

Sprawozdanie z laboratorium 3

Mikołaj Kubś 272662

30 kwietnia 2025

1 Cel zadania

Celem zadania jest stworzenie prostej aplikacji PWA z wykorzystaniem HTML, CSS i JS. Kolejnym krokiem był deployment i testy funkcjonalności.

1.1 Wprowadzenie

Progressive Web App to aplikacja internetowa uruchamiana tak jak zwykła strona internetowa, ale umożliwiająca stworzenie wrażenia działania jak natywna aplikacja mobilna (lub aplikacja desktopowa).

1.2 Tworzenie aplikacji PWA z wykorzystaniem HTML, JS i CSS

1.2.1 Kod HTML

Plik `index.html` definiuje prosty kod HTML aplikacji wraz z ikonami dla PWA i innymi informacjami:

```

lab03 > pwa-js > <> index.html > html > head > link
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="utf-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>PWA PIAC TEST</title>
7      <link rel="manifest" href="manifest.json" />
8      <link rel="stylesheet" href="style.css" />
9      <link rel="icon" href="favicon.ico" type="image/x-icon" />
10     <link rel="apple-touch-icon" href="images/icons/pwa-icon-152x152.png" />
11     <meta name="mobile-web-app-capable" content="yes" />
12     <meta name="apple-mobile-web-app-status-bar" content="#db4938" />
13     <meta name="theme-color" content="#db4938" />
14     <meta name="apple-mobile-web-app-title" content="PWA PIAC" />
15     <meta name="mobile-web-app-capable" content="yes" />
16     <meta name="apple-mobile-web-app-status-bar-style" content="default" />
17     <link rel="apple-touch-icon" href="images/icons/pwa-icon-152x152.png" />
18     <link
19       rel="apple-touch-startup-image"
20       href="images/icons/apple-splash-640x1136.png"
21       media="(device-
22 width: 320px) and (device-height: 568px)"
23     />
24     <link
25       rel="apple-touch-startup-image"
26       href="images/icons/apple-splash-750x1334.png"
27       media="(device-
28 width: 375px) and (device-height: 667px)"
29     />
30     <link
31       rel="apple-touch-startup-image"
32       href="images/icons/apple-splash-828x1792.png"
33       media="(device-
34 width: 414px) and (device-height: 896px)"
35     />

```

Rysunek 1: Kod index.html

1.2.2 Stylizacja CSS

Plik `style.css` definiuje kaskadowe arkusze stylów:

lab03 > pwa-js > style.css > html

```
1  html,
2  body {
3      font-family: sans-serif;
4      text-align: center;
5      height: 100%;
6      margin: 0;
7      padding: 0;
8      width: 100%;
9  }
10 .container {
11     margin: auto;
12     text-align: center;
13 }
14 main {
15     padding: 2rem;
16     font-size: 1rem;
17 }
18 .title {
19     font-size: 3rem;
20     animation: fadeIn 2s ease-in-out;
21 }
22 @keyframes fadeIn {
23     from {
24         opacity: 0;
25         transform: translateY(-20px);
26     }
27     to {
28         opacity: 1;
29         transform: translateY(0);
30     }
31 }
32 .dog-image {
33     width: 200px;
34     height: 200px;
```

1.2.3 Kod JavaScript

```
lab03 > pwa-js > js > js main.js > ...
1  import { initializeApp } from 'https://www.gstatic.com/firebasejs/10.6.0/firebase-app.js';
2  import { getMessaging, getToken, onMessage } from 'https://www.gstatic.com/firebasejs/10.6.0/firebase-messaging.js';
3
4  const firebaseConfig = {
5    apiKey: "AIzaSyAWLxcIG0vZSLD78nC2n2qGN0vuzul3rTE",
6    authDomain: "pwa-ts-c7c15.firebaseio.com",
7    projectId: "pwa-ts-c7c15",
8    storageBucket: "pwa-ts-c7c15.firebaseio.com",
9    messagingSenderId: "438732315703",
10   appId: "1:438732315703:web:bcc4ff82c8ce5e4b1f73b2",
11   measurementId: "G-7M93B09YP4"
12 };
13
14 const app = initializeApp(firebaseConfig);
15 const messaging = getMessaging(app);
16
17 const VAPID_KEY = "BDSUjXiUoetn8Qz6mmTlIEFL9Mv6Z-CCmuFofvAk1Uit5mn2TV4EMNHze1ei90PA6maIhR-e563M8XD5ffyiItY";
18 const MESSAGING_SW_PATH = '/pwa-js/firebase-messaging-sw.js';
19 const MESSAGING_SW_SCOPE = '/pwa-js/';
20 const MAIN_SW_PATH = '/pwa-js/sw.js';
21 const MAIN_SW_SCOPE = '/pwa-js/';
22
23 let firebaseMessagingSwReg = null;
24
25 if ("serviceWorker" in navigator) {
26   window.addEventListener("load", () => {
27     navigator.serviceWorker
28       .register(MAIN_SW_PATH, { scope: MAIN_SW_SCOPE })
29       .then(registration => {
30         console.log("Main PWA Service Worker registered:", registration);
31       })
32       .catch(error => {
33         console.error("Main PWA Service Worker registration failed:", error);
34       });
35   });
36 }
```

Rysunek 3: Fragment kodu main.js

```

lab03 > pwa-js > JS sw.js > ...
1  const basePath = '/pwa-js/';
2
3  const cacheName = "piac-pwa-v1";
4  const filesToCache = [
5    basePath,
6    basePath + 'index.html',
7    basePath + 'cats.html',
8    basePath + 'style.css',
9    basePath + 'js/main.js',
10   basePath + 'manifest.json',
11   basePath + 'images/icons/pwa-icon-192x192.png'
12 ];
13
14
15 self.addEventListener("install", event => {
16   console.log('[SW] Install event');
17   event.waitUntil(
18     caches.open(cacheName).then(cache => {
19       console.log('[SW] Caching core assets');
20       return cache.addAll(filesToCache);
21     })
22     .catch(error => {
23       console.error('[SW] Failed to cache core assets:', error);
24     })
25   );
26   self.skipWaiting();
27 });
28
29 self.addEventListener("activate", event => {
30   console.log('[SW] Activate event');
31   const cacheWhitelist = [cacheName];
32   event.waitUntil(
33     caches.keys().then(cacheNames => {
34       return Promise.all(
35         cacheNames.map(cache => {

```

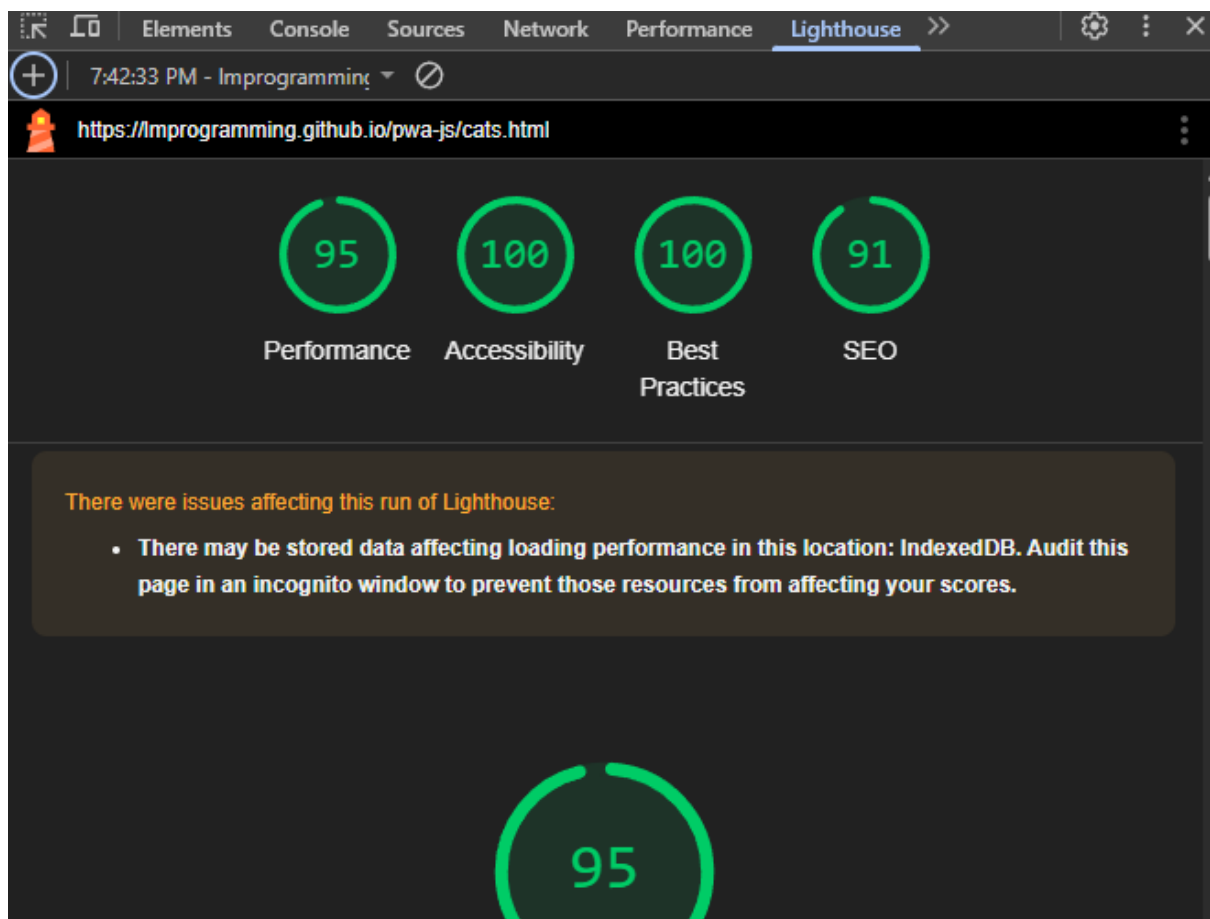
Rysunek 4: Fragment kodu sw.js - service worker

1.3 Dalsza implementacja

Dzięki wykorzystaniu kodu w instrukcji implementacja przebiegła bez większych problemów. Najwięcej problemów wynikło ze ścieżek, gdyż publikując stronę w GitHub Pages okazuje się, że jest głębiej w folderze.

Do ikon wykorzystano oba podane narzędzia: pwa-asset-generator i RealFaviconGenerator.

Rozbudowano stronę, dodając zdjęcia psów i podstronę ze zdjęciami kotów.



Rysunek 5: Wynik analizy Lighthouse

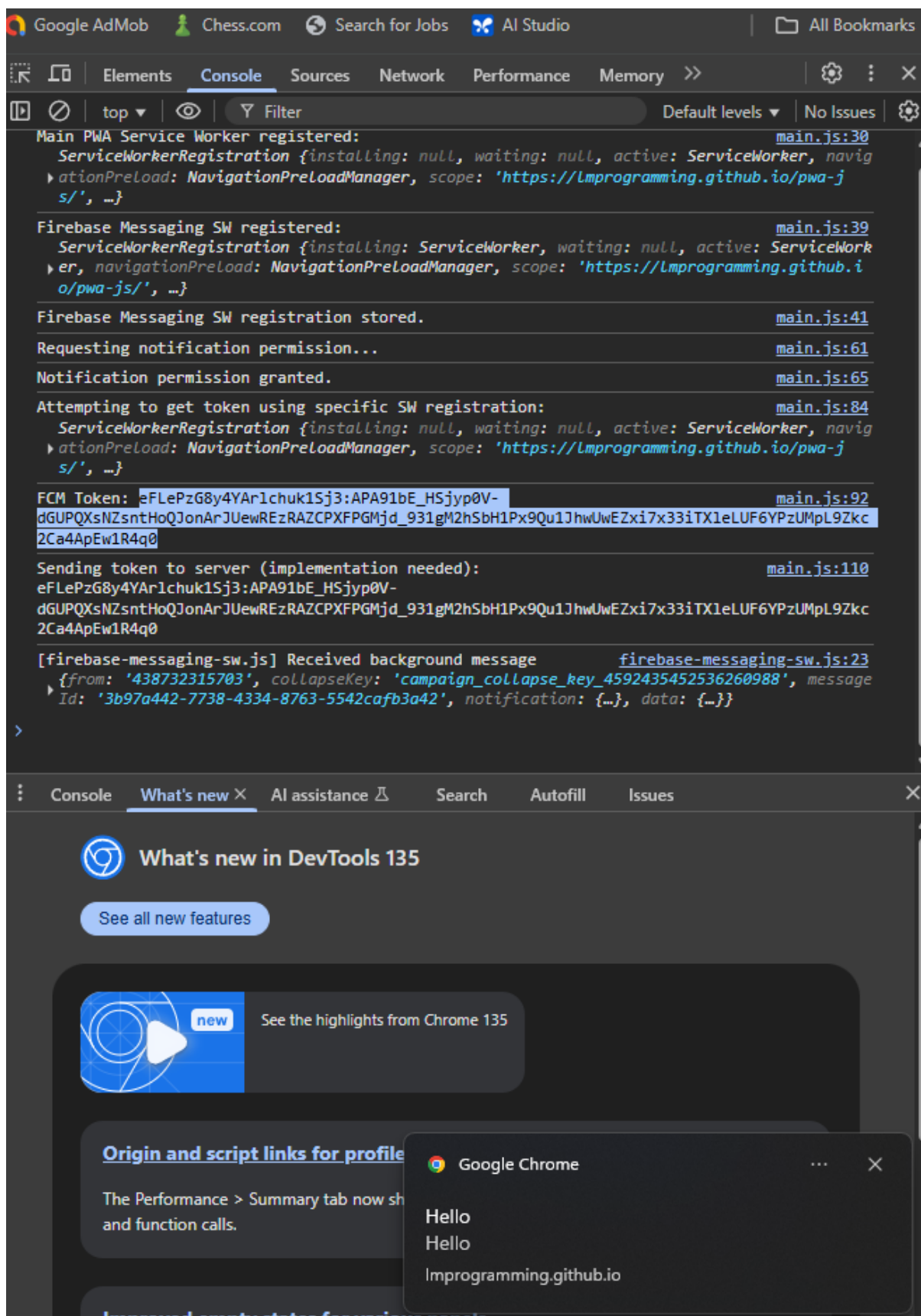


Service worker działa jak należy, cache’ując odpowiednie pliki. Po pobraniu aplikacji na telefon splash image się pokazuje.

Wdrożono aplikację w GitHub pages, co, jak wyżej wspomniano, wymagało naprawy ścieżek.

1.4 Powiadomienia

Sprawiły one najwięcej problemów implementacyjnych. Stworzono projekt w Firebase i skrypt `firebase-messaging-sw.js`. Po poprawie ścieżek w końcu zadziałało i dzięki wydrukowaniu tokenu FCM można było wysłać testową wiadomość z konsoli Firebase.



Rysunek 7: Powiadomienie