TECHNICAL DOCUMENTATION

Dashboard - EPITECH 2024

Alexandre Sauner - Axel Biehler

1. API

The API is a NodeJS REST API using express at its core. Routers such as authentication, services or instances can be found in the `routes` folder. These routers are used in the `main.js` file.

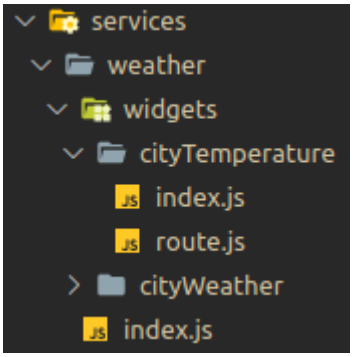
The API handles CORS.

The database used is Mongo. It is queried with Mongoose. Models can be found in the `database` folder.

Authentication is done using Json Web Tokens. Passwords are hashed using BCrypt. An `authMiddleware` is available in the `authentication` folder. This middleware makes sure a token is passed in the `Authorization` header. It verifies the token and adds the user identifier to the Request object under the `userId` key.

Services and widgets are automatically imported with dynamic requirements.

Services and widgets must be created in the `services` folder. Here is the file structure of a service `weather` with a `cityTemperature` widget:



`index.js` files contain metadata for either a service or a widget. route.js files export Express route function which are linked to widgets.

Each service must export a simple metadata object:

```
const path = require('path');

const metadata = {
  name: path.basename(__dirname),
  displayName: 'Weather',
  description: 'Use the OpenWeather API to get weather data.',
  needsOAuth: false,
};

module.exports = metadata;
```

Each widget must do the same:

```
const path = require('path');

const metadata = {
  name: path.basename(__dirname),
  displayName: 'City temperature',
  description: 'Display temperature for a city',
  params: [
    {
      name: 'city',
      type: 'string',
      placeholder: 'City name',
    },
  ],
};

module.exports = metadata;
```

A. Routes

a. Authentication

```
POST /auth/login
{
   "username": "alexandre",
   "password:" "password"
}

{
   "status": true/false,
   "error": "wrong password"
}

POST /auth/register
{
   "username": "alexandre",
   "password": "password"
}

{
   "status": true/false,
   "error": "password too short"
}
```

b. Services

```
GET /services

[
   {
       "name": "weather",
       ...
   },
   ...
]
```

c. Instances

```
GET /instances

{
    "status": true/false,
    "error": "example error"
    "instances": [
        {
            "_id": "618d7ddbbda85c8fa70e9e99",
            "service": "weather",
            "widget": "cityTemperature",
            "params": {
                "city": "Strasbourg",
                "refreshRate": 30
            }
        },
        ...
    ]
}

POST /instances

{
    "service": "weather",
    "widget": "cityTemperature",
    "params": {
        "city": "Paris",
        "refreshRate": 10
    }
}

{
    "status": true/false,
    "error": "example error"
}
```

```
GET
/instances/weather/cityTemperature/618d7ddbbda85c8fa
70e9e99

{
   "status": true,
   "temp": 7.76,
   "city": "Strasbourg",
   "country": "FR"
}

DELETE /instances/618d7ddbbda85c8fa70e9e99
{
   "status": true/false,
   "error": "example error"
}
```

2. Front-End

The front-end framework used is React. Useful npm packages used are:

- @mui/material
- react-grid-layout