ZÁRÓDOLGOZAT

Készítették:

Szabó Judit Anna - Pejkó Bálint - Várdai Balázs

Konzulens:

Farkas Zoltán

Miskolc

2024

Miskolci SZC Kandó Kálmán Informatikai Technikum

Miskolci Szakképzési Centrum

SZOFTVERFEJLESZTŐ- ÉS TESZTELŐ SZAK

Tesztelési dokumentáció

PrintFusion

Szabó Judit Anna - Pejkó Bálint - Várdai Balázs

2024

Bevezetés

Teszterv:

- Unit tesztek a backend alkalmazás tesztelésére
- Lighthouse chrome bővítmény a frontend tesztelésére

Unit teszt

Az **Unittest** egy olyan tesztelési módszer, amely segít ellenőrizni, hogy az alkalmazás részei megfelelnek-e a tervezett specifikációnak. Az Unittestek biztosítják az alkalmazás stabilitását és megbízhatóságát, segítve az új funkciók bevezetését és a hibák gyors felderítését. Az ASP.NET környezetben az Unittestek segítenek megőrizni az alkalmazás összhangját és kompatibilitását a különböző platformokon és eszközökön.

Unit tesztek:

```
private HttpClient client;
private static int commentId=185;
private static Hozzaszolasok dummyPostComment = new Hozzaszolasok
{
    UserId = "ee8b35cc-3cea-4445-b7c2-7b4022a3ae02",
    TermekId = 3,
    Leiras = "Ez a sor tesztelve lesz",
    Ertekeles = 1
};
```

Hozzászólás - GET

```
[Test]
0 references
public async Task HozzaszolasGetTest()
{
    HttpResponseMessage response = await client.GetAsync("/Hozzaszolas");
    Assert.AreEqual(HttpStatusCode.OK, response.StatusCode);
}
```

GET Kérés: A client.GetAsync("Hozzaszolas") kódsor egy aszinkron GET kérést indít a “Hozzaszolás” végpontra.
Válasz Ellenőrzése: A response változóban tárolt válasz státuszkódját ellenőrzi az Assert.AreEqual(HttpStatusCode.OK, response.StatusCode) állítás, hogy az OK-e (200), ami azt jelenti, hogy a végpont elérhető és helyesen működik.

```
[Test]
0 references
public async Task HozzaszolasGetByTermekTest()
{
    int termekId = 8;
    HttpResponseMessage response = await client.GetAsync("/Hozzaszolas/termek/" + termekId);
    Assert.AreEqual(HttpStatusCode.OK, response.StatusCode);
}
```

A HozzaszolasGetByTermekTest egy aszinkron tesztmetódus, amely a következő lépéseket tartalmazza:
Termékazonosító Beállítása: Egy termekId változó értéke 8, amely egy termék azonosítóját jelöli.
GET Kérés: A client.GetAsync("/Hozzaszolas/termek/" + termekId) kódsor egy aszinkron GET kérést indít, amely a termékazonosítóval kapcsolatos hozzászólásokat kéri le.
Válasz Ellenőrzése: Az Assert.AreEqual(HttpStatusCode.OK, response.StatusCode) állítás ellenőrzi, hogy a válasz státuszkódja OK-e (200), ami azt jelenti, hogy a kérés sikeres volt.

Hozzászólás- POST

```
[Test]
0 references
public async Task HozzaszolasPostTest()
{
    var json = JsonSerializer.Serialize(dummyPostComment);
    var data = new StringContent(json, Encoding.UTF8, "application/json");

    HttpResponseMessage response = await client.PostAsync("/Hozzaszolas", data);


    Assert.AreEqual(HttpStatusCode.OK, response.StatusCode, "Failed to post comment");
}
```

A HozzaszolasPostTest egy aszinkron tesztmetódus, amely a következő lépéseket hajtja végre:
Létrehozási Kérés: A client.PostAsync("/Hozzaszolas", data) kódsor egy aszinkron POST kérést indít a “Hozzaszolas” végpontra.
Válasz Ellenőrzése: Az Assert.AreEqual(HttpStatusCode.OK, response.StatusCode, "Failed to post comment") állítás ellenőrzi, hogy a válasz státuszkódja OK-e (200), ami azt jelenti, hogy a hozzászólás sikeresen létrejött.

Hozzászólás-PUT

```
[Test]
0 references
public async Task HozzaszolasPutTest()
{
    Hozzaszolasok dummyComment = new Hozzaszolasok
    {
        TermekId = 8,
        Leiras = "A sor tesztelve lett",
        Ertekeles = 3
    };

    var json = JsonSerializer.Serialize(dummyComment);
    var data = new StringContent(json, Encoding.UTF8, "application/json");

    HttpResponseMessage response = await client.PutAsync($"/Hozzaszolas/{commentId}", data);
    Assert.AreEqual(HttpStatusCode.OK, response.StatusCode, "Expected OK status code but got " + response.StatusCode + " ");
}
```

A HozzaszolasPutTest egy aszinkron tesztmetódus, amely a következő lépéseket hajtja végre: Frissítési Kérés: A client.PutAsync($"hozzaszolas/{commentId}", data) kódsor egy aszinkron PUT kérést indít a “hozzaszolas” végpontra, ahol a {commentId} a frissítendő hozzászólás azonosítóját jelöli.
Válasz Ellenőrzése: Az Assert.AreEqual(HttpStatusCode.OK, response.StatusCode, $"Expected OK status code but got {response.StatusCode}") állítás ellenőrzi, hogy a válasz státuszkódja OK-e (200), ami azt jelenti, hogy a frissítés sikeres volt.

Put teszt előtt:

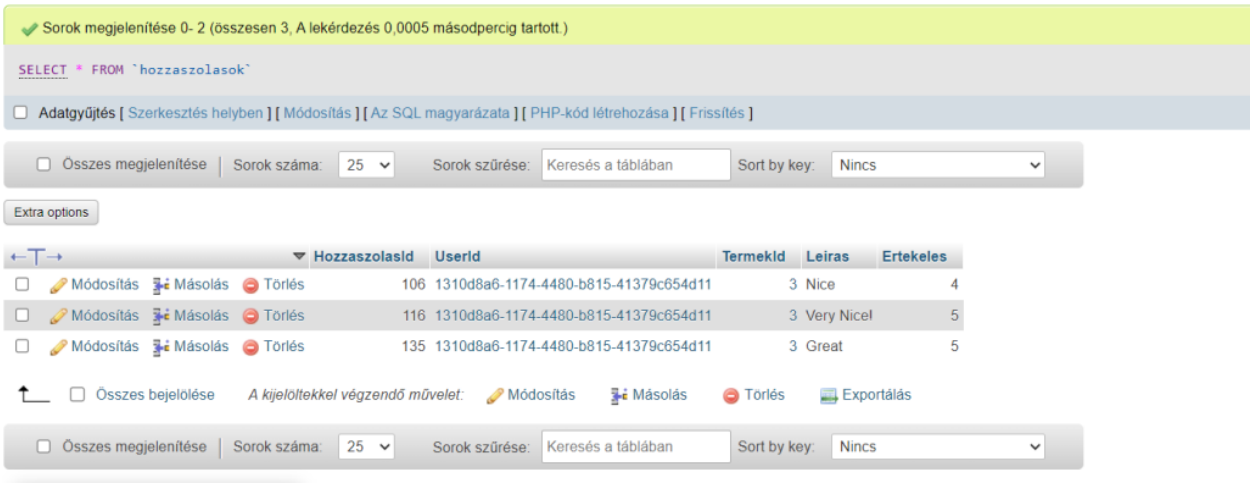
Put teszt után:

Hozzászólás – DELETE

A HozzaszolasDeleteTest egy aszinkron tesztmetódus, amely a következő lépéseket hajtja végre:
Törlési Kérés: A client.DeleteAsync($"Hozzaszolas/{commentId}") kódsor egy aszinkron törlési

```
[Test]
0 references
public async Task HozzaszolasDeleteTest()
{
    HttpResponseMessage response = await client.DeleteAsync($"/Hozzaszolas/{commentId}");
    Assert.AreEqual(HttpStatusCode.OK, response.StatusCode);
}
```

kérést indít a “Hozzaszolas” végpontra, ahol a {commentId} a törlendő hozzászólás azonosítóját jelöli.
Válasz Ellenőrzése: Az Assert.AreEqual(HttpStatusCode.OK, response.StatusCode) állítás ellenőrzi, hogy a válasz státuszkódja OK-e (200), ami azt jelenti, hogy a törlés sikeres volt.



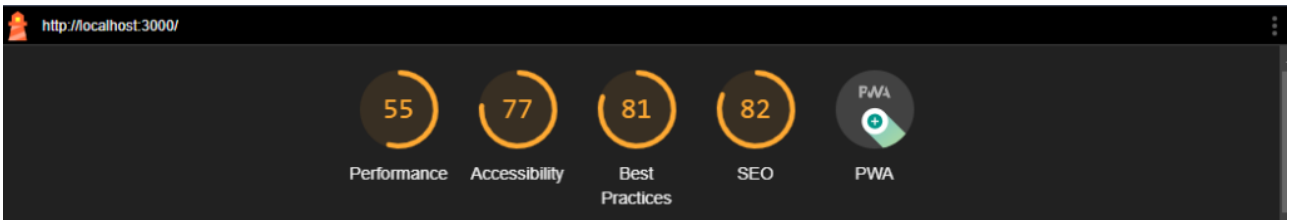
Lighthouse

A **Lighthouse** egy nyílt forráskódú automatizálási eszköz a webhelyek minőségének javítására. Bármilyen weboldalon futtatható, legyen az nyilvános vagy hitelesített. Ellenőrizni tudja a teljesítményt, a hozzáférhetőséget, a progresszív webes alkalmazásokat, a SEO-t és még sok mást.

Main page tesztelése:

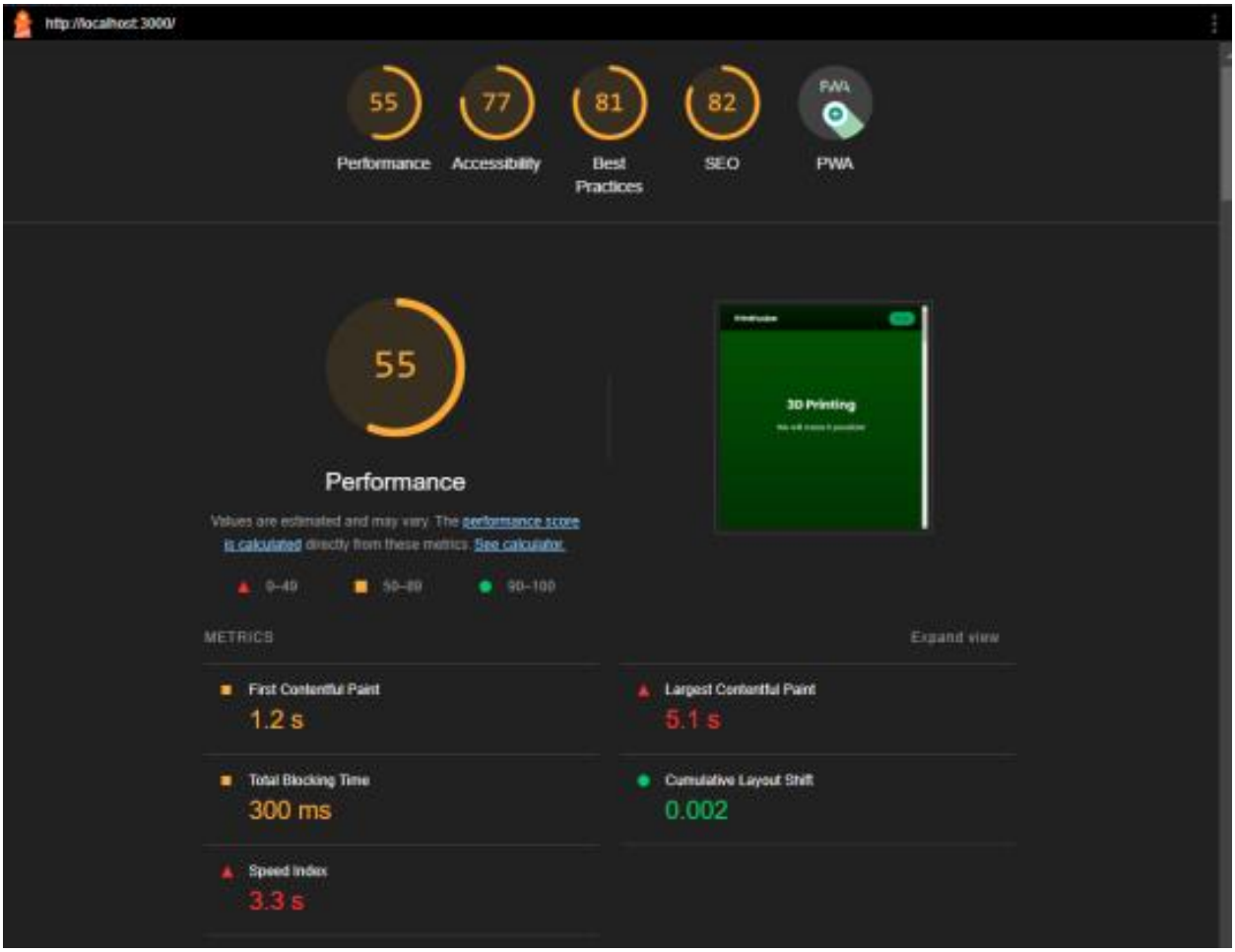
A frontend teljesítményének, hozzáférhetőségének, legjobb gyakorlatoknak, SEO-nak és progresszív webalkalmazás (PWA) jellemzőinek értékelésére a Lighthouse eszközt használtuk. A teszt a következő eredményeket hozta:

- Teljesítmény: 55/100
- Hozzáférhetőség: 77/100
- Legjobb gyakorlatok: 81/100
- SEO: 82/100
- PWA: Sikeres



Ezek az eredmények fontos betekintést nyújtanak a weboldalunk jelenlegi állapotába, és segítenek azonosítani a fejlesztésre szoruló területeket. A teljesítmény pontszám alacsonyabb, mint szeretnénk, ezért további optimalizálásra van szükség a betöltési idők és a futási teljesítmény javítása érdekében. A hozzáférhetőség és a SEO pontszámok viszont viszonylag magasak, ami azt jelzi, hogy a weboldalunk jól elérhető és barátságos a keresőmotorok számára. A PWA teszt sikeres volt, ami azt jelenti, hogy a weboldalunk megfelel a progresszív webalkalmazásokra vonatkozó alapvető követelményeknek. Ez biztosítja, hogy a felhasználók offline módban is élvezhessék a weboldal funkcióit, és jobb felhasználói élményt kapjanak mobil eszközökön.

Teljesítmény:

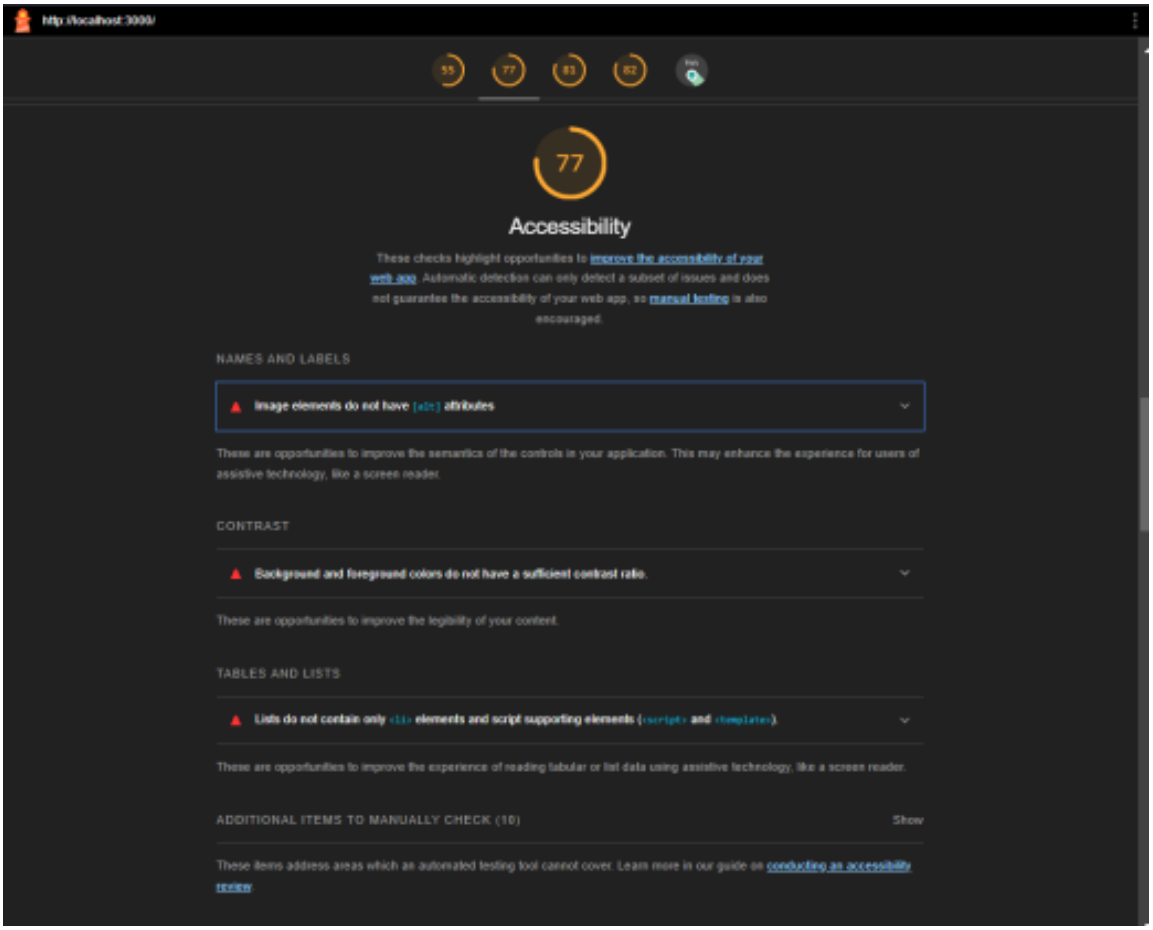


Teljesítmény pontszám: 55/100

- Első tartalmi festés (First Contentful Paint): 1.2 másodperc
- Teljes blokkolási idő (Total Blocking Time): 300 ms
- Sebesség index (Speed Index): 3.3 másodperc
- Legnagyobb tartalmi festés (Largest Contentful Paint): 5.1 másodperc
- Kumulatív elrendezési változás (Cumulative Layout Shift): 0.002

Ezek az eredmények azt mutatják, hogy a weboldalunk teljesítménye javításra szorul, különösen a betöltési idők és a vizuális stabilitás terén. A dokumentációban részletesen ismertetjük azokat a lépéseket, amelyeket a teljesítmény javítása érdekében tervezünk megtenni.

Hozzáférhetőség:

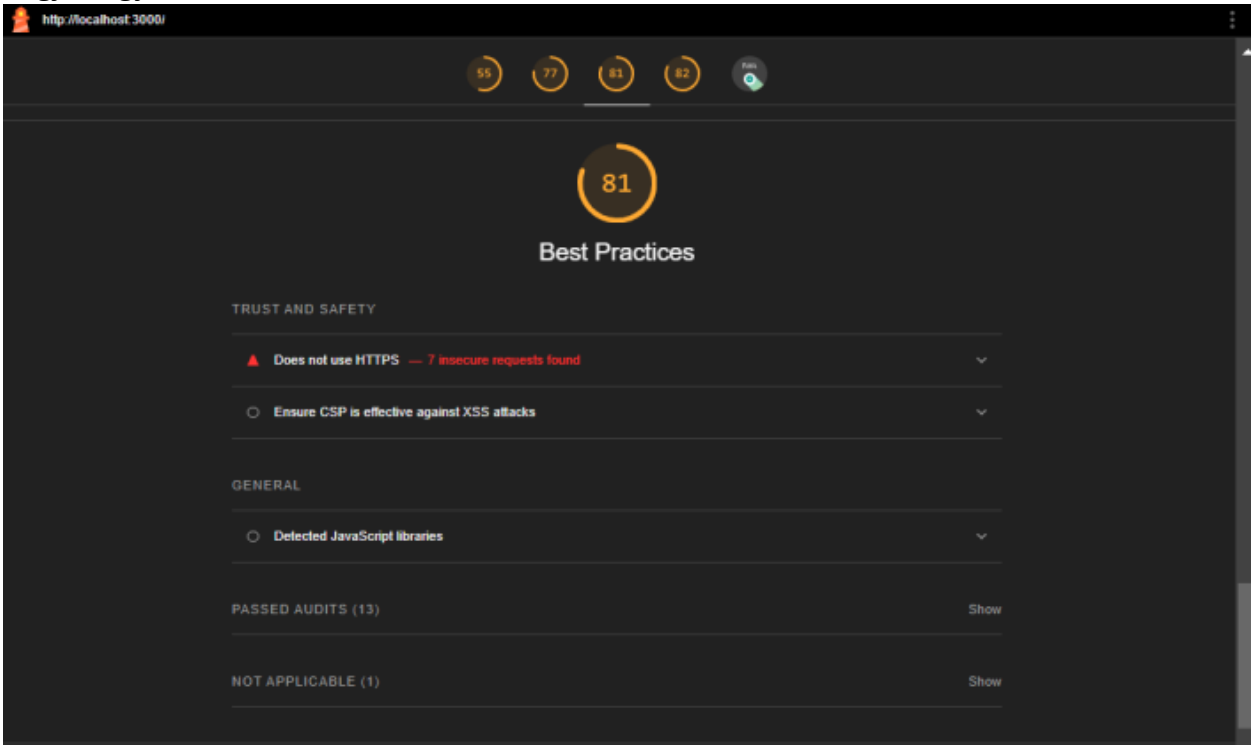


Hozzáférhetőségi pontszám: 77/100

- Nevek és címkék: A teszt azt jelzi, hogy vannak képelemek, amelyek nem rendelkeznek [alt] attribútumokkal. Ez javítható lehetőséget kínál az asszisztív technológiák, mint például képernyőolvasók számára.
- Kontraszt: A háttér és az előtér színei között nincs elegendő kontrasztarány, ami javítható lehetőséget kínál a tartalom olvashatóságának növelésére.
- Táblázatok és listák: A listák nem csak <li> elemeket és a scriptet támogató elemeket (<script> és <template>) tartalmaznak, ami javítható lehetőséget kínál a listák szemantikájának fejlesztésére.
- Manuálisan ellenőrizendő további elemek: Tíz további elem van, amelyeket manuálisan kell ellenőrizni. Ezek az elemek olyan területeket érintenek, amelyeket egy automatizált teszteszköz nem tud lefedni.

Ezek az eredmények azt mutatják, hogy a weboldalunk jól teljesít a hozzáférhetőség szempontjából, de vannak területek, ahol további fejlesztésre van szükség. A dokumentációban részletesen ismertetjük azokat a lépéseket, amelyeket a hozzáférhetőségi teljesítmény javítása érdekében tervezünk megtenni.

Legjobb gyakorlatok:

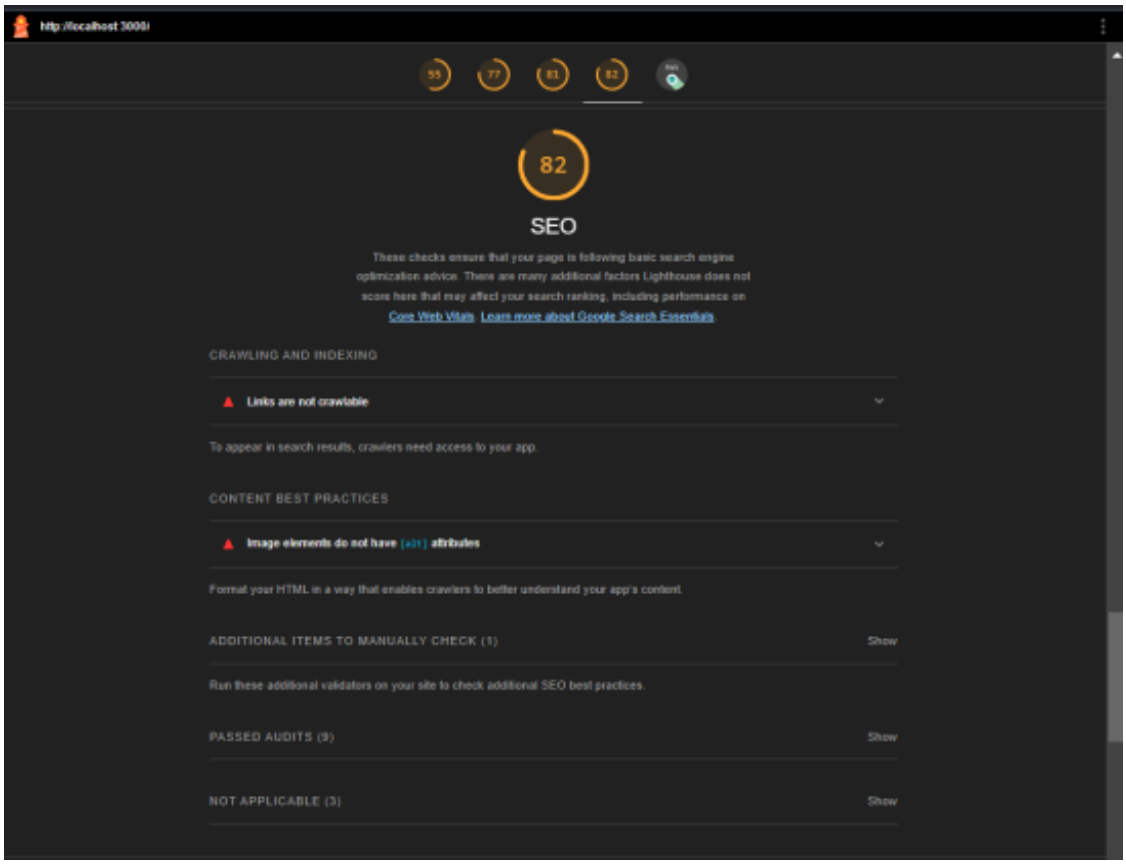


Legjobb gyakorlatok pontszám: 81/100

- Bizalom és biztonság: A teszt azt jelzi, hogy az oldal nem használ HTTPS-t, ami 7 biztonságtalan kérést eredményezett. Ezenkívül ajánlott biztosítani, hogy a CSP (Content Security Policy) hatékony legyen az XSS (Cross-Site Scripting) támadások ellen.
- Általános: A teszt észlelte a JavaScript könyvtárakat.
- Sikeresen teljesített auditok: A teszt során 13 elem sikeresen teljesítette az auditokat, amelyeket a dokumentáció részletesen ismertet.
- Nem alkalmazható elemek: Vannak olyan elemek, amelyek nem relevánsak az oldalunkra, és ezért nem érintik a „Legjobb gyakorlatok” pontszámot.

Ezek az eredmények azt mutatják, hogy a weboldalunk jól teljesít a legjobb gyakorlatok szempontjából, de vannak területek, ahol további fejlesztésre van szükség, különösen a biztonság terén. A dokumentációban részletesen ismertetjük azokat a lépéseket, amelyeket a biztonsági teljesítmény javítása érdekében tervezünk megtenni.

SEO:

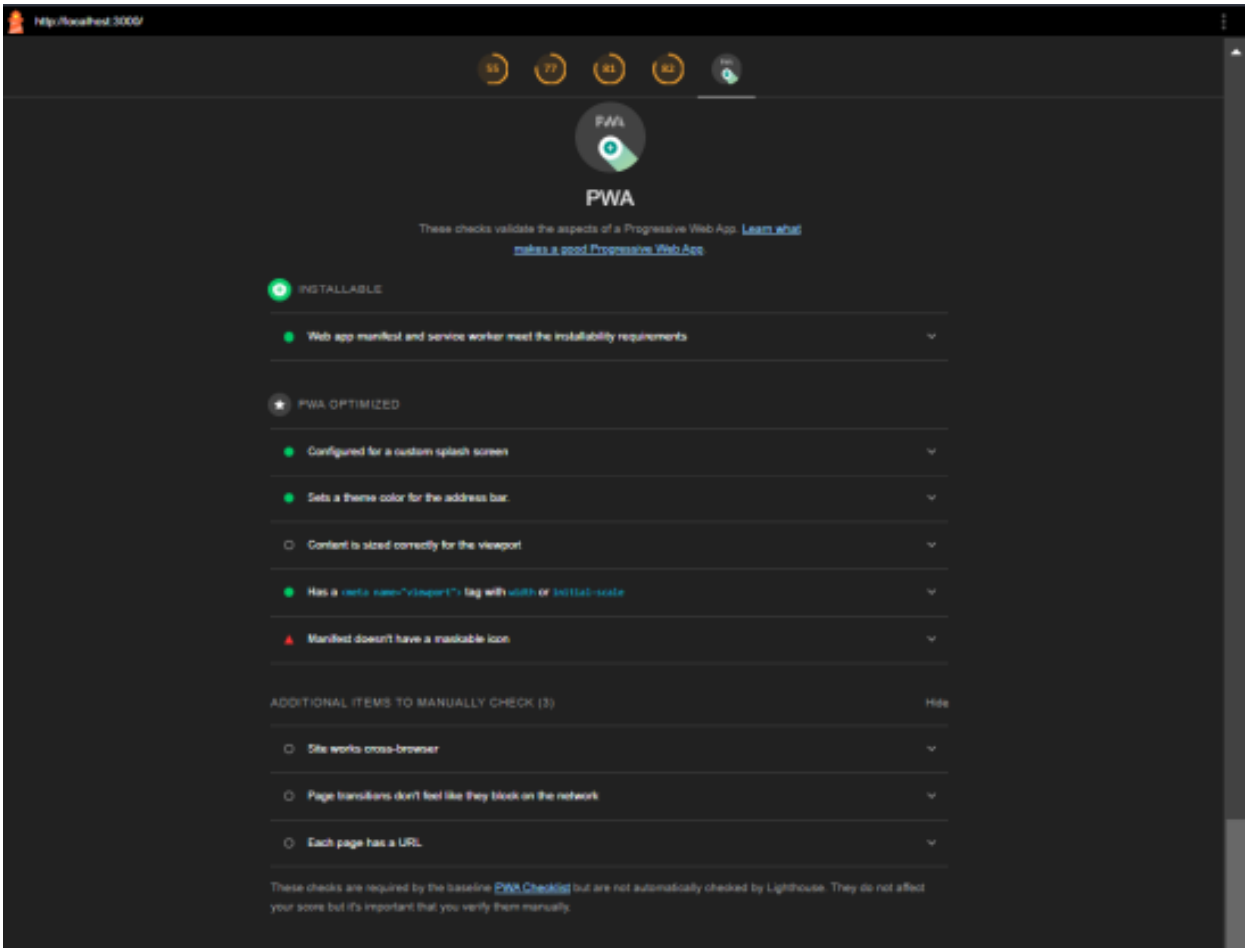


SEO Pontszám: 82/100

- Keresőmotorok indexelése és felderítése: A teszt azt jelzi, hogy vannak olyan linkek, amelyek nem kereshetők, ami akadályozhatja a keresőmotorokat az oldal tartalmának indexelésében.
- Tartalom legjobb gyakorlatok: Az elemzés megállapította, hogy vannak képelemek, amelyek nem rendelkeznek [alt] attribútumokkal, ami fontos a képek keresőmotorok általi értelmezéséhez és a hozzáférhetőség szempontjából.
- Manuálisan ellenőrizendő további elemek: A HTML fájlban található specifikációk, amelyek segítik a keresőrobotokat a tartalom jobb megértésében.
- Sikeresen teljesített auditok: A teszt során több elem is sikeresen teljesítette az auditokat, amelyeket a dokumentáció részletesen ismertet.
- Nem alkalmazható elemek: Vannak olyan elemek, amelyek nem relevánsak az oldalunkra, és ezért nem érintik a SEO pontszámot.

Ezek az eredmények azt mutatják, hogy a weboldalunk jól teljesít a SEO szempontjából, de vannak területek, ahol további fejlesztésre van szükség. A dokumentációban részletesen ismertetjük azokat a lépéseket, amelyeket a SEO teljesítmény javítása érdekében tervezünk megtenni.

PWA:



- Telepíthetőség: A webalkalmazás manifesztuma és a service worker megfelelnek a telepíthetőségi követelményeknek, ami lehetővé teszi a felhasználók számára, hogy az alkalmazást a kezdőképernyőjükre telepítsék.
- PWA optimalizálás: Az alkalmazás konfigurálva van egy egyedi indítóképernyőhöz, beállításra került egy témaszín az címsorhoz, és a tartalom megfelelően van méretezve a nézetablakhoz. Azonban a manifesztum nem rendelkezik maszkolható ikonnal, ami egy olyan terület, ahol javításra van szükség.
- Manuálisan ellenőrizendő további elemek: Két szolgáltatáscsoport létezik: Ez azt jelenti, hogy a service worker-ek megfelelően vannak konfigurálva a gyors és megbízható működés érdekében. Minden oldalnak van URL-je: Ez biztosítja, hogy az alkalmazás minden része könnyen elérhető és indexálható a keresőmotorok számára.

Ezek az eredmények azt mutatják, hogy az alkalmazásunk jól teljesít a PWA követelmények szempontjából, de vannak területek, ahol további fejlesztésre van szükség. A dokumentációban részletesen ismertetjük azokat a lépéseket, amelyeket a teljesítmény javítása érdekében tervezünk megtenni.