

UNIWERSYTET RZESZOWSKI
WYDZIAŁ NAUK ŚCISŁYCH I TECHNICZNYCH
INSTYTUT INFORMATYKI



Krystian Filipek
134907

Informatyka

Technologie Internetowe - JavaScript

Praca projektowa

Praca wykonana pod kierunkiem
dr. Katarzyna Garwol

Rzeszów 2025

Spis treści

1. Wprowadzenie.....	7
2. Opis założeń projektu.....	8
2.1. Wymagania funkcjonalne	8
2.2. Wymagania niefunkcjonalne	8
3. Opis struktury projektu	10
3.1. Użyte technologie.....	10
3.2. Struktura katalogów.....	10
3.3. Konwencje kodowania i organizacja stylów.....	10
3.4. Implementacja logiki w JavaScript.....	11
3.5. Responsywność (RWD).....	11
3.6. Prezentacja najważniejszych skryptów i ich fragmentów	11
3.6.1. Zarządzanie motywem (theme.js).....	11
3.6.2. Animacja powitalna sekcji Hero (animation.js).....	12
3.6.3. Pasek nawigacji (animation.js)	12
3.6.4. Konwerter liczb rzymskich (calc.js)	12
3.6.5. Interaktywna lista zadań (todo.js)	13
3.6.6. Obsługa formularza i system alertów (form.js)	14
4. Harmonogram realizacji projektu	15
4.1. Etapy realizacji projektu.....	15
4.2. System kontroli wersji.....	15
5. Prezentacja warstwy użytkowej projektu.....	16
5.1. Identyfikacja wizualna i stylistyka strony	16
5.2. Kolorystyka	16
5.3. Typografia.....	16
5.4. Prezentacja witryny	16
5.4.1. Pasek nawigacji.....	16
5.4.2. Sekcja Hero.....	17
5.4.3. Lista To-Do	17
5.4.4. Kalkulator rzymski.....	17
5.4.5. Formularz kontaktowy	18
5.4.6. Alert - Błąd	18
5.4.7. Alert - Sukces.....	19
5.4.8. Tryb ciemny	19
6. Instrukcja uruchomieniowa.....	21
7. Podsumowanie.....	22
Bibliografia	23

Spis rysunków	24
---------------------	----

1. Wprowadzenie

Strona internetowa "**VEXILLARIUS**" stworzona została na potrzeby drugiego projektu (Zakres **JavaScript**) z przedmiotu **Technologie Internetowe** na 2 roku Informatyki na **Uniwersytecie Rzeszowskim**. Strona jako projekt posiada skrypty napisane w języku **JavaScript**, które nadają witrynie pewne funkcjonalności. Całość spełnia podstawowe standardy w tworzeniu stron internetowych, którymi między innymi są **responsywność**, intuicyjny układ, jak i również płynne działanie umieszczonych na stronie skryptów.

The website "**VEXILLARIUS**" was created for the second project (JavaScript) in the **Internet Technologies** course for second-year Computer Science students at the **University of Rzeszów**. The project includes scripts written in **JavaScript**, which provide the site with certain functionalities. The entire project meets basic standards for website development, including **responsiveness**, an intuitive layout, and the smooth operation of scripts placed on the site.

2. Opis założeń projektu

Celem projektu było stworzenie wielofunkcyjnej strony internetowej o nazwie **Vexillarius**, będącej praktycznym sprawdzianem podstawowych umiejętności głównie z zakresu technologii **JavaScript**. Strona została zaprojektowana w sposób spójny wizualnie z myślą o zaimplementowanych skryptach.

Głównym założeniem projektu było przejście ze statycznego dokumentu HTML/CSS do interaktywnej witryny poprzez użycie skryptów. W ramach prac zaimplementowano ich kilka, przy czym część z nich to narzędzia takie jak kalkulator oraz lista zadań, które pozwalają na interakcję z użytkownikiem.

Rozwiązywanie problemu projektowego przebiegło w następujących krokach:

1. Przygotowanie struktury dokumentu HTML5.
2. Stworzenie stylów CSS3 wraz z tymi odpowiedzialnymi za podstawową responsywność witryny - **Media Queries**.
3. Implementacja skryptów w języku JavaScript odpowiedzialnych za warstwę użytkową strony.

2.1. Wymagania funkcjonalne

Projekt realizuje następujące funkcjonalności, które pozwalają użytkownikowi na interakcję z witryną:

Przełącznik trybu light/dark mode: Możliwość zmiany motywu strony (jasny/ciemny) za pomocą jednego przycisku zlokalizowanego w prawym górnym rogu paska nawigacyjnego.

Interaktywna lista zadań (To-Do): Lista pozwala użytkownikowi na wpisywanie i dodawanie zadań, jak i ich usuwanie za pomocą kliknięcia. Lista zadań wyświetlna jest za pomocą dynamicznej listy na ekranie.

Kalkulator liczb - arabskie/rzymeskie: Kalkulator przeliczający wpisane liczby arabskie na ich rzymskie odpowiedniki, prezentując wynik w czasie rzeczywistym.

Obsługa formularza kontaktowego: Sekcja z validacją pól (imię, e-mail, treść wiadomości). Skrypt sprawdza poprawność danych i informuje użytkownika o błędach lub powodzeniu wysłania wiadomości za pomocą autorskich komunikatów - *custom alerts*.

Dynamiczny pasek nawigacji: Nagłówek strony podczas scrollowania w dół zmienia swój wygląd poprzez pojawięcie się dolnego obramowania oddzielającego go od reszty strony.

2.2. Wymagania niefunkcjonalne

Środowisko testowe

Testy projektu odbywały się lokalnie na komputerze o przedstawionej poniżej specyfikacji.

- **System operacyjny:** Microsoft Windows 11 Pro
- **Model:** Lenovo Yoga 7 2-in-1 14IML9
- **Procesor:** Intel® Core™ Ultra 5-125H 4.5 GHz
- **Pamięć RAM:** 16GB DDR4
- **Karta graficzna:** Intel® Arc™ graphics (Zintegrowana)

- **Dysk:** Samsung MZAL81T0HDLB-00BL2 1TB
- **Rozdzielcość ekranu:** 2880 x 1800
- **IDE:** Visual Studio Code

Technologie: Strona zgodnie z przyjętymi wytycznymi powinna skupiać się na wykorzystaniu skryptów w języku **JavaScript**. Dozwolone jest natomiast wykorzystanie ich w formie strony internetowej, gdzie wykorzystane będą również inne technologie.

Wydajność: Strona powinna charakteryzować się krótkim czasem ładowania poprzez brak ciężkich bibliotek.

Kompatybilność: Witryna powinna wyświetlać się poprawnie zarówno na przeglądarkach wyposażonych w silnik Chromium, jak i Gecko.

Responsywność (RWD): Strona poprawnie wyświetla się na różnych szerokościach ekranu - od monitorów aż po urządzenia mobilne. Układ elementów dostosowuje się tak, aby zachować czytelność i wygodę klikania.

Spójność wizualna: Wykorzystanie zmiennych globalnych w kaskadowych arkuszach stylów w celu zarządzania kolorystyką oraz typografią nadaje stronie estetycznego, jednolitego wyglądu.

3. Opis struktury projektu

3.1. Użyte technologie

Strona internetowa **VEXILLARIUS** została wykonana przy użyciu natywnych technologii webowych. Zgodnie z założeniami projektowymi, zrezygnowano z zewnętrznych frameworków na rzecz czystego kodu.

HTML5 wykorzystano do stworzenia podstawowego szkieletu witryny. Zastosowanie znaczników takich jak `<header>`, `<nav>`, `<section>` oraz `<footer>` zapewnia poprawną strukturę dokumentu i wspiera jego dostępność (accessibility).

CSS wykorzystano do stworzenia warstwy wizualnej opartej na nowoczesnych standardach, w tym Flexbox do układów oraz zmienne do prostego zarządzania kolorami i typografią.

JavaScript wykorzystano do stworzenia skryptów odpowiadających za funkcjonalność całej strony. Zastosowano DOM (Document Object Model) oraz obsługę zdarzeń w czasie rzeczywistym.

3.2. Struktura katalogów

Projekt został podzielony na katalogi, co zapewnia separację struktury, stylów oraz skryptów. Ułatwia to konserwację kodu oraz jego ewentualną rozbudowę.

ti-project-2/	
css/ Arkusze stylów
style.css Główny arkusz stylów witryny
documentation/ Pliki źródłowe dokumentacji
html/ Struktura dokumentów
index.html Główny plik strony internetowej
javascript/ Logika aplikacji (skrypty)
animation.js Obsługa animacji i efektów scrolla
calc.js Logika kalkulatora rzymskiego
form.js Obsługa formularza i validacji
theme.js Skrypt zmiany motywu kolorystycznego
todo.js Obsługa interaktywnej listy zadań

3.3. Konwencje kodowania i organizacja stylów

W arkuszach stylów zastosowano konwencje zwiększające czytelność i uniwersalność kodu:

- **Zmienne CSS (`:root`)**: Definicje kolorów (tła, tekstu, akcentów) zostały wraz z typografią zamknięte w zmiennych. Pozwala to na błyskawiczną zmianę globalnej palety barw oraz umożliwiło implementację trybu `dark-mode` poprzez nadpisywanie wartości zmiennych w klasie nadrzędnej.
- **Skalowalne jednostki REM**: Bazowa wielkość czcionki została ustawiona na 50% (8px). Dzięki temu projekt jest łatwiej skalowalny, a jednostki REM pozwalają na zachowanie proporcji między elementami.
- **Box-sizing**: Globalne ustawienie `border-box` zapobiega problemom z obliczaniem szerokości elementów przy dodawaniu marginesów wewnętrznych (padding).

3.4. Implementacja logiki w JavaScript

Skrypty projektu zostały zaprojektowane w sposób modułowy. Każda funkcjonalność znajduje się w osobnym pliku, co zapobiega konfliktom nazw i ułatwia ewentualne debugowanie:

- **Manipulacja DOM:** Skrypty dynamicznie tworzą (np. lista zadań) lub modyfikują (np. alerty) elementy drzewa dokumentu.
- **Event Listeners:** Interakcja z użytkownikiem opiera się na nasłuchiwaniu zdarzeń, takich jak click, scroll czy keypress.
- **Powiadomienia:** Zastosowano funkcje czasu (setTimeout), aby kontrolować czas wyświetlania komunikatów błędów i sukcesów w przypadku formularza kontaktowego.

3.5. Responsywność (RWD)

Strona dostosowuje się do urządzeń mobilnych poprzez *Media Queries*. Zastosowano podejście, które zakłada płynne przejścia między szerokościami, tak aby strona wyglądała czytelnie na urządzeniach węższych.

- **Typografia:** Wykorzystano funkcję clamp(), która automatycznie dobiera wielkość czcionki w zależności od szerokości okna przeglądarki.
- **Pasek nawigacji:** Pasek nawigacji na telefonach zmienia swój układ na kolumnowy.
- **Flex-column:** Na mniejszych ekranach kontenery formularza i narzędzi (kalkulator, lista) zmieniają orientację z poziomej na pionową.

3.6. Prezentacja najważniejszych skryptów i ich fragmentów

W niniejszej sekcji przedstawiono implementację oraz krótką analizę najważniejszych skryptów **JavaScript**. Każdy fragment kodu został opatrzony krótkim komentarzem technicznym wyjaśniającym zasadę jego działania.

3.6.1. Zarządzanie motywem (theme.js)

Skrypt ten odpowiada za interakcję z użytkownikiem w zakresie wyboru preferowanego motywu wizualnego (Dark Mode / Light Mode).

Listing 3.1. Przełącznik motywów Dark Mode / Light Mode

```

1 var themeButton = document.getElementById('theme-button');
2 themeButton.onclick = function() {
3   var pageBody = document.body;
4   pageBody.classList.toggle('dark-mode');
5 }

```

Wyjaśnienie: Funkcja przypisana do zdarzenia onclick przycisku wywołuje metodę toggle() na liście klas dokumentu. Mechanizm automatycznie dodaje klasę .dark-mode przy jej braku lub usuwa ją, gdy jest już ona obecna. Powoduje to natychmiastową aktualizację wartości zmiennych CSS.

3.6.2. Animacja powitalna sekcji Hero (animation.js)

Skrypt odpowiedzialny za pojawienie się tytułu strony wykorzystuje funkcję czasu do płynnej zmiany właściwości CSS.

Listing 3.2. Implementacja animacji "fade-in-up"

```

1 var title = document.querySelector('.hero-content h1');
2 var position = 50;
3 var opacity = 0;
4 title.style.marginTop = position + "px";
5 title.style.opacity = opacity;
6 var timer = setInterval(function() {
7   position = position - 1;
8   opacity = opacity + 0.02;
9   title.style.marginTop = position + "px";
10  title.style.opacity = opacity;
11
12 if (position <= 0) {
13   clearInterval(timer);
14   title.style.marginTop = "0px";
15   title.style.opacity = "1";
16 }
17 }, 15);

```

Wyjaśnienie: Skrypt ręcznie steruje animacją przy użyciu metody `setInterval`, która wykonuje blok co 15 milisekund. W każdym kroku wartość zmiennej `position` maleje, powodując ruch tekstu w górę, zaś `opacity` rośnie, zwiększając widoczność. Po osiągnięciu docelowej pozycji, interwał jest czyszczony przez `clearInterval`, co zatrzymuje proces.

3.6.3. Pasek nawigacji (animation.js)

Skrypt ten zapewnia wizualne oddzielenie paska nawigacji od treści strony podczas jej przewijania.

Listing 3.3. Zmiana stylu nagłówka przy przewijaniu

```

1 var header = document.querySelector('header');
2 window.addEventListener('scroll', () => {
3   if (window.scrollY > 0) {
4     header.classList.add('scrolled');
5   } else {
6     header.classList.remove('scrolled');
7   }
8 });

```

Wyjaśnienie: Skrypt rejestruje słuchacza zdarzeń (`Event Listener`) dla obiektu `window`. Gdy wartość `scrollY` przekracza 0, do nagłówka dodawana jest klasa `.scrolled`, która w CSS odpowiada za wyświetlenie dolnego obramowania `border-bottom`.

3.6.4. Konwerter liczb rzymskich (calc.js)

Skrypt implementuje algorytm przeliczania liczb arabskich na liczby systemu rzymskiego. Z oczywistych względów nie wylistowano obsługi przycisków i deklaracji zmiennych.

Listing 3.4. Logika przeliczania liczb arabskich na rzymskie

```

1  function convert() {
2    var val = parseInt(numberInput.value);
3    if (isNaN(val) || val <= 0 || val > 3999) {
4      out.innerText = "Error";
5      return;
6    }
7
8    var numbers = [1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1];
9    var roman = ["M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V",
10      "IV", "I"];
11    var finalStr = "";
12
13    for (var i = 0; i < numbers.length; i++) {
14      while (val >= numbers[i]) {
15        finalStr = finalStr + roman[i];
16        val = val - numbers[i];
17      }
18    }
19    out.innerText = finalStr;
}

```

Wyjaśnienie: Algorytm sprawdza poprawność danych wejściowych (Zakres 1 - 3999), a następnie wykonuje pętle `for` do iteracji po tablicy zdefiniowanych progów liczbowych. Wewnętrzna pętla `while` odejmuje najwyższą możliwą wartość od liczby wejściowej i dopisuje odpowiadający jej znak rzymski do wyniku.

3.6.5. Interaktywna lista zadań (todo.js)

Skrypt demonstruje działanie listy zadań.

Listing 3.5. Dynamiczne dodawanie i usuwanie zadań

```

1  function addNewTask() {
2    var text = inputField.value;
3    if (text != "") {
4      var newItem = document.createElement('li');
5      newItem.className = 'todo-item';
6      newItem.innerHTML = "<span>" + text + "</span>";
7
8      newItem.onclick = function() {
9        this.remove();
10      };
11
12      myList.appendChild(newItem);
13      inputField.value = "";
14    }
15  }

```

Wyjaśnienie: Po aktywacji funkcji, skrypt tworzy w pamięci nowy element `li`, nadaje mu klasę stylu i wstrzykuje tekst pobrany z inputa. Każdemu elementowi przypisywana jest również funkcja `remove()`, wyzwalana po kliknięciu w dodany wcześniej element, co umożliwia selektywne usuwanie z listy.

3.6.6. Obsługa formularza i system alertów (form.js)

Skrypt odpowiada za weryfikację danych w formularzu oraz za wysyłanie powiadomień w przypadku błędnych danych lub poprawnie przesłanego formularza.

Listing 3.6. Walidacja w formularzu i generowanie powiadomień

```

1  function myAlert(msg, mode) {
2      var old = document.querySelector('.custom-alert');
3      if (old) { old.remove(); }
4
5      var box = document.createElement('div');
6      box.className = 'custom-alert' + mode;
7      box.innerHTML = "<span>" + (mode === 'success' ? 'EMOTKA1' : 'EMOTKA2') + "</span> " + msg;
8      document.body.appendChild(box);
9
10     setTimeout(function() { box.classList.add('show'); }, 50);
11     setTimeout(function() { box.classList.remove('show'); }, 3000);
12 }
13
14 var contactForm = document.getElementById('contact-form');
15
16 if (contactForm) {
17     contactForm.onsubmit = function(e) {
18         e.preventDefault();
19         var userEmail = document.getElementById('email').value;
20
21         if (userEmail.indexOf('@') == -1 || userEmail.length < 5) {
22             myAlert("Adres email jest nieprawidłowy!", "error");
23             return;
24         }
25         myAlert("Wiadomość została przesłana.", "success");
26         contactForm.reset();
27     };
28 }
```

Wyjaśnienie: Funkcja myAlert dynamicznie buduje okno powiadomienia w drzewie DOM, wykorzystując funkcję setTimeout do sterowania animacją pojawiania się i znikania klasy stylu. Skrypt do walidacji danych formularza blokuje domyślną wysyłkę formularza przy użyciu e.preventDefault() i prowadząc test poprawności adresu e-mail, imienia, wiadomości wywołując alert z odpowiednim statusem sukcesu albo błędu.

4. Harmonogram realizacji projektu

4.1. Etapy realizacji projektu

Proces tworzenia witryny był ciągiem mniejszych bądź większych etapów, takich jak:

1. Analiza wymagań i ułożenie wstępного schematu.
2. Wstępne stworzenie układu strony.
3. Implementacja rozwiązania przy użyciu technologii HTML5.
4. Wprowadzenie kaskadowych arkuszy stylów.
5. Dodanie skryptów JavaScript.
6. Drobne zmiany wizualne, takie jak dodanie drobnego skryptu dla paska nawigacyjnego.
7. Stworzenie dokumentacji technicznej.

4.2. System kontroli wersji

- Podczas tworzenia projektu wykorzystano system kontroli wersji **Git** poprzez aplikację **GitHub Desktop**. Do przechowywania kodu źródłowego wykorzystano repozytorium na platformie **GitHub**. Repozytorium dostępne jest pod adresem:

<https://github.com/Avsh11/web-technologies/tree/main/ti-project-2>

- Repozytorium zawiera pełną wersję projektu po ostatecznych poprawkach (w chwili oddania) i dośćpnie będzie w trybie publicznym ustalonym przez wykładowcę prowadzącego przedmiot.

5. Prezentacja warstwy użytkowej projektu

Rozdział poświęcony jest wizualnej prezentacji zrealizowanego projektu. Zanim przejdziemy do omówienia i prezentacji samych widoków strony, warto pochylić się nad ogólnymi założeniami, które definiowały ostateczny kształt samej strony.

5.1. Identyfikacja wizualna i stylistyka strony

Projekt strony utrzymany został w minimalistycznej estetyce, gdzie sama strona jest witryną typu *single page*, co oznacza, że nie zawiera ona podstron.

5.2. Kolorystyka

W projekcie wykorzystano minimalistyczną paletę barw, gdzie kolory pochodzą ze strony <https://colors.co/>. Całość zapewnia dobrą czytelność niezależnie od warunków oświetleniowych. Dzięki zastosowaniu pseudoklasy :root w pliku style.css, kolory mogą być szybko zmienione.

- **Tryb jasny (Light Mode):** Dominujący kolor - biel (#F8F9FA) jako tło, jak i kolor paska nawigacyjnego, głęboka czerń (#0B090A) dla tekstów i elementów interaktywnych oraz odcień szarości (#DEE2E6), pełniąca rolę separatorów, będący również kolorem stopki.
- **Tryb ciemny (Dark Mode):** Odwrócona paleta barw, gdzie tło stanowi czarny, a teksty i ikony przybierają odcień bielszy, co redukuje ostatecznie zmęczenie wzroku przy słabym oświetleniu.

5.3. Typografia

W projekcie zastosowano kilka krojów z biblioteki *Google Fonts*, które budują tożsamość wizualną strony:

- **Cinzel:** Krój dekoracyjny inspirowany antyczną rzymską epigrafiką. Został wykorzystany dla nagłówków sekcji oraz wyników kalkulatora. Całość nadaje elegancki, historyczny charakter.
- **Montserrat:** Nowoczesny, bezszeryfowy krój pisma o wysokiej czytelności. Wykorzystany jako główny font treściowy (body), w listach zadań oraz formularzach.

5.4. Prezentacja witryny

Prezentacja wyglądu strony, wraz z przykładowym działaniem funkcjonalności, przedstawiona będzie w poniższych podsekcjach.

5.4.1. Pasek nawigacji

Wygląd paska nawigacji przedstawiono na zdjęciu poniżej - **rys. 5.1**



Rys. 5.1. Panel dodawania nowego użytkownika

5.4.2. Sekcja Hero

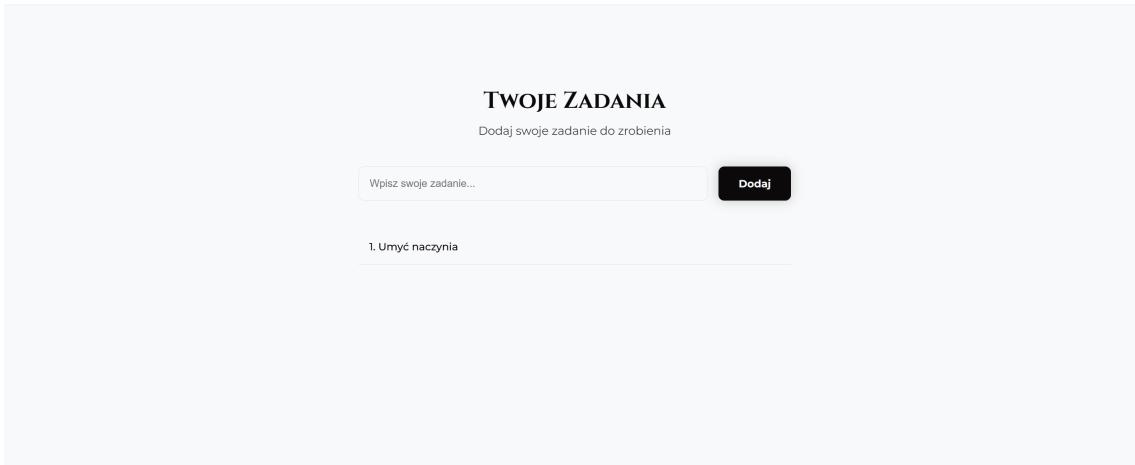
Wygląd sekcji Hero z nagłówkiem dekoracyjnym przedstawiono na zdjęciu poniżej - **rys. 5.2**



Rys. 5.2. Główna sekcja powitalna witryny (Hero)

5.4.3. Lista To-Do

Wygląd listy To-Do przedstawiono na zdjęciu poniżej - **rys. 5.3**



Rys. 5.3. Lista To-Do ze stworzonym zadaniem

5.4.4. Kalkulator rzymski

Kalkulator do konwersji liczb arabskich na liczby rzymskie przedstawiono na zdjęciu poniżej - **rys. 5.4**



Rys. 5.4. Kalkulator liczb rzymskich - 2026

5.4.5. Formularz kontaktowy

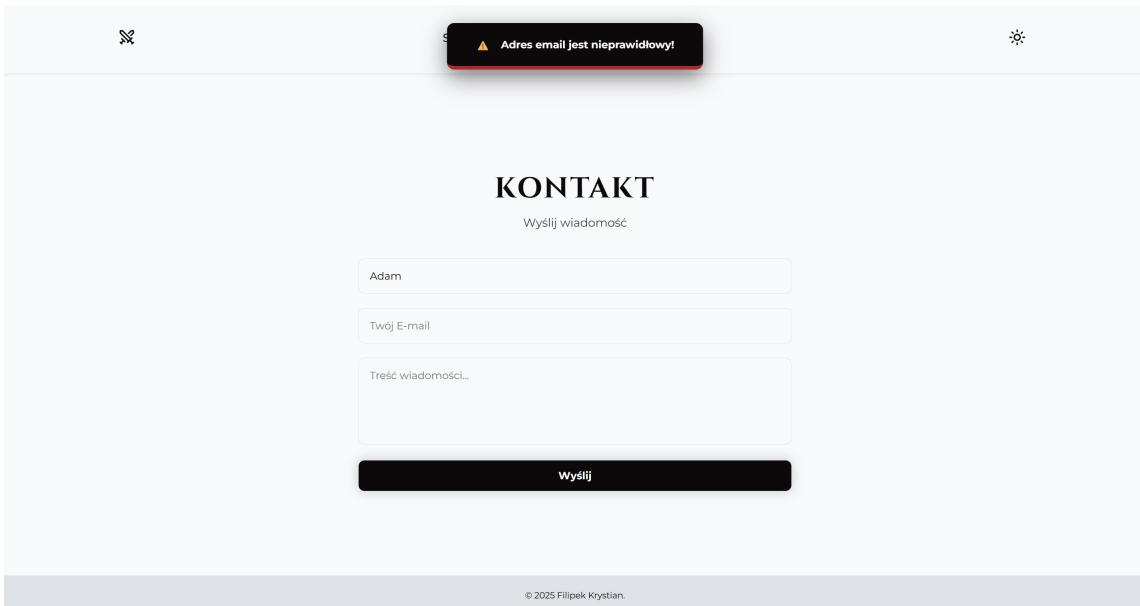
Formularz kontaktowy przedstawiony jest na zdjęciu poniżej - **rys. 5.5**

A screenshot of a contact form titled "KONTAKT". The page has a light gray background. At the top center, the title "KONTAKT" is displayed in bold capital letters. Below it, a smaller line of text reads "Wyślij wiadomość". There are three input fields: "Twoje Imię", "Twój E-mail", and "Treść wiadomości...". Each input field has a placeholder text inside. Below these fields is a large black "Wyślij" button. At the bottom of the form, there is a small gray footer bar with the copyright text "© 2025 Filipek Krystian.".

Rys. 5.5. Formularz kontaktowy

5.4.6. Alert - Błąd

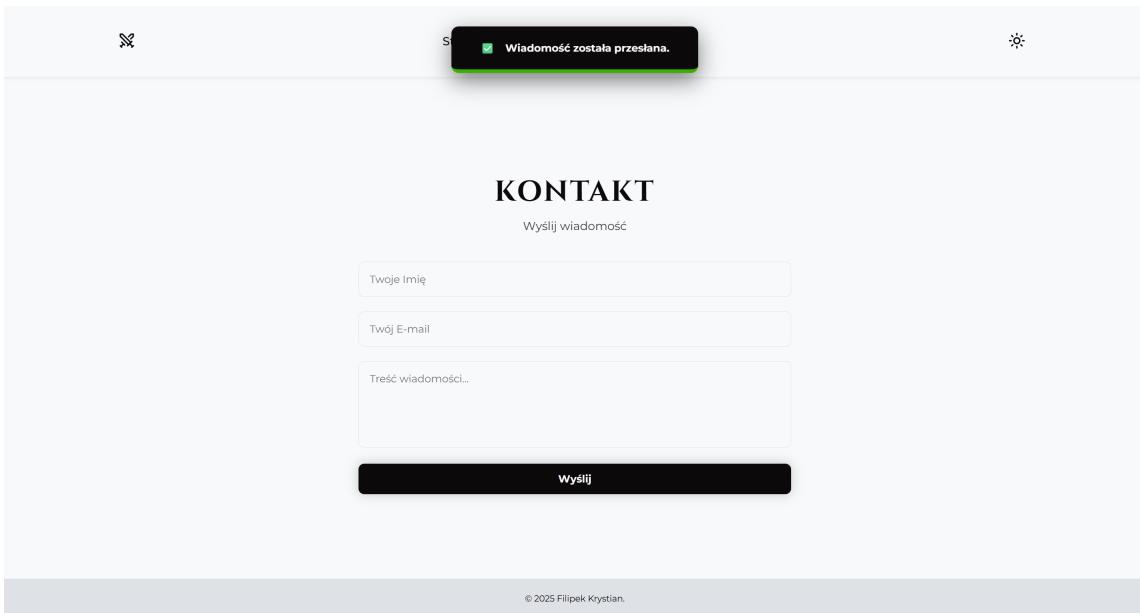
Powiadomienie (górną część strony) informujące o błędzie w danych formularza przedstawione jest na rysunku poniżej - **rys. 5.6**



Rys. 5.6. Powiadomienie o błędzie

5.4.7. Alert - Sukces

Powiadamienie (górna część strony) informujące o sukcesie przesłania formularza przedstawione jest na rysunku poniżej - **rys. 5.7**



Rys. 5.7. Powiadomienie o powodzeniu przesłania formularza

5.4.8. Tryb ciemny

Strona internetowa w ciemnym motywie przedstawiona jest na zdjęciu poniżej - **rys. 5.8**



Rys. 5.8. Tryb ciemny strony internetowej

6. Instrukcja uruchomieniowa

Aby poprawnie uruchomić witrynę w celach testowych, należy wykonać następujące kroki uprzednio przygotowując środowisko uruchomieniowe:

Wymagane środowisko

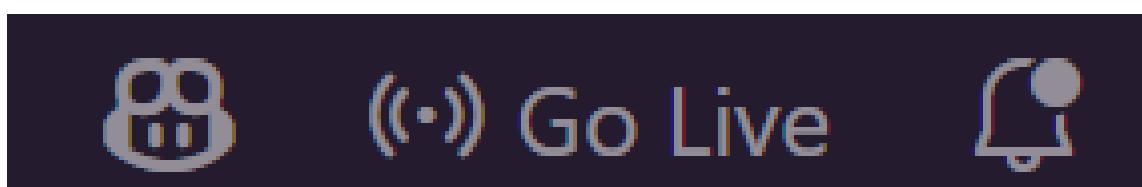
- **Proponowany system operacyjny:** Microsoft Windows / Linux / Apple MacOS
- **IDE:** Visual Studio Code
- **Rozszerzenia:** Live Server autorstwa: **Ritwicka Deya**
- **Repozytorium GitHub:** <https://github.com/Avsh11/web-technologies>
- **Proponowany system kontroli wersji:** GitHub Desktop App

Krok 1. - Konfiguracja środowiska testowego

1. Pobierz edytor Visual Studio Code.
2. Pobierz aplikację GitHub Desktop.
3. Uruchom aplikację GitHub Desktop i wybierz opcję sklonowania repozytorium w zakładce **File**.
4. Sklonowane repozytorium umieszczone ma zostać w wybranym, stworzonym przez użytkownika katalogu.

Krok 2. - Uruchomienie strony

1. Otwórz uprzednio skonfigurowaną aplikację Visual Studio Code.
2. Upewnij się, że poprawnie zainstalowałeś rozszerzenie Live Server.
3. Otwórz katalog, w którym sklonowałeś repozytorium.
4. Wejdź w plik `index.html`.
5. W prawym dolnym rogu **IDE** kliknij w przycisk `Go Live` jak na obrazku poniżej - **rys. 6.1**



Rys. 6.1. Przycisk Go Live rozszerzenia Live Server

7. Podsumowanie

Celem projektu było stworzenie responsywnej i wizualnie estetycznej witryny zawierającej skrypty zapewniające funkcjonalności dla strony. Umożliwić ma ona użytkownikowi łatwą nawigację przez zrozumiale rozmieszczone treści. Dzięki zastosowaniu kaskadowych arkuszy stylów i języka znaczników HTML5 jak i również języka JavaScript udało się stworzyć witrynę spełniającą powyższe kryteria.

Mimo pomyślnego zrealizowania większości najważniejszych funkcjonalności wedle etapów rozwoju witryna wciąż posiada miejsce na wiele innych równie ważnych funkcjonalności, o które można w przyszłości rozbudować ową witrynę. Oto niektóre z nich:

1. Umożliwić działanie formularza poprzez skrypty oraz implementacje języka PHP wraz z wybraną bazą danych.
2. Bardziej zaawansowane animacje na stronie.

Bibliografia

- [1] ChatGPT, 2025. Wykorzystano do wygenerowania dummy text. <https://openai.com/chatgpt>.
- [2] Coolors, 2025. <https://coolors.co/>.
- [3] Freepik, 2025. <https://www.freepik.com/>.
- [4] Google Fonts, 2025. <https://fonts.google.com/>.
- [5] Pexels, 2025. <https://www.pexels.com/>.
- [6] Stack Overflow, 2025. <https://stackoverflow.com/>.
- [7] Unsplash, 2025. <https://unsplash.com/>.
- [8] W3schools, 2025. <https://www.w3schools.com/>.

Spis rysunków

5.1	Panel dodawania nowego użytkownika	16
5.2	Główna sekcja powitalna witryny (Hero)	17
5.3	Lista To-Do ze stworzonym zadaniem	17
5.4	Kalkulator liczb rzymskich - 2026	18
5.5	Formularz kontaktowy	18
5.6	Powiadomienie o błędzie	19
5.7	Powiadomienie o powodzeniu przesłania formularza	19
5.8	Tryb ciemny strony internetowej	20
6.1	Przycisk Go Live rozszerzenia Live Server	21