Készítsünk egy Laravel Web API-t, amely:

a foglalásokhoz rögzítendő userId string típusú.

Egy bérlés több ékszert is tartalmazhat, A bérlés hossza minimum 3, maximum 21 nap lehet. A bérlés lemondása csak a

kezdete előtt legalább 3 nappal lehetséges. A cégnek az ékszerekből egyenlőre csak egy darab áll rendelkezésére, ne

engedjen átfedő foglalást rögzíteni a rental végponton POST eseténben. Mivel a frontenden még nincs végleges

megállapodás az üzleti logika működésében, a rentalitems végponton egyelőre nem szükséges az ellenőrzéssel

foglalkoznia. A rental végponton a GET és a GET/{id} végpontok listázzák a hozzájuk tartozó ékszereket is.

Végpontok:

/api/rentals/ (GET, POST, DELETE)

api/rentals/ – GET

[

{

"id": 0,

"userId": "string",

"startDate": "2025-03-30T16:35:30.726Z",

"endDate": "2025-03-30T16:35:30.726Z",

"rentalItems": [

{

"id": 0,

"rentalId": 0,

"jewelryId": 0,

"dailyPrice": 0

}

],

"duration": 0,

"totalCost": 0

}

]

api/rentals/{id} – GET

{

"id": 0,

"userId": "string",

"startDate": "2025-03-30T16:37:25.613Z",

"endDate": "2025-03-30T16:37:25.613Z",

"rentalItems": [

{

"id": 0,

"rentalId": 0,

"jewelryId": 0,

"dailyPrice": 0

}

],

"duration": 0,

"totalCost": 0

}

api/rentals – POST

{

"userId": "string",

"startDate": "2025-03-30T16:37:25.602Z",

"endDate": "2025-03-30T16:37:25.603Z",

"rentalItems": [

{

"jewelryId": 0,

"dailyPrice": 0

}

]

}

api/rentals/{id} – DELETE

api/rentalitems/ (GET, POST, PUT, DELETE)

api/rentalitems/ – GET

[

{

"id": 0,

"rentalId": 0,

"jewelryId": 0,

"dailyPrice": 0

}

]

api/rentalitems/{id} – GET

{

"id": 0,

"rentalId": 0,

"jewelryId": 0,

"dailyPrice": 0

}

api/rentalitems/{id} – PUT

{

"id": 0,

"rentalId": 0,

"jewelryId": 0,

"dailyPrice": 0

}

api/rentalitems/{id} – DELETE

-

Modellek (duration, totalCost – számított mező):

Rental{

id integer($int32)

userId\* string

minLength: 1

startDate\* string($date-time)

endDate\* string($date-time)

rentalItems [

nullable: true

RentalItem{...}]

duration integer($int32)

readOnly: true

totalCost number($double)

readOnly: true

}

RentalItem{

id integer($int32)

rentalId\* integer($int32)

jewelryId\* integer($int32)

dailyPrice number($double)

1**. composer create-project laravel/laravel ekszerberles\_api**

2. **.env**

DB\_CONNECTION=mysqlite

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=jewelry

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(): void

    {

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

        $table->id();

        $table->string('userId'); // nincs aláhúzás (\_) a mezőnévben

        $table->date('start\_date');

        $table->date('end\_date');

        $table->timestamps();

    } );

    }

\*\*

5. **php artisan make:migration create\_rental\_items\_table**

public function up(): void

{

    Schema::create('rental\_items', function (Blueprint $table) {

        $table->id();

        $table->unsignedBigInteger('rental\_id');

        $table->foreign('rental\_id')

              ->references('id')

              ->on('rentals')

              ->onDelete('cascade');

        $table->unsignedBigInteger('jewelry\_id'); // ajánlott: snake\_case

        $table->decimal('daily\_price', 8, 2);

        $table->timestamps();

    });

}

6**.php artisan migrate vagy(php artisan migrate:fresh)**

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

**Rental.php**<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

use Illuminate\Database\Eloquent\Relations\HasMany;

class Rental extends Model

{

protected $fillable = [

'userId',

'start\_date',

'end\_date',

];

// ✨ számított mező: duration

protected $appends = ['duration', 'totalCost'];

// ✏️ Kapcsolat rental\_items táblával

public function rentalItems(): HasMany

{

return $this->hasMany(RentalItem::class);

}

// 👉 Számított mező: duration (napok száma)

public function getDurationAttribute(): int

{

return \Carbon\Carbon::parse($this->startDate)->diffInDays($this->end\_date);

}

// 👉 Számított mező: totalCost (összes napi ár \* napok száma)

public function getTotalCostAttribute(): float

{

return $this->rentalItems->sum(function ($item) {

return $item->daily\_price \* $this->duration;

});

}

public function items()

{

return $this->hasMany(RentalItem::class);

}

}

8. **php artisan make:model RentalItem**

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

use Illuminate\Database\Eloquent\Relations\BelongsTo;

class RentalItem extends Model

{

  protected $fillable = [

    'rental\_id',

    'jewelry\_id',

    'daily\_price',

  ];

  // Kapcsolat a Rental modellel

  public function rental(): BelongsTo

  {

    return $this->belongsTo(Rental::class);

  }

    // Kapcsolat a Jewelry modellel

    public function jewelry(): BelongsTo

    {

        return $this->belongsTo(Jewelry::class);

    }

}

9. **php artisan make:model Jewelry**

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Jewelry extends Model

{

    protected $fillable = [

        'name',

        'description',

        'price',

    ];

}

10. **php artisan make:controller RentalController**

<?php

namespace App\Http\Controllers;

use App\Models\Rental;

use App\Models\RentalItem;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Validator;

use Illuminate\Support\Carbon;

class RentalController extends Controller

{

    // Összes bérlés lekérdezése

    public function index()

    {

        return Rental::with('items')->get();

    }

    // Egy konkrét bérlés lekérdezése

    public function show(string $id)

    {

        $rental = Rental::with('items')->find($id);

        if (!$rental) {

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

        }

        return response()->json($rental);

    }

    // Új bérlés létrehozása

    public function store(Request $request)

    {

        // Alap validáció

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

            'userId' => 'required|string',

            'start\_date' => 'required|date',

            'end\_date' => 'required|date|after:start\_date',

            'rentalItems' => 'required|array|min:1',

            'rentalItems.\*.jewelry\_id' => 'required|integer',

            'rentalItems.\*.daily\_price' => 'required|numeric|min:0',

        ]);

        if ($validator->fails()) {

            return response()->json(['error' => $validator->errors()], 422);

        }

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

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

        $duration = $start->diffInDays($end);

        // Feltétel: minimum 3, maximum 21 nap

        if ($duration < 3 || $duration > 21) {

            return response()->json(['error' => 'A bérlés hossza 3 és 21 nap között lehet.'], 422);

        }

        // Átfedés ellenőrzése minden jewelry\_id-re

        foreach ($request->rentalItems as $item) {

            $overlap = RentalItem::where('jewelry\_id', $item['jewelry\_id'])

                ->whereHas('rental', function ($query) use ($start, $end) {

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

                        $q->whereBetween('start\_date', [$start, $end])

                          ->orWhereBetween('end\_date', [$start, $end])

                          ->orWhere(function ($q2) use ($start, $end) {

                              $q2->where('start\_date', '<', $start)

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

                          });

                    });

                })

                ->exists();

            if ($overlap) {

                return response()->json(['error' => 'Átfedő foglalás: az adott ékszer már le van foglalva erre az időszakra.'], 422);

            }

        }

        // Bérlés mentése

        $rental = Rental::create([

            'userId' => $request->userId,

            'start\_date' => $request->start\_date,

            'end\_date' => $request->end\_date,

            'duration' => $duration,

        ]);

        // Rental items mentése

        foreach ($request->rentalItems as $item) {

            RentalItem::create([

                'rental\_id' => $rental->id,

                'jewelry\_id' => $item['jewelry\_id'],

                'daily\_price' => $item['daily\_price'],

            ]);

        }

        return response()->json($rental->load('items'), 201);

    }

    // Bérlés módosítása (alap validációval)

    public function update(Request $request, string $id)

    {

        $rental = Rental::find($id);

        if (!$rental) {

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

        }

        // Egyelőre csak dátumot és userId-t módosítunk (feltételek nélkül)

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

            'userId' => 'sometimes|required|string',

            'start\_date' => 'sometimes|required|date',

            'end\_date' => 'sometimes|required|date|after:start\_date',

        ]);

        if ($validator->fails()) {

            return response()->json(['error' => $validator->errors()], 422);

        }

        if ($request->has('start\_date') && $request->has('end\_date')) {

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

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

            $duration = $start->diffInDays($end);

            if ($duration < 3 || $duration > 21) {

                return response()->json(['error' => 'A bérlés hossza 3 és 21 nap között lehet.'], 422);

            }

            $rental->start\_date = $request->start\_date;

            $rental->end\_date = $request->end\_date;

        }

        if ($request->has('userId')) {

            $rental->userId = $request->userId;

        }

        $rental->save();

        return response()->json($rental);

    }

    // Bérlés törlése (csak ha legalább 3 nappal előtte)

    public function destroy(string $id)

    {

        $rental = Rental::find($id);

        if (!$rental) {

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

        }

        $now = Carbon::now();

        $start = Carbon::parse($rental->start\_date);

        // Feltétel: lemondás legalább 3 nappal előtte

        if ($start->diffInDays($now, false) < 3) {

            return response()->json(['error' => 'A bérlés csak legalább 3 nappal a kezdete előtt törölhető.'], 403);

        }

        $rental->delete();

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

    }

}

**11.** **php artisan make:controller RentalItemController**

<?php

namespace App\Http\Controllers;

use App\Models\RentalItem;

use Illuminate\Http\Request;

class RentalItemController extends Controller

{

    // RentalItem-ek listázása (opcionálisan szűrés rental\_id szerint)

    public function index(Request $request)

    {

        if ($request->has('rental\_id')) {

            return RentalItem::where('rental\_id', $request->rental\_id)->get();

        }

        return RentalItem::all();

    }

    // Új rental item létrehozása

    public function store(Request $request)

    {

        $validated = $request->validate([

            'rental\_id' => 'required|exists:rentals,id',

            'jewelryId' => 'required|integer',

            'daily\_price' => 'required|numeric|min:0',

        ]);

        $item = RentalItem::create($validated);

        return response()->json($item, 201);

    }

    // Egy rental item lekérdezése

    public function show($id)

    {

        return RentalItem::findOrFail($id);

    }

    // Rental item frissítése

    public function update(Request $request, $id)

    {

        $item = RentalItem::findOrFail($id);

        $item->update($request->only(['jewelryId', 'daily\_price']));

        return response()->json($item);

    }

    // Rental item törlése

    public function destroy($id)

    {

        RentalItem::destroy($id);

        return response()->noContent();

    }

}

12. Routes

**bootstrap/app.php!!!**

**api: \_\_DIR\_\_.’/../routes/api.php’,**

13.**api.php – létrehozni**

<?php

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\RentalController;

use App\Http\Controllers\RentalItemController;

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

Route::post('/rentals', [RentalController::class, 'store']);        // Új bérlés létrehozása

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

Route::put('/rentals/{id}', [RentalController::class, 'update']);   // Bérlés módosítása

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

// RentalItem végpontok

Route::get('/rental-items', [RentalItemController::class, 'index']); // ?rental\_id=1 szűréshez

Route::post('/rental-items', [RentalItemController::class, 'store']);

Route::get('/rental-items/{id}', [RentalItemController::class, 'show']);

Route::put('/rental-items/{id}', [RentalItemController::class, 'update']);

Route::delete('/rental-items/{id}', [RentalItemController::class, 'destroy']);

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

| MŰVELET | METHOD | ÚTVONAL | MEGJEGYZÉS |

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

| Összes bérlés lekérése | GET | /api/rentals | Minden rental |

| Új bérlés létrehozása | POST | /api/rentals | Új rental mentése |

| Egy bérlés lekérése | GET | /api/rentals/{id} | ID alapján |

| Bérlés frissítése | PUT | /api/rentals/{id} | ID alapján |

| Bérlés törlése | DELETE | /api/rentals/{id} | ID alapján |

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

| Összes rental item lekérése| GET | /api/rental-items | Minden rental item |

| Új rental item létrehozása | POST | /api/rental-items | Új rental item |

| Egy rental item lekérése | GET | /api/rental-items/{id} | ID alapján |

| Rental item frissítése | PUT | /api/rental-items/{id} | ID alapján |

| Rental item törlése | DELETE | /api/rental-items/{id} | ID alapján |

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

POST POST | /api/rentals | Új rental mentése

{

"userId": "abc123",

"start\_date": "2025-06-01",

"end\_date": "2025-06-05",

"rentalItems": [

{

"jewelry\_id": 1,

"daily\_price": 120.50

},

{

"jewelry\_id": 2,

"daily\_price": 85.75

}

]

}

update PUT  
 "userId": "abc123",

"start\_date": "2025-06-01",

"end\_date": "2025-06-05",

"rentalItems": [

{

"jewelry\_id":3,

"daily\_price": 120.50

}

]

}

add berles

{

"userId": "abc123",

"start\_date": "2025-06-01",

"end\_date": "2025-06-05",

"rentalItems": [

{

"jewelry\_id": 3,

"daily\_price": 120.50

}

]

}

rentalItems Add

{

"userId": "abc123",

"start\_date": "2025-07-01",

"end\_date": "2025-07-10",

"rentalItems": [

{

"jewelry\_id": 1,

"daily\_price": 150.00

},

{

"jewelry\_id": 2,

"daily\_price": 180.50

}

]

}