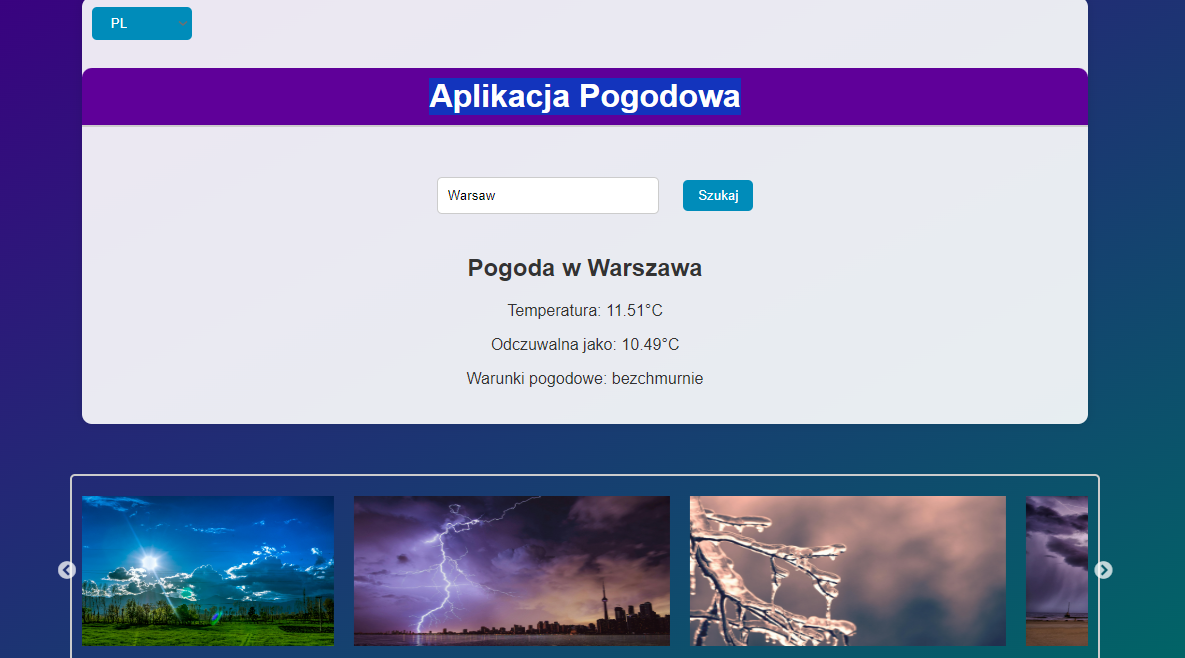
Logo, company name

Description automatically generated

# Dokumentacja Z przedmiotu: Wprowadzenie do technologii internetowych Temat: Projekt "Aplikacja Pogodowa"

**Opracowanie::** Mariia Moldavan, Nr Albumu: 68423

### **Opis projektu**

Projekt "Aplikacja Pogodowa" jest aplikacją webową, która umożliwia użytkownikom sprawdzenie aktualnej pogody dla wybranego miasta oraz przeglądanie galerii obrazów. Korzystając z API OpenWeatherMap, użytkownicy mogą szybko uzyskać informacje na temat temperatury, odczuwalnej temperatury i warunków pogodowych w wybranym miejscu. Dodatkowo aplikacja zawiera dynamiczną galerię obrazów, która automatycznie przewija się, prezentując różne obrazy.  
  
[](https://www.pexels.com/search/weather/)

Rysunek 1 Glowny wyglad aplikacji. Opracowane samodzielnie

*Zdziecia pobrane ze strony https://www.pexels.com/search/weather/*

### **Wykorzystane technologie**

Projekt został zrealizowany przy użyciu następujących technologii:

* **HTML**: Struktura i zawartość strony internetowej.
* **CSS**: Stylizacja interfejsu użytkownika.
* **JavaScript**: Interakcja z użytkownikiem, integracja z API OpenWeatherMap oraz zarządzanie galerią obrazów.
* **API OpenWeatherMap**: Pobieranie danych pogodowych dla podanych miast.
* **Slick.js**: Biblioteka JavaScript do tworzenia dynamicznych sliderów i karuzel obrazów.

### **Funkcjonalności**

1. **Sprawdzanie pogody**:
   * Użytkownik może wprowadzić nazwę miasta w 4 jezykach w pole tekstowym.
   * Po naciśnięciu przycisku "Szukaj", aplikacja wysyła zapytanie do API OpenWeatherMap, aby pobrać aktualne dane pogodowe dla podanego miasta.
   * Dane pogodowe, takie jak temperatura, odczuwalna temperatura i warunki pogodowe, są wyświetlane na stronie.  
       
      
2. **Przeglądanie galerii obrazów**:
   * Aplikacja wyświetla galerię 12 obrazów zawierającą różne zdjęcia.
   * Zdjęcia są automatycznie przewijane za pomocą slidera Slick, który wyświetla trzy obrazy na raz.
   * Użytkownik może przeglądać obrazy, przewijając w prawo lub lewo.

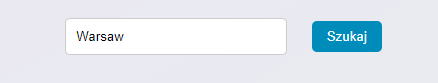
### **Funkcje**

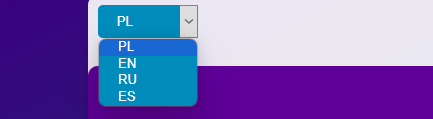
Projekt "Aplikacja Pogodowa" posiada następujące funkcje:

* **Pobieranie danych pogodowych**: Aplikacja korzysta z API OpenWeatherMap, aby uzyskać aktualne dane pogodowe dla wprowadzonego miasta.
* **Wyświetlanie danych pogodowych**: Po otrzymaniu danych pogodowych, aplikacja wyświetla je na stronie w czytelny sposób.
* **Przeglądanie galerii obrazów**: Użytkownicy mogą przeglądać różne obrazy w galerii, korzystając z funkcjonalności automatycznego przewijania dostarczonej przez bibliotekę Slick.js.

W projekcie "Aplikacja Pogodowa" występują dwie główne interaktywne elementy, które są wykorzystywane jako przyciski:

**W projekcie "Aplikacja Pogodowa" występują dwie główne interaktywne elementy, które są wykorzystywane jako przyciski:**

1. **Przycisk "Szukaj"**:  
   * Ten przycisk jest używany do rozpoczęcia wyszukiwania danych pogodowych dla wprowadzonej nazwy miasta.
   * Po kliknięciu tego przycisku aplikacja wysyła zapytanie do API OpenWeatherMap, aby pobrać aktualne dane pogodowe dla podanego miasta.  
       
       
     
2. **Przycisk wyboru języka**:  
   * Ten przycisk jest rozwijaną listą, która umożliwia użytkownikowi wybór 4 języka, w jakim chciałby otrzymywać informacje o pogodzie.
   * Po wybraniu nowego języka z listy, aplikacja automatycznie przystosowuje wyświetlane dane do wybranego języka.



### **Podsumowanie**

Projekt "Aplikacja Pogodowa" jest interaktywną aplikacją webową, która umożliwia użytkownikom sprawdzenie aktualnej pogody dla wybranego miasta oraz przeglądanie galerii obrazów. Główne cechy i zalety tego projektu obejmują:

1. **Prosta interakcja użytkownika**:
   * Aplikacja oferuje prosty i intuicyjny interfejs, który umożliwia użytkownikom łatwe korzystanie z funkcji sprawdzania pogody i przeglądania galerii obrazów.
   * Interaktywne przyciski "Szukaj" i przycisk wyboru języka umożliwiają użytkownikom szybkie i wygodne dostęp do funkcji aplikacji.
2. **Informacje o pogodzie na żądanie**:
   * Dzięki integracji z API OpenWeatherMap, użytkownicy mogą szybko uzyskać aktualne dane pogodowe dla dowolnego miasta na świecie.
   * Wyświetlane informacje zawierają kluczowe parametry, takie jak temperatura, odczuwalna temperatura i warunki pogodowe, co umożliwia użytkownikom uzyskanie kompletnego obrazu aktualnej sytuacji pogodowej.
3. **Przyjemne doświadczenie przeglądania obrazów**:
   * Aplikacja oferuje dynamiczną galerię obrazów, która automatycznie przewija się, prezentując różne obrazy.
   * Dzięki wykorzystaniu biblioteki Slick.js, użytkownicy mogą komfortowo przeglądać obrazy, korzystając z funkcjonalności przewijania w prawo i lewo.
4. **Dostosowanie do preferencji użytkownika**:
   * Aplikacja umożliwia użytkownikom wybór języka, w jakim chcieliby otrzymywać informacje o pogodzie, co zapewnia spersonalizowane doświadczenie użytkownika.
   * Wybór języka jest prosty i wygodny, dzięki rozwijanej liście języków dostępnej na stronie.

W sumie, "Aplikacja Pogodowa" jest praktycznym i przyjemnym narzędziem do sprawdzania pogody i przeglądania obrazów, które zapewnia użytkownikom łatwe dostęp do ważnych informacji i przyjemnych wrażeń wizualnych. Dzięki prostocie interfejsu i intuicyjnym funkcjom może być użytecznym narzędziem dla szerokiej grupy użytkowników.

### **1. Opis kodu HTML:**

1. **Struktura główna**:
   * Kod HTML definiuje strukturę głównej strony aplikacji.
   * Zawiera elementy takie jak nagłówek, sekcja wyszukiwania pogody, wyniki pogodowe oraz galeria obrazów.
2. **Formularz wyszukiwania pogody**:
   * Zawiera pole tekstowe do wprowadzenia nazwy miasta, przycisk "Szukaj" oraz rozwijaną listę języków.
3. **Sekcja wyników pogodowych**:
   * Wyświetla pobrane dane pogodowe dla wybranego miasta.
4. **Galeria obrazów**:
   * Przedstawia różne obrazy w formie przewijanej galerii.  
       
     Sam kod ‘index.html’:
5. <!DOCTYPE html>
6. <html lang="pl">
7. <head>
8. <meta charset="UTF-8">
9. <title>Aplikacja Pogodowa</title>
10. <link rel="stylesheet" href="styles.css"> <!-- Względem pliku CSS -->
11. <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.8.1/slick.min.css">
12. <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.8.1/slick-theme.min.css">
13. <style>
14. body {
15. font-family: Arial, sans-serif;
16. margin: 0;
17. padding: 0;
18. background: linear-gradient(to bottom right, #3a0080, #006666); /\* Gradientowe tło \*/
19. color: #fff; /\* Kolor tekstu na tle \*/
20. }
21. .container {
22. width: 80%;
23. margin: auto;
24. text-align: center;
25. padding-top: 50px;
26. background-color: rgba(255, 255, 255, 0.9); /\* Półprzezroczyste białe tło \*/
27. border-radius: 10px;
28. box-shadow: 0 0 10px rgba(0,0,0,0.1);
29. padding-bottom: 20px;
30. color: #333; /\* Zmiana koloru tekstu w kontenerze \*/
31. position: relative;
32. }
33. h1 {
34. margin-bottom: 30px;
35. padding: 10px;
36. border-bottom: 2px solid #ccc;
37. border-radius: 10px 10px 0 0;
38. background-color: #5f0099; /\* Kolor tła nagłówka \*/
39. color: #fff; /\* Zmiana koloru tekstu w nagłówku \*/
40. }
41. input[type="text"] {
42. width: 200px;
43. padding: 10px;
44. margin: 20px;
45. border: 1px solid #ccc;
46. border-radius: 5px;
47. }
48. button {
49. padding: 8px 15px; /\* Zmniejszenie rozmiaru przycisku \*/
50. background-color: #008CBA;
51. color: white;
52. border: none;
53. border-radius: 5px;
54. cursor: pointer;
55. }
56. button:hover {
57. background-color: #005f73;
58. }
59. #weatherResult {
60. margin-top: 20px;
61. }
62. .gallery-container {
63. width: 80%;
64. margin: auto;
65. margin-top: 50px;
66. border: 2px solid #ccc; /\* Dodanie ramki dla galerii \*/
67. border-radius: 5px;
68. padding: 10px;
69. }
70. .gallery {
71. display: flex;
72. justify-content: center;
73. }
74. .gallery img {
75. width: 200px;
76. height: 150px;
77. margin: 10px;
78. }
79. #language { /\* Styl dla rozwijanej listy \*/
80. width: 100px; /\* Zdefiniowanie rozmiaru przycisku \*/
81. padding: 8px 15px; /\* Zmniejszenie rozmiaru przycisku \*/
82. border: none;
83. border-radius: 5px;
84. background-color: #008CBA; /\* Kolor tła przycisku \*/
85. color: white;
86. cursor: pointer;
87. }
88. .language-container { /\* Styl dla kontenera przycisku wyboru języka \*/
89. position: absolute; /\* Pozycjonowanie absolutne \*/
90. top: 10px; /\* Położenie u góry \*/
91. left: 10px; /\* Położenie po lewej \*/
92. }
93. </style>
94. </head>
95. <body>
96. <div class="container">
97. <div class="language-container">
98. <select id="language">
99. <option value="pl">PL</option>
100. <option value="en">EN</option>
101. <option value="ru">RU</option>
102. <option value="es">ES</option>
103. </select>
104. </div>
105. <h1>Aplikacja Pogodowa</h1>
106. <input type="text" id="cityName" placeholder="Wpisz nazwę miasta">
107. <button onclick="getWeather()">Szukaj</button>
108. <div id="weatherResult"></div>
109. </div>
110. <div class="gallery-container">
111. <div id="gallery" class="gallery">
112. <!-- Zdjęcia galerii -->
113. </div>
114. </div>
115. <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
116. <script src="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.8.1/slick.min.js"></script>
117. <script src="script.js" defer></script>
118. <script>
119. // Obsługa zmiany języka
120. document.getElementById('language').addEventListener('change', function() {
121. // Pobierz wartość wybranego języka
122. var language = this.value;
123. // Wywołaj funkcję zmieniającą język
124. changeLanguage(language);
125. });
126. // Funkcja zmieniająca język
127. function changeLanguage(language) {
128. // Twój kod zmieniający język
129. console.log('Wybrany język: ' + language);
130. }
131. </script>
132. </body>
133. </html>

### **2. Opis kodu CSS ‘style.css’:**

**body, html**: Resetuje marginesy i padding dla elementów **body** i **html**, aby uniknąć domyślnych przesunięć w przeglądarce. Ustawia również styl czcionki na Arial lub podobną czcionkę bezszeryfową.

**background-color**: Ustawia tło strony na kolor szary (#f2f2f2).

**text-align**: Wyśrodkowuje tekst na stronie.

**.container**: Stylizuje kontener aplikacji, dodając margines na górze o wartości 50 pikseli.  
 **input[type="text"]**: Stylizuje pole tekstowe, ustawiając jego szerokość na 200 pikseli, dodając wewnętrzny margines (padding) i zewnętrzny margines (margin) oraz dodając ramkę o grubości 1 piksela i szarym kolorze.  
 **button**: Stylizuje przycisk, ustawiając jego wewnętrzny margines (padding), kolor tła, kolor tekstu, zaokrąglenie rogów i kursor myszy.

**button:hover**: Zmienia kolor tła przycisku na niebieski (#005f73) po najechaniu kursorem.  
 **#weatherResult**: Stylizuje sekcję wyników pogodowych, dodając margines na górze o wartości 20 pikseli.  
 **.gallery-container**: Stylizuje kontener galerii obrazów, ustawiając jego szerokość na 80% szerokości okna przeglądarki i wyśrodkowując go.  
 **.gallery**: Stylizuje galerię obrazów, ustawiając jej wyświetlanie na flex, aby elementy w galerii były wyśrodkowane, a także aby umożliwić dodawanie marginesów wokół obrazów.  
 **.gallery img**: Stylizuje obrazy w galerii, ustawiając ich szerokość na 200 pikseli, wysokość na 150 pikseli i dodając margines wokół każdego obrazu.

body, html {

    margin: 0;

    padding: 0;

    font-family: Arial, sans-serif;

    background-color: #f2f2f2;

    text-align: center;

}

.container {

    margin-top: 50px;

}

input[type="text"] {

    width: 200px;

    padding: 10px;

    margin: 20px;

    border: 1px solid #ccc;

    border-radius: 5px;

}

button {

    padding: 10px 20px;

    background-color: #008CBA;

    color: white;

    border: none;

    border-radius: 5px;

    cursor: pointer;

}

button:hover {

    background-color: #005f73;

}

#weatherResult {

    margin-top: 20px;

}

.gallery-container {

    width: 80%;

    margin: auto;

}

.gallery {

    display: flex;

    justify-content: center;

}

.gallery img {

    width: 200px; /\* Ustaw stałą szerokość obrazu \*/

    height: 150px; /\* Ustaw stałą wysokość obrazu \*/

    margin: 10px; /\* Dodaj margines wokół obrazu \*/

}

**3. Opis funkcji i ich działania w JavaScript ‘script.js’:**

1. **getWeather()**:
   * Ta funkcja jest asynchroniczna i służy do pobierania danych pogodowych dla określonego miasta z wykorzystaniem API OpenWeatherMap.
   * Pobiera nazwę miasta z pola tekstowego o id "cityName" oraz wybrany język z listy rozwijanej o id "language".
   * Tworzy URL zapytania do API OpenWeatherMap zawierającego nazwę miasta, klucz API, jednostki oraz wybrany język.
   * Wysyła zapytanie HTTP do API OpenWeatherMap za pomocą metody fetch().
   * W przypadku udanej odpowiedzi przetwarza dane JSON i wyświetla je na stronie za pomocą funkcji displayWeather().
   * Obsługuje błędy, wyświetlając informacje o nich na stronie oraz logując je w konsoli.
2. **displayWeather(data)**:
   * Ta funkcja przyjmuje obiekt danych pogodowych jako argument.
   * Wyświetla na stronie nazwę miasta, temperaturę, odczuwalną temperaturę oraz opis warunków pogodowych.
3. **DOMContentLoaded event**:
   * Nasłuchuje zdarzenia DOMContentLoaded, które występuje po załadowaniu całej struktury HTML.
   * Po wystąpieniu tego zdarzenia wywołuje funkcję loadGallery().
4. **loadGallery()**:
   * Funkcja ta jest wywoływana po załadowaniu struktury HTML lub po zmianie języka.
   * Pobiera wybrany język z listy rozwijanej o id "language".
   * Tworzy adresy URL obrazów galerii na podstawie wzorca.
   * Tworzy elementy obrazów i dodaje je do kontenera galerii.
   * Inicjalizuje slider Slick dla galerii.

Funkcje te są używane w aplikacji pogodowej do pobierania danych pogodowych dla określonego miasta, wyświetlania ich na stronie oraz obsługi galerii zdjęć. Dodatkowo, są obsługiwane błędy, które mogą wystąpić podczas pobierania danych.

// Funkcja asynchroniczna do pobierania danych pogodowych

async function getWeather() {

    // Pobiera nazwę miasta z pola tekstowego

    const cityName = document.getElementById('cityName').value;

    // Pobiera wybrany język z pola wyboru języka

    const language = document.getElementById('language').value;

    // Ustawia klucz API

    const apiKey = 'c8b6885dc4c8c7d15c1ed48775e91e57';

    // Utwórz URL zapytania do API OpenWeatherMap

    const url = `https://api.openweathermap.org/data/2.5/weather?q=${cityName}&appid=${apiKey}&units=metric&lang=${language}`;

    try {

        // Wykonuje zapytanie HTTP do API OpenWeatherMap

        const response = await fetch(url);

        // Sprawdza, czy odpowiedź jest poprawna

        if (!response.ok) {

            // Jeśli odpowiedź nie jest poprawna, wyświetl komunikat błędu

            throw new Error(`Miasto nie znalezione: ${response.status}`);

        }

        // Jeśli odpowiedź jest poprawna, pobierz dane JSON

        const data = await response.json();

        // Wyświetla dane pogodowe na stronie

        displayWeather(data);

    } catch (error) {

        // Jeśli wystąpił błąd, wyświetl komunikat błędu w konsoli i na stronie

        console.error('Błąd podczas pobierania danych pogodowych:', error);

        document.getElementById('weatherResult').innerText = 'Błąd podczas pobierania danych pogodowych. Sprawdź nazwę miasta.';

    }

    // Wywołaje funkcję do ładowania galerii

    loadGallery(); // Przeniesione wywołanie loadGallery() do getWeather()

}

// Funkcja do wyświetlania danych pogodowych na stronie

function displayWeather(data) {

    // Pobiera potrzebne dane z obiektu danych

    const { name, main, weather } = data;

    // Wyświetla dane pogodowe na stronie

    document.getElementById('weatherResult').innerHTML = `

        <h2>Pogoda w ${name}</h2>

        <p>Temperatura: ${main.temp}°C</p>

        <p>Odczuwalna jako: ${main.feels\_like}°C</p>

        <p>Warunki pogodowe: ${weather[0].description}</p>

    `;

}

// Nasłuchuj zdarzenia DOMContentLoaded, aby wykonać kod po załadowaniu struktury HTML

document.addEventListener('DOMContentLoaded', () => {

    // Pobierz element wyboru języka

    const languageSelect = document.getElementById('language');

    // Pobiera kontener galerii

    const gallery = document.getElementById('gallery');

    // Funkcja do ładowania galerii

    function loadGallery() {

        // Pobiera wybrany język z pola wyboru języka

        const selectedLanguage = languageSelect.value;

        // Wyczyśća zawartość galerii przed ponownym załadowaniem

        gallery.innerHTML = '';

        // Tablica zawierająca adresy URL obrazów w galerii

        const images = [

            'images/img1(1).jpg',

            'images/img1(2).jpg',

            'images/img1(3).jpg',

            'images/img1(4).jpg',

            'images/img1(5).jpg',

            'images/img1(6).jpg',

            'images/img1(7).jpg',

            'images/img1(8).jpg',

            'images/img1(9).jpg',

            'images/img1(10).jpg',

            'images/img1(11).jpg',

            'images/img1(12).jpg'

        ];

        // Iteruj przez tablicę obrazów i dodaj je do galerii

        images.forEach(image => {

            const imgElement = document.createElement('img');

            imgElement.src = image;

            imgElement.alt = `Image ${image}`;

            gallery.appendChild(imgElement);

        });

        // Inicjalizuj slider Slick dla galerii

        $('.gallery').slick({

            slidesToShow: 3,

            slidesToScroll: 1,

            autoplay: true,

            autoplaySpeed: 2000,

        });

    }

    // Nasłuchuj na zmianę języka i ponownie ładować galerię

    languageSelect.addEventListener('change', loadGallery);

    // Załaduje galerię po załadowaniu struktury HTML

    loadGallery();

});

Dzieki 😊