**Backend dokumentáció:**

* **db.js – Adatbázis kapcsolat konfiguráció**

Ez a fájl felelős a MySQL adatbázishoz való kapcsolódásért, amely a **BookMyTime** alkalmazás backend működésének alapját biztosítja. A fájl létrehozza az adatbázis kapcsolatot egy connection pool segítségével, így egyszerre több lekérdezés is hatékonyan kiszolgálható.

**Fő funkciók:**

* Környezeti változók betöltése (dotenv segítségével)
* MySQL kapcsolat létrehozása pool-on keresztül
* Kapcsolódási teszt lefuttatása indításkor
* Hiba esetén kilépés a folyamatból

**Részletes magyarázat**

* **mysql2/promise**: A mysql2 csomag ígéret-alapú (Promise) verzióját használjuk, amely modern async/await szintaxissal való együttműködést tesz lehetővé.
* **dotenv**: A .env fájlban tárolt környezeti változókat tölti be, így érzékeny adatok (pl. jelszavak) nem kerülnek közvetlenül a kódbázisba.
* **mysql.createPool({...})**: Létrehoz egy ún. "kapcsolat poolt", amely lehetővé teszi több adatbázis kapcsolat újrafelhasználását a backend kiszolgálásához.
* **Kapcsolati teszt**: A getConnection() metódus teszteli, hogy az adatbázis kapcsolat sikeresen létrejön-e. Siker esetén a kapcsolatot visszaadjuk a pool-nak (connection.release()), hiba esetén pedig a folyamat kilép egy hibaüzenettel.

**Exportálás**

A db változó module.exports segítségével elérhető a projekt más fájljaiból is. Ezáltal bárhol használhatjuk az adatbázis kapcsolatot egyszerűen



**Példa felhasználásra:**



* **server.js - A szerverindító fájl:**

A server.js a **BookMyTime** backend alkalmazásának belépési pontja. Ez a fájl indítja el az Express.js alapú HTTP szervert, konfigurálja az útvonalakat (route-okat), engedélyezi a CORS használatát, betölti a környezeti változókat, és beállítja az alapvető middleware-eket és hibafigyelést.

**Fő funkciók**

* Express szerver inicializálása
* JSON-kérések kezelése
* CORS engedélyezése
* Útvonalak csatolása különféle funkciókhoz (pl. hitelesítés, foglalások, vállalkozások)
* Hibafigyelés kezelése
* Szerver elindítása

**Részletes magyarázat**

* **express()**: Az Express keretrendszer példányosítása, amely lehetővé teszi HTTP kérések kezelését.
* **dotenv.config()**: A .env fájlban megadott környezeti változók betöltése (pl. adatbázis jelszó, email beállítások).
* **app.use(express.json())**: Lehetővé teszi a JSON típusú kérések feldolgozását.
* **app.use(cors())**: Engedélyezi a cross-origin (pl. frontend és backend különböző porton futása esetén) kéréseket.
* **Route-ok regisztrálása**:
* /api/auth – Felhasználói hitelesítési útvonalak
* /api/businesses – Vállalkozásokkal kapcsolatos műveletek
* /api/foglalasok – Foglalások kezelése
* /api/felhasznalo – Felhasználói műveletek
* /api/teszt – Tesztelési célú route-ok (pl. fejlesztéshez)
* **Gyökérút (/)**: Visszaigazolást küld, hogy a szerver fut.
* **Hibakezelő middleware**: Globálisan elfogja a hibákat és szabványos válaszüzenetet küld vissza.
* **Szerver indítása**: A szerver a localhost:5000 címen indul el, és erről konzolüzenet tájékoztat.
* authController.js – **autentikációs fájl**

**Fájl célja:**  
A authController.js fájl az **autentikációs (hitelesítési)** műveleteket kezeli: bejelentkezés, felhasználói regisztráció, vállalkozói regisztráció.

Fő funkciók:

* **Bejelentkezés (login)**
* **Felhasználó regisztrálása (registerUser)**
* **Vállalkozás regisztrálása (registerBusiness)**

Importált modulok:

* jsonwebtoken – JWT token generálás a bejelentkezéshez.
* bcrypt – Jelszavak titkosítása és ellenőrzése.
* db – Adatbázis kapcsolat a lekérdezésekhez.

login(req, res)

**Cél:** Bejelentkezés felhasználóként vagy vállalkozóként.

* Ellenőrzi, hogy az email és jelszó meg van-e adva.
* Lekérdezi az emailhez tartozó felhasználót (felhasznalo vagy vallalkozo).
* Összehasonlítja a jelszót a hash-elt jelszóval.
* Ha sikeres, JWT tokent generál 1 órás lejárattal.
* Válasz tartalmazza a tokent, lejárati időt, azonosítót és típust.

registerUser(req, res)

**Cél:** Új felhasználó vagy vállalkozó regisztrálása.

* Ellenőrzi, hogy minden mező ki van-e töltve.
* Érvényesíti, hogy a tipus mező "felhasznalo" vagy "vallalkozo".
* Megnézi, hogy az e-mail már regisztrálva van-e.
* Titkosítja a jelszót bcrypt-tel.
* Beszúrja az új felhasználót az adatbázisba.
* Sikeres regisztráció esetén visszaadja az új ID-t és visszajelzést.

registerBusiness(req, res)

**Cél:** Új vállalkozás regisztrálása, statisztikai rekord létrehozása.

* Ellenőrzi a kötelező mezőket.
* Megnézi, hogy az e-mail már regisztrálva van-e vállalkozóként.
* Titkosítja a jelszót.
* Új rekordot hoz létre a vallalkozo táblában.
* Létrehoz egy kapcsolódó statisztikai rekordot a statisztika táblában.
* Visszaküldi az új vállalkozói ID-t és sikert jelző üzenetet.

**4. authMiddleware.js**

**1. Fájl szerepe:**

Az authMiddleware.js fájl egy middleware függvényt tartalmaz, amelyet a felhasználói hitelesítés (authentikáció) céljából hoztak létre. Ez a middleware funkció lehetővé tenné a JWT (JSON Web Token) tokenek ellenőrzését, azaz hogy egy adott kérés jogosult felhasználótól származik-e.

**2. Technológiák és könyvtárak:**

* jsonwebtoken: A JWT tokenek létrehozására és ellenőrzésére szolgáló könyvtár.
* SECRET\_KEY: A JWT tokenek aláírásához használt titkos kulcs.

**3. Változók:**

|  |  |  |
| --- | --- | --- |
| **Változó neve** | **Típus** | **Leírás** |
| SECRET\_KEY | string | A JWT tokenek dekódolásához használt kulcs (jelenleg: 'titkoskulcs'). |

**4. Függvények:**

🔹 authenticateToken(req, res, next)

* **Leírás:** Middleware függvény, amely a kérés során kapott Authorization fejlécből ellenőrizné a JWT tokent.
* **Paraméterek:**
* req: A beérkező kérés objektum.
* res: A válasz objektum.
* next: A következő middleware hívásához szükséges függvény.
* **Jelenlegi állapot:** A függvény jelenleg üres, automatikusan hívja a next() függvényt, így **nem végez semmiféle ellenőrzést**.
* **Jövőbeli bővítés:** A kód jövőbeli kiegészítésre vár, hogy ténylegesen dekódolja és ellenőrizze a JWT tokent.

**5. Export:**

* A module.exports segítségével a authenticateToken függvény elérhető más fájlok számára (pl. route-oknál, ahol védeni kell egyes útvonalakat hitelesítés alapján).

**5. authRoutes.js**

**1. Fájl szerepe:**

Az authRoutes.js fájl a **Felhasználói és Vállalkozói hitelesítési és regisztrációs útvonalakat** tartalmazza. Ez a fájl biztosítja a kapcsolatot a frontend és a backend között, hogy a felhasználók és vállalkozók regisztrálhassanak, illetve bejelentkezhessenek az alkalmazásba.

**2. Technológiák és könyvtárak:**

* express: A Node.js alapú webkeretrendszer, amely a RESTful API útvonalakat biztosítja.
* dotenv: Környezeti változók betöltésére szolgáló könyvtár.
* authController: A regisztrációval és bejelentkezéssel kapcsolatos logikát kezelő kontroller.

**3. Függvények és útvonalak:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Módszer** | **Útvonal** | **Leírás** | **Controller/Függvény** |
| POST | /register | Regisztráció felhasználóként. Ellenőrzi a kért adatokat és regisztrálja a felhasználót. | registerUser |
| POST | /register-vallalkozo | Regisztráció vállalkozóként. Az adatok ellenőrzése után regisztrálja a vállalkozót. | registerBusiness |
| POST | /login | Felhasználói bejelentkezés. Ellenőrzi az adatokat, majd visszaadja a JWT tokent. | login |

**4. Middleware:**

A fájl jelenleg nem tartalmaz speciális middleware funkciókat, de később bővíthető hitelesítési ellenőrzéssel, pl. a authenticateToken middleware használatával.

**5. Függvények a controllerből:**

* **registerUser:** Felhasználó regisztrálása. A megadott adatokat (név, email, jelszó, telefon, tipus) validálja, és a megfelelő táblába (felhasznalo) beszúrja az új felhasználót.
* **registerBusiness:** Vállalkozó regisztrálása. A megadott adatokat (név, email, jelszó, telefon, adószám) validálja, és a megfelelő táblába (vallalkozo) beszúrja az új vállalkozót.
* **login:** Bejelentkezés. Ellenőrzi a felhasználó email-jét és jelszavát, majd ha helyesek az adatok, visszaad egy JWT tokent, amelyet a frontend használhat a további hitelesítési folyamatokhoz.

**6. Környezeti változók:**

* **dotenv.config()**: Betölti a környezeti változókat, például az adatbázis kapcsolatot vagy más érzékeny adatokat.

**7. Export:**

* A router objektumot exportálja, amely az alkalmazás más részein, például a server.js fájlban használható az autentikációs útvonalak regisztrálásához.

**6. bookingRoutes.js**

**Leírás:**

A bookingRoutes.js fájl az időpontfoglalási rendszert kezelő API végpontokat tartalmazza. Ezek a végpontok lehetővé teszik a szolgáltatások, vállalkozások és kategóriák lekérdezését, valamint a szabad időpontok megjelenítését és foglalását. Ezen kívül biztosítja a foglalások rögzítését és az időpontok státuszának frissítését is.

**Használt modulok:**

* **express**: A Express.js keretrendszert használjuk a HTTP végpontok kezelésére.
* **db**: Az adatbázis kapcsolathoz szükséges konfigurációt tartalmazza.
* **emailjs-com**: Email küldéséhez használt könyvtár, bár jelenleg nincs használva a kódban.

**Végpontok:**

1. GET /szolgaltatasok

* **Leírás**: A végpont szolgáltatásokat kér le a rendszerből vállalkozás ID vagy kategória alapján.
* **Query paraméterek**:
* vallalkozas\_id: A vállalkozás ID-ja.
* category: A szolgáltatás kategóriája.
* **Működés**: A végpont a megadott paraméterek alapján szűri a szolgáltatásokat. Ha egyik paraméter sincs megadva, akkor az összes szolgáltatás vissza lesz adva.
* **Válasz**:
* 200 OK: A szolgáltatások listája JSON formátumban.
* 500: Hiba esetén.

2. GET /vallalkozasok

* **Leírás**: A végpont visszaadja a vállalkozásokat kategória alapján.
* **Query paraméterek**:
* category: A vállalkozások kategóriája.
* **Működés**: Ha a category paraméter meg van adva, akkor csak az abba a kategóriába tartozó vállalkozások kerülnek visszaadásra.
* **Válasz**:
* 200 OK: A vállalkozások listája JSON formátumban.
* 500: Hiba esetén.

3. GET /business-categories

* **Leírás**: A végpont lekéri az összes elérhető vállalkozás kategóriát.
* **Működés**: A végpont az összes egyedi kategóriát adja vissza, amelyek a vallalkozas táblában szerepelnek.
* **Válasz**:
* 200 OK: A kategóriák listája JSON formátumban.
* 500: Hiba esetén.

4. GET /szabad-idopontok/:szolgaltatas\_id

* **Leírás**: A végpont a megadott szolgáltatás ID alapján lekéri a szabad időpontokat.
* **Paraméterek**:
* szolgaltatas\_id: A szolgáltatás ID-ja.
* **Működés**: Az időpontok státuszát ellenőrzi, és csak a szabad időpontokat adja vissza.
* **Válasz**:
* 200 OK: A szabad időpontok listája JSON formátumban.
* 500: Hiba esetén.

5. POST /foglalas

* **Leírás**: A végpont lehetővé teszi egy új foglalás létrehozását.
* **Kérelem adatai**:
* szolgaltatas\_id: A foglalni kívánt szolgáltatás ID-ja.
* ido\_id: A foglalni kívánt időpont ID-ja.
* felhasznalo\_id: A felhasználó ID-ja (ha nem vállalkozó).
* vallalkozas\_id: A vállalkozás ID-ja (ha nem felhasználó).
* foglalo\_tipus: A foglaló típusa (felhasznalo vagy vallalkozas).
* email: A felhasználó email címe.
* **Működés**: A végpont először ellenőrzi, hogy a foglaló nem próbálja-e lefoglalni saját vállalkozásához az időpontot. Ezután ellenőrzi, hogy az időpont szabad-e. Ha mindkét feltétel teljesül, létrejön a foglalás, és az időpont státusza frissítésre kerül.
* **Válasz**:
* 201 Created: A foglalás sikeresen rögzítve.
* 400 Bad Request: Hiba esetén (pl. már foglalt időpont).
* 500: Hiba esetén.

6. GET /szolgaltatasok/:category

* **Leírás**: A végpont a szolgáltatásokat kategória alapján kérdezi le.
* **Paraméterek**:
* category: A keresett szolgáltatás kategória.
* **Működés**: A végpont a megadott kategóriában található szolgáltatásokat adja vissza.
* **Válasz**:
* 200 OK: A szolgáltatások listája JSON formátumban.
* 500: Hiba esetén.

**7. businessRoutes.js - Vállalkozásokkal kapcsolatos API végpontok**

Ez a fájl az Express alkalmazásban található, és a vállalkozások kezelésére szolgáló API végpontokat tartalmazza. Az alábbiakban részletesen bemutatjuk az egyes végpontokat és azok funkcióit.

**1. GET /vallalkozasok**

**Leírás:**  
Ez a végpont a vállalkozások listáját adja vissza a megadott vallalkozo\_id alapján.

**Kérési paraméterek:**

* vallalkozo\_id (query paraméter) - A vállalkozó azonosítója, amely alapján a vállalkozások szűrésre kerülnek.

**Válasz:**

* **Sikeres válasz (200):** A lekérdezett vállalkozások listája.
* **Hibás válasz (400):** Hiányzó vállalkozó ID.

**2. POST /vallalkozasokHozzaadasa**

**Leírás:**  
Új vállalkozás hozzáadása a rendszerhez.

**Kérés törzse:**

* vallalkozas\_neve (string) - A vállalkozás neve.
* helyszin (string) - A vállalkozás helyszíne.
* nyitva\_tartas (string) - A nyitva tartás.
* category (string) - A vállalkozás kategóriája.
* vallalkozo\_id (int) - A vállalkozó ID-je.

**Válasz:**

* **Sikeres válasz (201):** A hozzáadott vállalkozás adatai.
* **Hibás válasz (400):** Hiányzó szükséges adat.

**3. GET /vallalkozasok/:id/details**

**Leírás:**  
A megadott id alapján egy adott vállalkozás részletes információit adja vissza.

**URL paraméterek:**

* id (paraméter) - A vállalkozás ID-je.

**Válasz:**

* **Sikeres válasz (200):** A vállalkozás részletes adatai.
* **Hibás válasz (400):** Hiányzó vállalkozás ID.
* **Hibás válasz (404):** Vállalkozás nem található.

**4. PUT /vallalkozasok/:id/modositas**

**Leírás:**  
A megadott ID alapján módosítja a vállalkozás helyszínét, nyitva tartását és kategóriáját.

**URL paraméterek:**

* id (paraméter) - A vállalkozás ID-je.

**Kérés törzse:**

* helyszin (string) - Az új helyszín.
* nyitva\_tartas (string) - Az új nyitva tartás.
* category (string) - Az új kategória.

**Válasz:**

* **Sikeres válasz (200):** A frissített vállalkozás adatai.
* **Hibás válasz (400):** Hiányzó szükséges adat.
* **Hibás válasz (404):** Vállalkozás nem található.

**5. POST /vallalkozasok/:id/szolgaltatasok**

**Leírás:**  
Szolgáltatás hozzáadása a vállalkozásnak.

**URL paraméterek:**

* id (paraméter) - A vállalkozás ID-ja.

**Kérés törzse:**

* szolgaltatas\_neve (string) - A szolgáltatás neve.
* idotartam (string) - A szolgáltatás időtartama.
* ar (int) - A szolgáltatás ára.

**Válasz:**

* **Sikeres válasz (201):** Az új szolgáltatás adatai.
* **Hibás válasz (400):** Hiányzó szükséges adat.

**6. GET /vallalkozasok/:id/szolgaltatasok**

**Leírás:**  
A vállalkozás összes szolgáltatásának lekérdezése.

**URL paraméterek:**

* id (paraméter) - A vállalkozás ID-ja.

**Válasz:**

* **Sikeres válasz (200):** A vállalkozás összes szolgáltatásának listája.
* **Hibás válasz (500):** Adatbázis hiba.

**7. POST /delete/:id**

**Leírás:**  
Vállalkozás törlésének végrehajtása.

**URL paraméterek:**

* id (paraméter) - A törlendő vállalkozás ID-ja.

**Válasz:**

* **Sikeres válasz (200):** A vállalkozás sikeres törlése.
* **Hibás válasz (404):** A vállalkozás nem található.

**8. POST /:id/add-idopont**

**Leírás:**  
Új időpont hozzáadása a vállalkozás szolgáltatásaihoz.

**URL paraméterek:**

* id (paraméter) - A vállalkozás ID-ja.

**Kérés törzse:**

* szabad\_ido (string) - Az új időpont (formátum: YYYY-MM-DD HH:mm:ss).
* szolgaltatas\_id (int) - A szolgáltatás ID-ja, amelyhez az időpont tartozik.

**Válasz:**

* **Sikeres válasz (200):** Az új időpont hozzáadása.
* **Hibás válasz (400):** Minden mezőt ki kell tölteni.
* **Hibás válasz (500):** Hiba történt az időpont hozzáadása közben.

**9. GET /vallalkozo-profile**

**Leírás:**  
A vállalkozó profiljának lekérése email alapján.

**Kérési paraméterek:**

* email (query paraméter) - A vállalkozó email címe.

**Válasz:**

* **Sikeres válasz (200):** A vállalkozó profiladatai.
* **Hibás válasz (400):** Hiányzó email paraméter.
* **Hibás válasz (404):** A vállalkozó nem található.

**10. POST /update-bio**

**Leírás:**  
A vállalkozó bio adatainak frissítése.

**Kérés törzse:**

* email (string) - A vállalkozó email címe.
* bio (string) - Az új bio.

**Válasz:**

* **Sikeres válasz (200):** A bio sikeres frissítése.
* **Hibás válasz (400):** Hiányzó email vagy bio.
* **Hibás válasz (500):** Hiba történt a bio frissítése közben.

**11. POST /update-user**

**Leírás:**  
A vállalkozó személyes adatainak (név, cím) frissítése.

**Kérés törzse:**

* email (string) - A vállalkozó email címe.
* nev (string) - A vállalkozó új neve.
* cim (string) - Az új cím.

**Válasz:**

* **Sikeres válasz (200):** A személyes adatok sikeres frissítése.
* **Hibás válasz (400):** Hiányzó adat.
* **Hibás válasz (500):** Hiba történt a személyes adatok frissítése közben.

**8. UserRoutes.js**

A UserRoutes.js fájl az alkalmazás része, amely a felhasználói adatokat kezeli. Az API végpontok a felhasználói profil lekérdezését és jelszó módosítását támogatják. Az alkalmazás a Node.js környezetben működik, és az Express.js keretrendszert használja. Az adatokat MySQL adatbázisból kérdezi le és módosítja.

1. Cél és Funkció

A UserRoutes.js célja két fő feladat ellátása:

* A felhasználó profiljának lekérése az azonosítója alapján.
* A felhasználó jelszavának módosítása a jelenlegi jelszó és az új jelszó megadásával.

Fő funkciók:

* **Profil lekérdezése**: A felhasználó profilját (név, email, telefonszám) adja vissza, ha a felhasználó létezik az adatbázisban.
* **Jelszó módosítása**: A felhasználó jelszavát megváltoztathatja, ha megadja a jelenlegi jelszót és az új jelszót, amelyet biztonságosan hash-elünk és frissítünk az adatbázisban.

2. Felhasznált Technológiák

* **Node.js**: JavaScript futtatókörnyezet.
* **Express.js**: Web keretrendszer, amely az HTTP végpontok kezelését biztosítja.
* **bcryptjs**: Jelszavak biztonságos hash-elésére és ellenőrzésére szolgáló könyvtár.
* **MySQL**: Relációs adatbázis, amely tárolja a felhasználói adatokat.

3. API Végpontok

3.1 GET /profil/:id

**Leírás**

Ez a végpont lehetővé teszi a felhasználó profiljának lekérdezését az ID alapján. A profil tartalmazza a felhasználó nevét, email címét és telefonszámát.

**Paraméterek:**

* **Felhasználói ID** (URL paraméter): A felhasználó egyedi azonosítója (felhasznalo\_id), amely az URL-ben szerepel.

3.2 POST /jelszo-valtoztatas/:id

**Leírás**

Ez a végpont lehetővé teszi a felhasználó jelszavának módosítását. A felhasználónak meg kell adnia a jelenlegi jelszót és az új jelszót.

**Paraméterek:**

* **Felhasználói ID** (URL paraméter): A felhasználó egyedi azonosítója (felhasznalo\_id), amely az URL-ben szerepel.
* **Törzs (Body)**:
* **jelszo** (string): A felhasználó jelenlegi jelszava.
* **ujJelszo** (string): Az új jelszó.

4. Funkcionális Leírás

4.1 Felhasználói profil lekérése

* A kérés megérkezése után a szerver az URL-ből kinyeri a felhasználó ID-ját.
* Az adatbázisból lekérdezi a felhasználó adatait (név, email, telefonszám) a felhasználó ID-ja alapján.
* Ha a felhasználó megtalálható, visszaadja az adatokat JSON formátumban.
* Ha a felhasználó nem található, a szerver 404-es hibát ad vissza.
* Ha a kérés érvénytelen vagy az ID hiányzik, a szerver 400-as hibát ad vissza.

4.2 Jelszó módosítása

* A kérés megérkezése után a szerver az URL-ből kinyeri a felhasználó ID-ját, és a kérés törzséből a jelenlegi és az új jelszót.
* Az adatbázisból lekérdezi a felhasználó jelenlegi jelszavát.
* A jelenlegi jelszót a bcryptjs könyvtár segítségével összehasonlítja a megadott jelszóval.
* Ha a jelenlegi jelszó helyes, az új jelszót biztonságosan hash-eli és frissíti az adatbázisban.
* Ha a jelenlegi jelszó helytelen, a szerver 401-es hibát ad vissza.
* Ha a felhasználó nem található, a szerver 404-es hibát ad vissza.
* A sikeres jelszó módosítás után a rendszer visszaadja a megfelelő üzenetet.

5. Biztonsági Szempontok

* A jelszó módosítása során minden jelszót a bcryptjs segítségével hash-elünk, hogy megakadályozzuk a jelszavak nyílt tárolását.
* A szerver csak akkor engedélyezi a jelszó módosítását, ha a megadott jelenlegi jelszó helyes.
* A felhasználói adatok biztonságos lekérdezéséhez az adatbázis lekérdezés paraméterezett lekérdezéseket használ, hogy megelőzze az SQL injection támadásokat.

**9. Backend tesztek:**

* **bookingRoutes.js tesztek:**

**1. Szolgáltatások lekérdezése (GET /api/booking/szolgaltatasok)**

Leírás: Lekérdezi az elérhető szolgáltatásokat vállalkozás ID vagy kategória alapján.

Kérés:  
GET <http://localhost:5000/api/booking/szolgaltatasok?vallalkozas_id=1>

Sikeres válasz:

[  
 {  
 "id": 1,  
 "nev": "Hajvágás",  
 "ar": 5000,  
 "ido": 30,  
 "vallalkozas\_id": 1,  
 "vallalkozas\_neve": "Szépségszalon Anna"  
 }  
]

**2. Vállalkozások listázása (GET /api/booking/vallalkozasok)**

Leírás: Visszaadja az összes vagy szűrt (pl. kategória alapján) vállalkozást.

Kérés:  
GET <http://localhost:5000/api/booking/vallalkozasok?category=Fodrászat>

Sikeres válasz:

[  
 {  
 "id": 1,  
 "vallalkozas\_neve": "Szépségszalon Anna",  
 "category": "Fodrászat",  
 "telefonszam": "+36201234567",  
 "cim": "Budapest, Fő utca 1."  
 }  
]

**3. Kategóriák lekérdezése (GET /api/booking/business-categories)**

Leírás: Az adatbázisban szereplő egyedi kategóriák lekérése.

Kérés:  
GET <http://localhost:5000/api/booking/business-categories>

Sikeres válasz:

[  
 { "category": "Fodrászat" },  
 { "category": "Masszázs" },  
 { "category": "Fogászat" }  
]

**4. Szabad időpontok lekérdezése (GET /api/booking/szabad-idopontok/:szolgaltatas\_id)**

Leírás: Egy adott szolgáltatás szabad időpontjainak lekérdezése.

Kérés:  
GET <http://localhost:5000/api/booking/szabad-idopontok/1>

Sikeres válasz:

[  
 {  
 "ido\_id": 12,  
 "szabad\_ido": "2025-05-10 10:30",  
 "statusz": "szabad"  
 },  
 {  
 "ido\_id": 13,  
 "szabad\_ido": "2025-05-10 11:00",  
 "statusz": "szabad"  
 }  
]

**5. Foglalás létrehozása (POST /api/booking/foglalas)**

Leírás: Egy felhasználó időpontot foglal egy szolgáltatásra.

Kérés:  
POST <http://localhost:5000/api/booking/foglalas>  
Content-Type: application/json

Törzs:

{  
 "szolgaltatas\_id": 1,  
 "ido\_id": 12,  
 "felhasznalo\_id": 5,  
 "vallalkozas\_id": 1,  
 "foglalo\_tipus": "ugyfel",  
 "email": "teszt@pelda.hu"  
}

Sikeres válasz:

{  
 "message": "Foglalás sikeresen rögzítve"  
}

Hibás válasz példa (időpont nem elérhető):

{  
 "message": "Az időpont már nem elérhető."  
}

* **businessRoutes tesztek:**

**GET /vallalkozasok?vallalkozo\_id={id}**

Leírás: Lekérdezi az adott vállalkozóhoz tartozó vállalkozásokat.

Válaszkód: 200 OK

Válasz:

[  
 {  
 "id": 1,  
 "vallalkozas\_neve": "Teszt Bt.",  
 "helyszin": "Budapest",  
 "nyitva\_tartas": "08:00-16:00",  
 "category": "fodrászat",  
 "vallalkozo\_id": 2  
 }  
]

**POST /vallalkozasokHozzaadasa**

Leírás: Új vállalkozás hozzáadása.

Várható válaszkód: 201 Created

Válasz példa:

{  
 "id": 3,  
 "vallalkozas\_neve": "Új Cég",  
 "helyszin": "Szeged",  
 "nyitva\_tartas": "10:00-18:00",  
 "category": "kozmetika",  
 "vallalkozo\_id": 1  
}

**GET /vallalkozasok/{id}/details**

Leírás: Lekérdezi a megadott vállalkozás részletes adatait.

Válaszkód: 200 OK

Válasz:

{  
 "id": 3,  
 "vallalkozas\_neve": "Új Cég",  
 "helyszin": "Szeged",  
 "nyitva\_tartas": "10:00-18:00",  
 "category": "kozmetika",  
 "vallalkozo\_id": 1  
}

**PUT /vallalkozasok/{id}/modositas**

Leírás: Frissíti egy vállalkozás adatait.

Válaszkód: 200 OK

Válasz:

{  
 "id": 3,  
 "helyszin": "Győr",  
 "nyitva\_tartas": "09:00-17:00",  
 "category": "masszázs"  
}

**POST /vallalkozasok/{id}/szolgaltatasok**

Leírás: Szolgáltatás hozzáadása a megadott vállalkozáshoz.

Válaszkód: 201 Created

Válasz:

{  
 "id": 5,  
 "szolgaltatas\_neve": "Hajvágás",  
 "idotartam": 30,  
 "ar": 5000,  
 "vallalkozas\_id": 3  
}

* **UserRoutes tesztek:**

**1. GET /profil/:id**

Egy adott felhasználó adatainak lekérése az `id` alapján.

Paraméterek (URL):  
 - :id – felhasználó azonosítója (kötelező)

***Példa kérés:***

GET /profil/1

***Válaszok:***

* Sikeres lekérés:

***{***

***'nev': 'Kiss Péter',***

***'email': 'kiss.peter@example.com',***

***'telefonszam': '+36123456789'***

***}***

* Hiányzó ID:

***{'error': 'Hiányzó ID'}***

* Felhasználó nem található:

***{'error': 'Nincs ilyen felhasználó'}***

**2. POST /profil/:id**

Egy adott felhasználó adatainak frissítése.

Paraméterek (URL):  
 - :id – felhasználó azonosítója (kötelező)  
Törzs (JSON):  
 {

"nev": "Új Név",

"email": "uj.email@example.com",

"telefonszam": "+3611111111"

}

***Példa kérés:***

POST /profil/1

***Válaszok:***

* Sikeres frissítés:

***{'message': 'Sikeres frissítés'}***

* Hiányzó mezők:

***{'error': 'Minden mező kitöltése kötelező'}***

* Felhasználó nem található:

***{'error': 'Felhasználó nem található'}***

**3. DELETE /felhasznalo**

Felhasználó törlése, ha nincs kapcsolódó foglalás.

Törzs (JSON):  
 {

"felhasznalo\_id": 1,

"foglalo\_tipus": "ugyfel"

}

***Példa kérés:***

DELETE /felhasznalo

***Válaszok:***

* Sikeres törlés:

***{'message': 'Profil sikeresen törölve!'}***

* Kapcsolódó foglalás miatt nem törölhető:

***{'error': 'Nem törölhető a profil, mert van kapcsolódó foglalás.'}***

* Szerverhiba:

***{'error': 'Szerverhiba történt a profil törlésekor.'}***

**4. POST /jelszo-valtoztatas/:id**

Felhasználó jelszavának megváltoztatása.

Paraméterek (URL):  
 - :id – felhasználó azonosítója (kötelező)  
Törzs (JSON):  
 {

"jelszo": "regiJelszo123",

"ujJelszo": "ujJelszo456"

}

***Példa kérés:***

POST /jelszo-valtoztatas/1

***Válaszok:***

* Sikeres jelszócsere:

***{'message': 'Jelszó sikeresen frissítve'}***

* Hibás jelenlegi jelszó:

***{'error': 'Hibás jelenlegi jelszó'}***

* Felhasználó nem található:

***{'error': 'Felhasználó nem található'}***

* Hiányzó mezők:

***{'error': 'Hiányzó adat(ok)'}***

* **AuthRoutes.js tesztek:**

**POST /register**

Leírás: Új felhasználó regisztrációja.

* Kérés (Request):
* - Content-Type: application/json
* - Body :

{

'nev': 'Teszt Elek',

'email': 'teszt@example.com',

'jelszo': 'titkos123',

'telefonszam': '06201234567'

}

* Válasz (Response):
* - HTTP státusz: 201
* - JSON válasz: {'message': 'Sikeres regisztráció', 'userId': 1}

**POST /register-vallalkozo**

Leírás: Új vállalkozó regisztrációja.

* Kérés (Request):
* - Content-Type: application/json
* - Body példa:

{

'nev': 'Cég Bt.',

'email': 'vallalkozo@example.com',

'jelszo': 'cegesjelszo123',

'telefonszam': '06201112233',

'cegnev': 'Cég Bt.',

'szolgaltatas': 'fodrász'

}

* Válasz (Response):
* - HTTP státusz: 201
* - JSON válasz: {'message': 'Vállalkozó sikeresen regisztrálva', 'userId': 2}

**POST /login**

Leírás: Bejelentkezés felhasználónév és jelszó alapján.

* Kérés (Request):
* - Content-Type: application/json
* - Body példa:

{

'email': 'teszt@example.com',

'jelszo': 'titkos123'

}

* Válasz (Response):
* - HTTP státusz: 200
* - JSON válasz: {'message': 'Sikeres bejelentkezés', 'token': 'jwt\_token\_itt', 'userId': 1}
* **AuthController.js tesztek:**

**1. Bejelentkezés (POST /login)**

Ez a végpont lehetővé teszi a felhasználók és vállalkozók számára a bejelentkezést.

**Kérés:**

POST <http://localhost:3000/api/auth/login>

Body (JSON):

*{*

*"email": "nagy.istvan@gmail.com",*

*"jelszo": "Jelszo123"*

*}*

Válasz:

{

"message": "Sikeres bejelentkezés felhasználóként.",

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiZW1haWwiOiJuYWd5LmlzdHZhbkBnbWFpbC5jb20iLCJ0aXB1cyI6ImZlbGhhc3puYWxvIiwiaWF0IjoxNzQ2NDQxNzk5LCJleHAiOjE3NDY0NDUzOTl9.XNqH62rh1LSHAXArvfZ35peVOz3ElFBdmrxOspHhsis",

"expirationTime": 1746445399814,

"tipus": "felhasznalo",

"id": 1

}

2. Regisztráció (POST /register)

Ez a végpont új felhasználók vagy vállalkozók regisztrációját teszi lehetővé.

Kérés:

POST <http://localhost:3000/api/auth/register>

Body (JSON):

{

"name": "Teszt Elek",

"email": "teszt@example.com",

"password": "titkosjelszo",

"phone": "+3612345678",

"tipus": "felhasznalo"

}

Válasz:

{

"success": true,

"message": "Sikeres regisztráció.",

"user": {

"id": "662f3bd8f7c9aa3f9e0e52d1",

"name": "Teszt Elek",

"email": "teszt@example.com",

"phone": "+3612345678",

"tipus": "felhasznalo"

}