VIDZEME UNIVERSITY OF
APPLIED SCIENCES
ENGINEERING FACULTY

WEB APPLICATION FOR NEARBY MCDONALDS

COURSE WORK

Author: Gustavs Oto Cers
Stud. id Nr.: IT21024

VALMIERA 2023

CONTENT

INTRODUCTION

McDonald’s stands as the top runner of fast-food restaurants in the United States, with approximately 13500 locations nationwide and 40000 global restaurants in 118 countries serving an impressive 69 million people daily. With more financial options and food delivery apps reshaping how Americans eat, McDonald's continues to be the favorite choice for many people who want a quick, familiar, and tasty meal (David, 2022). However, not always does every customer know what alternative McDonalds restaurants are in their city. In addition to that not all locations have the same weather. For instance, some individuals may feel happier in warm and sunny weather, while others may experience mood declines on rainy days (Emily 2022).

The goal is to enhance the user experience by integrating geospatial data for McDonald's locations and real-time weather.

Objectives include improving the user interface, providing real-time weather data, enabling radius selection, ensuring data accuracy, and optimizing data presentation to help users make informed decisions regarding local McDonald's locations and their weather.

The documentation covers app overview, technical details, and data accuracy, to ensure that users understand the app's purpose, how it works, and the reliability of data.

APPLICATION OVERVIEW

GATHERING SPATIAL DATA

To get live weather data for web application Open-Meteo is used. It is an open-source weather API with free access for non-commercial use and no API Key required (Open-Meteo, 2023a). Documentation is easy to navigate and extra variables like humidity, soil temperature, weather code are easy to attach to the received data. It automatically generates the necessary URL so there is no need to create a request server (Open-Meteo, 2023b).

However, to gather McDonalds location data, reverse engineering was needed to find McDonalds API from their official website using their location finder (McDonalds 2023). Also, problems with fetching data directly from the API made it to where an HTTP request server for it needed to be created to act as middle ground between McDonalds and the user.

Data from these APIs is passed to the application in JSON format which is later initialized to create markers, display weather data and radius of the area for the search.

COMPONENTS

Application consists of total of 5 components: index.php, app.js, test.php, styles.css and node.js server for requests.

Node.js server listens on port 3000 and handles GET requests made to /mcdonalds-locations API. When a client requests McDonald's locations near him, the script extracts query parameters (latitude, longitude, radius, maxResults, country, and language) and constructs an API request to McDonald's servers. Upon receiving the data, it responds with the retrieved information or an error message if something goes wrong.

Index.php serves as the main HTML document for the web application. It defines the structure and layout of the web page. This file includes essential elements such as the map container, radio buttons for selecting a search radius, and a container for displaying real-time weather information. It links to external resources like the Leaflet library, which is crucial for creating interactive maps. Additionally, it imports the primary JavaScript logic from "app.js," which enables the web application's core functionality.

Test.php is a PHP script that interfaces with the McDonald's location API. It receives user-defined parameters like latitude, longitude, and radius through the URL query string. It then constructs an API request and handles the response. This script processes the data, extracts location details, and formats them into JSON for display on the map.

App.js is a JavaScript file responsible for the core functionality of the web application. It initializes and configures the Leaflet map, setting its initial view coordinates and adding map tiles from OpenStreetMap. It also manages user interactions, allowing users to click on the map to retrieve real-time weather data at the selected location as well as retrieve McDonald's restaurant locations based on the selected search radius in miles. The script displays weather information, McDonald's locations, and interactive markers on the map, providing an engaging user experience.

Styles.css contains CSS rules for styling the web application created in "index.php." It defines the visual layout, including elements like fonts, colors, margins, and positioning. These styles ensure a user-friendly and visually appealing interface, enhancing the overall user experience.

USERS GUIDE

To start using this web application you need to follow these steps:

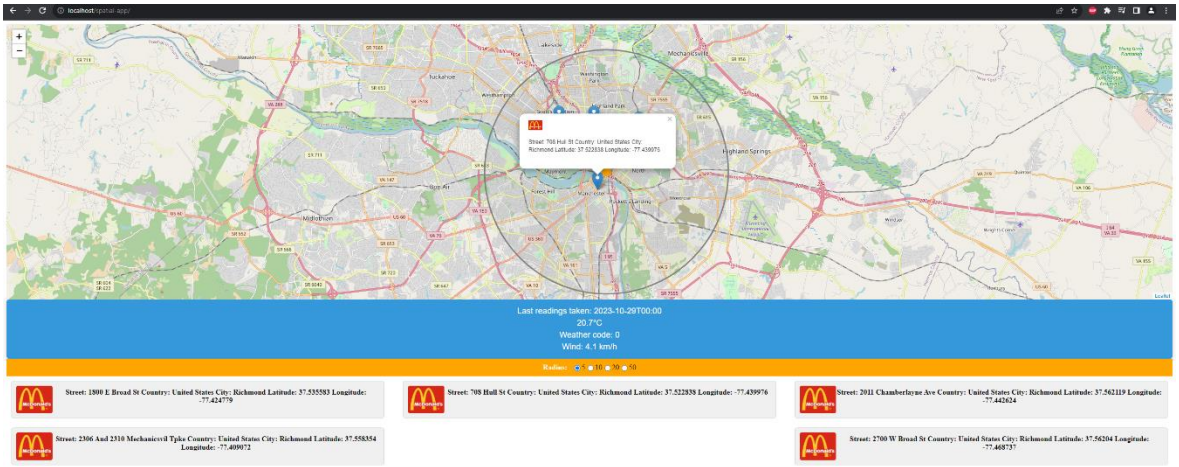
1. Create necessary components and put them in WAMP /www folder.
2. Start up the WAMP server.
3. Open localhost on web browser and open the folder.
4. To view nearby McDonalds restaurants, click on map (only works for United States region because the API does not display McDonalds in other countries).
5. After clicking orange pointer appears with blue markers that represent nearby McDonalds. (If you click on them, they show their information).
6. Based on selected radius a circle of search is drawn.
7. Weather data is displayed for the location.
8. If there are nearby McDonalds, they will be displayed in blocks which when clicked on will popup the blue marker for it.
9. When clicking on new location old markers will be deleted.

LIMITATIONS

As previously mentioned, this web application only works when searching in United States region because API comes from there. Latvia has its own McDonalds website to which you can not make API requests with longitude and latitude. In addition to that the McDonalds API does not return restaurants further than 50 miles radius or more than 30 restaurants for the currently selected range. The live weather data is updated every one hour so data is not always

END RESULT

The final product is an application that seamlessly integrates geospatial data from different sources and presents it to users in an intuitive and sophisticated manner, simplifying their access to valuable information.



1.3.1 Picture. Web application being used.

REFERENCES

1. David Chang, (08.10.2022), Here Are the Most Popular Fast Food Brands in America, available: https://www.fool.com/the-ascent/personal-finance/articles/here-are-the-most-popular-fast-food-brands-in-america/
2. Emily Swaim, (12.08.2022), Yes, Weather Can Affect Mood and Energy — and So Can Climate Change, available: https://www.healthline.com/health/mental-health/weather-and-mood
3. Open-Meteo, 2023, Free Weather API, available: https://open-meteo.com/
4. Open-Meteo, 2023, docs, available: https://open-meteo.com/en/docs
5. McDonalds, 2023, restaurant locator, available: https://www.mcdonalds.com/us/en-us/restaurant-locator.html

ATTACHMENTS

index.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Simple Spatial Data App</title>
    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css" />
    <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div id="map"></div>
    <div id="data-list"></div>
    <div class="buttons">
        <label>Radius:</label>
        <input type="radio" name="radius" value="5" checked> 5
        <input type="radio" name="radius" value="10"> 10
        <input type="radio" name="radius" value="20"> 20
        <input type="radio" name="radius" value="50"> 50
    </div>
    <div id="locationsContainer"></div>

    <script src="app.js"></script>
</body>
```

node.js

```
const express = require('express');
const fetch = require('node-fetch');

const app = express();
const port = 3000;
```

```
app.get('/mcdonalds-locations', async (req, res) => {
  try {
    const latitude = req.query.latitude;
    const longitude = req.query.longitude;
    const radius = req.query.radius || 50;
    const maxResults = req.query.maxResults || 30;
    const country = req.query.country || 'us';
    const language = req.query.language || 'en-us';
    const apiUrl = `https://www.mcdonalds.com/googleappsv2/geolocation?` +
                   `latitude=${latitude}&longitude=${longitude}&` +
                   `radius=${radius}&maxResults=${maxResults}&` +
                   `country=${country}&language=${language}`;

    const response = await fetch(apiUrl);
    const data = await response.json();
    res.json(data);
  } catch (error) {
    res.status(500).json({ error: 'Error fetching data from McDonald\'s API' });
  }
});

app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

styles.css

```
#map {
  width: 100%;
  height: 600px;
}

#locationsContainer {
  display: flex;
  flex-wrap: wrap;
```

```
    justify-content: space-between;
}

.Box {
    background-color: #3498db;
    color: #fff;
    padding: 10px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
    text-align: center;
    font-family: Arial, sans-serif;
}

.Box p {
    margin: 5px;
    font-size: 18px;
}

.buttons {
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: orange;
    padding: 10px;
}

.buttons label {
    margin-right: 10px;
    font-weight: bold;
    color: white;
}

.location {
```

```
    background-color: #f0f0f0;
    padding: 10px;
    margin: 10px;
    border: 1px solid #ddd;
    border-radius: 5px;
    width: 32%;
    display: inline-block;
    vertical-align: top;
    text-align: center;
    box-sizing: border-box;
    position: relative;
    display: flex;
    flex-direction: column;
}

.locations {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    justify-content: space-between;
}

.location p {
    margin: 5px 0;
    font-weight: bold;
    text-align: center;
    flex: 4;
}

.location-logo {
    max-width: 10%;
    height: auto;
    flex: 1;
```

```
}
```

app.js

```
//Leaflet map with a specified center and zoom level
const map = L.map('map').setView([39.8002401, -101.1848659], 5);

//OpenStreetMap layer
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    maxZoom: 19,
}).addTo(map);

//For deleting markers
const markers = [];
let currentMarker = null;
let circle = null;

// When clicked on map
map.on('click', function(e) {
    const selectedRadius = document.querySelector('input[name="radius"]:checked').value;
    const latitude = e.latlng.lat;
    const longitude = e.latlng.lng;

    // Fetch weather data based on the clicked location
    fetch(`https://api.open-
meteo.com/v1/forecast?latitude=${latitude}&longitude=${longitude}&hourly=temperatur
e_2m,weathercode,windspeed_10m`, {
        method: 'GET',
    })
    .then(response => response.json())
    .then(data => {
        const temperature = data.hourly.temperature_2m[0];
        const weatherCode = data.hourly.weathercode[0];
        const windspeed = data.hourly.windspeed_10m[0];
        const time = data.hourly.time[0];
```

```
        // <div> block that displays fetched weather data
        const locationInfo = document.getElementById('data-list');
        locationInfo.innerHTML = `
            <div class="Box">
                <p>Last readings taken: ${time} </p>
                <p> ${temperature}°C </p>
                <p> Weather code: ${weatherCode} </p>
                <p> Wind: ${windspeed} km/h</p>
            `;
        if (circle) {
            map.removeLayer(circle);
        }
    })
    .catch(error => {
        console.error('Error fetching weather data:', error);
    });

    // Removes existing blue markers
    markers.forEach(marker => {
        map.removeLayer(marker);
    });
    //removes orange marker
    markers.length = 0;
    if (currentMarker) {
        map.removeLayer(currentMarker);
    }

    // Fetching McDonalds data

fetch(`test.php?radius=${selectedRadius}&latitude=${latitude}&longitude=${longitude}`,
{
        method: 'GET',
```

```
    })
    .then(response => response.json())
    .then(data => {
        const locationsContainer = document.getElementById('locationsContainer');
        locationsContainer.innerHTML = '';
        data.forEach(location => {
            // makes <div> block for each fetched location
            const locationDiv = document.createElement('div');
            locationDiv.classList.add('location');
            const locationInfoHTML = `
                <div class="locations">
                    <img src="./logo.png" alt="Logo" class="location-logo">
                    <p>Street: ${location.addressLine1}  Country: ${location.addressLine4}
City: ${location.addressLine3}  Latitude: ${location.latitude}  Longitude:
${location.longitude}</p>
                </div>
            `;
            locationDiv.innerHTML = locationInfoHTML;
            locationsContainer.appendChild(locationDiv);
            const marker = L.marker([location.latitude, location.longitude])
                .bindPopup(locationInfoHTML)
                .addTo(map);

            markers.push(marker);
            locationDiv.querySelector('p').addEventListener('click', () => {
                marker.openPopup();
            });
        });

        // Adds an orange marker for the clicked location
        currentMarker = L.marker([latitude, longitude], { icon: orangeIcon }).addTo(map);
        const group = new L.featureGroup(markers);
        map.fitBounds(group.getBounds());
```

```
        const radiusInMeters = selectedRadius * 1609.34;
        circle = L.circle([latitude, longitude], {
            radius: radiusInMeters,
            color: 'gray',
            fillColor: 'gray',
            fillOpacity: 0.1,
        }).addTo(map);
    })
    .catch(error => {
        console.error('Error fetching locations:', error);
    });
});

// Cursom marker for current location (orange)
const orangeIcon = L.divIcon({
    className: 'custom-icon',
    iconSize: [20, 20],
    html: '<div style="background-color: orange; width: 20px; height: 20px; border-radius:
50%;"></div>',
});
```