Sprawozdanie z laboratorium 1

Mikołaj Kubś 272662

14 kwietnia 2025

1 Cel zadania

Celem zadania było zapoznanie się z procesem tworzenia prostych stron Single Page Application (SPA), a także integracją z Azure Static Web i Github Pages. Dodatkowo wykorzystano technikę leniwego ładowania i mechanizm reCAPTCHA.

1.1 Wprowadzenie

Single Page Application (SPA) to rodzaj aplikacji internetowej, w której nawigacja odbywa się poprzez asynchroniczne ładowanie poszczególnych elementów strony, takich jak sekcje lub całe widoki, bez konieczności przeładowywania całej strony. Dzięki temu użytkownik doświadcza płynniejszej interakcji, ponieważ zmiany w interfejsie są natychmiastowe i nie wymagają pełnego odświeżenia przeglądarki.

W aplikacjach SPA cała zawartość jest zazwyczaj ładowana jednorazowo przy pierwszym wejściu na stronę, a późniejsze interakcje z użytkownikiem prowadzą do dynamicznej aktualizacji wyświetlanych danych za pomocą JavaScript. To podejście pozwala na szybsze działanie aplikacji oraz lepsze wykorzystanie zasobów sieciowych, ponieważ jedynie zmieniane elementy są przesyłane między serwerem a klientem.

1.2 Tworzenie aplikacji SPA z wykorzystaniem HTML i JS

1.2.1 Kod HTML

Plik `index.html` definiuje prosty kod HTML aplikacji:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>SPA PIAC TEST</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <header class="header">
      <ul class="Header-links-ul">
        <li class="header-link" id="about-link">About Me</li>
        <li class="header-link" id="contact-link">Contact</li>
        <button id="theme-toggle">Toggle Theme</button>
      </ul>
    </header>
    <main>
      <h1 class="title">Hello World!</h1>
      <p>
        Lorem Ipsum is simply dummy text of the printing and typesetting
        industry...
      </p>
    </main>
    <script src="js/router.js"></script>
  </body>
</html>
```

Rysunek 1: Kod index.html

Kod wewnątrz <main> jest podmieniany przez `router.js`.

1.2.2 Stylizacja CSS

Plik `style.css` definiuje kaskadowe arkusze stylów:

```css
html,
body {
    font-family: sans-serif;
    text-align: center;
    height: 100%;
    margin: 0;
    padding: 0;
    width: 100%;
}

header {
    display: flex;
    justify-content: space-around;
    align-items: center;
}

.Header-links-ul {
    width: 60%;
    list-style: none;
    display: flex;
    justify-content: space-around;
}

.header-link {
    padding: 0.4rem;
    border-radius: 2px;
    cursor: pointer;
}

.header-link:hover {
    border-bottom: 1px solid white;
}
```

1.2.3 Kod JavaScript

Plik `router.js` zarządza nawigacją w aplikacji SPA:

```
let pageUrls = {
    about: '/index.html?about',
    contact: '/index.html?contact',
    gallery: '/index.html?gallery'
};
function OnStartUp() {
    popStateHandler();
}
OnStartUp();
document.querySelector('#about-link').addEventListener('click', (event) => {
    let stateObj = { page: 'about' };
    document.title = 'About';
    history.pushState(stateObj, "about", "?about");
    RenderAboutPage();
});
document.querySelector('#gallery-link').addEventListener('click', (event) => {
    let stateObj = { page: 'gallery' };
    document.title = 'Gallery';
    history.pushState(stateObj, "gallery", "?gallery");
    RenderGalleryPage();
});
document.querySelector('#contact-link').addEventListener('click', (event) => {
    let stateObj = { page: 'contact' };
    document.title = 'Contact';
    history.pushState(stateObj, "contact", "?contact");
    RenderContactPage();
});
function RenderGalleryPage() {
    document.querySelector('main').innerHTML = `
        <br />
        <br />
        <br />
        <br />
        <br />
        <br />
        <br />
        <br />
        <br />
```

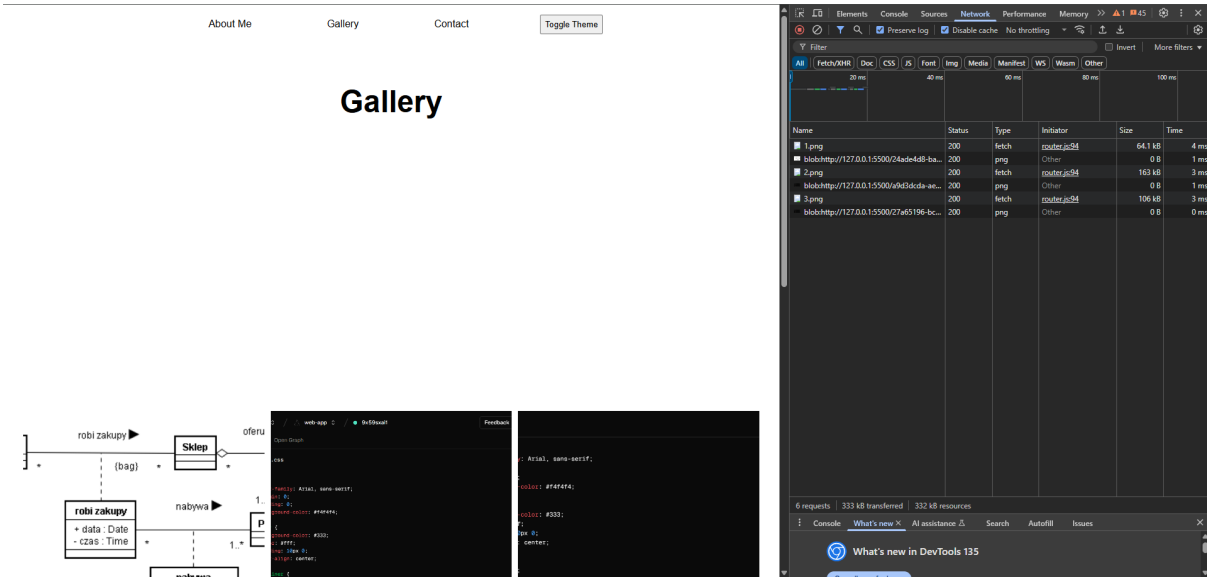
Rysunek 3: Fragment kodu router.js

1.2.4 Leniwe ładowanie

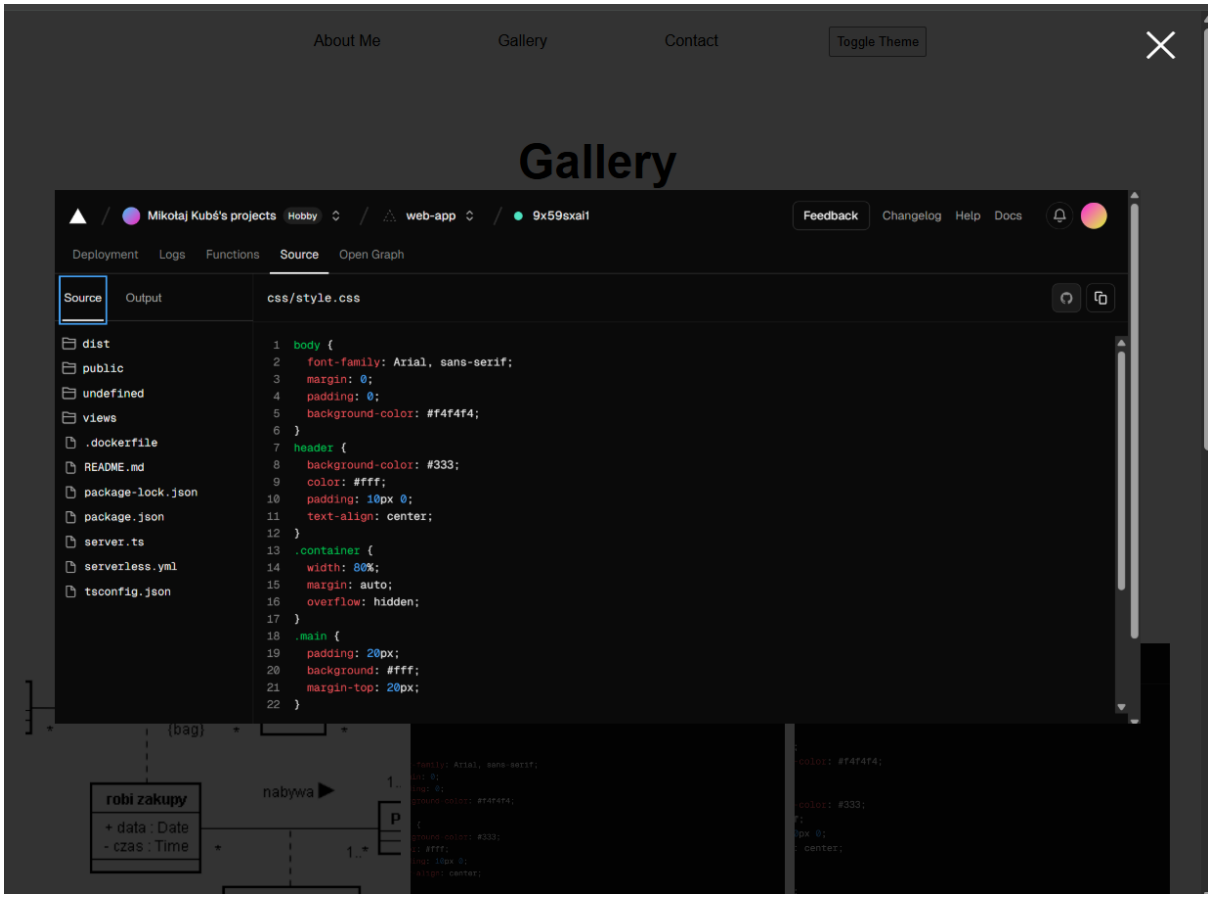
Do implementacji leniwego ładowania obrazków wykorzystano Intersection Observer API. Obrazki są tworzone dynamicznie, a atrybut `src` jest przypisany dopiero, gdy obrazki powinny być widoczne. Obrazki są ładowane jako Blob.

```
function RenderGalleryPage() {

    const observer = new IntersectionObserver(async (entries, obs) => {
        for (const entry of entries) {
            if (entry.isIntersecting) {
                const img = entry.target;
                const imagePath = img.dataset.srcPath;

                obs.unobserve(img);

                try {
                    const response = await fetch(imagePath);

                    if (!response.ok) {
                        throw new Error(`HTTP error! Status: ${response.status} for ${imagePath}`);
                    }

                    const imageBlob = await response.blob();
                    const objectURL = URL.createObjectURL(imageBlob);

                    img.src = objectURL;

                    img.onload = () => {
                        img.classList.add('loaded');
                        img.alt = `Image ${parseInt(img.alt.match(/\d+/)[0])}`;
                        URL.revokeObjectURL(objectURL);
                    };

                    img.onerror = () => {
                        img.alt = `Image ${parseInt(img.alt.match(/\d+/)[0])} (load error)`;
                        URL.revokeObjectURL(objectURL);
                    };

                } catch (error) {
                    img.alt = `Image ${parseInt(img.alt.match(/\d+/)[0])} (fetch error)`;
                }
            }
        }
    }, observerOptions);
```

Rysunek 4: Fragment kodu odpowiedzialnego za leniwe ładowanie



Rysunek 5: Galeria w aplikacji



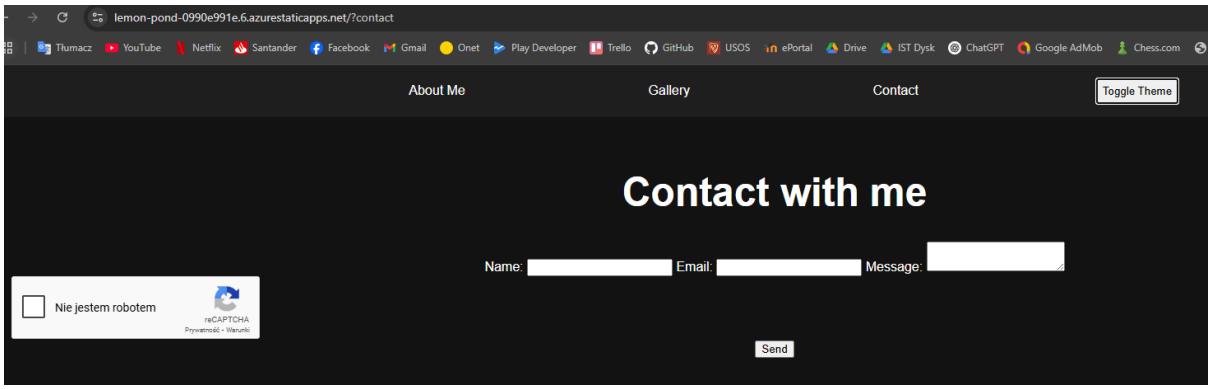
Rysunek 6: Modalne okno obrazka

1.2.5 Implementacja reCAPTCHA

Należało przypisać domenę w systemie reCAPTCHA Google. Grecaptcha renderuje widget Google, a w form jej wysłanie jest przechwycone i sprawdzone, czy grecaptcha nie zwraca błędu.

```
console.log('Contact form loaded, rendering reCAPTCHA...');

const recaptchaContainer = document.getElementById('recaptcha');
if (recaptchaContainer) {
    try {
        grecaptcha.render('recaptcha', {
            'sitekey': '6LeIxAcTAAAAAJcZVRqyHh71UMIEGNQ_MXjiZKhI'
        });
        console.log('reCAPTCHA rendered.');
    } catch (error) {
        console.error("Error rendering reCAPTCHA:", error);
        recaptchaContainer.innerHTML = "reCAPTCHA failed to load.";
    }
} else {
    console.error("reCAPTCHA container not found");
}

const formElement = document.getElementById('contact-form');
const errorAlert = formElement.querySelector('.alert-danger');
const successAlert = formElement.querySelector('.alert-success');

if (formElement) {
    formElement.addEventListener('submit', function (event) {
        event.preventDefault();

        console.log('Form submit event triggered');
        errorAlert.style.display = 'none';
        successAlert.style.display = 'none';

        const nameInput = formElement.querySelector('#name');
        const emailInput = formElement.querySelector('#email');
        const messageInput = formElement.querySelector('#message');

        if (!nameInput.value.trim() || !emailInput.value.trim() || !messageInput.value.trim()) {
            console.log('Standard field validation failed');
            errorAlert.querySelector('p').textContent = 'Failed! Please fill in all required fields (Name, Email, Message).';
```

Rysunek 7: Fragment kodu odpowiedzialnego za mechanizm reCAPTCHA



Rysunek 8: Strona kontakt w aplikacji

1.3 Wdrożenie aplikacji w środowisku chmurowym

1.3.1 Azure Static Web Apps

Azure Static Web Apps oferuje darmową usługę hostingową dla aplikacji SPA. Kroki wdrożenia:

1. Zaloguj się na `https://portal.azure.com`.
2. Utwórz nową aplikację statyczną, podając szczegóły projektu.
3. Połącz aplikację z repozytorium GitHub.
4. Wdróż aplikację i sprawdź jej działanie.

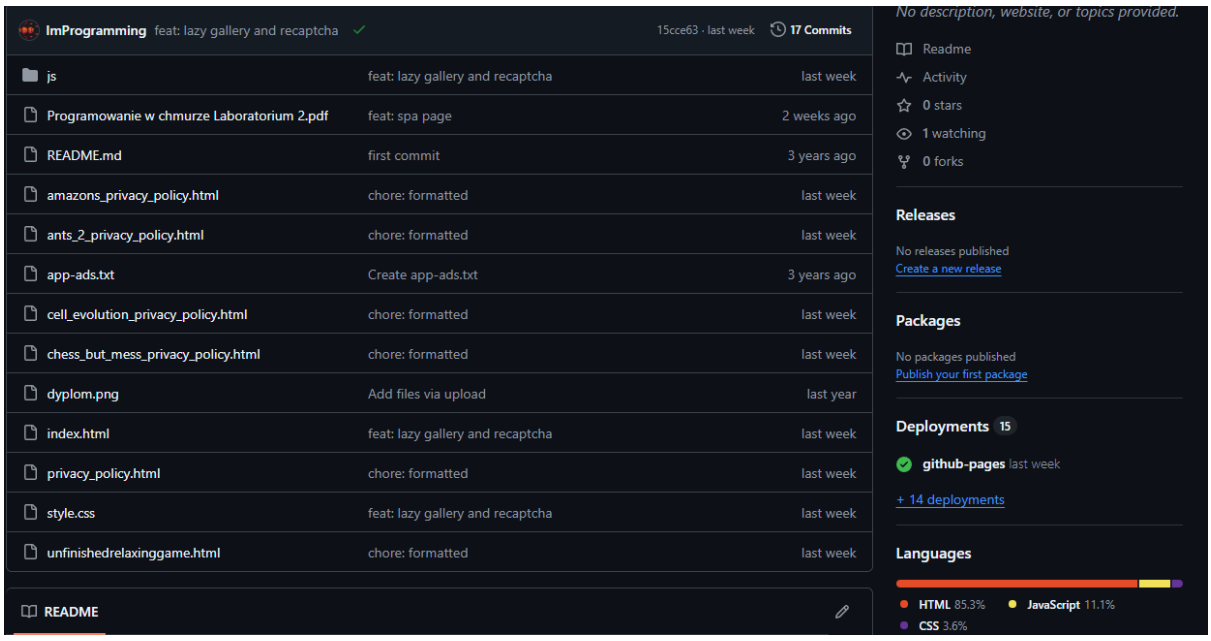
Wdrożenie aplikacji zadziałało bezproblemowo. Po zcommitowaniu kodu na GitHub i poczekaniu, aż Azure to przetworzy, zmieni się hostowana wersja aplikacji.

1.3.2 GitHub Pages

GitHub Pages umożliwia hostowanie aplikacji SPA. Kroki wdrożenia:

1. Utwórz repozytorium na GitHubie.
2. Skonfiguruj GitHub Pages w ustawieniach repozytorium.
3. Sprawdź dostępność aplikacji pod adresem `https://username.github.io/repository`.

Różne pliki były hostowane na `https://github.com/lmProgramming/lmProgramming.github.io` od dawna - głównie informacji dla botów reklamowych Google'a oraz polityki prywatności gier autora. Dodanie nowej struktury strony przebiegło bezproblemowo.



Rysunek 9: Udany deployment na Github pages.

1.4 Testowanie aplikacji

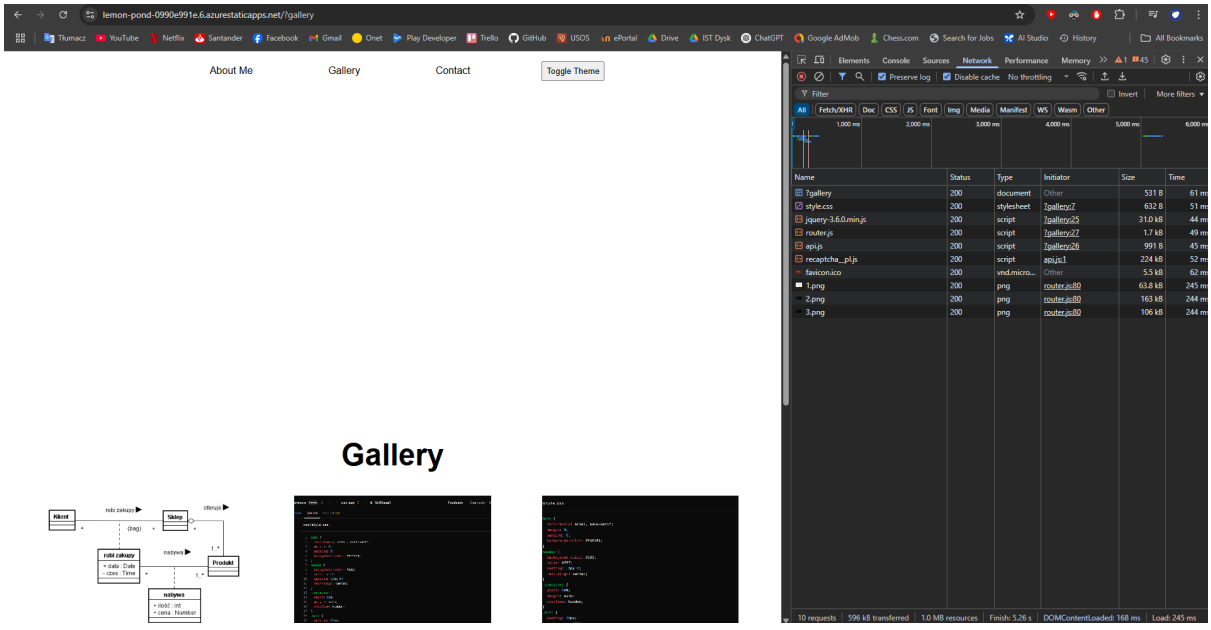
Przeprowadzone testy aplikacji:

- Sprawdź nawigację między stronami.
- Zweryfikuj poprawność ładowania obrazów w galerii.
- Przetestuj walidację formularza kontaktowego.
- Użyj narzędzia Lighthouse w ChromeDevTools do analizy wydajności.

1.4.1 Nawigacja

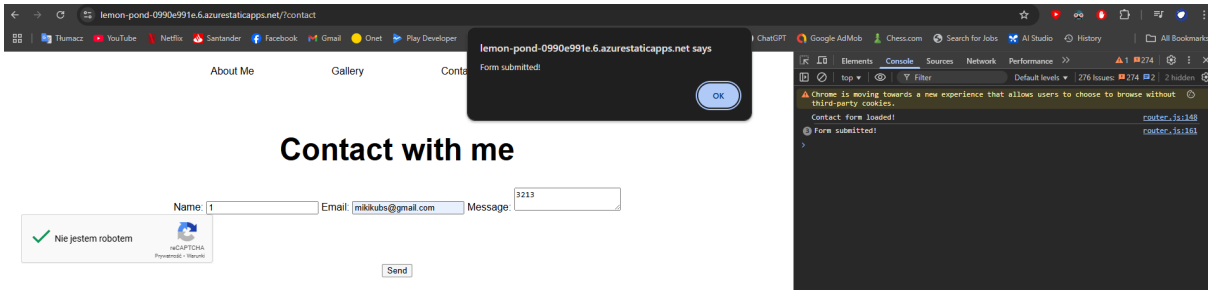
Nawigacja działa bezproblemowo, wszystkie stany są zapisane w historii. Nawigacja jest praktycznie natychmiastowa.

1.4.2 Leniwe ładowanie



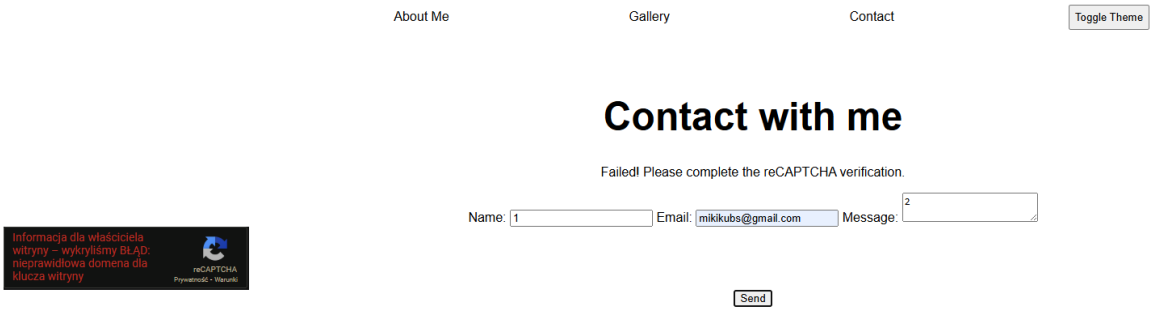
Rysunek 10: Tylko 3 z 9 obrazków załadowana, co oznacza, że leniwe ładowanie działa.

1.4.3 ReCAPTCHA



Rysunek 11: Sukces wysłania formularza z reCAPTCHA.

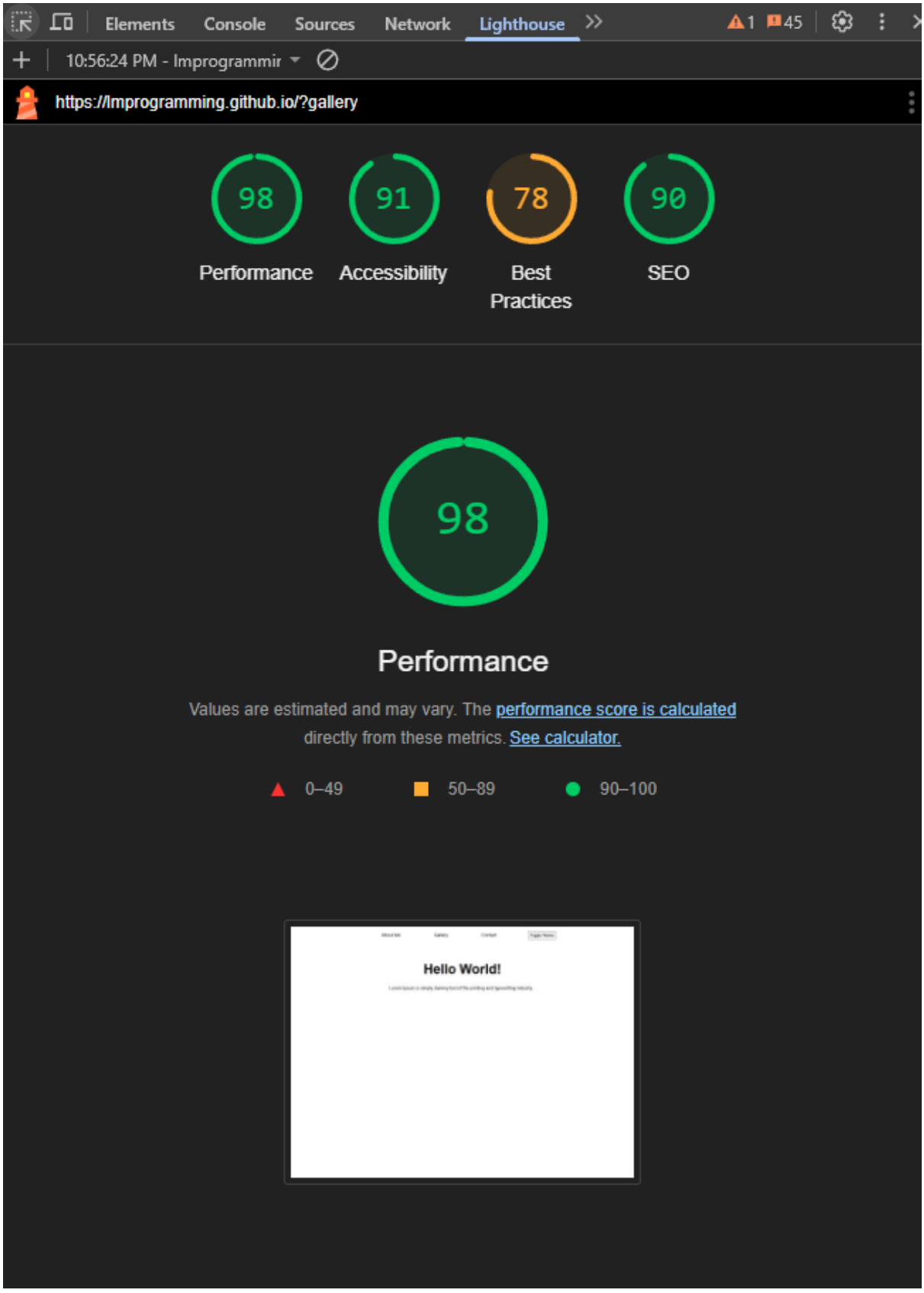
Przez test aplikacji na innej domenie, mechanizm reCAPTCHA nie może działać - wysłanie formularza nie uda się.



Rysunek 12: Nieudane wysłanie formularza z reCAPTCHA.

1.4.4 Lighthouse

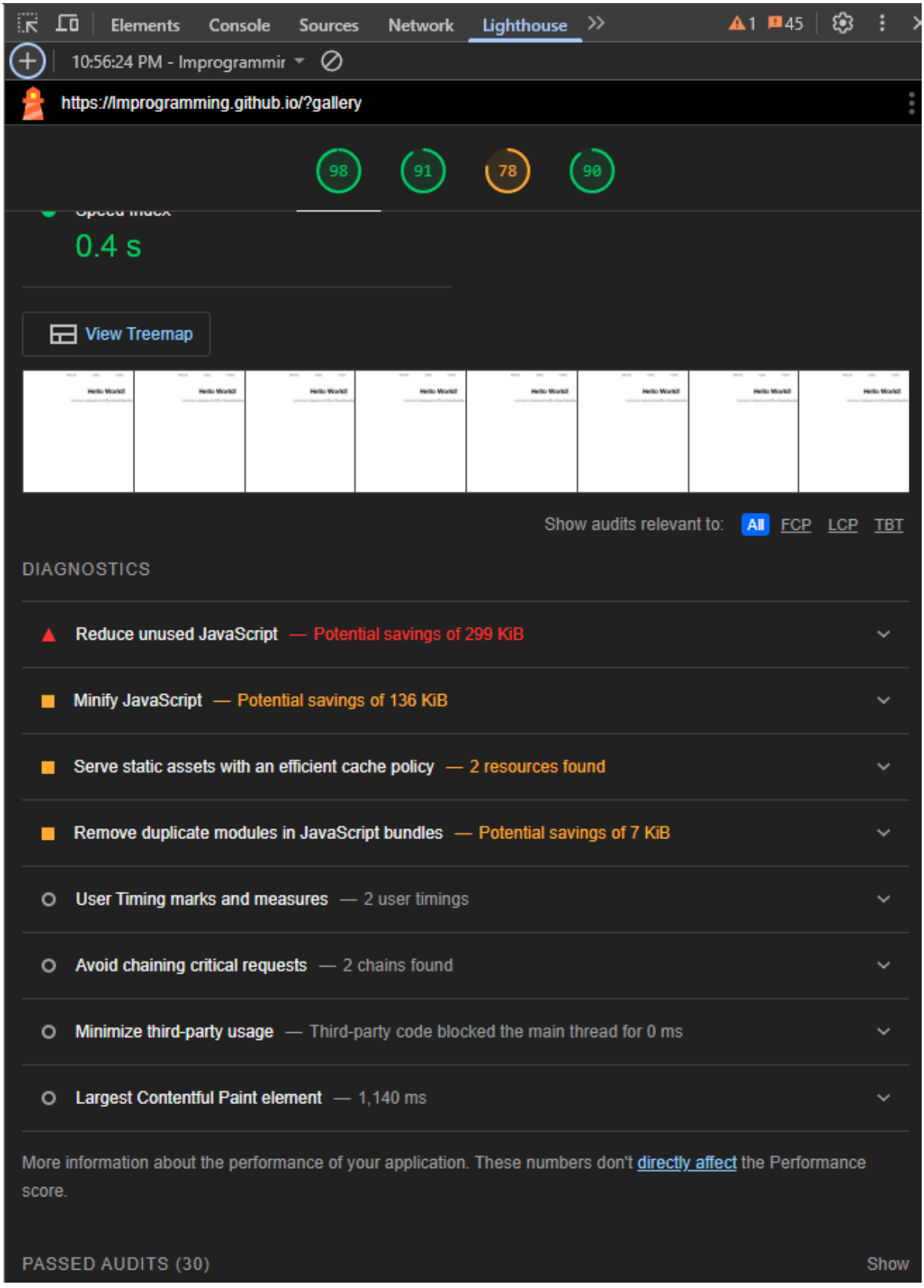
Lighthouse pomaga mierzyć i diagnozować problemy z wydajnością ładowania, interaktywnością (TBT), dostępnością (ważne dla wszystkich użytkowników), SEO (kluczowe dla SPA) i ogólnymi dobrymi praktykami webowymi.



Rysunek 13: Wynik analizy Lighthouse.

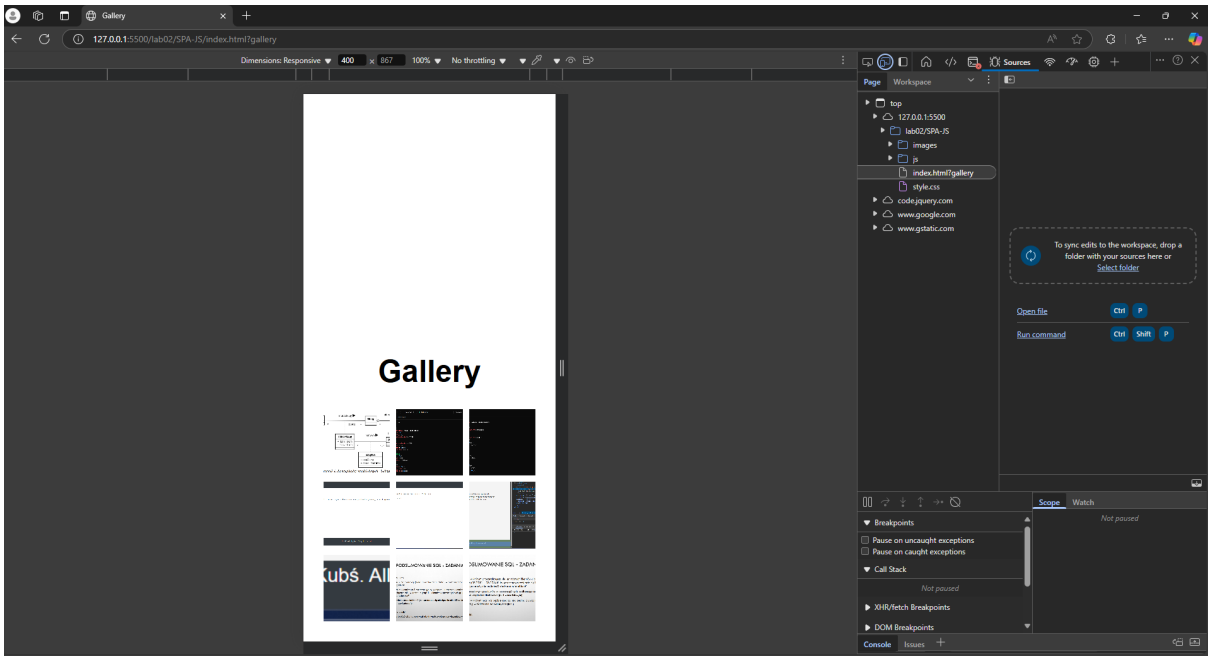
Wynik analizy performance to aż 98/100, co oznacza wynik bardzo dobry, nie ma

potrzeby go polepszać.



Rysunek 14: Szczegóły analizy Lighthouse.

1.4.5 Inna przeglądarka i rozdzielczość



Rysunek 15: Poprawne wyświetlanie strony w przeglądarce Edge, symulując wymiary telefonu

2 Wyzwania związane z SPA

- Routing po stronie klienta
- Problemy z zarządzaniem stanu aplikacji
- Optymalizacja pierwszego załadowania strony (skoro wymaga pobrania większej liczby stron na początku)
- Zwłaszcza kiedyś występowały problemy z SEO i SPA, teraz ten problem jest mniejszy dzięki lepszym robotom Google
- Bezpieczeństwo - większa ilość logiki po stronie klienta otwiera nowe możliwości ataku