Vámosi László Ádám
SZOFT-II-N

MyRoomify Backend Fejlesztési Feljegyzés

---

1. Adatbázis-migrációk elkészítése

Elkészültek a fő adatmodellek migrációi:

Létrehozott táblák

- **users**
    - admin / recepciós / vendég szerepkör támogatás
    - alap felhasználói adatok
- **rooms**
    - szobanév, kapacitás, leírás, ár, felszereltség
    - státuszmező: *available, inactive, maintenance*
- **bookings**
    - foglalási adatok (érkezés, távozás, vendégadatok)
    - foglalás típusa: *online, telefonos, helyszíni*
    - foglalási státusz: *pending, confirmed, cancelled, no_show, completed*
    - fizetési státusz: *unpaid, paid, refunded*
    - szoba- és felhasználókapcsolatok
- **reviews**
    - értékelés, csillagszám, hozzászólás, moderációs állapot
- **images**
    - képfájl neve, tartozás adott szobához (room_id)

A migrációkhoz a megfelelő foreign key kapcsolatok, indexelések is létre lettek hozva.

Vámosi László Ádám
SZOFT-II-N

2. Modellek létrehozása kapcsolatokkal

Elkészültek a Laravel Eloquent modellek, a következő kapcsolatokkal:

- **User**
    - hasMany(Bookings)
    - hasMany(Reviews)
- **Room**
    - hasMany(Images)
    - hasMany(Bookings)
    - hasMany(Reviews)
- **Booking**
    - belongsTo(User)
    - belongsTo(Room)
- **Review**
    - belongsTo(User)
    - belongsTo(Room)
- **Image**
    - belongsTo(Room)

Model szinten konfigurálva lettek:

- fillable mezők
- típuskonverziók (casts)
- konstansok a státuszokhoz
- alap query scope-ok (pl. aktív szobák lekérdezése)

---

3. API végpontok konfigurálása

A routes/api.php fájlban létrehoztam az összes, jelen szakaszban szükséges backend végpontot.

Szobák

- GET /api/rooms – szobák listázása
- POST /api/rooms – új szoba létrehozása
- PUT /api/rooms/{id} – szoba módosítása
- DELETE /api/rooms/{id} – szoba törlése
- PUT /api/rooms/{id}/status – szoba státusz módosítása
- POST /api/rooms/{id}/images – képek feltöltése adott szobához

Foglalások

- GET /api/bookings – foglalások listázása
- POST /api/bookings – új foglalás létrehozása
- PUT /api/bookings/{id} – foglalás adatainak frissítése
- PUT /api/bookings/{id}/status – foglalási státusz módosítása
- PUT /api/bookings/{id}/payment-status – fizetési státusz frissítése
- GET /api/bookings/user/{user_id} – adott felhasználó foglalásainak lekérése

A route-okhoz a megfelelő middleware-ek és szerepkör-jogosultságok előkészítése is megtörtént.

---

4. Elkészült CRUD controllerek

Elkészültek az alábbi controllerek a hozzájuk tartozó teljes CRUD logikával, validációkkal és hibakezeléssel.

RoomController

- szobák listázása
- új szoba mentése
- módosítás (PUT /api/rooms/{id})
- törlés
- státuszváltás (PUT /api/rooms/{id}/status)
- képfeltöltés adott szobához

BookingController

- foglalások listázása
- foglalás létrehozása
- adatfrissítés (PUT /api/bookings/{id})
- státuszkezelés (pending → confirmed → completed / cancelled / no_show)
- fizetési státusz frissítése
- dátumütközés-ellenőrzés implementálása
- felhasználó foglalásainak lekérése

ReviewController

- értékelések listázása
- új értékelés elmentése
- szoba szerinti értékelések
- felhasználó értékelései
- saját vagy admin által történő törlés

---

5. Request validációk

A projekthez külön FormRequest osztályokat készítettem:

- StoreRoomRequest
- UpdateRoomRequest
- StoreBookingRequest
- UpdateBookingRequest
- StoreReviewRequest

A validációk lefedik:

- kötelező mezők ellenőrzése
- dátumformátum ellenőrzés
- foglalási időintervallum szabályok
- csillagszám korlátozása 1–5 között
- képfájl-validációk (méret, kiterjesztés)

---

6. Üzleti logika és hibakezelés

A backendben elkészült több alapvető üzleti folyamat:

Dátumütközés vizsgálata foglalásnál

A BookingController ellenőrzi, hogy a szoba az adott intervallumban foglalt-e.

Szoba státusz alapján foglalhatóság

Inactive vagy maintenance állapotú szoba nem foglalható.

Foglalási státusz

- pending → confirmed
- confirmed → completed / cancelled / no_show