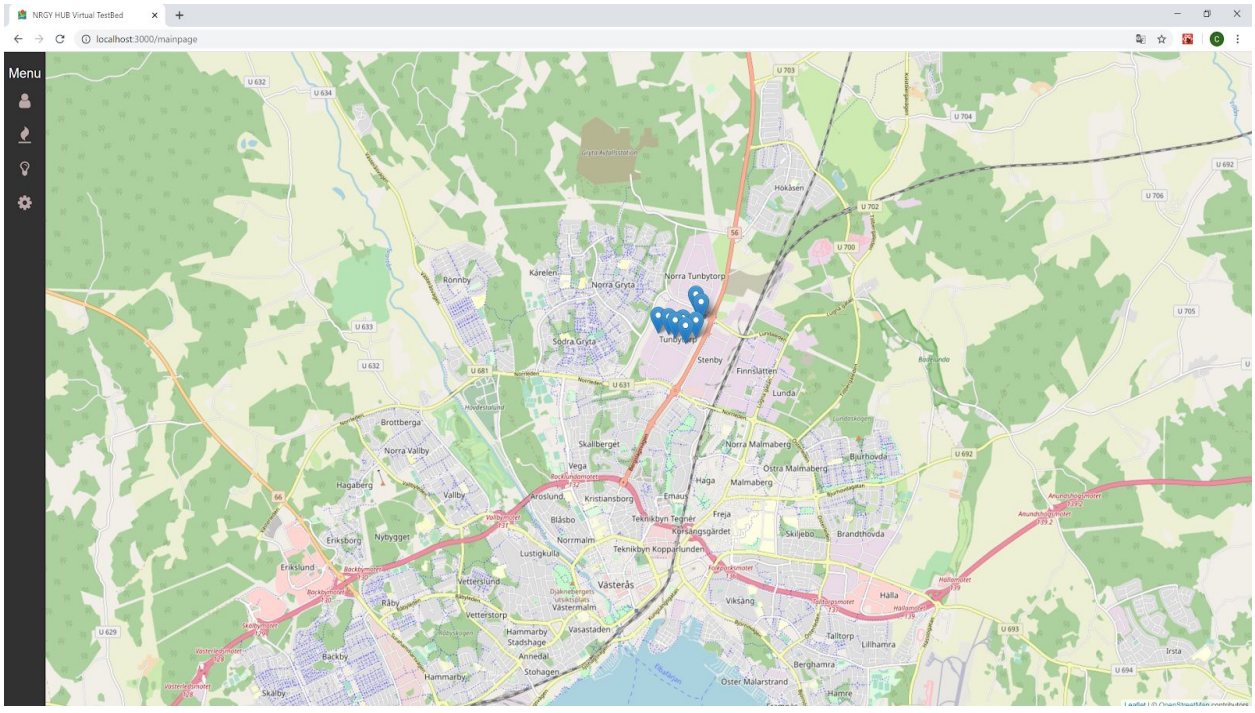
sidenav on the left side of both the main page and the admin page, there the user can navigate the thought the website or logout.



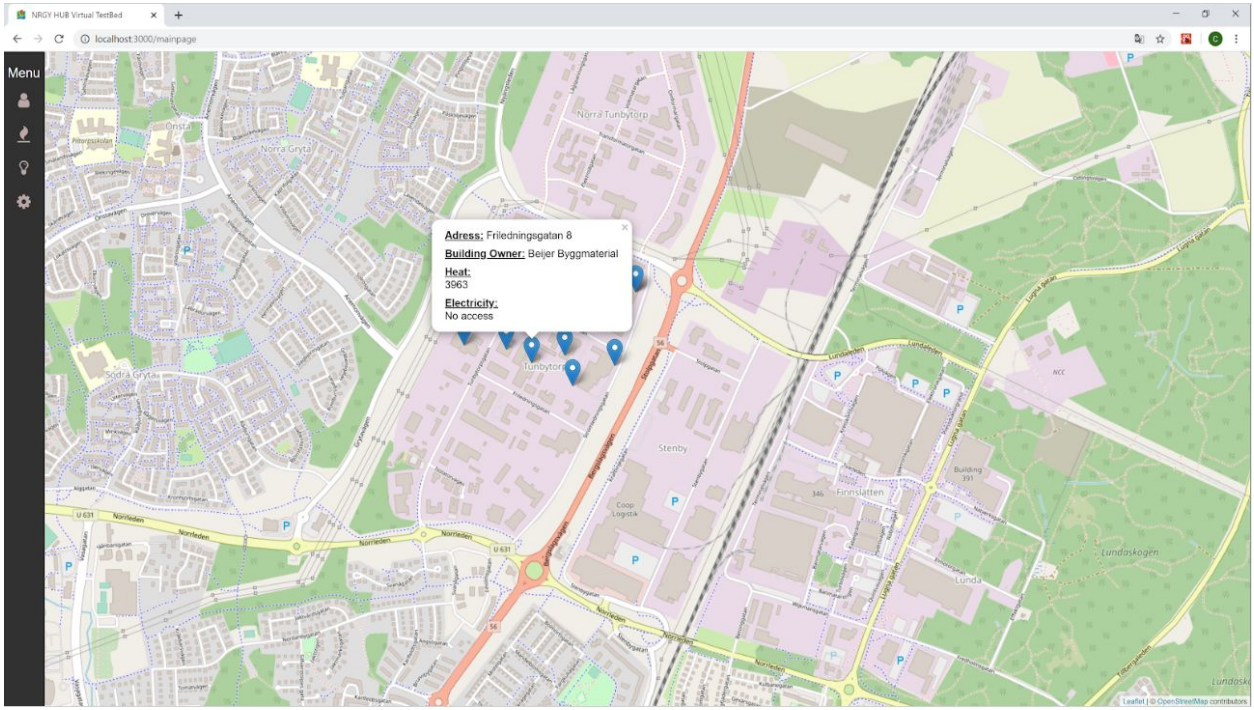
Picture 2. Login page



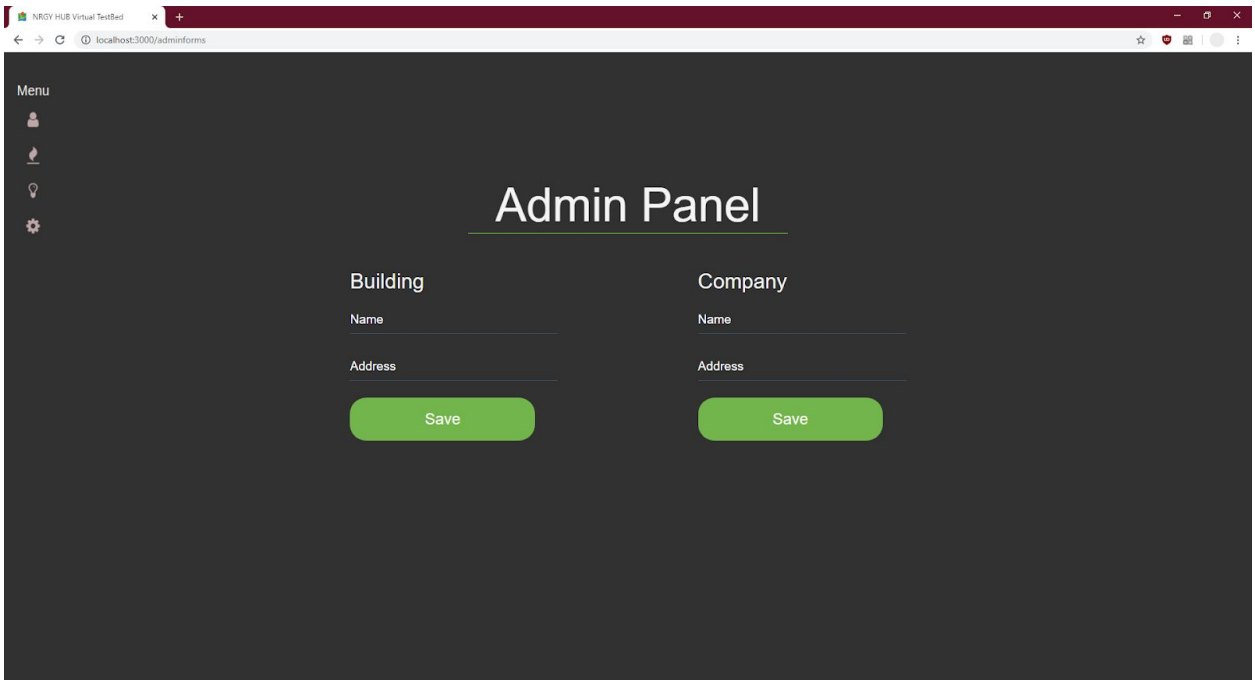
Picture 3. Registration page



Picture 4. Main view



Picture 5. Popup with information



Picture 6. Admin panel

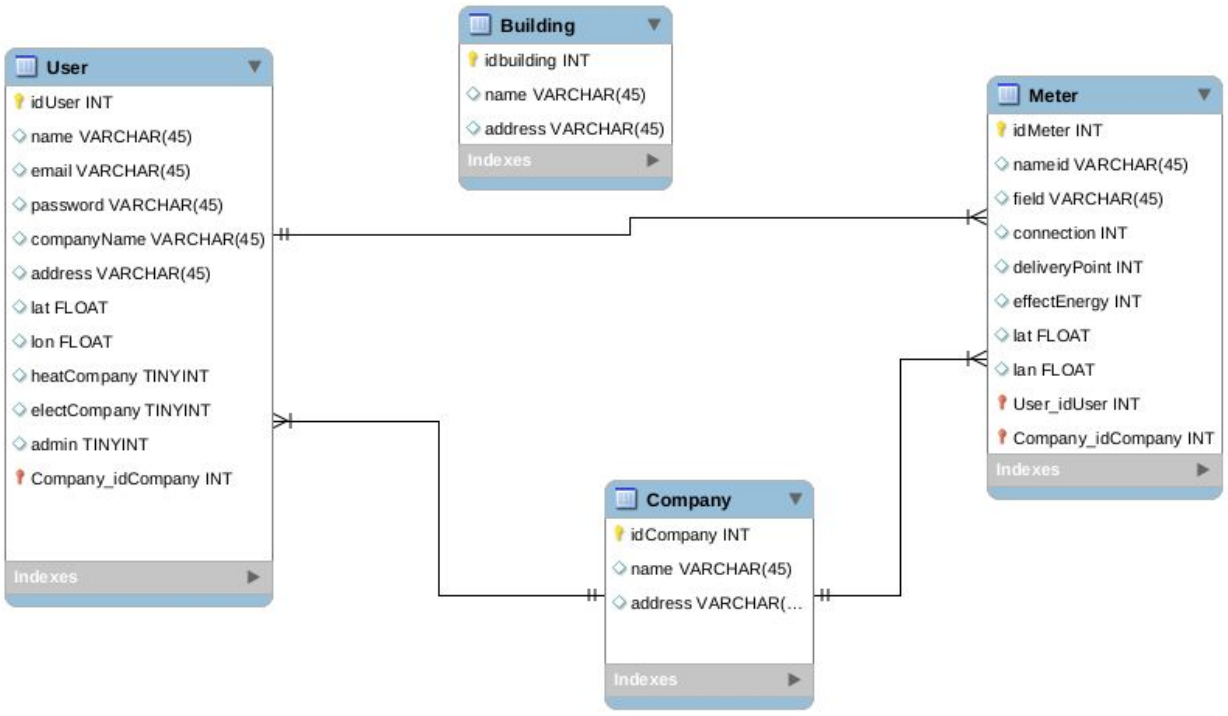
4.2. Interaction between user and system

The NRGY HUB system will be used by registered users. After the log in or register depending on the case, the website will ask the NRGY user tabel if the account is present. After the log in, the website will show the map over Västerås and the different markers that have been created. When the user clicks on the marker, a request will be sent to suitable route, data will be returned as a response and then displayed to user. Depending on the user permission the marker will then show a pop-up containing the information that this specific user has access to. If the user cannot access a specific content, the text “No access” will be displayed beside that information.

5. Design Description

5.1. Class Diagram

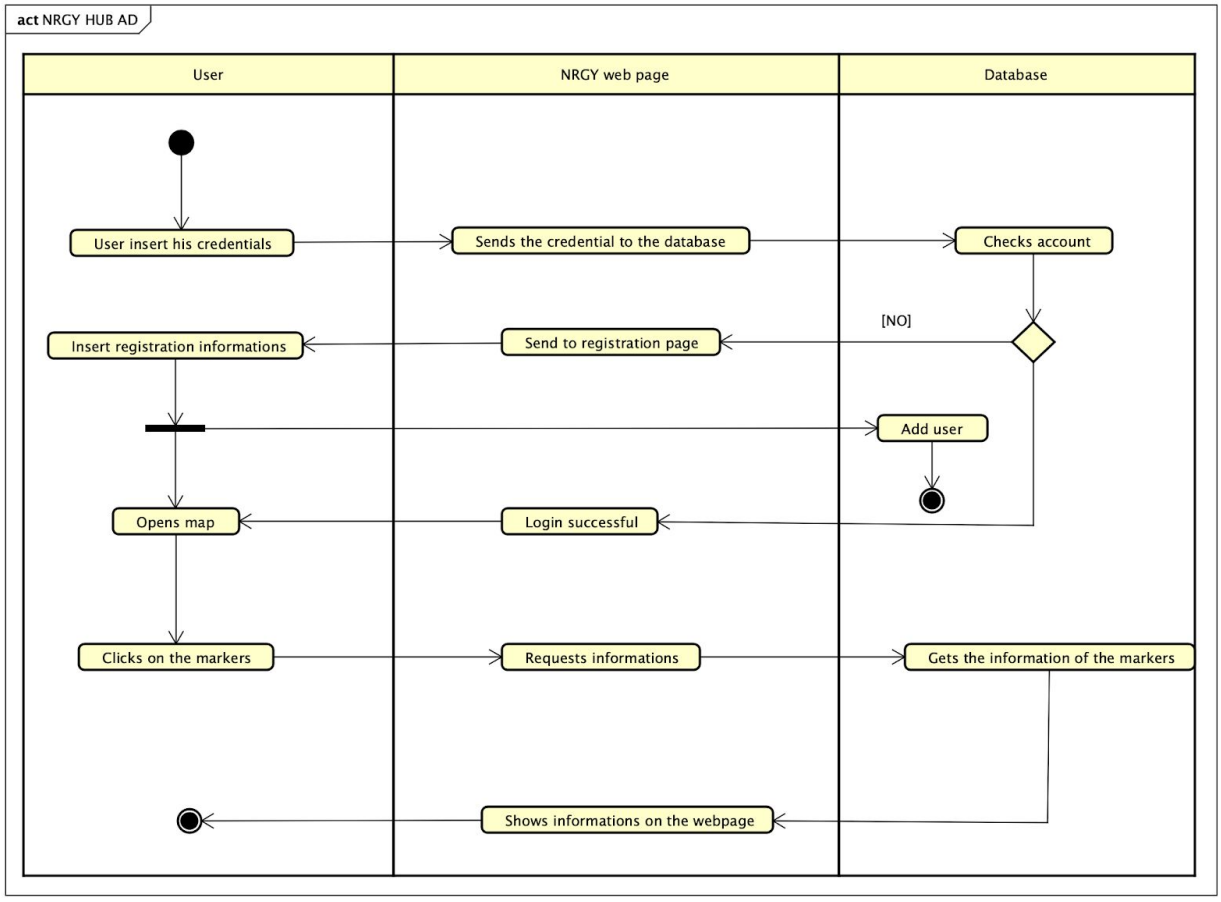
For the project, the modeling of the NRGY HUB main class diagram shows the attributes that are going to be used for User, Company, Building and Meter (Picture 7). User and Company table are connected with Meter table with relationship that one user can see multiple meters. The same thing is for the company table. Company and User tables are also connected in a relationship that one user can have one company, but one company can have many users.



Picture 7. Entity relationship diagram for NRGY HUB system

5.2. Activity Diagram

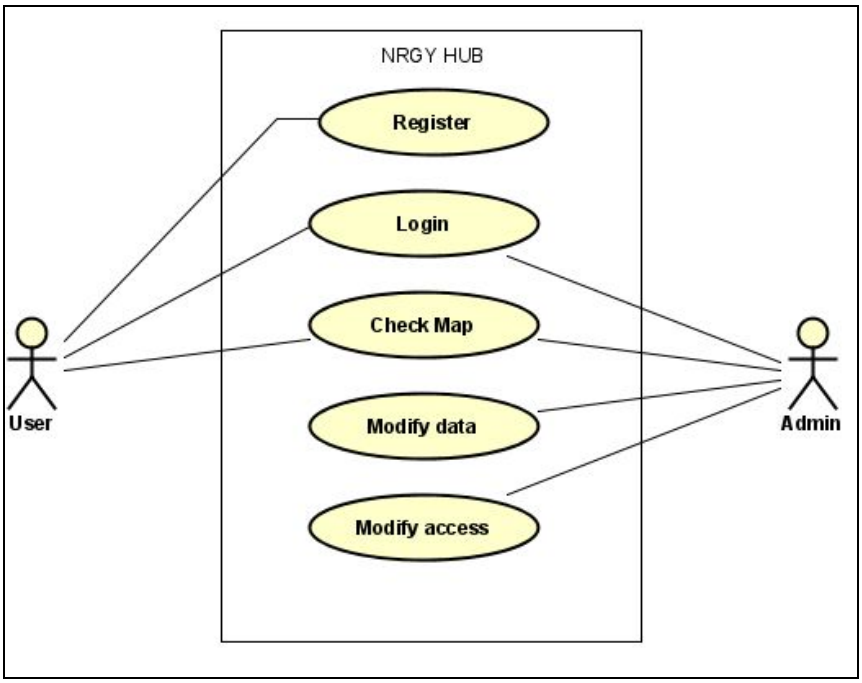
The activity diagram shows the control flow from a starting point in the website login, and the paths that will take through the web page and the database collections while its been executed (Picture 8).



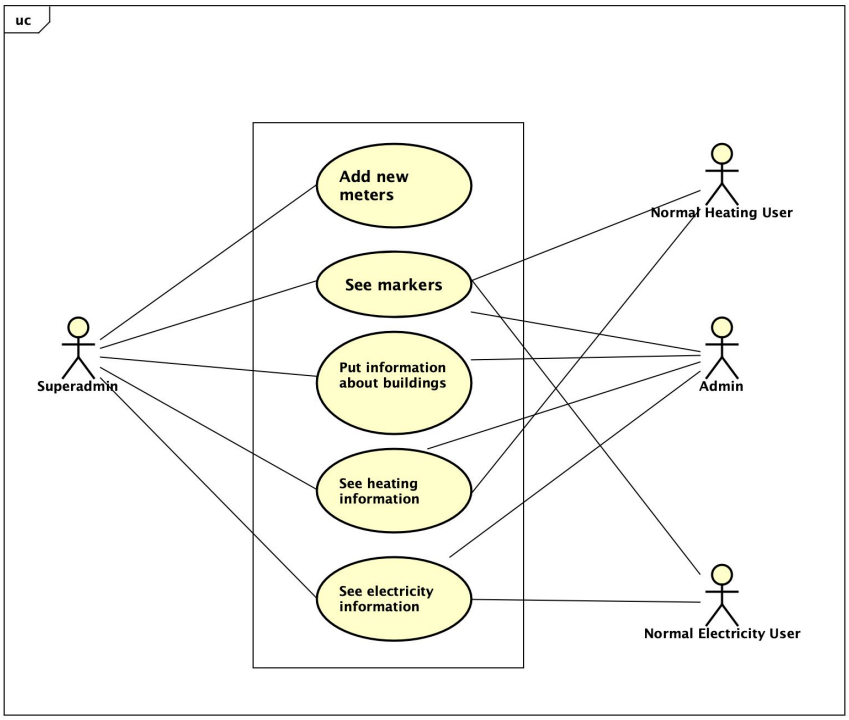
Picture 8. Activity diagram of NRGY HUB system

5.3. Use-case diagram

Use-case shows the interaction that takes part the user on the system, the same as the ones for the administrator. They both have different and share one activity. The user registers and logins to be able to check the map and interact with it. On the other side the administrator after the login, no need to register since their ids are already made on the collections, they can modify the data and access for the system (Picture 9 and Picture 10).



Picture 9. Use-case diagram for NRGY HUB system



Picture 10. Use-case diagram of the different privileges that each user has