* composer telepítése

**Parancssor:**

composer global require laravel/installer

**mappa cmd :**

laravel new <projekt neve>

ha nincs laravel => composer create-project laravel/laravel **04\_databaseCars**

* php artisan serve

**User táblák törlése:**

migration mappa tartalmának törlése

/config/sanctum.php törlése

/app/Models/User.php törlése

/database/factories/UserFactory.php törlése

composer remove laravel/sanctum

php artisan clear

routes/api.php:

// Route::middleware('auth:sanctum')->get('/user', function (Request $request) {

//     return $request->user();

// });

**Model osztály létrehozás:**

php artisan make:model Task -m

**Migráció:**

database\migrations\2022\_03\_22\_112156\_create\_tasks\_table.php:

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

            $table->id();

**$table->string('title');**

**$table->string('description');**

            $table->timestamps();

        });

<https://laravel.com/docs/9.x/migrations#available-column-types>

.env:

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=**taskmanager**

DB\_USERNAME=root

DB\_PASSWORD=

php artisan migrate

**Tesztadatok készítése:**

php artisan make:factory TaskFactory -m Task

<https://fakerphp.github.io/>

database\seeders\DatabaseSeeder.php:

public function run()

    {

        // \App\Models\User::factory(10)->create();

**\App\Models\Task::factory(50)->Create();**

    }

php artisan db:seed

**Kontroller létrehozás:**

php  artisan make:controller TaskController --api -m Task

**Routok megadása:**

routes\api.php:

**Route::apiResource('tasks',TaskController::class);**

**Kontroller függvények:**

\app\Http\Controllers\TaskController.php:

use App\Models\Task;

use Exception;

use Illuminate\Http\Request;

**use Illuminate\Support\Facades\Validator;**

 public function index()

    {

**$tasks = Task::all();  // összes rekord**

**return response()->json($tasks);**

    }

public function show($id)

    {

**try {**

**$task = Task::find($id);**

**if (!empty($task)){**

**return response()->json($task); // minden ok**

**} else {**

**return response()->json(['message'=>'Item not found.'],404);**

**}**

**} catch(Exception $e) {**

**return response()->json(['message'=>'Database error.'],400);**

**}**

    }

 public function store(Request $request)

    {

        // validálás

**$v = $this->fieldValidation($request);**

**if ($v != '') {**

**return response()->json($v,400);**

**}**

**try{**

**$task = new Task();**

**$task->title = $request->input('title');**

**$task->description = $request->input('description');**

**$task->save(); // insert into**

**return response()->json(**

**[**

**'message' => 'Item was created',**

**'id' => $task->id**

**],201**

**);**

**} catch(Exception $e) {**

**return response()->json(['message'=>'Database error.'],400);**

**}**

    }

public function update(Request $request, $id)

    {

**// validálás**

**$v = $this->fieldValidation($request);**

**if ($v != '') {**

**return response()->json($v,400);**

**}**

**try {**

**if (Task::where('id','=',$id)->exists()){**

**$task = Task::find($id);**

**$task->title = $request->input('title');**

**$task->description = $request->input('description');**

**$task->save(); // update**

**return response()->json(['message'=>'Item was updated.'],200);**

**} else {**

**return response()->json(['message'=>'Item not found.'],404);**

**}**

**} catch(Exception $e) {**

**return response()->json(['message'=>'Database error.'],400);**

**}**

    }

 public function destroy($id)

    {

**try{**

**if (Task::where('id','=',$id)->exists()){**

**$task = Task::find($id);**

**$task->delete();**

**return response()->json(['message'=>'Item was deleted.'],200);**

**}else {**

**return response()->json(['message'=>'Item not found.'],404);**

**}**

**} catch(Exception $e) {**

**return response()->json(['message'=>'Database error.'],400);**

**}**

    }

**public function fieldValidation(Request $request)**

**{**

**//** [**https://laravel.com/docs/9.x/validation#available-validation-rules**](https://laravel.com/docs/9.x/validation#available-validation-rules)

**$validator = Validator::make(**

**$request->all(),**

**[**

**'title' => 'required|min:10',**

**'description' => 'required'**

**],**

**[**

**'title.required' => 'A title kötelező!',**

**'title.min' => 'A title legalább 10 karakter kell, hogy legyen!',**

**'description.required' => 'A description kötelező!'**

**]**

**);**

**if ($validator->fails()) {**

**return $validator->messages();**

**}**

**return '';**

**}**