Festmeny API  
  
Egy Web API-t készítünk Laravel-ben, amely:

* tárolja a bérléseket adatbázisban,
* támogatja az új bérlés hozzáadását,
* lekéri az összes vagy egy adott bérlést,
* bérlést töröl,
* figyelembe veszi a **megadott üzleti szabályokat**.
* A bérlés kezdete nem lehet korábbi, mint a holnapi nap.
* 2. A bérlés időtartama legalább 3 nap kell legyen.
* 3. A bérlés időtartama legfeljebb 90 nap lehet. 4. Ugyanazt a festményt az adott időszakban nem lehet többször lefoglalni.

Azaz nem lehet átfedés két bérlés között ugyanarra a PaintingId-ra.

**1. LARAVEL PROJEKT LÉTREHOZÁSA**

composer create-project laravel/laravel painting\_api

.env fájlban:

makefile

MásolásSzerkesztés

DB\_CONNECTION=mysql

DB\_DATABASE=painting

DB\_USERNAME=root

DB\_PASSWORD= # vagy amit használsz

**2. MIGRÁCIÓ ÉS MODELL: Rental**

php artisan make:model Rental -m (migrációt is)

php artisan make:controller RentalItemController –resource

php artisan make:model Rental -mcr /model, controller, migration

**3. migr**

public function up()

{

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

$table->id(); // id

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

$table->unsignedBigInteger('painting\_id'); // festmény ID

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

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

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

$table->timestamps();

});

}

**4. Futtasd a migrációt:**

php artisan migrate

**5. ROUTE-ok beállítása (routes/api.php)**

use App\Http\Controllers\RentalController;

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

use App\Http\Controllers\RentalController;

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

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

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

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

**6. CONTROLLER LÉTREHOZÁSA**

php artisan make:controller RentalController --api

**7. RENTAL CONTROLLER LOGIKA**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\Rental;

use Illuminate\Support\Carbon;

use Illuminate\Support\Facades\Validator;

class RentalController extends Controller

{

    // GET /api/berlesek - Összes bérlés lekérése

    public function index()

    {

        return Rental::all(); // Visszaadja az összes bérlést

    }

    // GET /api/berlesek/{id} - Egy bérlés lekérése ID alapján

    public function show($id)

    {

        $rental = Rental::find($id); // Bérlés keresése ID alapján

        if (!$rental) {

            return response()->json(['error' => 'A bérlés nem található.'], 404);

        }

        return $rental; // Megtalált bérlés visszaadása

    }

    // POST /api/berlesek - Új bérlés létrehozása

    public function store(Request $request)

    {

        // Adatok validálása

        $validator = Validator::make($request->all(), [

            'Uid' => 'required|integer',

            'PaintingId' => 'required|integer',

            'StartDate' => 'required|date',

            'EndDate' => 'required|date|after:StartDate',

            'DailyPrice' => 'required|integer|min:0',

        ]);

        if ($validator->fails()) {

            return response()->json(['errors' => $validator->errors()], 400);

        }

        // Dátumok átalakítása Carbon objektummá

        $start = Carbon::parse($request->StartDate);

        $end = Carbon::parse($request->EndDate);

        $today = Carbon::today();

        // Üzleti szabály 1: kezdőnap legalább holnap legyen

        if ($start->lte($today)) {

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

        }

        // Üzleti szabály 2: legalább 3 napos bérlés

        if ($start->diffInDays($end) < 3) {

            return response()->json(['error' => 'A bérlésnek legalább 3 naposnak kell lennie.'], 400);

        }

        // Üzleti szabály 3: legfeljebb 90 napos bérlés

        if ($start->diffInDays($end) > 90) {

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

        }

        // Üzleti szabály 4: nincs átfedés ugyanarra a festményre

        $overlap = Rental::where('PaintingId', $request->PaintingId)

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

                $query->whereBetween('StartDate', [$start, $end->subDay()])

                      ->orWhereBetween('EndDate', [$start->addDay(), $end]);

            })

            ->exists();

        if ($overlap) {

            return response()->json(['error' => 'Erre a festményre már van lefoglalás az adott időszakban.'], 400);

        }

        // Ha minden rendben, új bérlés mentése

        $rental = Rental::create([

            'Uid' => $request->Uid,

            'PaintingId' => $request->PaintingId,

            'StartDate' => $request->StartDate,

            'EndDate' => $request->EndDate,

            'DailyPrice' => $request->DailyPrice,

        ]);

        return response()->json($rental, 201); // Sikeres mentés válasza

    }

    // DELETE /api/berlesek/{id} - Bérlés törlése

    public function destroy($id)

    {

        $rental = Rental::find($id);

        if (!$rental) {

            return response()->json(['error' => 'A bérlés nem található.'], 404);

        }

        $rental->delete(); // Törlés az adatbázisból

        return response()->json(['message' => 'A bérlés sikeresen törölve lett.']);

    }

}

**8. Rental model (app/Models/Rental.php)**

<?php

namespace App\Models; // A modell a megfelelő névtérben található.

use Illuminate\Database\Eloquent\Model; // Az Eloquent Model osztályt használjuk az ORM működéséhez.

use Illuminate\Database\Eloquent\Casts\Attribute; // Az Attribute osztályt a számolt tulajdonság (TotalPrice) létrehozásához importáljuk.

class Rental extends Model // A Rental modell az Eloquent Model osztályból öröklődik.

{

// Meghatározzuk, hogy mely mezők tömegesen kitölthetők (pl. a create() metódus használatakor).

protected $fillable = [

'uid', // A bérlő azonosítója

'painting\_id', // A bérelt festmény azonosítója

'start\_date', // Bérlés kezdőnapja

'end\_date', // Bérlés zárónapja

'daily\_price', // Napi bérleti díj

];

// A típuskonverziókat itt állítjuk be: a start\_date és end\_date mezőket automatikusan DateTime-ként kezeli a Laravel.

protected $casts = [

'start\_date' => 'datetime',

'end\_date' => 'datetime',

];

// A total\_price mezőt számolt mezőként definiáljuk (nem kerül adatbázisba mentésre).

protected function totalPrice(): Attribute

{

return Attribute::make(

// GETer: ha start\_date és end\_date létezik, kiszámoljuk a különbség napokban \* napi ár.

get: fn () => $this->start\_date && $this->end\_date

? $this->start\_date->diffInDays($this->end\_date) \* $this->daily\_price

: null, // Ha nincs mindkét dátum, akkor null értéket adunk vissza.

);

}

}

**9. Teszt**

+--------+------------------------+--------------------------+----------------------------+

| Módszer| URL | Cél | Küldendő adat (body) |

+--------+------------------------+--------------------------+----------------------------+

| GET | /api/berlesek | Összes bérlés lekérése | - |

| GET | /api/berlesek/{id} | Egy bérlés lekérése | - |

| POST | /api/berlesek | Új bérlés létrehozása | JSON body (lásd lent) |

| DELETE | /api/berlesek/{id} | Bérlés törlése | - |

+--------+------------------------+--------------------------+----------------------------+

**A) GET /api/berlesek**

* **Method**: GET
* **URL**: http://localhost:8000/api/berlesek

**B) GET /api/berlesek/{id}**

* **Method**: GET
* **URL**: http://localhost:8000/api/berlesek/1 (pl. 1-es ID)

**C) POST /api/berlesek – Új bérlés létrehozása**

* **Method**: POST
* **URL**: http://localhost:8000/api/berlesek
* **Body**:
  + Típus: JSON

{

"Uid": 1,

"PaintingId": 3,

"StartDate": "2025-05-28",

"EndDate": "2025-06-05",

"DailyPrice": 1000

}

**D) DELETE /api/berlesek/{id}**

* **Method**: DELETE
* **URL**: http://localhost:8000/api/berlesek/1

**Törli az ID=1 bérlést**