HotelFlow

Algoritmus specifikáció

Szabó Máté 13.D

Monostori Márk György 13.D

2025.11.05.

Tartalom

1. Bevezetés

Ez a dokumentum a HotelFlow rendszer API–jának a technikai specifikációját tartalmazza.

A rendszer két szereplőt támogat: Vendég (User) és Szállásadó (Hotel). A rendszer REST architektúrát követ; a végpontok /api/... prefixszel érhetők el. A védett végpontokhoz token alapú autentikáció szükséges; a tokeneket a bejelentkezés adja vissza.

Minden időpont a backendben UTC-ben tárolódik; a frontend felelős az átalakításért a lokális megjelenítéshez. A dokumentum alapján a backend könnyen rekonstruálható és fejleszthető bármely erre alkalmas keretrendszerben. (pl.: Django, Laravel, Node.js, ASP.NET)

2. Bejelentkezés (Login)

- **Cél**: Felhasználó azonosítása és token generálása.
- **UI Interakció:**
    1. Felhasználó a login oldalon látja az email és jelszó mezőket, valamint a Bejelentkezés gombot.
    2. Kitölti a mezőket és rákattint a gombra.
    3. Hiba esetén (üres mező, rossz jelszó) a mezők felett hibaüzenet jelenik meg: „Érvénytelen email vagy jelszó”.
    4. Sikeres bejelentkezés esetén:
    5. A felhasználót átirányítja a dashboardra vagy kezdőoldalra.
    6. A token tárolódik a frontenden (localStorage / cookie) a további API hívásokhoz.
- Bemenet: email, password
- Kimenet: access_token, token_type
- UI kimenet: hibaüzenet vagy dashboard megjelenítése
- **Eljárás:**
    1. A rendszer ellenőrzi, hogy a `Request` tartalmazza az `email` és `password` mezőket.
    2. Ha bármelyik hiányzik → hibaüzenet visszaadása.
    3. Kikeresi az adatbázisból (`users` tábla) az adott `email`-hez tartozó felhasználót.
    4. Ha nincs ilyen user → hibás hitelesítés.
    5. Ellenőrzi a jelszó egyezőségét a tárolt hash (`passwordHash`) és a beérkező jelszó között.
    6. Ha nem egyezik → hibás hitelesítés.
    7. Ha minden rendben van:

8. Generál egy API token-t a felhasználónak (Sanctum `createToken`).
9. Visszaadja a tokent és a felhasználó azonosító adatait JSON formátumban.

3. Regisztráció (Register)

- **Cél:** Új felhasználó létrehozása.
- **UI Interakció:**
    1. A regisztrációs oldalon kiválasztja a felhasználó, hogy szállodaként vagy vendégként szeretne regisztrálni.
    2. Felhasználó a regisztrációs oldalon látja a mezőket: **név**, **email**, **jelszó**, **szálloda esetén a szállodának az adatait.**
    3. Kitöltés után a **Regisztráció** gombbal küldi el az adatokat.
    4. Hibák:
        - Üres mező → mező mellett piros hibaüzenet
        - Már létező email → „Ez az email már foglalt”
    5. Sikeres regisztráció után:
    6. Megjelenik a „Regisztráció sikeres!” üzenet.
    7. Felhasználót átirányíthatjuk a **login oldalra**.
- **Bemenet:** name, email, password, role, location-hotel esetén, type-hotel esetén
  Kimenet: user objektum, message
  UI kimenet: sikerüzenet + átirányítás loginra
- **Eljárás**:
    1. Ellenőrzi, hogy minden kötelező mező (`name`, `email`, `password`, `role`) jelen van és valid.
    2. Ellenőrzi, hogy az `email` nem foglalt az adatbázisban.
    3. Ha minden valid:
    4. A jelszót hash-eli (`Hash::make`)
    5. Létrehoz egy új `User` rekordot az adatbázisban.
    6. Visszaadja az új felhasználó adatait (ID, név, email, role, stb.).

4. Szállásfoglalás (Booking)

Cél: Felhasználó szobát foglal a szállodában.

UI Interakció:

- Felhasználó a **szállodai oldal** listájából kiválasztja a kívánt hotelt.
- Megnyitja a hotel részleteit: **szobák, árak, képek, szolgáltatások**.
- Kiválasztja a szobákat és a foglalás dátumát.
- Opció: extra szolgáltatások kiválasztása (pl. reggeli, wellness).

- Foglalás gomb: `Foglalás` → AJAX POST request az API-hoz.
- Hibák:
  1. Lejárt dátum, nem elérhető szoba → üzenet a felhasználónak
- Sikeres foglalás:
  1. Megjelenik a „Foglalás sikeres!” üzenet
  2. Átirányítás a **saját foglalások oldalra** vagy megjelenítés modalban
  3. Check-in token később a foglalás részleteinél látható

Bemenet: user_id, rooms_id, startDate, endDate, services
Kimenet: booking objektum, status
UI kimenet: sikerüzenet, foglalás részletek, navigáció a saját foglalásokhoz

- **Eljárás:**
  1. Ellenőrzi a beérkező foglalási adatokat (dátumok, szobák, felhasználó létezése).
  2. Létrehoz egy `Booking` rekordot az `bookings` táblában.
  3. Létrehozza a kapcsolót a szobákhoz a `bookingsRelation` táblában.
  4. Ha tartalmaz szolgáltatásokat → kapcsolja a `servicesRelation` táblához.
  5. Generálhat check-in és check-out tokeneket a szobákhoz.
  6. Foglalás státusza kezdetben `pending`.
  7. Visszaadja a foglalás adatait JSON formátumban.

5. Szállodai oldal – foglalás visszaigazolása

- **Cél:** Szálloda visszaigazolja vagy elutasítja a foglalást.
- **UI Interakció:**
  1. Szállodai admin a **foglalások listája** oldalon látja az összes `pending` státuszú foglalást.
  2. A foglalás részletei tartalmazzák: **vendég neve, dátumok, szoba, extra szolgáltatások**.
  3. Gombok: **Visszaigazolás**, **Mégse**.
  4. Klikk után AJAX request a `confirm` vagy `cancel` action-re.
  5. Visszaigazolás után:
  6. A státusz a listában **confirmed** vagy **cancelled** lesz.
  7. Értesítés a felhasználónak e-mailben.
  8. Hiba esetén a felületen **piros hibaüzenet** jelenik meg.

Bemenet: booking_id, action
Kimenet: booking status
UI kimenet: státusz frissítés, üzenet a felhasználónak

- **Eljárás:**
  1. Ellenőrzi, hogy a foglalás létezik (`bookings` tábla).
  2. Ha `confirm` → állítja a foglalás státuszát `confirmed`.
  3. Ha `cancel` → állítja a státuszt `cancelled`.
  4. Frissíti a foglalás dátumát (`updatedAt`) és esetleg e-mail értesítést küld a felhasználónak.
  5. A foglalás státuszát a frontenden és mobil appban is lekérdezhetjük.

6. Kártyás beléptető rendszer (QR kód és RFID)

Cél: A vendégek be és kiléptetés, szállodai recepciós alkalmazása nélkül. Egyszerűen és gyorsan.

Eljárás:

1. A vendég érkezéskor leolvassa az emailben kapott QR kódot
2. A fő egység a saját adatbázisából megkeresi a foglalás azonosítót, ami a QR kódban volt.
3. A foglalásban kiosztott szobákhoz tartozó kártyákat az egység kiadja
4. A vendég a kártyával tud belépni a szoba ajtaján, ami e kártya adatait MQTT protokollon keresztül juttatja el a főegységen futó ellenőrző szoftvernek. Ha helyes a kártya száma, akkor ugyan ezzel a metódussal kapja vissza a terminál és nyitja a zárat.
5. Kilépéskor a belépéshez használt QR kóddal tudja leadni a kártyát.

Logikai elv: Terminál → MQTT → Ellenőrző program → MQTT → Terminál

7. Szoba felvétele a szállodához

- **Cél:** Szálloda adminisztrátor hozzáad egy új szobát a hotelhez.
- **UI Interakció:**
  1. Szállodai admin a **Hotel kezelése** oldalon látja a **Szobák** listáját.
  2. Gomb: **Új szoba hozzáadása**
  3. **Form**:
     - `Név` (pl. „Deluxe szoba”)
     - `Leírás`

        - `Ár/éj`
        - `Kapacitás`
        - `Alapár`
    4. Képek feltöltése (opcionális)
    5. Kitöltés után **Mentés** gomb:
        - Backend ellenőrzi a kötelező mezőket
        - Létrehoz egy új `rooms` rekordot
        - Ha vannak képek, azokat feltölti az `images` táblába és kapcsolja a `imagesRelation`-nel
        - Visszaadja az új szoba adatait
- **Bemenet:** hotel_id, name, description, pricePerNight, capacity, basePrice, images
  Kimenet: szoba objektum, képek listája
  UI kimenet: sikerüzenet, szoba listában frissítés

8. Szálloda adatbővítése / szerkesztése

- **Cél:** Szálloda adminisztrátor frissíti a hotel adatait, képeket ad hozzá, ratingeket, szolgáltatásokat.
- **UI Interakció:**
    1. Szállodai admin a **Hotel szerkesztése** oldalon látja a mezőket:
        1. Név
        2. Leírás
        3. Típus (hotel, apartman, villa…)
        4. Csillagbesorolás
        5. Képek kezelése (feltöltés, törlés)
        6. Szolgáltatások hozzáadása / törlése
        7. Cím / lokáció
    2. Admin módosítja az adatokat, feltölti képeket.
    3. Mentés gomb:
        - Backend ellenőrzi az inputot
        - Frissíti a `hotels` rekordot
        - Új képek → `images` tábla + `imagesRelation`
        - Szolgáltatások → `services` + `servicesRelation`
    4. Sikeres mentés után:
        - Visszajelzés: „Hotel adatai frissítve”
        - A hotel részletei a frontenden azonnal frissülnek

- **Bemenet:** hotel_id, name, description, type, starRating, location, images, services
  Kimenet: hotel objektum, frissített képek és szolgáltatások
  UI kimenet: sikerüzenet, frissített hotel oldalon megjelenés

9. Szoba elérhetőségének ellenőrzése

- **Cél:**
  - Megállapítani, hogy egy szoba a megadott időintervallumban (startDate – endDate) **szabad-e**, vagy már létezik hozzá ütköző foglalás.
- **Bemenet:**
  - room_id – a választott szoba azonosítója
  - startDate – foglalás kezdete
  - endDate – foglalás vége
- **Kimenet:**
  - Szoba foglalható / nem foglalható (boolean)
  - Ütköző foglalások listája (opcionális)
- **Eljárás leírása:**
  1. A rendszer megkapja a szoba azonosítóját és a kért időszakot.
  2. Az adatbázisban megkeresi az adott szobához tartozó foglalásokat (bookingsRelation + bookings).
  3. A foglalások közül azokat vizsgálja, amelyek:
     - státusza **confirmed** vagy **pending**,
     - **átfednek** az új foglalás időszakával.
  4. Két foglalás akkor fed át, ha teljesül az alábbi feltétel:
     - existing.startDate <= new.endDate ÉS existing.endDate >= new.startDate
  5. Ha talál a rendszer akár egyetlen ütköző foglalást:
  6. az eredmény: **szoba nem foglalható**.
  7. Ha nincs ütközés:
     - az eredmény: **szoba szabad és foglalható**.
  8. **Kivételes esetek**
     - Ha a szoba nem létezik → hibaüzenet (404)
     - Ha a dátumok hibásak → hiba (400)

10. Foglalás árkalkulációja

- **Cél:**
  - A foglalás teljes árának kiszámítása a kiválasztott szobák, azok kapacitása, az ott tartózkodó vendégek száma, a foglalás időtartama és az igénybe vett szolgáltatások alapján.
- **Bemenet:**
  - rooms[] – szobák ID-listája
  - services[] – szolgáltatások ID-listája
  - startDate, endDate – foglalási időszak
  - guests[] – vendégek listája (név, idNumber, dateOfBirth)
- **Kimenet:**
  - Teljes ár (integer)
  - Részletes bontás (szobaár, szolgáltatásár, napok száma)
- **Eljárás leírása:**
  - A rendszer megkapja a foglaláshoz tartozó szobák listáját.
  - Meghatározza a foglalás hosszát napokban:
    - napok = endDate - startDate
  - A rendszer betölti az összes érintett szoba adatait:
    - basePrice
    - pricePerNight
    - capacity
  - Betölti a vendégek számát:
    - vendégekSzáma = guests tömb hossza
  - Minden szobánál kiszámolja a szoba árát:
    - Szoba árképzés logikája:
      - A szoba ára =
        (basePrice × vendégek száma) + (pricePerNight × napok száma)
      - Ha több szoba van, a rendszer összeadja az egyes szobákra számolt árakat.
  - Ha a felhasználó igénybe vesz szolgáltatásokat:
    - betölti a services táblából az árakat,
    - összeadja őket:
      - szolgáltatások_ára = összeg(services.price)
  - A teljes ár:
    - totalPrice = összesSzobaÁr + szolgáltatások_ára
  - A rendszer a kalkulált értéket elmenti a bookings.totalPrice mezőbe.

- **Kivételes esetek:**
  - Ha a dátumok hibásak → hiba (400)
  - Ha vendégek száma meghaladja a szobák összkapacitását → hiba (409)
  - Ha egy szoba nem létezik → hiba (404)