Grill finder weboldal

dokumentáció

Készítette: Kovács Dániel

2025. április 22.

Dátum: 2025. április 22.

Tartalomjegyzék

1. Bevezetés

A Grill Finder egy modern webalkalmazás, amelynek célja, hogy egy Grill forgalmazó cég megtudja mutatni a nála készülő grilleket. Az alkalmazás különlegessége, hogy a keresési élmény hasonló a népszerű Tinder platformhoz: a felhasználók balra vagy jobbra húzással jelezhetik érdeklődésüket az egyes grillek iránt.

A rendszert egy cég fogja üzemeltetni (a megrendelő), amely az adatbázisban szereplő grilleket manage-eli, illetve üzemelteti az oldalt. A cél egy felhasználóbarát, vizuálisan vonzó és gyorsan reagáló felület létrehozása, amely a React technológiára épül. A projekt során kiemelt figyelmet fordítottunk az intuitív felhasználói élményre, a reszponzív kinézet és a skálázhatóságra.

A dokumentáció célja, hogy bemutassa az alkalmazás főbb komponenseit, technológiai hátterét, és irányelveket adjon a további fejlesztésekhez.

2. Használt technológiák

2.1. Frontend

React: A projekt frontendje a React könyvtárra épül, amely a modern webfejlesztés egyik legelterjedtebb és legdinamikusabban fejlődő JavaScript-alapú eszköze. A React lehetővé teszi újrafelhasználható komponensek fejlesztését, ami különösen előnyös egy olyan alkalmazás esetén, amely interaktív és reszponzív felhasználói élményt kíván nyújtani.

A React-et a Facebook fejlesztette ki, és széles körben használják egyoldalas alkalmazások (SPA - Single Page Application) létrehozásában. Komponensalapú szemlélet segített az egyes funkciók, például a grillfiókok, a keresőfelület és a navigációs menü strukturált kialakításában.

A React virtual DOM rendszere gyors újrarenderelést biztosít, így a felhasználói interakciók (például a „balra-jobbra húzás") azonnal jelennek meg a felületen. Emellett a React lehetőséget ad állapotkezelésre (pl. `useState`, `useEffect` hook-okkal), ami fontos volt az interaktív UI viselkedésének nyomon követéséhez.

JavaScript: A fejlesztés során JavaScript nyelvet használtam (mert ezt jobban szeretem mint a react saját script nyelvét), amely a web egyik alapköve. A nyelv rugalmassága és széleskörű eszköztára lehetővé tette, hogy gyorsan reagáljunk a változó igényekre, és könnyedén integráljuk a különféle külső könyvtárakat és API-kat.

A JavaScript aszinkron működési lehetőségei (`async/await`, `fetch`, stb.) kulcsfontosságúak voltak az adatok hatékony lekéréséhez és megjelenítésé-

hez. Emellett az eseménykezelés és DOM manipulációk is gördülékenyen működtek a React keretrendszerrel összhangban.

2.2. Backend

Node.js: A szerveroldalt Node.js-ben segítségével íródott meg, mivel jól támogatja az aszinkron működést és gyors válaszidőt biztosít, valamint a mongoDB adatbáziskezeléssel ezt tudtam a legjobban párosítani. A REST API-k segítségével a frontend egyszerűen kommunikálhatott a backenddel. Az Express keretrendszert használtuk a végpontok kialakításához.

MongoDB: Az adatokat MongoDB-ben tároltuk, mivel a dokumentumalapú NoSQL adatbázis jól illeszkedik a dinamikus adatszerkezetekhez. A JSON formátum könnyen kezelhető volt a Node.js környezetben, és rugalmasan bővíthető maradt az adatmodell.

2.3. Fejlesztési környezet és eszközök

Visual Studio Code: Azért erre esett a választás, mert testreszabható, és rengeteg hasznos bővítménnyel rendelkezik JavaScript és React fejlesztéshez.

Fejlesztői gép:

Komponens	**Részletek**
Operációs rendszer	Windows 10 Pro 64-bit
Processzor (CPU)	Intel Core i5 6300U @ 2.40GHz (Skylake-U/Y, 14nm) – 41°C
Memória (RAM)	8,00 GB Single-Channel DDR3 (11-11-11-28)
Alaplap	LENOVO 20FMS6UU08 (U3E1)
Grafika	1920×1080 @ 60Hz (Wide viewing angle & High density FlexView Display) Intel HD Graphics 520 (Lenovo)
Tárhely	238 GB SAMSUNG MZ7LN256HMJP-000L7 SSD (SATA) – 35°C
Optikai meghajtó	Nincs optikai meghajtó észlelve
Hang	Realtek High Definition Audio

1. táblázat. Fejlesztői gép specifikációi

GitHub: A verziókövetéshez GitHub-ot használtam. A repository-k segítségével több verzió is visszakereshető, illetve a változások könnyen nyomon

követhetők voltak a fejlesztés során (github pages alkalmazására is gondoltam, viszont nem találtam ingyenes HOST-ert a backendnek, így lemondtam róla).

3. Fontosabb funkciók

3.1. Frontend

Login.jsx:A bejelentkezési oldal lehetővé teszi a felhasználók számára, hogy hitelesítsék magukat e-mail és jelszó megadásával. A beküldött adatok egy POST kérés formájában a szerverhez kerülnek, amely érvényesítés után visszaad egy token-t, amit a böngésző `localStorage`-ban tárol. Sikeres bejelentkezés után a felhasználó automatikusan átirányításra kerül a dashboard felületre. A komponens valós időben kezeli a hibákat és a betöltési állapotot.

Register.jsx: A regisztrációs felületen lehet új fiókot létrehozni. Fontos, hogy e-mailcímet is megadjunk, ami formátum ellenőrzésen is átmegy.

AddGrill.jsx: Ez az Admin-ok kezelési felülete. itt tudnk hozzáadni Grilleket, vagy elvenni. Csak akkor jelenik meg, ha a felhasználó admin típusú (ezt manuálisan kell beállítani az adatbázisban.)

Modify.jsx: Itt lehet módosítani a grill-ek adatait. Mást nagyon nem kell tudni róla.

UserDashboard.jsx: Az oldal betöltésekor lekéri az összes grill-t. Amikor a felhasználó megnyomja a "like" gombot, a komponens elküld egy POST kérést a szervernek az adott grill ID-jével, amely frissíti a grill `likes` és `isLiked` mezőit. Ezután a felhasználó animációval a következő grillhez kerül. Ha a felhasználó inkább tovább szeretne lépni, anélkül, hogy kedvelné az adott grillt, a "skip" gombot használhatja. Mindkét esetben az animáció iránya (`left` vagy `right`) és a belépés/kilépés állapota szabályozza a vizuális áttűnést.

3.2. Backend végpontok

Express és Middleware: Az alkalmazás az `express` könyvtárral van létrehozva, amely a REST API működését biztosítja. A `cors` csomag engedélyezi a különböző domainek közötti kommunikációt, míg a `express.json()` a JSON típusú kérés testek feldolgozását végzi.

Mongoose Kapcsolat: A `mongoose.connect()` metódus csatlakozik a MongoDB adatbázishoz, a kapcsolat sikerességéről vagy hibájáról konzol üzenet

tájékoztat.

auth Middleware: Ez a köztes réteg ellenőrzi a JWT tokent. Ha a token hiányzik, érvénytelen vagy lejárt, a rendszer hibával válaszol. Sikeres ellenőrzés után a dekódolt token adatokat a `req.userData`-ban elérhetővé teszi.

adminAuth Middleware: Meghosszabbítja az `auth` működését, és csak admin jogosultságú felhasználók számára engedélyezi a hozzáférést bizonyos végpontokhoz.

/api/register: Felhasználó regisztrációs végpont. Ellenőrzi, hogy étterem típusú felhasználó esetén meg van-e adva az étterem neve és címe. Az új felhasználó alapértelmezetten nem admin.

/api/login: Bejelentkezési végpont, ahol a rendszer ellenőrzi az email és jelszó párost, majd sikeres bejelentkezés esetén egy óráig érvényes JWT tokent küld vissza.

/api/grills (POST): Admin jogosultságot igénylő végpont új grill létrehozására. A kérésben szerepelnie kell a grill nevének, képének URL-jének és leírásának.

/api/grills (GET): Hitelesített felhasználók számára elérhető végpont, amely visszaadja az összes grill objektumot. Minden grillhez tartozik egy `isLiked` mező, amely azt jelzi, hogy az adott felhasználó kedvelte-e már azt.

/api/grills/:id/like: Ez a végpont lehetővé teszi egy grill kedvelését vagy kedvelésének visszavonását. Frissíti a grill `likes` számát, és a felhasználó `likedGrills` tömbjét.

/api/grills/:id (PUT): Csak adminok számára elérhető végpont, amellyel egy meglévő grill adatai módosíthatók. Minden mező (`name`, `picture`, `desc`) kötelező.

/api/grills/:id (DELETE): Egy adott grill törlését végzi, csak admin jogosultsággal. A grill törlése után minden felhasználónál eltávolítja az adott grill ID-ját a `likedGrills` mezőből is.

4. Továbbfejlesztési lehetőségek

Az alkalmazás jelenlegi formájában stabil és jól működő alapokat biztosít, azonban több irányban is továbbfejleszthető a funkcionalitás, felhasználói élmény és adminisztrációs lehetőségek szempontjából. Az alábbiakban néhány javasolt fejlesztési irányt sorolunk fel:

- **Képfeltöltés támogatása:** Jelenleg a grill képeket URL-en keresztül lehet megadni. Egy képfeltöltő felület beépítése (pl. drag-and-drop vagy fájl kiválasztása) jelentősen növelné a felhasználói élményt.
- **Értékelési rendszer:** A „like" funkción túl lehetőség lenne szöveges értékelések vagy csillagos értékelési rendszer bevezetésére.
- **Grill szűrési és keresési lehetőségek:** A felhasználók számára lehetőséget lehetne adni grillek szűrésére típus, árkategória vagy egyéb jellemzők alapján.
- **Match rendszer:** Amennyiben egy grill több felhasználónál is népszerű, lehetne „match" alapú értesítést küldeni, hasonlóan társkereső alkalmazásokhoz.
- **Push értesítések:** Új grillek, akciók vagy események megjelenésekor automatikus értesítések küldése.
- **Admin jogosultság kezelő felület:** Az admin jogosultságokat jelenleg manuálisan kell beállítani. Egy admin felület, ahol ez dinamikusan módosítható, megkönnyítené a jogosultságkezelést.
- **Mobilalkalmazás fejlesztése:** A jelenlegi webes alkalmazás alapul szolgálhatna egy React Native alapú mobilalkalmazás létrehozásához is.
- **Felhasználói profiloldal:** A felhasználók saját profiloldalon láthatnák a korábban kedvelt grilleket, beállításaikat és statisztikáikat.
- **Statisztikák és admin dashboard:** Egy részletes admin dashboard, amely statisztikákat mutat a felhasználói aktivitásról, kedvelt grillekről stb.

5. Összefoglalás

Összegezve: A Grill Finder webalkalmazás célja, hogy a felhasználók könnyedén és élményszerűen találjanak rá a számukra legszimpatikusabb luxus grillekre.

Az alkalmazás főbb funkciói közé tartozik a regisztráció, bejelentkezés, grill-adatlapok közötti navigálás "like" és "skip" lehetőséggel, valamint az adminisztrátori jogosultságokhoz kötött grill-hozzáadás, módosítás és törlés. Az intuitív, kártya-alapú felhasználói felület és az interaktív működés révén a Grill Finder élményszerű és felhasználóbarát platformot kínál.

A rendszer REST API-kon keresztül kommunikál a backenddel, biztonságos autentikációt biztosít JWT tokenek segítségével, és hatékony állapotkezeléssel, animációkkal és reszponzív dizájnnal nyújt teljeskörű felhasználói élményt.

A dokumentáció célja, hogy átfogó képet adjon a fejlesztés során használt technológiákról, a rendszer komponenseiről, és iránymutatást nyújtson a további fejlesztésekhez és karbantartáshoz.