Készítsünk egy Laravel Web API-t, amely:

új irodabérléseket rögzít,

meglévő bérléseket lekérdez (egy vagy több),

bérléseket töröl.

**Adattáblák**

users: felhasználók (csak az azonosítójuk szükséges egyelőre)

offices: irodák (csak az azonosítójuk szükséges egyelőre)

rentals: a bérlések adatai (ez a fő tábla)

**Bérlés mezők:**

Mezőnév Típus Leírás

id int Egyedi azonosító (kulcs)

uid int A bérlő felhasználó ID-ja

officeId int A bérelt iroda ID-ja

startDate DateTime Bérlés kezdete

endDate DateTime Bérlés vége

dailyRate int Napi díj (€)

baseFee int Alapdíj (€)

totalPrice - Számolt érték, nem kerül adatbázisba (NotMapped)

**API végpontok**

GET /api/berlesek – összes bérlés

GET /api/berlesek/{id} – adott bérlés

POST /api/berlesek – új bérlés rögzítése (üzleti szabályokkal)

DELETE /api/berlesek/{id} – bérlés törlése

**Üzleti szabályok** új bérléshez (POST):

Kezdőnap ≥ holnapi nap

Bérlés hossza ≥ 5 nap

Bérlés hossza ≤ 90 nap

Az adott irodát nem lehet párhuzamosan kétszer lefoglalni

1**. composer create-project laravel/laravel office-sphere-api**

2. **.env**

DB\_CONNECTION=mysql

# DB\_HOST=127.0.0.1

# DB\_PORT=3306

# DB\_DATABASE=office\_sphare

# DB\_USERNAME=root

# DB\_PASSWORD=

3. **Generál egy titkos kulcsot** (APP\_KEY)

**php artisan key:generate**

4.Elsőként hozzuk létre a **rentals táblát.**

**php artisan make:migration create\_rentals\_table**

\*\*\*

public function up()

{

Schema::create('rentals', function (Blueprint $table) {

$table->id(); // id (PK)

$table->unsignedBigInteger('uid'); // felhasználó azonosítója

$table->unsignedBigInteger('office\_id'); // iroda azonosító

$table->date('start\_date');

$table->date('end\_date');

$table->integer('daily\_rate');

$table->integer('base\_fee');

$table->timestamps();

});

}

\*\*\*

5**.php artisan migrate**

6.**php artisan make:model Rental**

7. Itt **számolhatjuk a totalPrice** értékét

\*\*\*

use Illuminate\Database\Eloquent\Model;

use Illuminate\Database\Eloquent\Casts\Attribute;

class Rental extends Model

{

protected $fillable = [

'uid',

'office\_id',

'start\_date',

'end\_date',

'daily\_rate',

'base\_fee',

];

protected $appends = [

'total\_price',

];

// totalPrice = base\_fee + dailyRate \* napok száma

protected function totalPrice(): Attribute

{

return Attribute::make(

get: function () {

$days = $this->start\_date->diffInDays($this->end\_date);

return $this->base\_fee + ($this->daily\_rate \* $days);

},

);

}

}

\*\*\*

8.**Routes routes/api.php h**a nincs létrehozni

+++++++++++++ BEILLESZTENI A **bootstrap/app.php**-ba a +++++++++++++

**api: \_\_DIR\_\_.'/../routes/api.php',** ++++++++++++++++

**return Application::configure(basePath: dirname(\_\_DIR\_\_))**

\*\*\*

->withRouting(

web: \_\_DIR\_\_.'/../routes/web.php',

api: \_\_DIR\_\_.'/../routes/api.php',

commands: \_\_DIR\_\_.'/../routes/console.php',

health: '/up',

)

\*\*\*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Kérés → Route (útvonal) → Controller → Modell → Adatbázis → Válasz

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conrollerrel:

\*\*\*

use App\Http\Controllers\RentalController;

Route::get('/berlesek', [RentalController::class, 'index']); // 🔹 Összes bérlés

Route::get('/berlesek/{id}', [RentalController::class, 'show']); // 🔹 Egy adott bérlés

VAGY EZ EGYSZERŰBB

Route::apiResource('berlesek', RentalController::class);

Route::apiResource('irodak', OfficeController::class);

\*\*\*

9. **CONTROLLER**

**php artisan make:controller RentalController**

\*\*\*

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\Rental;

class RentalController extends Controller

{

public function index()

{

return response()->json(Rental::all());

}

public function show($id)

{

return response()->json(Rental::findOrFail($id));

}//

}

\*\*\*

10. **ÚJBÉRLÉS - store() metódus a RentalController**

Route::post('/berlesek', [RentalController::class, 'store']); VAGY

\*\*\*

use Illuminate\Http\Request;

use App\Models\Rental;

use Illuminate\Support\Carbon;

public function store(Request $request)

{

// 1️⃣ Érvényesítés: ellenőrizzük, hogy minden szükséges mező megvan-e és megfelelő-e

$validated = $request->validate([

'uid' => 'required|integer', // Bérlő azonosítója

'office:id' => 'required|integer', // Iroda azonosítója

'start\_date' => 'required|date', // Bérlés kezdete (dátumként)

'end\_date' => 'required|date', // Bérlés vége (dátumként)

'daily\_rate' => 'required|integer|min:0', // Napi díj (nem lehet negatív)

'base\_fee' => 'required|integer|min:0', // Alapdíj (nem lehet negatív)

]);

// A dátumokat átalakítjuk Carbon objektummá, hogy tudjunk velük számolni

$start = Carbon::parse($validated['startDate']);

$end = Carbon::parse($validated['endDate']);

$days = $start->diffInDays($end); // Különbség napokban

//Üzleti szabály: kezdődátum nem lehet korábbi, mint holnap

if ($start->isBefore(Carbon::tomorrow())) {

return response()->json(['error' => 'A bérlés kezdőnapja nem lehet korábbi, mint a holnapi nap.'], 400);

}

// Üzleti szabály: legalább 5 napos legyen a bérlés

if ($days < 5) {

return response()->json(['error' => 'A bérlésnek legalább 5 naposnak kell lennie.'], 400);

}

// Üzleti szabály: legfeljebb 90 napos lehet a bérlés

if ($days > 90) {

return response()->json(['error' => 'A bérlés nem lehet hosszabb 90 napnál.'], 400);

}

// Ellenőrizzük, hogy van-e átfedés másik bérléssel ugyanarra az irodára

$overlap = Rental::where('office\_id', $validated['office\_id'])

->where(function ($query) use ($start, $end) {

$query->whereBetween('start\_date', [$start, $end]) // kezdőnap beleesik az időszakba

->orWhereBetween('end\_date', [$start, $end]) // zárónap beleesik az időszakba

->orWhere(function ($query) use ($start, $end) {

$query->where('start\_date', '<=', $start) // lefedi az egész időszakot

->where('end\_date', '>=', $end);

});

})->exists();

if ($overlap) {

// Ha van átfedés, visszatérünk hibával

return response()->json(['error' => 'Ez az iroda már foglalt a megadott időszakban.'], 400);

}

// Ha minden rendben, elmentjük az új bérlést

$rental = Rental::create($validated);

// isszaküldjük a mentett adatokat JSON válaszként

return response()->json($rental, 201); // 201 Created státusz

}

\*\*\*

11**. DELETE**

Route::delete('/berlesek/{id}', [RentalController::class, 'destroy']); VAGY

\*\*\*

public function destroy($id)

{

// Megkeressük a bérlést az ID alapján, vagy 404-es hibát dobunk, ha nincs ilyen

$rental = Rental::findOrFail($id);

// öröljük az adatbázisból

$rental->delete();

// Visszajelzés: nincs tartalom, de sikeres törlés (204 No Content)

return response()->json(null, 204);

}

\*\*\*