JRCZ Temperature Monitoring Application

Team 7 - Celsius

Authors:

- Daniel Bartha
- Bartosz Adamczyk
- Fomum Pristen
- Emil Hristov
- Ivan Karev
- Michal Jakubowski
- Tygo van der Linde

Introduction

Our Laravel application aims to provide a clear visualization of room records inside the JRCZ Reasearch Centre. The purpose of this application is to help monitor past temperatures in order to analyze and reduce energy waste.

The application consists of th following main features:

```
- Import features, to import
csv formats into the application

- Room selector and datepicker, collaborating with eachother

- Visualization, the application is designed to visualize the imported data

- There are six charts representing temperatures, CO2 levels, outside
temperatures, minimum, maximum and average temperatures

- An authentication feature is also included in the application

- Only the Administrator account has the proper authorization to register new
users

- Guest accounts can view the dashboard but are not able to import data or create
accounts

- Translation feature from English to Dutch and vice-versa
```

Target Audience

Our target audience consists of:

- Our client, Loek van der Linde
- Technical services of HZ
- Laboratory managers of the JRCZ
- HZ teachers
- HZ students
- Potential visitors

Value and benefits of our product

Our application visualizez room records and outside temperatures in order to provide a better understanding of temperature changes.
The application simplifies the task of importing and analyzing data.
Our database is designed in such a way that it simplifies the task of importing data and structures the data in a way that makes it easy for the users to visualize data for selected dates and rooms.

Installation and Setup

Dependencies

- Laravel
- PHP
- Composer
- NPM package manager
- AdminLTE
- Black Dashboard Template

Application Setup

In order to use the code for our application the following steps are necessary.

Intsall composer

```
composer install
```

Install NPM

```
npm install
```

Copy `.env` file

```
cp .env.example .env
```

Create an application key

```
php artisan key:generate
```

Migrate and seed

```
php artisan migrate
```

```
php artisan db:seed
```

API Documentation

Data retrieval from database

First we organize the data for ease of use

```
    public function getRoomsAndTemperatures()
    {
        $data = [];
        $rooms = Room::all();
        foreach ($rooms as $room) {
            $roomTimes = [];
            foreach ($room->roomTime as $roomTime) {
                $seperatedTime = explode(' ', $roomTime->time);
                array_push($roomTimes, [
                    'dateAndTime' => [
                        'date' => $seperatedTime[0],
                        'time' => $seperatedTime[1]
                    ],
                    'co2' => $roomTime->co2,
                    'temperature' => $roomTime->temperature
                ]);
            }
            $roomData = [
                'room_name' => $room->room_number,
                'roomTimes' => $roomTimes
```

```
            ];
            array_push($data, $roomData);
        }
        $outsideTemperatureImport = OutsideTemperatureImport::all()->sort();
        $outsideTemperatures = [];
        foreach ($outsideTemperatureImport as $outsideTemperature) {
            $seperatedTime = explode(' ', $outsideTemperature->time);
            array_push($outsideTemperatures, [
                'dateAndTime' => [
                    'date' => $seperatedTime[0],
                    'time' => $seperatedTime[1]
                ],
                'outside_temperature' => $outsideTemperature->outside_temperature
            ]);
        }
        array_push($data, $outsideTemperatures);
        return $data;
    }
```

Then we retrieve the data with the usage of asynchronous functions

```
async function getData(url) {
    try {
        console.log('processing data');
        let response = await fetch(url);
        let data = await response.json();
        console.log('data retrieved');
        return data;
    } catch (err) {
        console.error("Error: ", err);
    }
}
```

```
async function loadDataForRoom(roomPicked) {
    roomTemperature = await getData('/co2-and-temperature-data');
    outsideTemperatures = roomTemperature[roomTemperature.length - 1];
    room1 = roomTemperature[roomPicked]['roomTimes'];
}
```

Outside Temperature API

This is an external API from weerlive.nl
Our implementation can be found in the "Code Structure and Organization" section, below.

Code Structure and Organization

Our application uses the standard MVC (Model-View-Controller) architecture.

Each 5 minutes, there is an API automatically called in /outsideTemperature path that retrieves outside temperature data (1) and inserts them into the database to outside_temperatures table (2).

(1) OutsideTemperatureController

```
public function create()
{
    $client = new \GuzzleHttp\Client();
    $response = $client->request('GET', 'https://weerlive.nl/api/json-data-
10min.php?key=f1de1e8df9&locatie=Middelburg');
    $request = json_decode($response->getBody()->getContents())->liveweer[0];
    $this->store($request);
}
```

(2) OutsideTemperatureController

```
public function store($request)
{
    $outsideTemperature = new \App\Models\OutsideTemperature();
    $request->time = $this->convertTime($request->time);
    $outsideTemperature->time = $request->time;
    $outsideTemperature->temperature = $request->temp;
    $outsideTemperature->image = $request->image;
    $outsideTemperature->save();
}
```

web.php

```
Route::resource("outsideTemperature",
\App\Http\Controllers\OutsideTemperatureController::class);
```

When entering, refreshing the page/subpage, the latest data retrieved in the backend from the database is displayed in the sidebar of our page.

```
<div class="max-w-6xl mx-auto sm:px-6 lg:px-8">
    <?php
        $outsideTemperatures = \App\Models\OutsideTemperature::all()-
>sortByDesc('time')->first();
        echo '<span style="color:white"> Middelburg: </span>';
        echo '<img id="weatherIcon" style="width: 1rem; filter: invert(1);
```

```
vertical-align: baseline;" src="asset("images")/iconen-
weerlive/'.$outsideTemperatures->image.'.svg">';
        echo '<span style="color: white">'.$outsideTemperatures-
>temperature.'</span>';
        echo '<img id="celsiusIcon" style="width: 1rem; filter: invert(1);
vertical-align: top" src="asset("images")/iconen-weerlive/celsius.svg">';
    ?>
    <script>
        document.getElementById('weatherIcon').src = '{{asset("images")}}/iconen-
weerlive/{{$outsideTemperatures->image}}.svg';
        document.getElementById('celsiusIcon').src = '{{asset("images")}}/iconen-
weerlive/celsius.svg';
    </script>
</div>
```

Configuration and Customization

By utilising ChartJS, our application allows for easy customization of the charts from within the code.

For example:

```
let chartTemperature = document.getElementById('chartContainerTemperature');
    let tempChart = new Chart(chartTemperature, {
        type: "line",
        data: {
            labels: [],
            datasets: [
                {
                    label: "Temperature",
                    data: [],
                    borderWidth: 1,
                    borderColor: "rgb(114,101,197)",
                    backgroundColor: "rgba(95,79,217,0.4)",
                    fill: 'origin'
                }
            ]
        },
    });
```

Colors, labels and data can easily be changed.

Error Handling

Errors are properly handled within our application.

```
We handle 404 and 500 errors along with additional ones. All of which update you
on current status and helps you get redirected to the Home page.
```

```
Validation and error prevention is also present in our application on the Log In
page and registration page.
```

```
System flashing is also implemented to notify the user in case there is a lack of
data after importing and also during visualization.
```

Performance

Potential bottlenecks may occur when there is an abundance of data in the database. This can menifest in longer loading times.

Importing could result in bottlenecks due to timeout.