Energy utility platform implemented using Node.js and React

The application is based on building 2 portals, one for the regular user and one for the administrator. The regular user, the client is able to log in and see all his devices. Also, a client can choose a date and observe the consumption of the targeted device on that day. The administrator has 4 pages, one for the devices, one for the users and two more for creating a device and a user. The administrator can edit, delete devices and users and assign devices to users.

React frontend application

Architecture:

* The application is divided in two parts: components and pages.
* The components contain folders with tsx files and scss files to create html elements as: cards, charts, modals, forms, tables and navigation bar.
* The pages contain the same type of files to create and style the application pages: login, register, devices, users and a not found page.
* The not found page is used to show a regular user that he cannot visualize a page in which only an administrator has access.
* The charts and tables are created using the reactstrap Table component and recharts library.

Entry page with login and register options:

Shape

Description automatically generated with medium confidence

Admin devices page with modal activated for editing a device:

Graphical user interface

Description automatically generated

Client page with chart showing the device consumption for a specified day:

Graphical user interface

Description automatically generated

Device and user creation for an admin:

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Node.js backend application

The backend application is built using the Nestjs framework for Node.js. The framework creates an MVC structured application with a 3 layered architecture.

For the database, the server used is MySql, this is accessed in the application using the TypeOrm library, which is used for managing the interaction with the database, basically managing the repository layer, which is using the model classes from the entities folder.

The service layer is constructed to provide and manage all the database interaction which comes from the controller defined endpoints. There is a service built for each model.

There are controllers made for each type of service, having defined all the CRUD operations needed. Also there is a controller and service used for authentication. At this stage of the application, the authentication is done with a post request, which takes as a body the name and password of a user and tries to find the pair in the database.

Database diagram

Graphical user interface

Description automatically generated

The DB design is made mirroring the models of the application. Each entity has an id of type uuid as primary key. A user has as fields a name and a password which will be used for logging in and an isAdmin field to decide the pages the user has access to.

A device has a description, address which will be the location of the device and a maximum hourly consumption. Also, a device will need a userId as a foreign key, since a device will be assigned to a user.

Each device can will be assigned an energy consumption, which will be computed hourly. That means that an object of type energy consumption will have the deviceId as a foreign key, the consumption of the device and the timestamp of the consumption.

Deployment diagram

Diagram

Description automatically generated