Dokumentáció

Tartalomjegyzék:

[Bevezetés 2](#_Toc194843448)

[Felhasznált technológiák 3](#_Toc194843449)

[Kódolási konvenciók 4](#_Toc194843450)

[Controllerek 7](#_Toc194843451)

[Általános működés 12](#_Toc194843452)

[Osztályok 13](#_Toc194843453)

[Tesztelés 40](#_Toc194843454)

[Ismert hibák 45](#_Toc194843455)

[Fejlesztési lehetőségek 46](#_Toc194843456)

[Fejlesztők 47](#_Toc194843457)

# Bevezetés

Egy Webshop létrehozása, amely bakelit lemezek értékesítésére szolgál.

Az adatbázishoz REST API, illetve Firebase biztosítja.  
Frontendnek egy asztali és egy webes alkalmazás állrendelkezésre.  
Az asztali alkalmazásból adminisztrálhatóak a lemezek, a webes felületen böngészhetőek.

# Felhasznált technológiák

Backend  
REST API  
 Firebase (user kezeléshez)

Készítéshez használt programok

Dia  
Visual Studio 2022  
Visual Studio Code  
Swagger

Frontend

Angular

Készítéshez használt programok  
Paint

Visual Studio Code

# Kódolási konvenciók

A kódot git verziókezelővel használjuk.

Alapkönyvtárak

api

assets

database

desktop

web

Az api backend, a web a webes frontend.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Végpont** | **Metódus** | **Leírás** | **Bemenet** | **Kimenet** |
| /items | get | Tételek lekérdezése | Nincs bemenet. | Lista a tételekről (id, productId, price, quantity) |
| /items | post | Tételek hozzáadása | { productId: number, price: number, quantity: number } | Az új tétel adatai (id, productId, price, quantity) |
| /items{id} | get | Egy tétel lekérdezése | id (szám) | A tétel adatai (id, productId, price, quantity) |
| /items{id} | put | Tétel szerkesztése | |  | | --- | |  |  |  | | --- | | { productId: number, price: number, quantity: number } | | A frissített tétel adatai (id, productId, price, quantity) |
| /items{id} | delete | Tétel törlése | |  | | --- | |  |  |  | | --- | | id (szám) | | Nincs kimenet (sikeres törlés esetén). |
| /orders | get | Rendelések lekérése | Nincs bemenet. | |  | | --- | | Lista a rendelésekről (id, userId, date, items) |  |  | | --- | |  | |
| /orders | post | Rendelések hozzáadása | { userId: string, items: [{productId: number, quantity: number}] } | A rendelés adatai (id, userId, date, items) |
| /orders{id} | get | Egy rendelés lekérése | id (szám) | |  | | --- | |  |  |  | | --- | | A rendelés adatai (id, userId, date, items) | |
| /orders{id} | put | Rendelés szerkesztése | { userId: string, items: [{productId: number, quantity: number}] } | A frissített rendelés adatai (id, userId, date, items) |
| /orers{id} | delete | Rendelés törlése | id (szám) | Nincs kimenet (sikeres törlés esetén). |
| /orders{userid} | get | Rendelés lekérése személy szerint | userId (string) | A rendelés adatai (id, userId, date, items) |
| /orders/clean-invalid-items | delete | Törli a nem létező tételeket | Nincs bemenet. | Nincs (sikeres törlés esetén). |
| /products | get | Termékek lekérése | Nincs bemenet. | |  | | --- | |  |  |  | | --- | | Lista a termékekről (id, title, price, genre) | |
| /products | post | Termékek hozzáadása | |  | | --- | | { title: string, artist: string, price: number, genre: string } |  |  | | --- | |  | | |  | | --- | | Az új termék adatai (id, title, artist, price, genre) |  |  | | --- | |  | |
| /products{id} | get | Egy termék lekérése | |  | | --- | | id (szám) |  |  | | --- | |  | | |  | | --- | | A termék adatai (id, title, artist, price, genre) |  |  | | --- | |  | |
| /profucts{id} | put | Termék szerkesztése | |  | | --- | | { title: string, artist: string, price: number, genre: string } |  |  | | --- | |  | | |  | | --- | | A frissített termék adatai (id, title, artist, price, genre) |  |  | | --- | |  | |
| /products{id} | delete | Termék törlése | |  | | --- | | id (szám) |  |  | | --- | |  | | |  | | --- | | Nincs kimenet (sikeres törlés esetén) |  |  | | --- | |  | |

# Controllerek

**ItemsController**

Az ItemsController egy REST API, amely az "Item" entitás kezelését teszi lehetővé. Az API lehetőséget biztosít az adatok lekérésére, módosítására, létrehozására és törlésére.

**Végpontok**

**1. GET /api/Items**

**Leírás:** Lekéri az összes elérhető tételt az adatbázisból.

**2. GET /api/Items/{id}**

**Leírás:** Lekér egy adott tételt az ID alapján.

**Paraméterek:**

id (int, URL-ben): A keresett tétel azonosítója.

**Hibakódok:**

404 Not Found: Ha a megadott azonosítóval nem létezik tétel.

**3. POST /api/Items**

**Leírás:** Új tétel létrehozása.

**Válasz:**

201 Created: A tétel sikeresen létre lett hozva.

400 Bad Request: Ha az adatok hibásak.

**4. PUT /api/Items/{id}**

**Leírás:** Egy meglévő tétel módosítása.

**Paraméterek:**

id (int, URL-ben): A módosítandó tétel azonosítója.

**Válasz:**

204 No Content: A tétel sikeresen módosítva lett.

400 Bad Request: Ha az ID nem egyezik.

404 Not Found: Ha a tétel nem található.

**5. DELETE /api/Items/{id}**

**Leírás:** Egy tétel törlése az ID alapján.

**Paraméterek:**

id (int, URL-ben): A törlendő tétel azonosítója.

**Válasz:**

204 No Content: A tétel sikeresen törölve lett.

404 Not Found: Ha a tétel nem létezik.

**OrdersController**

Az OrdersController az API rendelésekhez kapcsolódó műveleteit kezeli. Lehetővé teszi a rendelések lekérdezését, létrehozását, frissítését és törlését.

**Végpontok**

**1. GET /api/Orders - Minden rendelés lekérdezése**

**Leírás:**  
Visszaadja az összes rendelést, beleértve az ahhoz tartozó tételeket és termékeket.

**Válasz:**

200 OK: Az összes rendelés listája

500 Internal Server Error: Hiba történt a lekérdezés során

**2. GET /api/Orders/{id} - Egy adott rendelés lekérdezése**

**Leírás:**  
Visszaadja a megadott azonosítójú rendelést, beleértve az ahhoz tartozó tételeket és termékeket.

**Válasz:**

200 OK: A kért rendelés adatai

404 Not Found: A rendelés nem található

**3. GET /api/Orders/user/{userId} - Egy adott felhasználó rendeléseinek lekérdezése**

**Leírás:**  
Visszaadja egy adott felhasználó rendeléseit.

**Válasz:**

200 OK: A felhasználó rendeléseinek listája

404 Not Found: A felhasználónak nincs rendelése

**4. PUT /api/Orders/{id} - Egy rendelés módosítása**

**Leírás:**  
Módosítja a megadott rendelést.

**Törzs:**

Order objektum a módosított adatokkal

**Válasz:**

204 No Content: Sikeres módosítás

400 Bad Request: Az azonosító nem egyezik

404 Not Found: A rendelés nem található

**5. POST /api/Orders - Új rendelés létrehozása**

**Leírás:**  
Létrehoz egy új rendelést.

**Törzs:**

Order objektum az új rendelés adataival

**Válasz:**

201 Created: A rendelés sikeresen létrejött

**6. DELETE /api/Orders/{id} - Egy rendelés törlése**

**Leírás:**  
Törli a megadott azonosítójú rendelést.

**Válasz:**

204 No Content: Sikeres törlés

404 Not Found: A rendelés nem található

**7. DELETE /api/Orders/clean-invalid-items - Érvénytelen tételek törlése**

**Leírás:**  
Törli azokat a rendelési tételeket, amelyek nem kapcsolódnak létező termékhez.

**Válasz:**

200 OK: A törölt tételek száma

404 Not Found: Nem voltak érvénytelen tételek

**ProductsController**

A ProductsController az API termékekhez kapcsolódó műveleteit kezeli. Lehetővé teszi a termékek lekérdezését, létrehozását, frissítését és törlését.

**Végpontok**

**1. GET /api/Products - Minden termék lekérdezése**

**Leírás:**Visszaadja az összes terméket az adatbázisból.

**Válasz:**

**200 OK**: Az összes termék listája.

**500 Internal Server Error**: Hiba történt a lekérdezés során.

**2. GET /api/Products/{id} - Egy adott termék lekérdezése**

**Leírás:**Visszaadja a megadott azonosítójú terméket.

**Válasz:**

**200 OK**: A kért termék adatai.

**404 Not Found**: A termék nem található.

**3. PUT /api/Products/{id} - Egy termék módosítása**

**Leírás:**Módosítja a megadott terméket.

**Törzs:**

Product objektum a módosított adatokkal.

**Válasz:**

**204 No Content**: Sikeres módosítás.

**400 Bad Request**: Az azonosító nem egyezik.

**404 Not Found**: A termék nem található.

**4. POST /api/Products - Új termék létrehozása**

**Leírás:**Létrehoz egy új terméket.

**Törzs:**

Product objektum az új termék adataival.

**Válasz:**

**201 Created**: A termék sikeresen létrejött.

**5. DELETE /api/Products/{id} - Egy termék törlése**

**Leírás:**Törli a megadott azonosítójú terméket.

**Válasz:**

**204 No Content**: Sikeres törlés.

**404 Not Found**: A termék nem található.

# Általános működés

A REST API HTTP kéréseket fogad, melyek tartalmazzák a műveletekhez szükséges megfelelő adatokat. A kényes műveletek végpontjai védettek, autentikáció szükséges a használatukhoz. Ilyenek a kijelentkezés, új termék (pl. bakelit lemez) felvétele, termék adatainak módosítása, termék törlése, rendelés leadása. A termékek és rendelések lekérdezésének végpontjai publikusak, ezek használatához autentikáció nem szükséges.

Az adatokat JSON formátumban fogadja és feldolgozza. A vezérlést kontrollerek valósítják meg, minden adatkezelési csoportnak (autentikáció, termékek, rendelések) külön kontrollere van, itt történik az adatfeldolgozás.

A kontrollerek modellekkel vannak kapcsolatban, amelyek az adatkezelésért felelősek. Minden adatkezelési csoportnak (felhasználó, termék, rendelés) külön modellje van, itt történik az adatok adatbázisból való kiolvasása, illetve az adatok kiírása adatbázisba.

A modellek adatbázis táblákkal vannak kapcsolatban, melyek az adatok tárolásáért felelősek. Az adatbázis táblák adatait a modellek kezelik.

# Osztályok

**ProductService**

Feladata a termékekkel kapcsolatos HTTP-kérések kezelése az API-val való kommunikáció során.

**Metódusok:**

* **getProducts()**

Ez a függvény a szerveren tárolt összes terméket lekéri. Ehhez egy GET kérés segítségével elküldi a kérést a következő címre: [**https://localhost:7025/api/Products**](https://localhost:7025/api/Products).  
A szerver erre válaszként visszaküldi az összes terméket egy tömbben. Ez azt jelenti, hogy egy listát kapunk, amelyben minden egyes elem egy-egy termék adatait tartalmazza.

Bemenet:  
Nincs bemeneti adat.

Kimenet:  
Egy tömb, amely tartalmazza az összes terméket. Minden termék objektum a következő mezőket tartalmazhatja.

* **getProductById(id: number)**

Ez a függvény egy adott termék adatainak lekérésére szolgál. Egy GET kérést küld a szervernek az alábbi címre: [https://localhost:7025/api/Products/{id}](https://localhost:7025/api/Products/%7bid%7d). Az {id} helyére annak a terméknek az azonosítója kerül, amelyet le szeretnénk kérni. A szerver válasza egy objektum, amely a kiválasztott termék adatait tartalmazza.

Bemenet:id (szám): A lekérdezni kívánt termék azonosítója.

Kimenet:  
A lekérdezett termék adatait tartalmazó objektum.

* **addProduct(product: any)**

Ez a függvény egy új terméket küld a szerverre, amelyet el kell menteni az adatbázisba. Ehhez egy POST kérés segítségével elküldi a termék adatait a következő címre: <https://localhost:7025/api/Products>. A szerver a kapott adatokat feldolgozza, elmenti az új terméket, és válaszként visszaküldi az elmentett termék információit.

Bemenet:  
A termék adatai, amelyeket a kliens küld el.

Kimenet:  
A szerver válasza az elmentett termék adataival, beleértve a generált id-t.

* **updateProduct(id: number, product: any)**

Ez a függvény egy meglévő termék adatainak frissítésére szolgál. Egy PUT kérés segítségével elküldi a módosított adatokat az alábbi címre: [https://localhost:7025/api/Products/{id}](https://localhost:7025/api/Products/%7bid%7d). Az {id} helyére annak a terméknek az azonosítója kerül, amelyet módosítani szeretnénk. A szerver frissíti a megfelelő adatokat, és válaszként visszaküldi a módosított termék információit.

Bemenet:id (szám): A módosítani kívánt termék azonosítója.

A módosított termék adatai, amelyeket a kliens küld el.

Kimenet:  
A szerver válasza a módosított termék adataival.

* **deleteProduct(id: number)**

Ez a függvény egy adott termék törlésére szolgál. Egy DELETE kérés segítségével elküldi a törlési kérelmet a következő címre: https://localhost:7025/api/Products/{id}. Az {id} helyére annak a terméknek az azonosítója kerül, amelyet törölni szeretnénk. Ha a szerver sikeresen törölte a terméket, akkor visszaigazolást küld.

Bemenet:

id (szám): A törölni kívánt termék azonosítója.

Kimenet:  
A törlés sikerességét visszaigazoló üzenet

**OrdersService**

Feladata a rendelésekkel kapcsolatos HTTP-kérések kezelése az API-val való kommunikáció során.

* **getOrders()**

Ez a függvény az adminisztrátorok számára az összes rendelést lekéri. Ehhez egy GET kérés segítségével elküldi a kérést az alábbi címre: https://localhost:7025/api/Orders. A szerver válasza egy tömb, amely tartalmazza az összes rendelést.

* **Bemenet:**  
  Nincs bemeneti adat.
* **Kimenet:**  
  Egy tömb, amely tartalmazza az összes rendelést.
* **getOrdersForUser()**

Ez a függvény a felhasználó saját rendeléseit lekéri. Először a felhasználói adatokat lekéri az AuthService segítségével, majd egy GET kérést küld a következő címre: https://localhost:7025/api/Orders/user/{userId}, ahol a {userId} helyére a bejelentkezett felhasználó azonosítója kerül. A szerver válasza egy tömb, amely tartalmazza a felhasználó összes rendelését.

Bemenet:  
Nincs bemeneti adat.

Kimenet:  
A felhasználó összes rendelése egy tömbben.

* **getOrdersByUserId(userId: string)**

Ez a függvény egy adott felhasználó rendeléseit lekéri a felhasználó ID-ja alapján. Ehhez egy GET kérés segítségével elküldi a kérést a következő címre: https://localhost:7025/api/Orders/user/{userId}. A szerver válasza egy tömb, amely tartalmazza az adott felhasználó összes rendelését.

Bemenet:userId (string): A felhasználó azonosítója.

Kimenet:  
A felhasználó összes rendelése egy tömbben.

* **deleteOrder(id: number)**

Ez a függvény egy adott rendelés törlésére szolgál. Egy DELETE kérés segítségével elküldi a törlési kérelmet a következő címre: https://localhost:7025/api/Orders/{id}, ahol az {id} helyére a törlendő rendelés azonosítója kerül.

Bemenet:id (szám): A törölni kívánt rendelés azonosítója.

Kimenet:  
A törlés sikerességét visszaigazoló üzenet.

* **placeOrder(orderData: any)**

Ez a függvény egy új rendelést küld a szerverre, amelyet el kell menteni az adatbázisba. Ehhez egy POST kérés segítségével elküldi a rendelés adatait a következő címre: https://localhost:7025/api/Orders.

Bemenet:orderData (objektum): A rendelés adatainak objektuma.

Kimenet:  
A szerver válasza az elmentett rendelés adataival.

**ConfigService**

Feladata a konfigurációs beállítások és egyéb globális információk kezelése.

* **getConfig()**

Ez a függvény lekéri az alkalmazás konfigurációs beállításait. Ehhez egy GET kérést küld a következő címre: https://localhost:7025/api/Config. A szerver válaszként visszaküldi az összes konfigurációs adatot, amelyet az alkalmazás használ.

**CartService**

Feladata a vásárlói kosár kezelésére és az API-val való kommunikációra vonatkozó műveletek ellátása.

* **getCart()**

Ez a függvény visszaadja a kosár tartalmát egy Observable formájában.

Bemenet:  
Nincs bemenet.

Kimenet:  
Observable<any[]>: A kosárban lévő termékek listája, amely a BehaviorSubject-on keresztül érhető el. A kosárban található termékek adatai frissülnek, ha a kosár módosul.

* **addProduct(product: any)**

Ez a függvény egy terméket ad hozzá a kosárhoz. Ha a termék már létezik, akkor növeli a mennyiségét, ha nem, akkor hozzáadja a kosárhoz.

Bemenet:product: any: A hozzáadni kívánt termék objektuma, amely tartalmazza a szükséges adatokat, például az id-t, nevet, árat stb.

Kimenet:Nincs közvetlen kimenet, de a kosár állapota frissül és a változások elmentődnek a localStorage-ba, és a BehaviorSubject figyelői értesítést kapnak a változásról.

* **deleteProduct(product: any)**

Ez a függvény eltávolít egy terméket a kosárból. Ha a termék mennyisége nagyobb, mint 1, akkor csökkenti a mennyiséget, egyébként törli a terméket.

Bemenet:  
product: any: Az eltávolítani kívánt termék objektuma, amely tartalmazza az id-t, amely alapján azonosítani tudjuk a törlendő terméket.

Kimenet:  
Nincs közvetlen kimenet, de a kosár állapota frissül, az új kosáradatok elmentődnek a localStorage-ba, és a BehaviorSubject figyelői értesítést kapnak a változásról.

* **clearCart()**

Ez a függvény kiüríti a kosarat.

Bemenet:Nincs bemenet.

Kimenet:  
Nincs közvetlen kimenet, de a kosár állapota teljesen kiürül, és a BehaviorSubject figyelői értesítést kapnak a változásról. Az új kosáradatok elmentődnek a localStorage-ba.

* **getCartItemCount()**

Ez a függvény visszaadja a kosárban lévő összes termék mennyiségét.

Bemenet:  
Nincs bemenet.

Kimenet:  
number: A kosárban található termékek összesített mennyisége.

* **updateCart()**

Ez a privát függvény frissíti a kosár tartalmát, és eltárolja azt a localStorage-ben. Az Observable-t is frissíti, így a kosár változásai automatikusan frissülnek az alkalmazásban.

* **loadCart()**

Ez a függvény betölti a kosár tartalmát a localStorage-ból, ha az elérhető, és frissíti a kosár állapotát a BehaviorSubject-ban.

Bemenet:  
Nincs bemenet.

Kimenet:  
Nincs közvetlen kimenet, de a kosár állapota frissül a localStorage-ból betöltött adatokkal, és a BehaviorSubject figyelői értesítést kapnak a változásról

* **placeOrder(orderData: any)**

Ez a függvény egy új rendelést küld a szerverre, amelyet el kell menteni az adatbázisba.

Bemenet:  
orderData: any: A rendelés adatainak objektuma, amely tartalmazza a kosárban lévő termékek adatait és más szükséges információkat a rendelésről.

Kimenet:  
Observable<any>: A szerver válasza az elmentett rendelés adataival, amelyet a kliens kap vissza a rendelés sikeres létrehozása után.

**AuthService**

Feladata a felhasználók autentikálása és jogosultságaik kezelése, beleértve a bejelentkezést, regisztrációt, jelszó visszaállítást és adminisztrátori jogosultságok kezelését.

* **getIsAdmin()**

Ez a függvény visszaadja, hogy a bejelentkezett felhasználó admin-e.

Bemenet:  
Nincs bemenet.

Kimenet:  
 BehaviorSubject<boolean> – true, ha az admin státusz be van állítva, különben false.

* **getIsLoggedUser()**

Ez a függvény visszaadja, hogy van-e bejelentkezett felhasználó.

Bemenet:  
Nincs bemenet.

Kimenet:  
BehaviorSubject<boolean> – true, ha van bejelentkezett felhasználó, különben false.

* **getUsers()**

Lekéri az összes felhasználót az adatbázisból.

Bemenet:  
Nincs, de a kérés tartalmaz egy Authorization fejléccel ellátott token-t.

Kimenet:  
Observable<any> – Az összes felhasználó adata (JSON formátumban).

* **setUserClaims(uid: any, claims: any)**

Felhasználói jogosultságok beállítása a Firebase-ben.

Bemenet:uid: string – A felhasználó egyedi azonosítója.

claims: any – A beállítandó jogosultságok objektuma.

Kimenet:Observable<any> – A szerver válasza a frissített jogosultságokkal.

* **updateUser(displayName: any, phoneNumber: any, email: any)**

Frissíti a bejelentkezett felhasználó adatait.

Bemenet:  
displayName: string – A felhasználó új neve.

phoneNumber: string – A felhasználó telefonszáma.

email: string – A felhasználó e-mail címe.

Kimenet:  
Observable<any> – A frissített felhasználói adatok válasza.

* **getLoggedUser()**

Visszaadja a bejelentkezett felhasználó adatait.

Bemenet:  
Nincs bemenet.

Kimenet:  
BehaviorSubject<any> – A bejelentkezett felhasználó adatai.

* **googleAuth()**

Google fiókkal történő bejelentkezést indít.

Bemenet:  
Nincs bemenet.

Kimenet:  
Promise<any> – A hitelesítés eredménye.

* **signOut()**

Kijelentkezteti a felhasználót és visszairányítja a login oldalra.

Bemenet:  
Nincs bemenet.

Kimenet:  
 Promise<void> – A kijelentkezés befejezése után végrehajtódik.

* **signUpMailPassword(email: string, password: string)**

Regisztráció email és jelszó alapján.

Bemenet:  
email: string – A felhasználó email címe.

password: string – A felhasználó jelszava.

Kimenet:  
Promise<void> – Sikeres regisztráció esetén végrehajtja a navigációt és beállítja az állapotot.

* **signInMailPassword(email: string, password: string)**

Bejelentkezés email és jelszó alapján.

Bemenet:  
email: string – A felhasználó email címe.

password: string – A felhasználó jelszava.

Kimenet:  
Promise<void> – Sikeres bejelentkezés után a felhasználó a kezdőoldalra kerül.

* **forgotPassword(email: any)**

Elküldi a jelszó visszaállító emailt a megadott címre.

Bemenet:  
email: string – A felhasználó email címe.

Kimenet:  
Promise<void> – Ha sikeres, akkor a felhasználó emailt kap a visszaállítási linkkel.

* **getUserData()**

Visszaadja az aktuálisan bejelentkezett felhasználó adatait.

Bemenet:  
Nincs bemenet.

Kimenet:  
Observable<any> – Az autentikált felhasználó adatai.

**UserComponent:**

* **Logged User lekérése:**

A komponens a getLoggedUser metódussal lekéri az aktuálisan bejelentkezett felhasználót az AuthService szolgáltatástól.

Az eredményt tárolja a loggedUser változóban, és ha van bejelentkezett felhasználó, akkor további műveletek történnek

* **Felhasználók lekérése:**

Ha létezik bejelentkezett felhasználó, a komponens meghívja a getUsers metódust az AuthService-től, amely lekéri az összes felhasználót.

Az adatokat a users változóban tárolja.

* **Claims beállítása:**

A setCustomClaims metódus segítségével beállítható egy felhasználó jogosultsága, amelyet az AuthService-hez küld, ahol a setUserClaims metódus segítségével módosítják a felhasználói jogosultságokat.

* **Jogosultság módosítása (change metódus):**

A change metódusban egy adott felhasználó (UID alapján) jogosultságát módosítja a claims alapján. A jogosultságok módosítása az AuthService segítségével történik.

* **Bemenet és kimenet:**

Bemenet:

uid: A felhasználó azonosítója.

claims: A felhasználóhoz tartozó új jogosultságok.

Kimenet:

Módosított felhasználói jogosultságok, amelyeket az AuthService-től kapott válasz alapján frissítenek.

* **Metódusok:**

**getLoggedUser()**

A bejelentkezett felhasználó adatait lekéri az AuthService-től.

**getUsers()**

Az összes felhasználó adatát lekéri az AuthService-től.

**setCustomClaims(uid: any, claims: any)**

A megadott felhasználó UID-ja és az új jogosultságok (claims) alapján beállítja a felhasználó jogosultságait az AuthService segítségével.

**change(uid: any)**

Módosítja a felhasználó jogosultságait a uid és az új claims alapján.

**UsComponent:**

Ennek a komponensnek nincsen funkcionalitása.

**SettingsComponent:**

Ennek a komponensnek nincsen funkcionalitása.

**RegisterComponent:**

* **Űrlapmezők és Jelszóellenőrzés:**

A komponens biztosít egy regisztrációs űrlapot, ahol a felhasználók megadhatják nevüket, email címüket és a jelszavukat.

A felhasználó képes a jelszót kétszer megadni (jelszó és jelszó megerősítése), és ha a két jelszó nem egyezik, figyelmeztetés jelenik meg.

* **Regisztráció:**

A regisztrációs folyamat akkor kezdődik el, amikor a felhasználó megnyomja a regisztráció gombot.

Az email és jelszó adatokat az AuthService signUpMailPassword() metódusa használja a Firebase felhasználói regisztráció végrehajtásához.

* **Jelszavak Összehasonlítása:**

A register() metódusban a jelszavak összehasonlítása történik.

Ha a két jelszó nem egyezik, a felhasználó értesítést kap, és a regisztráció nem történik meg.

* **Bemenet és Kimenet:**

Bemenet:

name (string): A felhasználó neve.

email (string): A felhasználó email címe.

password (string): A felhasználó jelszava.

passwordConfirm (string): A felhasználó által másodszor megadott jelszó (megerősítés).

Kimenet:

A regisztráció sikeres végrehajtása esetén a felhasználó regisztrálódik, és a rendszer átirányíthatja a bejelentkezéshez vagy más oldalra, amennyiben van ilyen implementálva.

* **Metódusok:**

**register():** A felhasználói adatokat ellenőrzi, és ha minden helyes, elindítja a regisztrációs folyamatot a backend rendszeren keresztül.

**ProductsComponent:**

* **Termékek lekérése és megjelenítése:**

A komponens a ConfigService segítségével lekéri az összes terméket, és a products és filteredProducts változókban tárolja őket.

A filteredProducts változó a termékek szűrt listáját tartalmazza, amelyet a felhasználó kereshet.

* **Termékek keresése:**

A searchProducts() metódus segítségével a felhasználó kereshet a termékek között.

A keresés a termék címére, előadójára és műfajára vonatkozik.

A keresési eredmények az filteredProducts változóban jelennek meg.

* **Kosárba helyezés:**

A addCart() metódus lehetővé teszi, hogy a felhasználó hozzáadja a kívánt terméket a kosárhoz a CartService segítségével.

* **Vásárlás megerősítése modálablakban:**

A openPurchaseModal() metódus megnyitja a vásárlási modált, amely tartalmazza a kiválasztott terméket.

A felhasználó a modálablakon keresztül megerősítheti a vásárlást, amely a confirmPurchase() metódus segítségével történik.

A vásárlás után a modálablak bezárul a closeModal() metódussal.

* **Bemenet és Kimenet:**

Bemenet:

searchQuery (string): A keresési kifejezés, amelyet a felhasználó beír a keresőmezőbe.

Kimenet:

A termékek szűrt listája, amely a filteredProducts változóban tárolódik.

Kosárba helyezett termékek, amelyeket a CartService kezel.

* **Metódusok:**

**searchProducts()**

Keresést végez a termékek között a felhasználó által megadott kifejezés alapján.

A találatok a filteredProducts változóban jelennek meg.

**addCart(product: any)**

A megadott terméket hozzáadja a kosárhoz a CartService segítségével.

**openPurchaseModal(product: any)**

Megnyitja a vásárlási modálablakot a kiválasztott termékkel.

**closeModal()**

Bezárja a vásárlási modálablakot.

**confirmPurchase()**

A kiválasztott terméket hozzáadja a kosárhoz és bezárja a modálablakot.

**ProductListComponent:**

* **Termékek lekérése:**

A komponens a ProductService segítségével lekéri az összes terméket, és a products és filteredProducts változókban tárolja őket.

* **Termék szerkesztése (editProduct metódus):**

A editProduct() metódus segítségével szerkeszthetjük a termékek adatait.

A termék frissítése a ProductService segítségével történik az updateProduct() metódussal.

* **Új termék hozzáadása (addProduct metódus):**

Az addProduct() metódus segítségével új terméket adhatunk hozzá a listához.

A termékek adatainak elküldése a ProductService addProduct() metódusán keresztül történik.

* **Termék törlése (deleteProduct metódus):**

A deleteProduct() metódus segítségével törölhetjük a terméket a listából.

A törlés a ProductService deleteProduct() metódusával történik.

* **Bemenet és Kimenet:**

Bemenet:

newProduct (object): Az új termék adatai, amelyet a felhasználó hozzáad a listához.

product (object): A szerkesztett vagy törölt termék, amelyet a felhasználó kiválasztott.

Kimenet:

products (array): Az összes termék, amely a backendről érkezett.

filteredProducts (array): A szűrt termékek, amelyeket a felhasználó keresési feltételeknek megfelelően megjeleníthet.

* **Metódusok:**

**loadProducts()**

A termékek betöltése a backendről a ProductService segítségével.

Az adatokat tárolja a products és filteredProducts változókban.

**editProduct(product: any)**

A kiválasztott terméket szerkeszti, az új adatokat a ProductService updateProduct() metódusával frissíti.

**addProduct()**

Új terméket ad hozzá a listához, az adatokat a ProductService addProduct() metódusán keresztül küldi el a backendnek.

**deleteProduct(product: any)**

Törli a kiválasztott terméket a listából, a törlés a ProductService deleteProduct() metódusával történik.

**NewComponent**

* **Album borítójának kiválasztása (onAlbumCoverSelected metódus):**

A felhasználó fájlt választhat az album borítójához.

A kiválasztott fájlt előnézetként jeleníti meg a komponens.

* **Zenei fájlok hozzáadása (onFileSelected metódus):**

A felhasználó fájlokat tölthet fel (maximum 15 fájl).

Az egyes fájlokat a megfelelő indexnél tárolja a fileInputs tömbben.

* **Új fájl mező hozzáadása (addFileInput metódus):**

A felhasználó új fájl mezőt adhat hozzá, ha még nincs 15 fájl.

A fileInputs tömb hossza növekszik, és az új mező üresen kerül hozzáadásra.

* **Fájl mező eltávolítása (removeFileInput metódus):**

A felhasználó eltávolíthat egy fájl mezőt, ha már több mint 3 mező van.

A fileInputs tömbben a megfelelő indexű mezőt törli.

* **Album hozzáadása a kosárhoz (addToCart metódus):**

Az album címét és borítóját, valamint a feltöltött fájlokat (melyek nem null értékek) kosárba helyezi.

Ha az album címe üres, figyelmeztetést küld.

* **Bemenet és Kimenet:**

Bemenet:

albumTitle (string): Az album címe, amelyet a felhasználó ad meg.

fileInputs (array): A feltöltött fájlok, melyek a fileInputs tömbben tárolódnak.

Kimenet:

Kosárba helyezett album (object): Az album címét, borítóját és a feltöltött fájlokat tartalmazó objektum.

* **Metódusok:**

**onAlbumCoverSelected(event)**

A felhasználó album borítóját választja ki. A kiválasztott fájl előnézete megjelenik.

**onFileSelected(event, index)**

A felhasználó zenei fájlokat választ ki. A kiválasztott fájlokat a megfelelő indexhez rendeljük.

**addFileInput()**

Új fájl mezőt ad hozzá, ha még nincs 15 fájl.

**removeFileInput(index)**

Eltávolít egy fájl mezőt, ha már több mint 3 mező van.

**addToCart()**

Az albumot és a fájlokat kosárba helyezi, ha az album címe érvényes.

**NavbarComponent**

* **Bejelentkezett felhasználó kezelése:**

A komponens lekéri, hogy a felhasználó be van-e jelentkezve az AuthService segítségével, és ennek megfelelően frissíti a isLoggedIn változót.

* **Kosár darabszámának kezelése:**

A kosárban lévő elemek számát a CartService-ből lekérve tárolja a itemCount változóban, amelyet a navigációs sávon megjelenít.

* **Nyelvváltás:**

A felhasználó átválthatja az alkalmazás nyelvét magyar (hu) és angol (en) között. A TranslateService használatával biztosítja a fordítást.

* **Kijelentkezés:**

A felhasználó kijelentkezését az AuthService signOut metódusával kezeli, és a felhasználó sikeres kijelentkezése után egy üzenetet logol a konzolra.

* **Bemenet és Kimenet:**

Bemenet:

Nincs közvetlen bemenete, mivel az adatok lekérése és kezelése a szolgáltatásokból történik (AuthService, CartService).

Kimenet:

isLoggedIn (boolean): A felhasználó be van jelentkezve vagy sem.

itemCount (number): Kosárban lévő termékek darabszáma.

currentLanguage (string): A jelenlegi nyelv (magyar vagy angol).

* **Metódusok:**

**ngOnInit()**

A komponens inicializálásakor lekéri a bejelentkezett felhasználó állapotát és a kosár tartalmát.

Beállítja az alapértelmezett nyelvet magyarra.

**switchLanguage()**

A nyelvet magyar és angol között váltja. Ha a jelenlegi nyelv hu, akkor angolra (en) vált, ha pedig angolra van állítva, visszaállítja magyarra.

**logout()**

Kijelentkezéskor hívja az AuthService signOut metódusát, és sikeres kijelentkezés után a konzolra logolja a sikeres műveletet.

**LoginComponent**

* **Google bejelentkezés:**

A googleAuth metódus segítségével a felhasználó Google fiókjával jelentkezhet be. Ha a bejelentkezés sikeres, akkor átirányítja a felhasználót az alkalmazás kezdőlapjára.

Hibakezelés is történik, amennyiben probléma merül fel a Google bejelentkezés során.

* **Email/jelszó alapú bejelentkezés:**

A signIn metódus használatával a felhasználó bejelentkezhet az email-jelszó párosával. Ha a bejelentkezés sikeres, a felhasználó átirányításra kerül a kezdőlapra.

Ha a bejelentkezés sikertelen, akkor a komponens egy hibaüzenetet jelenít meg.

* **Elfelejtett jelszó:**

Az forgotPassword metódus segítségével a felhasználó elfelejtett jelszót kérhet az email-címéhez. A komponens hívja az AuthService megfelelő metódusát a jelszó visszaállításához.

* **Bemenet és Kimenet:**

Bemenet:

email (string): A felhasználó email címe a bejelentkezéshez.

password (string): A felhasználó jelszava a bejelentkezéshez.

Kimenet:

emailPassError (boolean): Ha true, akkor hibát jelez a jelszóval való bejelentkezésnél.

emailPassMessage (string): A hibaüzenet, amely megjelenik, ha probléma történik a bejelentkezés során.

* **Metódusok:**

**googleAuth()**

A felhasználó Google fiókjával történő bejelentkezést kezeli.

Ha sikeres, a felhasználó átirányításra kerül a kezdőlapra.

**signIn()**

Email és jelszó alapú bejelentkezést végez.

Ha sikeres, átirányítja a felhasználót a kezdőlapra. Ha hiba történik, hibát jelenít meg.

**forgotPassword()**

* A felhasználó által megadott email címhez jelszó visszaállítást kér.

**HomeComponent**

* **Dia Show:**

ngAfterViewInit() metódus a komponens életciklusa után biztosítja a dia-show inicializálását. A diák közötti váltás történik a showNextSlide és showPrevSlide metódusokkal.

Az @ViewChildren dekorátor használatával az összes dia elem elérhető, és az index alapján vált a dia.

* **Felhasználói bejelentkezés és admin státusz kezelése:**

ngOnInit() metódusban figyeli, hogy be van-e jelentkezve a felhasználó és admin státuszú-e.

A userName a bejelentkezett felhasználó nevét tartalmazza, ha a felhasználó be van jelentkezve.

Az admin státusz a isAdmin változóban tárolódik, és ha a felhasználó admin, akkor különleges jogosultságok lehetnek elérhetők számára.

* **Bemenet és Kimenet:**

nincsenek bemenetek vagy kimenetek.

* **Metódusok**

Ebben azosztályban vannak metódusok, de az alapvető logika nem tartalmaz adatbevitelt vagy komplex műveleteket.

**BasketComponens**

* **Kosár betöltése (loadCart):**

A loadCart metódus segítségével a komponens betölti a kosár tartalmát a localStorage-ból és frissíti a termékek számát.

* **Kosár frissítése (updateLocalStorage):**

Az updateLocalStorage metódus biztosítja, hogy minden módosítás után a kosár tartalmát elmentsük a localStorage-ba.

* **Kosár darabszámának frissítése (updateItemCount):**

A updateItemCount metódus összesíti a kosárban lévő termékek mennyiségét és frissíti a megjelenített darabszámot.

* **Termék törlése a kosárból (deleteProduct):**

A deleteProduct metódus lehetővé teszi, hogy a felhasználó eltávolítson egy terméket a kosárból, ha az egy adott ID-val rendelkezik.

* **Fizetés pop-up megjelenítése (openPaymentPopup):**

Az openPaymentPopup metódus megjeleníti a fizetési pop-up-ot.

* **Fizetés pop-up bezárása (closePaymentPopup):**

A closePaymentPopup metódus bezárja a fizetési pop-up-ot.

* **Fizetés folyamatának indítása (proceedToPayment):**

A proceedToPayment metódus ellenőrzi, hogy a felhasználó kiválasztotta-e a szállítási módot, majd továbblép a fizetési részletek megadásához.

* **Fizetési adatok pop-up bezárása (closePaymentDetailsPopup):**

A closePaymentDetailsPopup metódus bezárja a fizetési részletek pop-up-ot.

* **Fizetés megerősítése (confirmPayment):**

A confirmPayment metódus ellenőrzi, hogy minden kötelező adat kitöltésre került-e, és ha igen, véglegesíti a rendelést.

* **Kosár ürítése (clearCart):**

A clearCart metódus teljesen üríti a kosarat, és eltávolítja a mentett adatokat a localStorage-ból.

* **Sikeres rendelés kezelés (closeSuccessPopup):**

A closeSuccessPopup metódus kezeli a sikeres rendelés utáni folyamatokat, mint a rendelés adatainak mentése és a kosár kiürítése.

* **Bemenet és Kimenet:**

Bemenet:

cart: A kosárban lévő termékek listája.

user: A bejelentkezett felhasználó adatai.

paymentMethod: A felhasználó által választott fizetési mód.

deliveryMethod: A felhasználó által választott szállítási mód.

cardNumber: A bankkártya száma.

cardExpiry: A bankkártya lejárati ideje.

cardCVC: A bankkártya CVC kódja.

shippingAddress: A szállítási cím (ha szükséges).

Kimenet:

itemCount: A kosárban lévő termékek mennyisége.

cart: A frissített kosár, amely tartalmazza az összes módosítást.

orderSuccessPopupVisible: Ha igaz, akkor a sikeres rendelés pop-up-ja látható.

* **Metódusok:**

**loadCart()**

Betölti a kosár tartalmát a localStorage-ból és frissíti a kosár termékeinek darabszámát.

**updateLocalStorage()**

A kosár frissítése a localStorage-ban.

**updateItemCount()**

A kosár darabszámának frissítése.

**deleteProduct(productId: number)**

Eltávolít egy terméket a kosárból a megadott productId alapján.

**openPaymentPopup()**

Megnyitja a fizetési pop-up-ot.

**closePaymentPopup()**

Bezárja a fizetési pop-up-ot.

**proceedToPayment()**

Ellenőrzi, hogy a szállítási módot kiválasztották-e, és ha igen, továbblép a fizetési részletekhez.

**closePaymentDetailsPopup()**

Bezárja a fizetési részletek pop-up-ot.

**confirmPayment()**

Ellenőrzi a fizetési adatokat, és ha minden rendben van, megerősíti a rendelést.

**clearCart()**

Kiüríti a kosarat és törli a localStorage-ból.

**closeSuccessPopup()**

Kezeli a sikeres rendelés utáni folyamatokat, mint a rendelés adatainak mentése és a kosár kiürítése.

**AccountComponent**

* **Felhasználói adatok lekérése:**

A felhasználó adatai (pl. email, név, telefonszám, megerősítés státusza stb.) betöltésre kerülnek a AuthService segítségével.

A isAdmin státusz is lekérdezésre kerül a felhasználó adatainál.

* **Rendelések lekérése:**

Az admin felhasználók számára az összes rendelés lekérdezésre kerül, míg nem admin felhasználók csak a saját rendeléseiket láthatják.

A rendeléseket az OrdersService segítségével kérdezzük le a felhasználó ID-ja alapján.

* **Jelszó módosítása:**

A changePassword metódus segítségével a felhasználó új jelszót kérhet. A jelszó módosítása egy prompttal történik.

* **Bemenet és Kimenet:**

Bemenet:

Nincs közvetlen bemenet az alapértelmezett működés során, mivel a felhasználói adatok és rendelések az autentikációval és a háttérrendszerrel (pl. Firebase) történnek.

Kimenet:

userData: A felhasználó adatai (email, név, telefonszám, email megerősítés státusza, fotó URL, stb.)

orders: A felhasználó rendelései (az adminok esetében az összes rendelés).

isAdmin: Boolean érték, amely jelzi, hogy a felhasználó admin jogosultsággal rendelkezik-e.

* **Metódusok:**

**ngOnInit()**

A komponens inicializálásakor lekérdezi a felhasználói adatokat, majd a felhasználó ID-jával betölti a rendeléseket. Admin felhasználók számára az összes rendelést, míg nem admin felhasználók számára csak a saját rendeléseiket jeleníti meg.

**loadOrders(userId: string)**

Betölti a felhasználó rendeléseit az OrdersService segítségével. Ha a felhasználó admin, akkor az összes rendelést lekéri, különben csak a saját rendeléseit.

**changePassword()**

A felhasználó új jelszót adhat meg, amit a rendszer módosít. Az új jelszót egy prompt segítségével kérdezi le.

**AboutComponent:**

Ennek a komponensnek nincsen funkcionalitása.

# Tesztelés

* **API Tesztelés Swagger segítségével**

Az API működését Swagger UI segítségével teszteltem. A Swagger automatikusan generált dokumentációt biztosít az API végpontjairól és azok paramétereiről, valamint lehetőséget biztosít azok interaktív tesztelésére.

* **Tesztelés folyamata:**

API Dokumentáció: A Swagger UI biztosítja az API teljes dokumentációját, amely tartalmazza az összes elérhető végpontot, azok metódusait (GET, POST, PUT, DELETE) és a hozzájuk tartozó paramétereket.

* **Interaktív Tesztelés:**

Az API végpontok tesztelése a Swagger UI-ban történt. Itt különböző HTTP kéréseket küldtem az API felé, például a GET /products lekérést a termékek listázására, és a POST /products végpontot új termék hozzáadásához.

* **Válaszok Validálása:**

A Swagger automatikusan validálja a válaszokat az API specifikációja szerint. Minden végpont válasza helyesen tartalmazza a megfelelő státuszkódokat és adatokat, biztosítva, hogy az API a várt eredményeket adja vissza.

* **Hibakezelés:**

A hibás kérések, például a hiányzó vagy érvénytelen paraméterek tesztelésekor az API megfelelő hibakódot (pl. 400) és részletes hibaüzenetet adott vissza.

* **Tesztelés Eredménye:**

Minden végpont sikeresen működött a tesztelés során, és az API válaszai megfeleltek a dokumentációban leírtaknak. Az összes tesztelt végpont helyesen kezeli a kéréseket, és a válaszok a megfelelő formátumban érkeznek.

Példa:

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

* **Frontend tesztelése**

A frontend működésének ellenőrzése manuálisan történt, ahol az alkalmazás összes fő funkcióját teszteltem a böngészőben.

* **Oldalak betöltése**

Az alkalmazás minden oldala megfelelően betöltődik.

A navigációs menü helyesen működik, és minden link a megfelelő oldalra irányít.

* **Felhasználói hitelesítés**

A regisztráció és bejelentkezés sikeresen működik.

A Google-fiókkal való bejelentkezés is megfelelően végrehajtható.

A kijelentkezés után a felhasználó visszakerül a kezdőlapra.

* **Termékek és kosár kezelése**

A termékek helyesen jelennek meg a főoldalon.

A kosárba helyezés és a kosár megtekintése megfelelően működik.

A kosár tartalmának módosítása (termék törlése, mennyiség változtatása) megfelelően frissíti az adatokat.

* **Rendelés leadása**

A rendelési folyamat végigvihető hiba nélkül.

A szállítási és fizetési adatok megadása után a rendelés sikeresen rögzítésre kerül.

A képen szöveg, képernyőkép, Betűtípus, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

* **Admin funkciók (ha van)**

Az admin felület elérhető, és csak admin jogosultsággal rendelkező felhasználók férhetnek hozzá.

Az admin megfelelően tud új terméket hozzáadni és meglévőt szerkeszteni és törölni.

A képen szöveg, képernyőkép, Betűtípus, sor látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

* **Reszponzivitás**

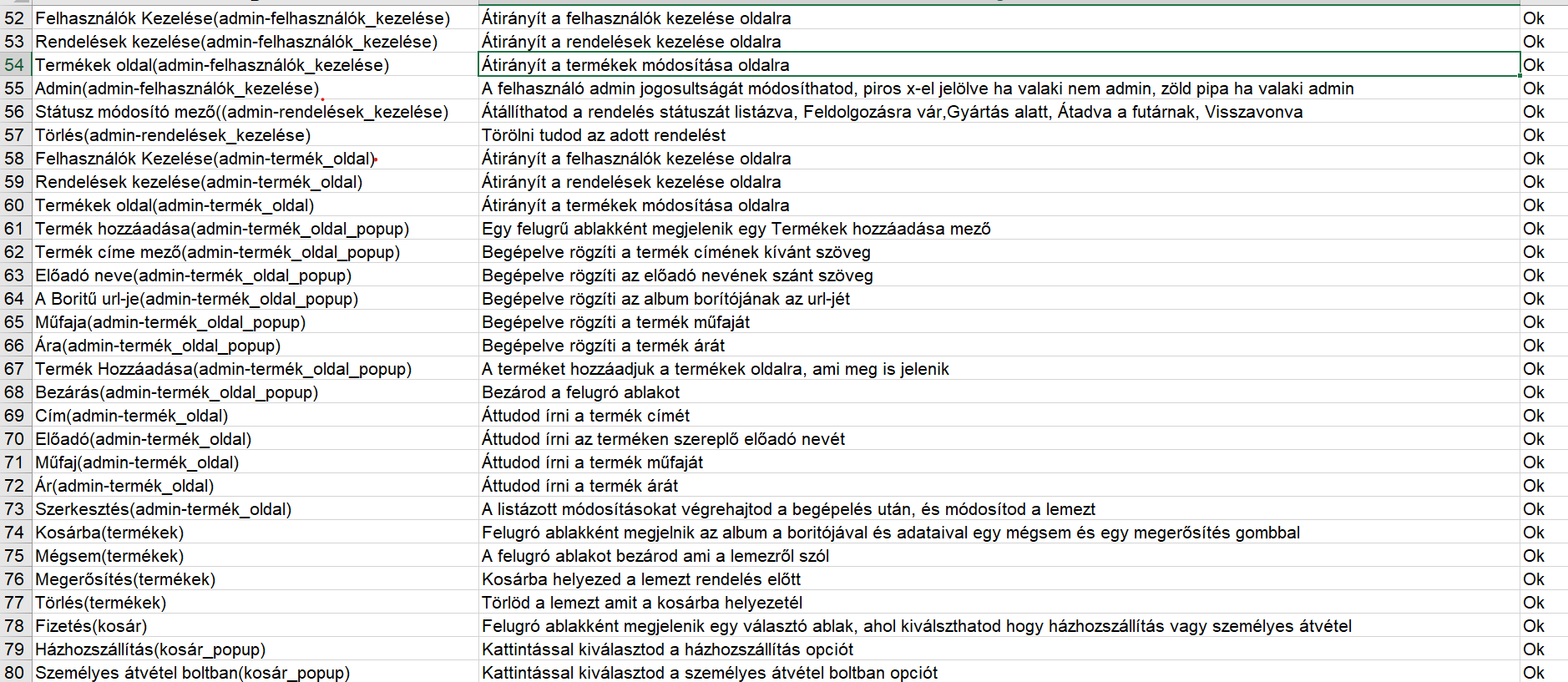
Az alkalmazás mobilon és asztali böngészőben is megfelelően működik.

Az elemek helyesen igazodnak kisebb képernyőkön.

A képen szöveg, képernyőkép, tervezés látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

* **Tesztelés eredménye:**

A manuális tesztelés során az alkalmazás minden fő funkciója megfelelően működött. Az oldal betöltődése gyors, a felhasználói interakciók helyesen működnek, és nincs észlelhető kritikus hiba.

# Ismert hibák

* **A saját lemez hozzáadása funkció nem működik teljesen**  
  Leírás: A felhasználók nem tudnak zenét hozzáadni a saját lemezükhöz, így az oldal ezen funkciója nem működik megfelelően.  
  Lehetséges megoldás: Ellenőrizni kell, hogy a zene hozzáadása megfelelően kommunikál-e a backenddel, illetve szükség lehet az adatmodell vagy az API végpontok frissítésére.

# Fejlesztési lehetőségek

* **Zene hozzáadása a saját lemezhez**  
  Leírás: Jelenleg a felhasználók nem tudnak zenét hozzáadni a saját lemezeikhez. Egy fejlesztési lehetőség az lenne, hogy ezt a funkciót teljes mértékben implementáljuk.  
  Lehetséges megoldás: Ellenőrizni kell, hogy a frontend megfelelően küldi-e az adatokat a backend felé, és szükség esetén módosítani kell az API-t és az adatbázist.

# Fejlesztők

* Nagy Gergő – Frontend fejlesztés, Angular integráció
* Fülöp Ervin – Backend fejlesztés, API implementáció, adatbázis kezelés, Firebase kezelés, dokumentáció írása
* Juhász Zsombor – Oldal forditása, Oldal tesztelése, A tesztelés megírása