# Kizöldítjük a Földet!

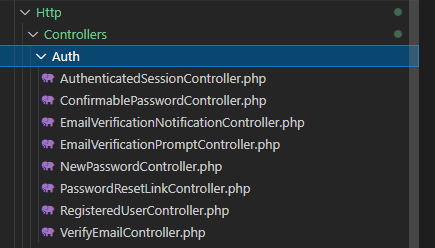
## projekt megvalósításának lépési Laravellel

### Alapfeltételek:

PHP 8 + Adatbázis (XAMPP csomag tartalmazza, DB: MySQL)

Töltsd le a kiindulási laravel projektet innen: <https://github.com/csefikatalin/alap_laravel.git>

A projekt tartalmazza az autentikciót is.



Eztuán indíthatjuk a projektünket:

**composer install**

**xamp indítása**

**localhost/phpmyadmin: adatbázis létrehozása**

**.env.example** fájl tartalmának másolása .env fájlba

**.env** módosítása, hogy a kapcsolódási paraméterek legyenek élesek (DB kapcsolódási paraméterek beállítása)

**php artisan serve**

**A projekt rendszerint a localhost:8000 proton érhető el.**

## feladat

Kizöldítjük a Földet! címmel weblapot készítünk, amelyben rögzítheti minden osztály minden diákja, hogy milyen tevékenységeket végzett a fenntarthatóság kapcsán.

A felhasználók és a lehetséges tevékenységek az adatbázisban előre rögzítve vannak, tehát nem feladat ezek rögzítésére felület létrehozása.

Az osztályokat pedig a JS kódban rögzítjük.

**Lehetséges tevékenységek:**

* kerékpárral jöttem iskolába
* rollerrel jöttem iskolába
* 10 km-t gyalogoltam buszozás helyett
* ültettem fát
* ültettem virágot
* ültettem egyéb növényt
* kevesebb vizet használtam a fürdéshez
* összeszedtem a szemetet egy közterületen, erdőben, stb
* tartós szatyorba vásároltam, nem nylonba
* nem használtam egyszer használatos műanyagot
* tartós csomagolású terméket vásároltam – pl üvegbe vettem a tejet, nem használtam pet palackot,
* környezetbarát csomagolású terméket vásároltam
* kevesebb húst ettem a héten
* ökológiai gazdaságból származó élelmiszert vettem
* kirándultam, szabadban töltöttem időt a héten
* kevesebb ruhát/terméket vásároltam a héten, hogy fenntarthatóbb legyen a környeztünk!
* önkénteskedtem a Greenpeace-nél, vagy más zöld szervezetnél

Minden tevékenységhez tartozik egy pontszám is.

Ez alapján versenyt hirdethetünk az osztályok között a  **Legzöldebb Osztály** címért.

Amikor a felhasználó egy tevékenységet rögzít a rendszerben, akkor növekszik az osztály összpontszáma a kiválasztott tevékenységnek megfelelően.

Az oldal megjeleníti az összes osztály összes pontszámát egy táblázatban (+++ illetve Google Chart-tal),

A táblázat alatt egy listában jelennek meg a rögzített tevékenységek.

++A listát lehet szűrni osztály szerint, illetve rendezni pontszám szerint

++++Az oldalhoz tartozik egy admin felület, ahol az osztályhoz tartozó tanár jóvá tudja hagyni a tevékenységeket. (ez nincs kész 🡪 kellene a User táblába egy isTanar boolean)

Minden ***új bejegyzéshez***  tartozik:

* + - osztály
    - bejegyzés neve
    - állapot (jóváhagyásra vár/jóváhagyott)

### Táblák

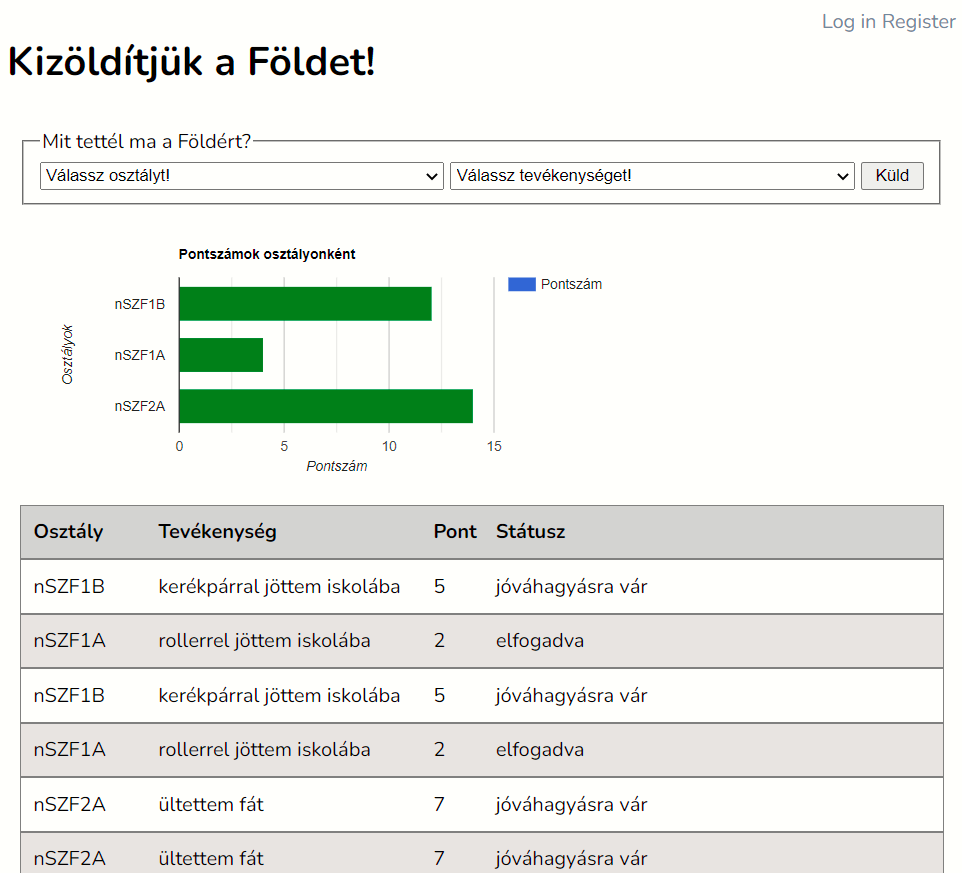
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BEJEGYZES |  | TEVEKENYSEG |  | USER |
| id |  | tevekenyseg\_id |  | id |
| tevekenyseg\_id |  | tevekenyseg\_nev |  | name |
| osztaly\_id |  | pontszam |  | password |
| allapot |  |  |  | email |
|  |  |  |  | osztaly\_id |
|  |  |  |  |  |

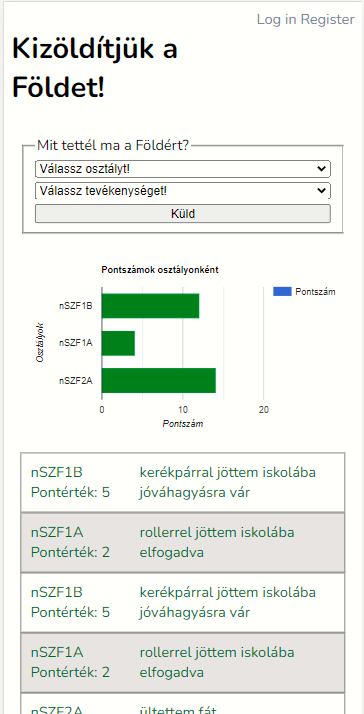
#### API végpontok és feladatok

GET all /bejegyzesek Visszaadja az összes bejegyzést a tevékenység nevével és pontszámával együtt

GET osztaly\_id /bejegyzesek/osztaly\_id Visszaadja egy adott osztály összes bejegyzését a tevékenység nevével és pontszámával együtt

POST /bejegyzes Új bejegyzést rögzíthetünk a táblában





## Lépésről lépésre

1. Indítsuk el a Xamp-ot.
2. Hozzunk létre egy **-–** adatbázist
3. Ellenőrizzük, hogy a .env fájlban ez az adatbázisnév van megadva!

A képen szöveg, monitor, képernyő, képernyőkép látható

Automatikusan generált leírás

### Adatbázis táblák létrehozása

1. Létre kell hozni egy-egy modelt a táblákhoz

php artisan make:model Tevekenyseg -m

php artisan make:model Bejegyzes -m

1. Létre kell hozni egy-egy controllert is a táblához

php artisan make:controller TevekenysegController -m

php artisan make:controller BejegyzesController -m

**Fájl elérési helyek:**

migrációs fájlok: database/migrations

model fájlok: app/Models

controller fájlok: app/Http/Controllers

view fájlok: resources/views

1. **database/migration** útvonalon létrejött egy új migrációs fájl egy CreateValamilyennevTable osztállyal.   
   Az **up()** metódusában megadjuk a táblákban használni kívánt mezőneveket a típusukkal és a méretükkel együtt. A mezőtípusokhoz itt találunk útmutatót: <https://laravel.com/docs/7.x/migrations>

TEVEKENYSEG

   public function up()

    {

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

            $table->id();

            $table->timestamps();

            $table->string('tevekenyseg\_nev',50);

            $table->integer('pontszam');

        });

    }

BEJEGYZES

   public function up()

    {

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

             $table->id();

            $table->timestamps();

            $table->integer('allapot'); //1 - jóváhagyva, 0 - nincs jóváhagyva

            $table->integer('tevekenysegId');

            $table->integer('osztalyokID');

        });

    }

1. A tábla migrálása:   
   php artisan migrate
2. Ha módosul a tábla mezője, akkor vissza kell vonni:   
   php artisan migrate:rollback  
   Majd módosítani a tábla mezőit és újra futtatni (3. 4. pont)
3. ~~Tábla módosítása: Új migrációt kell létrehozni, de most update-tel:  
   php artisan make:migration update\_user\_table  
   Az up() metódusba kell írni a módosítást, az új mezőt.~~
4. ~~Majd a módosítást is migrálni kell  
   php artisan migrate~~

### Api útvonalak előállítása

1. routes/web.php fájlban a Route-olás elkészítése
   1. Csatoljuk a BejegyzésekControllert és a tevekenysegekControllert!

/\*\* Bejegyzések \*/

Route::get('/bejegyzesek', [BejegyzesekController::class, 'index']);

Route::get('/bejegyzesek/{osztaly\_id}', [BejegyzesekController::class, 'osztaly']);

Route::post('/bejegyzes', [BejegyzesekController::class, 'store']);

/\*\*Tevekenységek \*/

Route::get('/tevekenysegek', [TevekenysegController::class, 'index']);

### A Bejegyzes.php – Modell

    public function tevekenyseg()

    {

        return $this->hasOne(Tevekenysegek::class, 'id', 'tevekenysegId');

    }

### A TevekenysegController

    public function index()

    {

        // $tasks = response()->json(Task::all());

        $tevekenysegek = response()->json(Tevekenysegek::all());

        return $tevekenysegek;

    }

### BejegyzesController

    public function index()

    {

        $bejegyzesek = response()->json(Bejegyzesek::with('tevekenyseg')->get());

        return $bejegyzesek;

    }

    public function osztaly($id)

    {

        $bejegyzesek = Bejegyzesek::with('tevekenyseg')->where('osztalyokID', $id)->get(); //azon bejegyzések listája (összekapcs. a Tevekenyseg táblával), ahol az osztalyID a bemenő osztály id-val megegyezik 🡪 tehát adott osztály által beszúrt bejegyzések listája

        return $bejegyzesek;

    }

    public function store(Request $request)

    {

        echo $request;

        $bejegyzesek = new Bejegyzesek();

        $bejegyzesek->allapot = 0;

        $bejegyzesek->tevekenysegId = $request->tevekenysegId;

        $bejegyzesek->osztalyokID = $request->osztalyokID;

        $bejegyzesek->save();

        return Bejegyzesek::find($bejegyzesek->id);

    }

# Frontend

### lépés: A html fájlok fejlécébe helyezd el a tokenhez tartozó meta taget:

 <meta name="csrf-token" content=<?php $token = csrf\_token();

                                    echo $token; ?>>

### lépés: Abban a scriptben, ahol példányosítod az ajax hívások osztályát, hozz létre egy token constanst, és ezzel példányosítsd az ajax hívások osztályát.

  const token = $('meta[name="csrf-token"]').attr("content");

    const myAjax = new MyAjax(token);

### lépés: A MyAjax osztály konstruktora:

 class MyAjax {

        constructor(token) {

            this.token = token;

        }

### lépés: a PUT, POST, DELETE metódusok esetében az ajax hívás fejlécébe el kell helyezni a tokent a headers-ben:

 postAjax(apivegpont, ujtermek) {

            ujtermek.\_token = this.token;

            $.ajax({

                headers: { "X-CSRF-TOKEN": this.token },

                url: apivegpont,

                type: "POST",

                data: ujtermek,

                success: function (result) {},

                error: function (result) {

                    console.log(result);

                },

            });

        }

ÉS kész. Ezzel elértük azt, hogy csak a session által generált token birtokában tudjuk módosítani az adatokat.

## Globális változó

const osztalyokTmb = [

    "Válassz tevékenységet!",

    "nSZF1A",

    "nSZF1B",

    "nSZF2A",

    "nIRU1A",

    "nIRU1B",

    "nIRU2A",

];

## MyAjax osztály

Feladata az api kérések indítása és a válaszok fogadása, továbbküldése.

class MyAjax {

    constructor(token) {

        this.token = token;

    }

    getAjax(apivegpont, myCallback) {

        const tomb = [];

        $.ajax({

            url: apivegpont,

            type: "GET",

            success: function (result) {

                result.forEach((element) => {

                    tomb.push(element);

                });

                myCallback(tomb);

            },

        });

    }

    postAjax(apivegpont, adat) {

        adat.\_token = this.token;

        $.ajax({

            headers: { "X-CSRF-TOKEN": this.token },

            url: apivegpont,

            type: "POST",

            data: adat,

            success: function (result) {

                console.log(result);

            },

        });

    }

}

## UrlapKezeles osztály

Feladata, hogy a feltöltse a legördülő listákat adattal, illetve, hogy kezelje a küldés gombra kattintást.

class UrlapKezeles {

    constructor() {

        this.osztalyokSelect = $("#osztalyok");

        this.tevekenysegekSelect = $("#tevekenysegek");

        console.log(this.tevekenysegekSelect);

        this.optionOsztalyokFeltolt(osztalyokTmb);

        $("#kuld").on("click", () => {

            const adat = {

                osztalyokID: $("#osztalyok").val(),

                tevekenysegId: $("#tevekenysegek").val(),

            };

            console.log(adat);

            this.kattintasTrigger(adat);

        });

    }

    kattintasTrigger(adat) {

        let event = new CustomEvent("kuldKattintas", {

            detail: adat, //ezzel adatokat tudok átadni

        });

        window.dispatchEvent(event);

    }

    //tevékenységek feltöltése

    optionTevekenysegFeltolt(tomb) {

        console.log(tomb);

        tomb.forEach((element) => {

            $("#tevekenysegek").append(

                `<option value='${element.id}'>${element.tevekenyseg\_nev}</option>`

            );

        });

    }

    optionOsztalyokFeltolt(tomb) {

        console.log(tomb);

        tomb.forEach((element, index) => {

            this.osztalyokSelect.append(

                `<option value='${index}'>${element}</option>`

            );

        });

    }

}

## Bejegyzes osztály

Feladata, hogy megjelenítsen egyetlen bejegyzést

class Bejegyzes {

    constructor(adat) {

        this.adat = adat;

        const szuloelem = $("#adatok");

        let allapot = adat.allapot === 0 ? "jóváhagyásra vár" : "";

        const elem = `<div class="bejegyzes">

               <div class="osztaly">${osztalyokTmb[adat.osztalyokID]}</div>

               <div class="tevekenyseg">${

                   adat.tevekenyseg.tevekenyseg\_nev

               }</div>

               <div class="pontszam">${adat.tevekenyseg.pontszam}</div>

               <div class="allapot">${

                   adat.allapot === 0 ? "jóváhagyásra vár" : "elfogadva"

               }</div>

       </div>`;

        szuloelem.append(elem);

        this.node = szuloelem.children(".bejegyzes:last-child");

    }

}

## Bejegyzesek osztály

Feladata, hogy példányosítson annyi bejegyzést, amennyit a szervertől válaszban kaptunk.

class Bejegyzesek {

    constructor() {}

    bejegyzesFeldolgoz(tomb) {

        tomb.forEach((elem) => {

            new Bejegyzes(elem);

        });

    }

}

## A főprogram:

$(function () {

    const token = $('meta[name="csrf-token"]').attr("content");

    const myAjax = new MyAjax(token);

    const urlapKezeles = new UrlapKezeles();

    const bejegyzesek = new Bejegyzesek();

    let bejegyzesekVegpont = "/bejegyzesek";

    let tevekenysegekVegpont = "/tevekenysegek";

    myAjax.getAjax(bejegyzesekVegpont, bejegyzesek.bejegyzesFeldolgoz);

    myAjax.getAjax(bejegyzesekVegpont, adatfeldolgoz);

    myAjax.getAjax(tevekenysegekVegpont, urlapKezeles.optionTevekenysegFeltolt);

    $(window).on("kuldKattintas", (event) => {

        let adat = event.detail;

        let bejegyzesVegpont = "/bejegyzes";

        myAjax.postAjax(bejegyzesVegpont, adat);

        myAjax.getAjax(bejegyzesekVegpont, bejegyzesek.bejegyzesFeldolgoz);

        myAjax.getAjax(bejegyzesekVegpont, adatfeldolgoz);

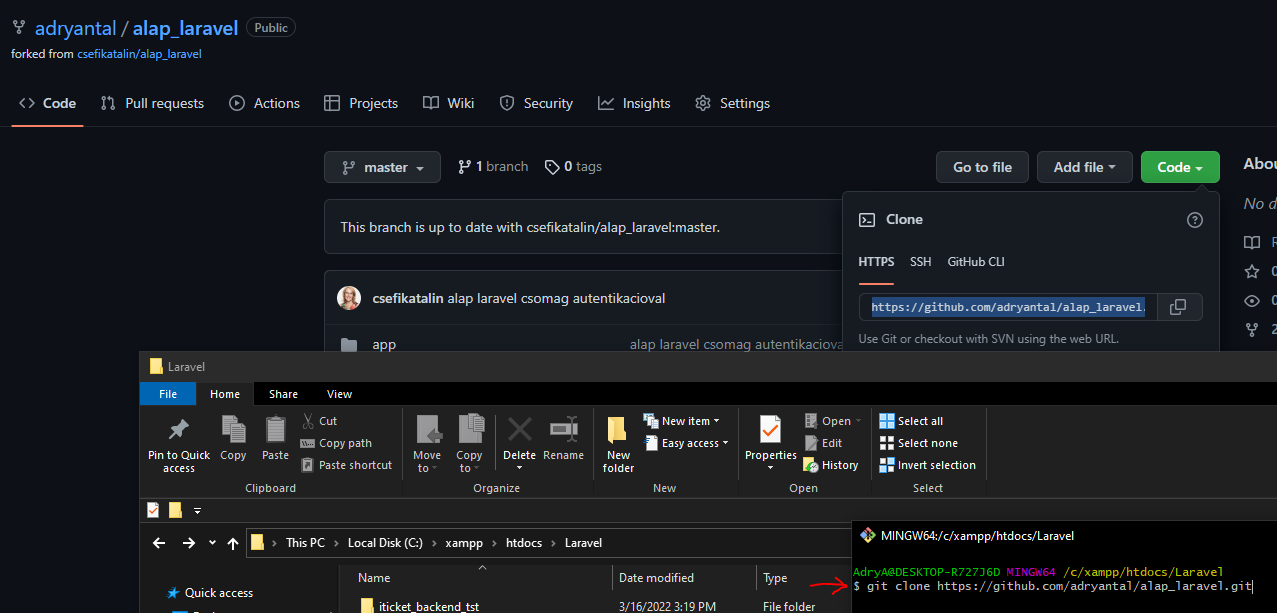
    });

});

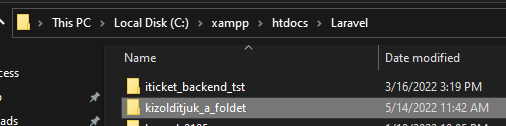
Saját megoldási menet vázlata

## BACKEND

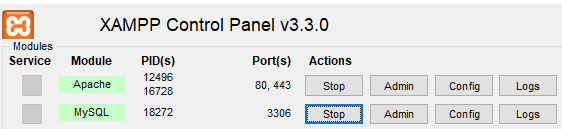
# Projekt klónozása lokális könyvtárba és előkészítése



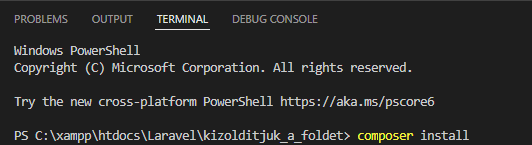
Tetszőlegesen átnevezem a létrejött lokális mappát:



Elindítjuk a XAMPP-ot.

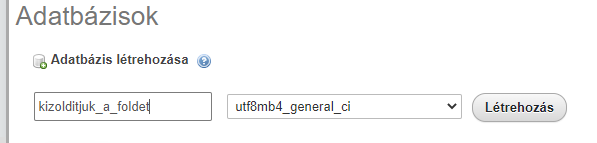


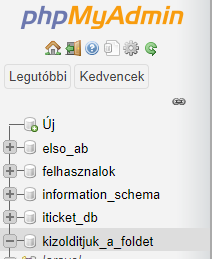
A függőségek telepítése végett futtatjuk a VS Studio Code terminálján keresztül a *composer install* parancsot:



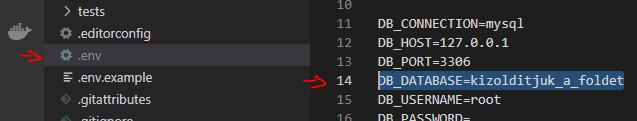
# .env fájl lérehozása, módosítása

Adatbázis létrehozása a <https://localhost/phpmyadmin/> alatt:





.env.example 🡪 CTRL C+ CTRL V🡪 a létrejvő copy fájl átnevezése .env-re 🡪 DB\_DATABASE értékét átírjuk az adatbázisnévre



# Model és migration

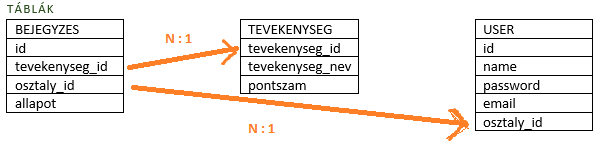
### Táblakapcsolatok

Egy tevékenységhez több bejegyzés tartozhat, míg egy bejegyzés csak egy tevékenységhez kapcsolódhat, ezért: **BEJEGYZES : TEVEKENYSEG 🡪 N : 1**

Egy tevékenységhez több bejegyzés tartozhat, míg egy bejegyzés csak egy tevékenységhez kapcsolódhat.

Egy user/osztály több bejegyzést is beszúrhat, míg egy bejegyzést csak egy személy (osztály) vihet fel, , ezért:

**BEJEGYZES : USER 🡪 N : 1**



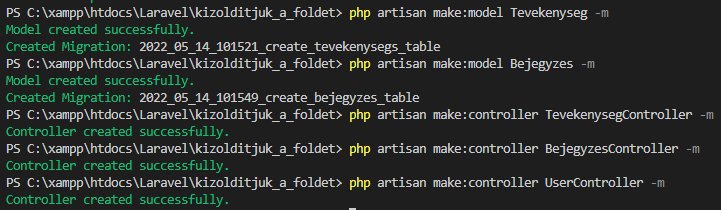
Megjegyzés: a feladat leírása szerint így lenne jó, de a leírt megoldásból az derül ki, hogy a USER táblára valójában nincs is szükség.

### Létre kell hozni egy-egy modelt a táblákhoz

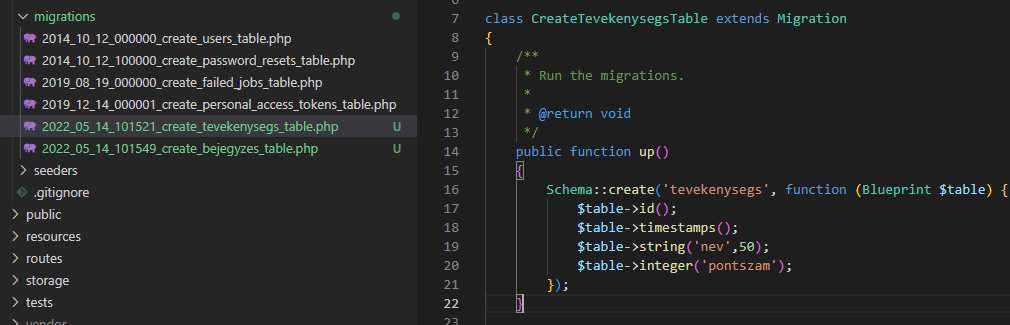
* + php artisan make:model Tevekenyseg -m
  + php artisan make:model Bejegyzes -m
  + A User model már alapból létezik, a Laravel default-ként adja

### Létre kell hozni controllert is a tábláKhoz

* + php artisan make:controller TevekenysegController -m
  + php artisan make:controller BejegyzesController -m
  + php artisan make:controller UserController -m



### Tevékenységek tábla



public function up()

    {

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

            $table->id();

            $table->timestamps();

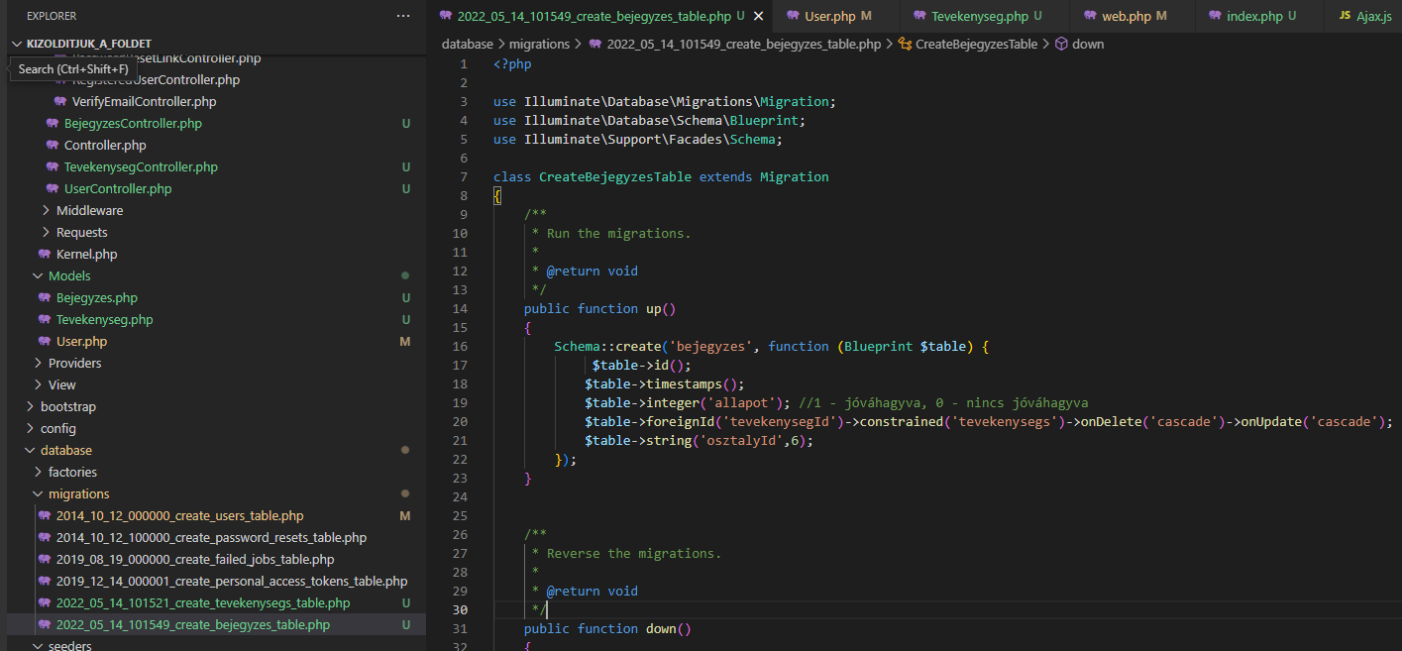
            $table->string('nev',50);

            $table->integer('pontszam');

        });

    }

### Bejegyzések tábla



  public function up()

    {

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

             $table->id();

            $table->timestamps();

            $table->integer('allapot'); //1 - jóváhagyva, 0 - nincs jóváhagyva

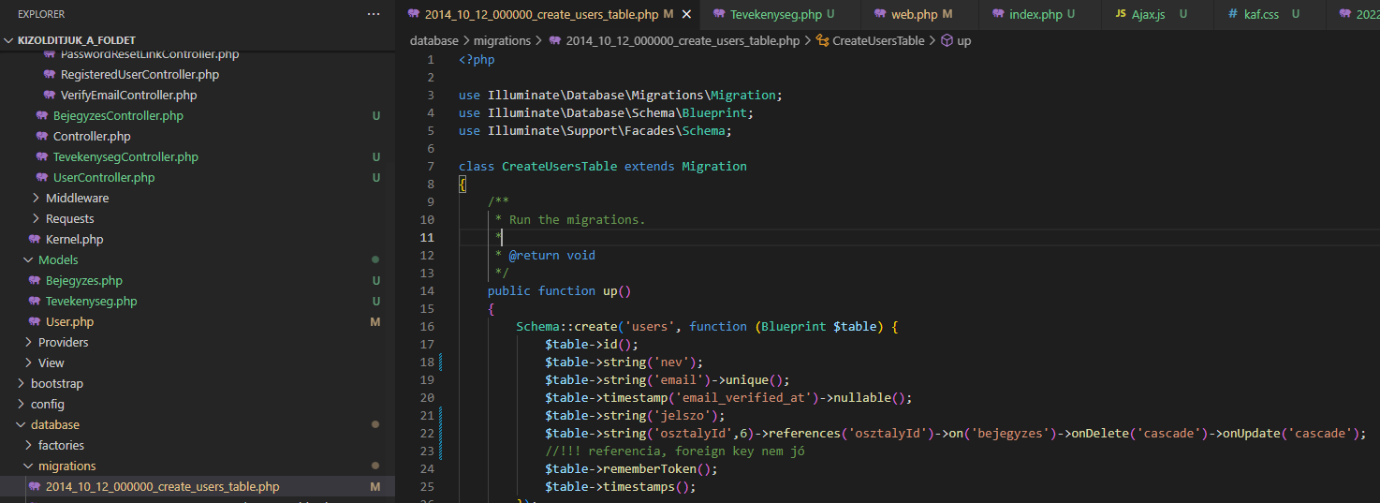
            $table->foreignId('tevekenysegId')->constrained('tevekenysegs')->onDelete('cascade')->onUpdate('cascade');

            $table->string('osztalyId',6);

        });

    }

### UserS Tábla



    public function up()

    {

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

            $table->id();

            $table->string('nev');

            $table->string('email')->unique();

            $table->timestamp('email\_verified\_at')->nullable();

            $table->string('jelszo');

            $table->string('osztalyId',6)->references('osztalyId')->on('bejegyzes')->onDelete('cascade')->onUpdate('cascade'); //!!! referencia, foreign key-es változat nem jó

            $table->rememberToken();

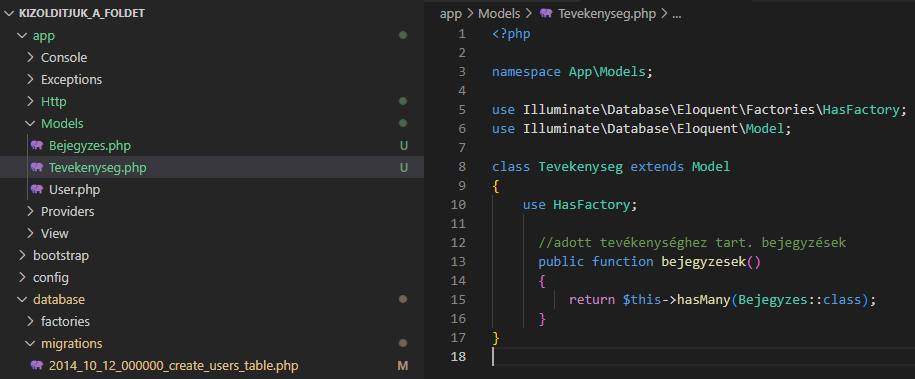
            $table->timestamps();

        });

    }

### Táblakapcsolatok beállítása a model-ekben

Tevekenyseg : Bejegyzes 🡪 1:N



class Tevekenyseg extends Model

{

    use HasFactory;

      //adott tevékenységhez tart. bejegyzések

      public function bejegyzesek()

      {

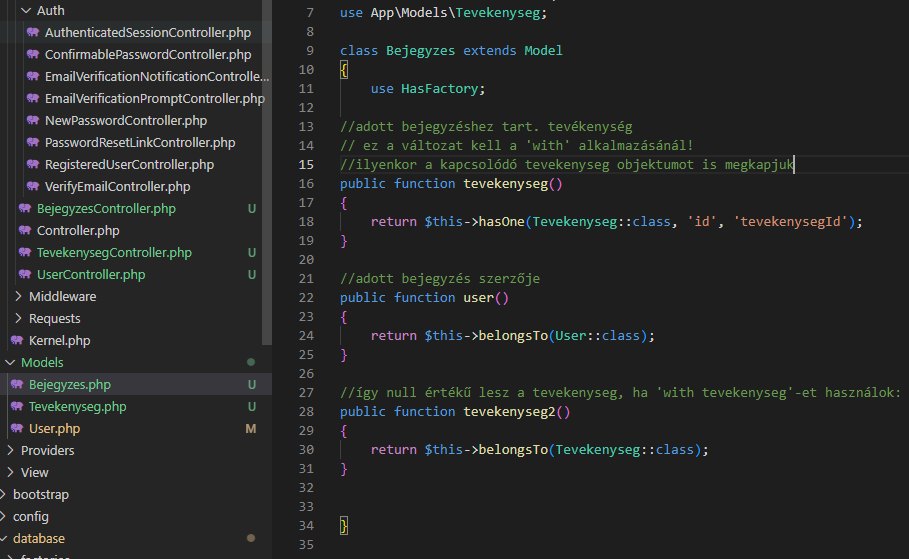
          return $this->hasMany(Bejegyzes::class);

      }

}

Bejegyzes:Tevekenyseg 🡪 N:1

Bejegyzes : User 🡪 N:1



class Bejegyzes extends Model

{

    use HasFactory;

//adott bejegyzéshez tart. tevékenység

// ez a változat kell a 'with' alkalmazásánál!

//ilyenkor a kapcsolódó tevekenyseg objektumot is megkapjuk

public function tevekenyseg()

{

    return $this->hasOne(Tevekenyseg::class, 'id', 'tevekenysegId');

}

//adott bejegyzés szerzője

public function user()

{

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

}

//így null értékű lesz a tevekenyseg, ha 'with tevekenyseg'-et használok:

public function tevekenyseg2()

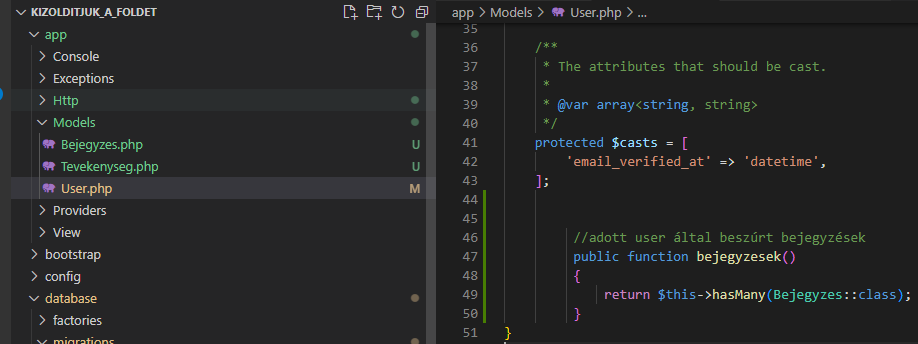
{

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

}

}

User : Bejegyzes 🡪 1:N



 //adott user által beszúrt bejegyzések

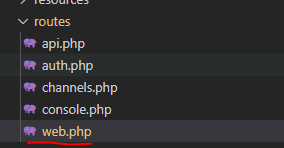
         public function bejegyzesek()

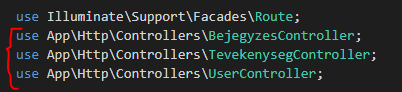
         {

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

         }

# ÚTVONALAK





/\* Bejegyzések \*/

Route::get('/bejegyzes/osszes', [BejegyzesController::class, 'index']); //összes bejegyzés kilistázása a tevékenységobjektumokkal együtt

Route::get('/bejegyzesek/{osztalyId}', [BejegyzesController::class, 'osztalyBejegyzesei']); //adott osztály által beszúrt bejegyzések

Route::post('/bejegyzes/beszur', [BejegyzesController::class, 'store']); //új bejegyzés beszúrása

/\* Tevekenységek \*/

Route::get('/tevekenyseg/osszes', [TevekenysegController::class, 'index']);

Route::get('/osztalyok', [UserController::class, 'osztalyok']); //osztályId-k lekérése a dropdown-ba történő betöltéshez

/\* Diagramhoz \*/

Route::get('/statisztika', [BejegyzesController::class, 'statisztika']);

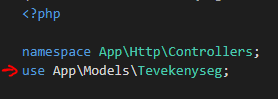
/\*Plussz\*/

//bejegyzés módosítása

//bejegyzés törlése

# Controllerek

### Tevekenysegcontroller



class TevekenysegController extends Controller

{

    //összes tevékenység

    public function index()

    {

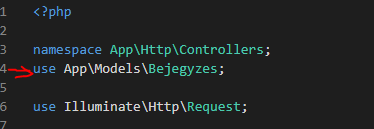
        $tevekenysegek = response()->json(Tevekenyseg::all());

        return $tevekenysegek;

    }

}

### BEJEGYZESCONTROLLER



class BejegyzesController extends Controller

{

    //összes bejegyzés a tevékenységobjektumokkal együtt

    public function index()

    {

        $bejegyzesek = response()->json(Bejegyzes::with('tevekenyseg')->get());

        return $bejegyzesek;

    }

   //adott osztály által beszúrt bejegyzések listája

    public function osztalyBejegyzesei($osztalyId)    {

        $bejegyzesek = Bejegyzes::with('tevekenyseg')->where('osztalyId', $osztalyId)->get();

                //azon bejegyzések listája (összekapcs. a Tevekenyseg táblával), ahol az osztalyID a bemenő osztály id-val egyezik meg

        return $bejegyzesek;

    }

   //új bejegyzés beszúrása

    public function store(Request $request)

    {

        echo $request;

        $bejegyzes = new Bejegyzes();

        $bejegyzes->allapot = 0;

        $bejegyzes->tevekenysegId = $request->tevekenysegId;

        $bejegyzes->osztalyId = $request->osztalyId;

        $bejegyzes->save();

        return Bejegyzes::find($bejegyzes->id);

    }

}

 //összpontszám lekérdezése osztályonként

    public function statisztika() {

        $osszPontszamOsztalyonkent = response()->json(DB::table('bejegyzes AS b') //query buildert indítok

        ->join('tevekenysegs AS t','b.tevekenysegId','t.id') //összekapcsolom a tevekenysegs táblával

        ->select('osztalyId', DB::raw('SUM(t.pontszam) as osszpontszam'))->groupBy(DB::raw("osztalyId")) //pontszám szummázása osztályonként

        ->get()

        );

        return $osszPontszamOsztalyonkent;

    }

### USercontroller

<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\DB;

use Illuminate\Http\Request;

use App\Models\User;

class UserController extends Controller

{

    public function osztalyok(){

        $osztalyok = response()->json(DB::table('users')

        ->select('osztalyId')

        ->distinct()->get()

        );

        return $osztalyok;

    }

}

# MIGRÁCIÓ

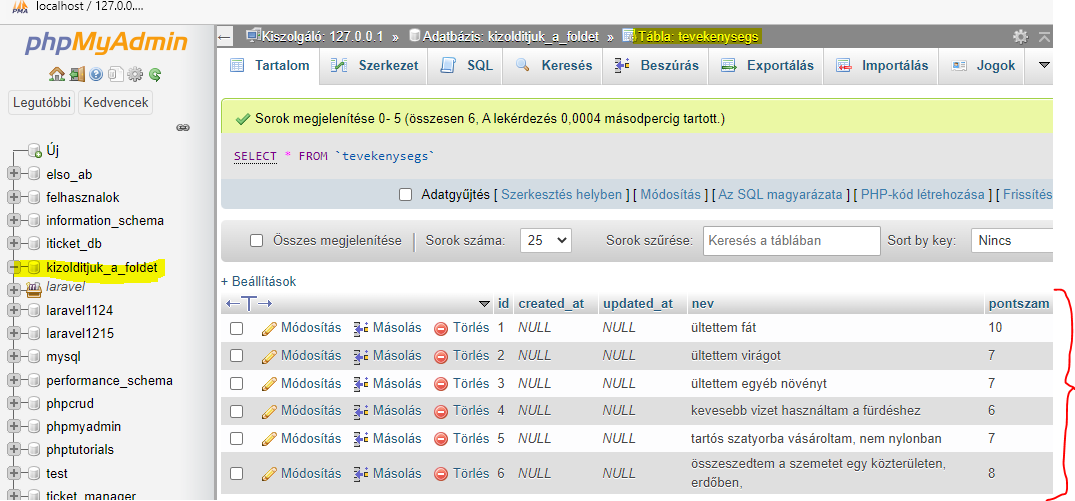
php artisan migrate

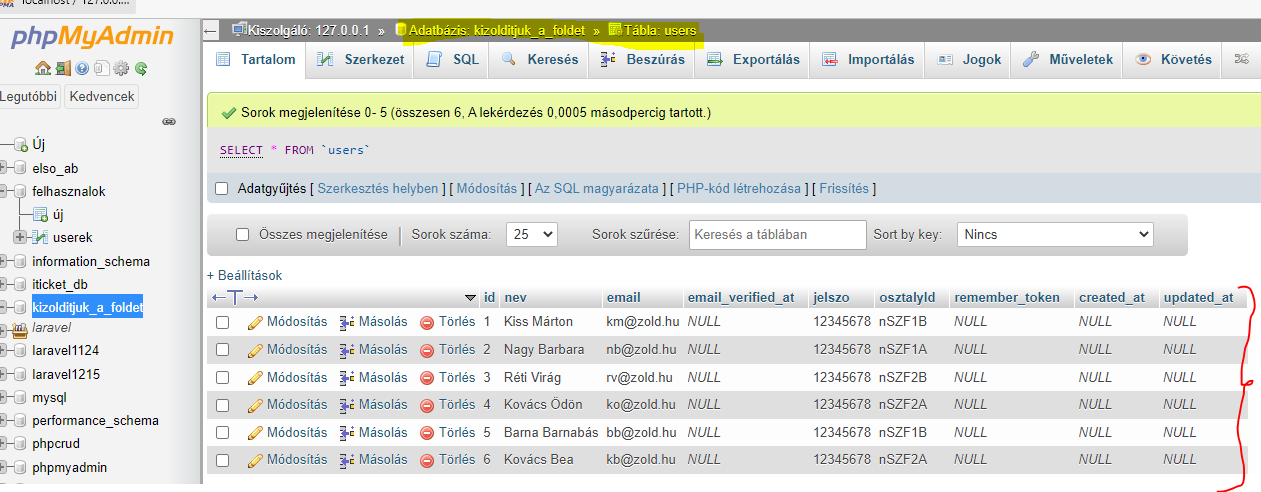
Változtatás esetén: php artisan migrate:fresh

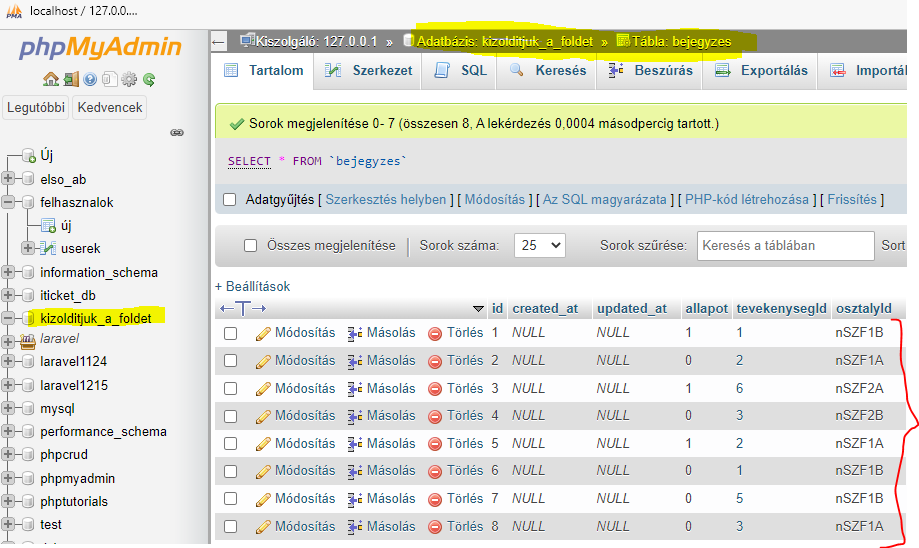
! SQL-t érdemes lementeni minden fresh előtt, amennyiben az adatbázis már tartalmaz tesztadatokat, és utólag beimportálni őket (a változtatások figyelembevételével)

# BACKEND TESZTELÉSE

Tesztadatok felvitele



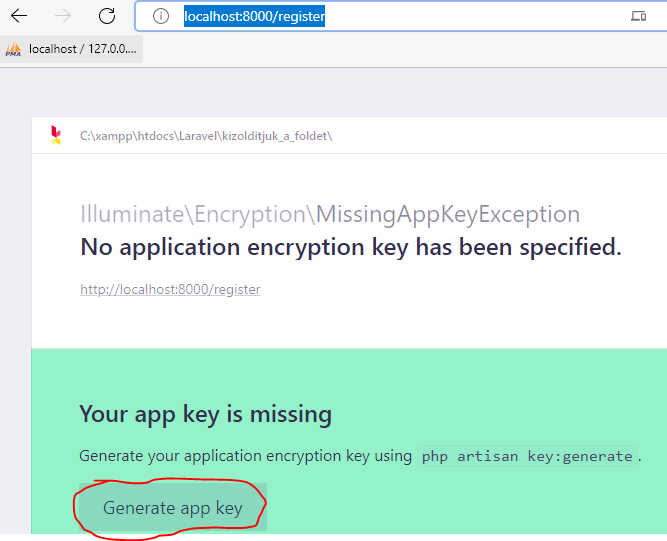


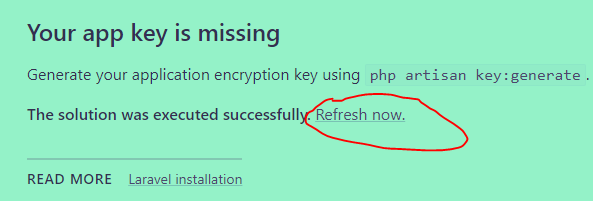


Beírjuk a terminálba *a php artisan serve* parancsot:

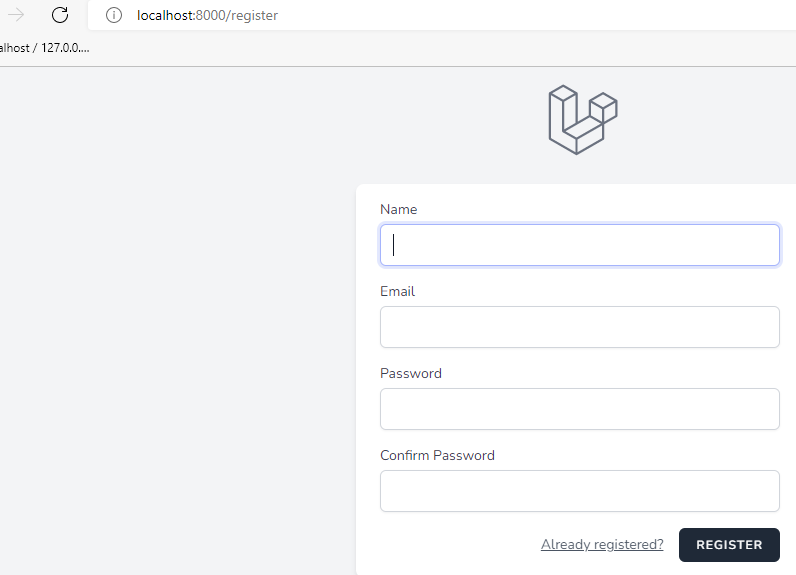


Beírom pl. a http://localhost:8000/register linket, mivel az a Laravel beépített authentikációs szolgáltatásának előzetes installációja miatt elérhető:





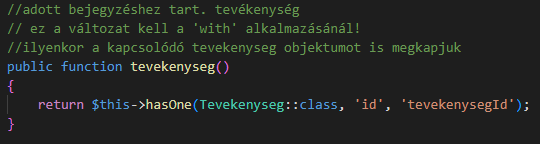
Működik:



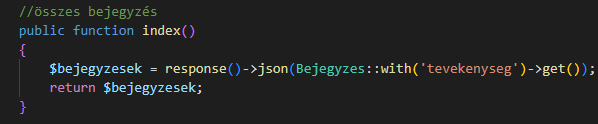
### Útvonalak tesztelése:

**Controllerben a ’with’ ill. model-ben a hasOne haszálatával:**

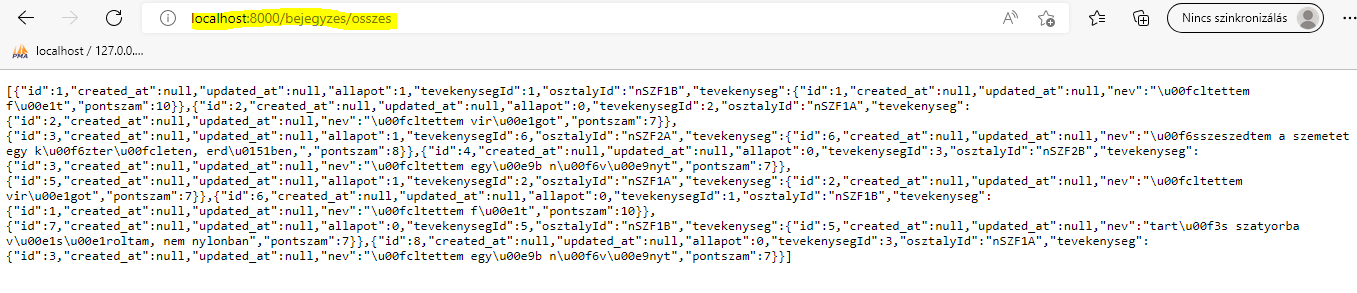
model:



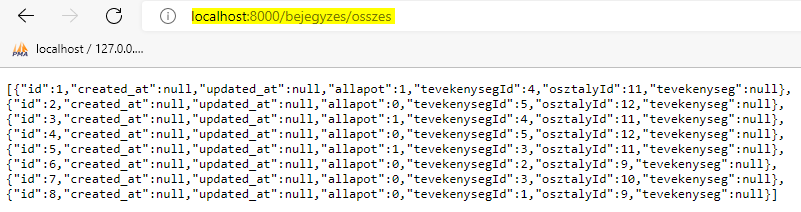
Controller:



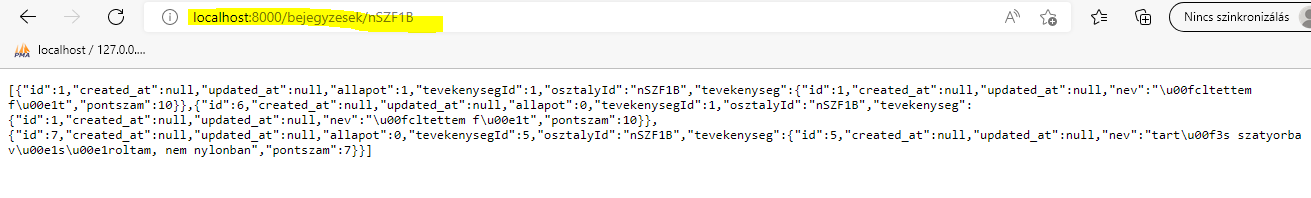
eredmény, összes bejegyzés a kapcsolódó tevékenységobjektumokkal együtt:



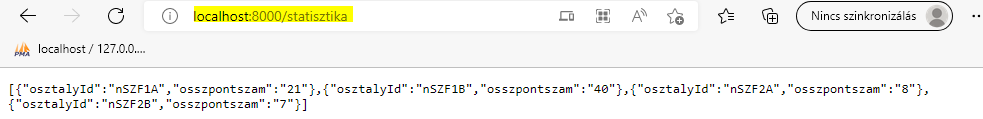
(ez akkor van, ha a ’with’-et használom a Controllerben, és a viszont Modelben csak simán a belongsTo-t; a tevekenysegek objektum null értékű lesz)



adott osztály bejegyzései:



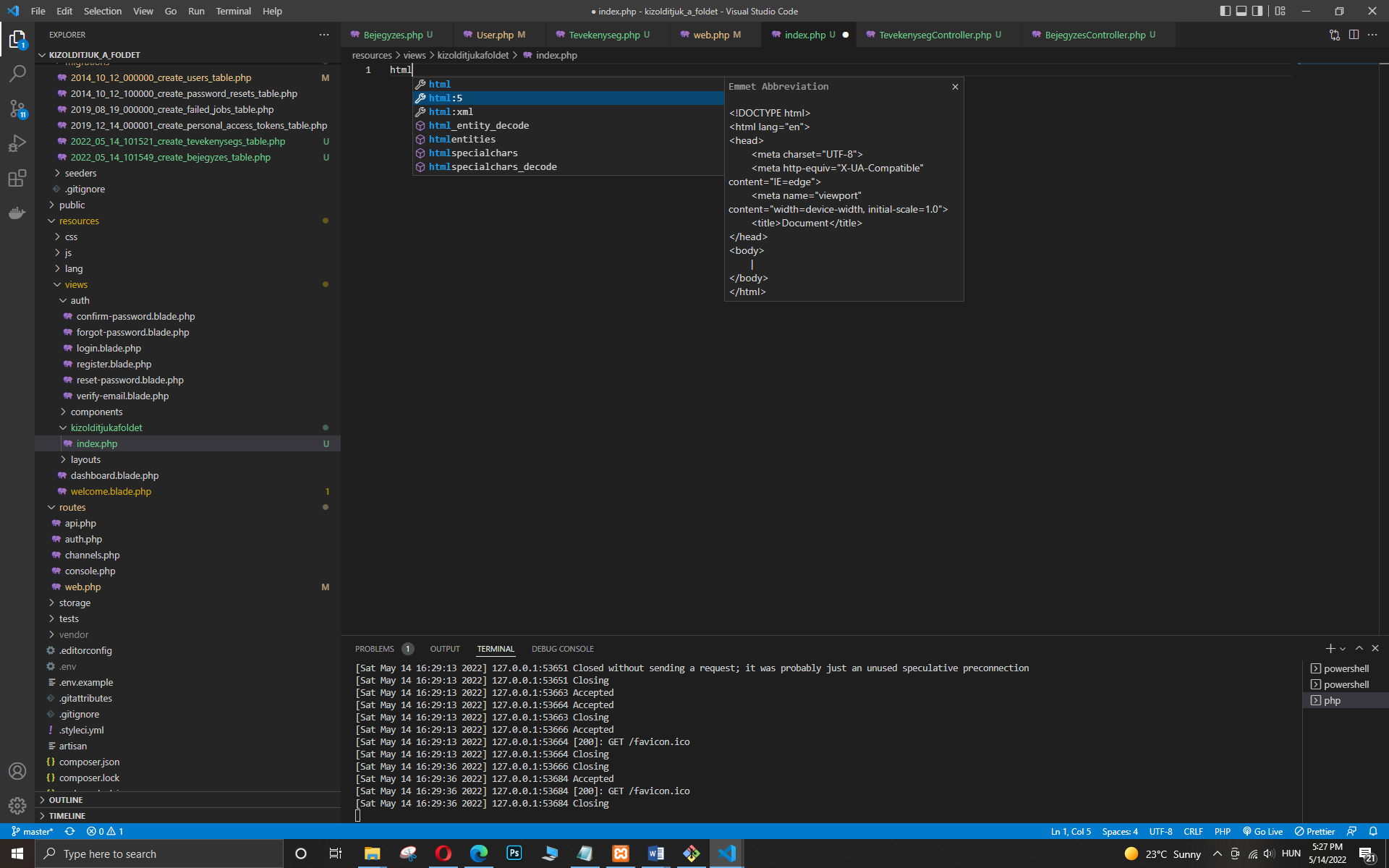
statisztika



## FRONTEND

Visual Studio Code 🡪 új html sablon:

az üres php fájlba beírjuk, hogy html, majd CTRL + SPACE 🡪 html5



### A TOKEN META TAG ELHELYEZÉSE A HTML FÁJL FEJLÉCÉBEN

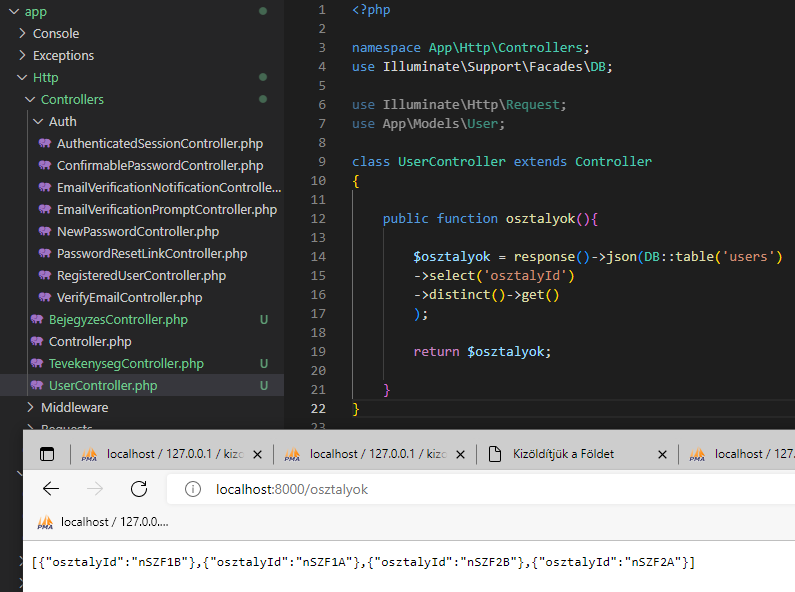
A <head></head közé> kell beilleszteni:

 <meta name="csrf-token" content=<?php $token = csrf\_token();

                                    echo $token; ?>>

Az osztályadatoknak is az adatbázisból kellene jönni, nem pedig tömbben megadni őket, ezért én az előbbi változathoz igazítom a megoldásom. A helyes megoldás valójában egy külön Osztály model létrehozását kívánná meg, amely össze van kapcsolva a User modellel. Most vegyük úgy, hogy az *adatbázisban szereplő userek osztályai* fogják alkotni az osztályválasztékot, amelyet beimportálunk az Osztályok legördülő listába. Ehhez össze kell gyűjtenem a létező osztályokat, mondjuk a User controlleren keresztül.

Route::get('/osztalyok', [UserController::class, 'osztalyok']);



### A diagram elkészítése

Az útvonal:

/\* Diagramhoz \*/

Route::get('/statisztika', [BejegyzesController::class, 'statisztika']);

A BejegyzésController függvény:

  //összpontszám lekérdezése osztályonként

    public function statisztika() {

        $osszPontszamOsztalyonkent = response()->json(DB::table('bejegyzes AS b') //query buildert indítok

        ->join('tevekenysegs AS t','b.tevekenysegId','t.id') //összekapcsolom a tevekenysegs táblával

        ->select('osztalyId', DB::raw('SUM(t.pontszam) as osszpontszam'))->groupBy(DB::raw("osztalyId")) //pontszám szummázása osztályonként

        ->get()

        );

        return $osszPontszamOsztalyonkent;

    }

JS:

Chart.js

class Chart{

constructor(){

google.charts.load('current', {packages: ['corechart', 'bar']});

google.charts.setOnLoadCallback(drawStacked);

let apiVegpontStatisztika = '/statisztika';

let adatTomb=[];

const token = $('meta[name="csrf-token"]').attr("content");

const myAjax = new MyAjax(token);

myAjax.getAjax(apiVegpontStatisztika,adatTomb,function(adatTomb){

    const dataArray=new Array();

    dataArray.push(['Osztály','Összpontszám',{ role: 'style' }]);// fejlécek

    adatTomb.forEach(element => {

        dataArray.push([element.osztalyId,Number(element.osszpontszam),'color: green']); //csak az értékeket pakolom át a kulcsok kihagyásával

    });

    drawStacked(dataArray);

});

function drawStacked(dataArray) {

      var chartData = google.visualization.arrayToDataTable(dataArray);

      var options = {

        title: 'Összpontszám osztályonként',

        chartArea: {width: '50%'},

        isStacked: true,

        hAxis: {

          title: 'Pontszám',

          minValue: 0,

        },

        vAxis: {

          title: 'Osztály'

        }

      };

      var chart = new google.visualization.BarChart(document.getElementById('chart\_div'));

      chart.draw(chartData, options);

    }

}

}

Main.js

//oldal betöltődésekor

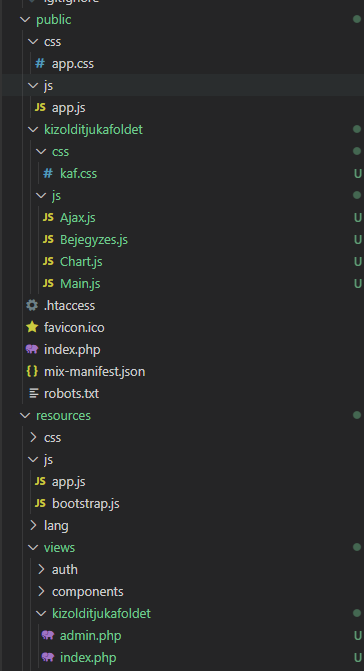
$(function(){

  new Urlapkezeles();

  new Chart();

});

##### Mappastruktúra



### index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Kizöldítjük a Földet</title>

    <!-- BOOTSTRAP - Latest compiled and minified CSS -->

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">

    <!-- BOOTSTRAP - Latest compiled JavaScript -->

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>

    <link rel="stylesheet" type="text/css" media="screen" href="\..\kizolditjukafoldet\css\kaf.css" />

    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>

    <script src='https://code.jquery.com/jquery-3.6.0.min.js'></script>

    <script src="kizolditjukafoldet\js\Ajax.js"></script>

    <script src="kizolditjukafoldet\js\Bejegyzes.js"></script>

    <script src="kizolditjukafoldet\js\Chart.js"></script>

    <script src="kizolditjukafoldet\js\Main.js"></script>

    <meta name="csrf-token" content=<?php $token = csrf\_token();

                                    echo $token; ?>>

</head>

<body>

<!-- nav -->

<nav class="navbar navbar-expand-sm navbar-light bg-success">

      <div class="container-fluid ">

        <a class="navbar-brand " href="#">Menü</a>

        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

          <span class="navbar-toggler-icon"></span>

        </button>

        <div class="collapse navbar-collapse " id="navbarSupportedContent">

          <ul class="navbar-nav ms-auto">

            <li class="nav-item">

              <a class="nav-link active" aria-current="page" href="#">Bejelentkezés</a>

            </li>

            <li class="nav-item">

              <a class="nav-link" href="#">Regisztráció</a>

            </li>

          </ul>

        </div>

      </div>

    </nav>

    <!-- header -->

    <header>

        <h1>Kizöldítjük a Földet</h1>

    </header>

    <article>

    <!-- form -->

        <form method="POST" enctype="multipart/form-data">

            <fieldset>

                <legend>Mit tettél ma a Földért?</legend>

                <div id="formfejlec">

                    <div>

                        <select name="osztalyok" id="osztalyok">

                            <option value="0">--Válassz osztályt--</option>

                        </select>

                    </div>

                    <div>

                        <select name="tevekenysegek" id="tevekenysegek">

                            <option value="0">--Válassz tevékenységet--</option>

                        </select>

                    </div>

                    <div>

                        <input type="button" id="kuld" name="kuld" value="Küld" >

                    </div>

                </div>

            </fieldset>

        </form>

        <!-- chart -->

        <div id="chart\_div"></div>

       <!-- kereses mezo -->

        <input class="form-control" id="kereses" type="text" placeholder="Keresés..">

        <!-- szures -->

        <div>

         <select name="szures-osztalyra" id="szures-osztalyra">

                            <option value="0">--Válassz osztályt--</option>

             </select>

         </div>

          <!-- tablazat -->

        <table class="table table-bordered table-striped table-success">

            <thead>

                <tr>

                 <th id='osztalyId'>Osztály</th>

                    <th id='tevekenyseg.nev'>Tevékenység</th>

                    <th id='tevekenyseg.pontszam' >Pont</th>

                    <th id='allapot' >Státusz</th>

                </tr>

            </thead>

            <tbody id="bejegyzes-lista">

            </tbody>

        </table>

    </article>

</body>

</html>

### css

@charset "UTF-8";

\* {

    box-sizing: border-box;

}

body{

    margin:auto;

    max-width: 900px;

}

article{

    margin-top: 30px;

}

fieldset,legend, nav{

    border: 1px solid lightgreen;

    border-radius: 4px;

}

legend{

    background-color: rgb(127, 255, 212,0.2);

    padding: 5px;

}

form, #kereses, nav, #chart\_div, #szures-osztalyra{

    margin-bottom: 30px;

}

#szures-osztalyra{

    width: 100%;

}

#formfejlec{

    display: grid;

    grid-template-columns: repeat(3,auto);

    align-content: space-around;

    justify-content: space-between;

    padding: 10px;

}

.pontinfo, .allapotinfo{

    display: none;

}

th:hover{

    cursor: pointer;

}

@media screen and (max-width: 768px) {

    #formfejlec{

        display:flex;

        flex-direction: column;

        gap: 5px;

}

#formfejlec select, #formfejlec input[type='button']{

    width: 100%;

}

.pontinfo, .allapotinfo{

    display: block;

}

th, .pont, td:last-child{

    display:none;

}

}

##### ajax.js

class MyAjax {

    constructor(token) {

        this.token = token;

    }

    /\*GET - ADATOK LEKÉRÉSE\*/

    //apiEndPoint: ahol megkapja a szervertől (ill. azon keresztül az adatbázistól) a (frissített ill. adott esetben szűrt) adatokat

    getAjax(apiEndPoint, array, myCallback) {

        array.splice(0, array.length); //tömb ürítése, hogy többszöri lefutáskor ne legyen hozzáfűzés

        $.ajax({

            url: apiEndPoint,

            type: "GET", //GET metódussal lekéri az adatokat az API végpontról, és egy result tömbbe teszi

            success: function (result) {

                result.forEach((element) => {

                    array.push(element); //a result összes elemét beletöltjük egy tömbbe

                });

                myCallback(array); //ha a tömb már teljesen feltöltődött, átadható paraméterként egy függvénynek

            },

        });

        console.log("GET OK");

    }

    /\*POST - új adat felvitele az AB-ba API végponton keresztül\*/

    postAjax(apiEndPoint, newData) {

        newData.\_token = this.token;

        $.ajax({

            headers: { "X-CSRF-TOKEN": this.token },

            url: apiEndPoint,

            type: "POST",

            data: newData,

            success: function (result) {

                console.log("POST success");

            },

        });

    }

    /\*DELETE - adott id-jú adat törlése az AB-ból API végponton keresztül\*/

    deleteAjax(apiEndPoint, id) {

        $.ajax({

            headers: { "X-CSRF-TOKEN": this.token },

            url: apiEndPoint + "/" + id,

            type: "DELETE",

            success: function (result) {

                console.log("DEL success");

            },

        });

    }

    /\*PUT - adott id-jú adat módosítása az AB-ban API végponton keresztül \*/

    putAjax(apiEndPoint, newData, id) {

        newData.\_token = this.token;

        $.ajax({

            headers: { "X-CSRF-TOKEN": this.token },

            url: apiEndPoint + "/" + id,

            type: "PUT",

            data: newData,

            success: function (result) {

                console.log("PUT success");

            }

        });

    }

}

### bejegyzes.js

//bejegyzés megjelenítése, adatok a megfelelő selectorokba történő betöltése

class BejegyzesElem {

    constructor(adat) {

        this.adat = adat; //adat: egy bejegyzésrekord

        this.kontener = $('#bejegyzes-lista');

        const allapot = adat.allapot===0 ? 'jóváhagyásra vár' : 'elfogadva';

        const elem =

            "<tr class='bejegyzes'> <td><div class='osztalyinfo'>" +  adat.osztalyId +

                "</div><div class='pontinfo'> Pontérték: " + adat.tevekenyseg.pontszam +

                "</div></td><td><div class='tevekenyseginfo'>" +

                adat.tevekenyseg.nev +"</div> <div class='allapotinfo'>" + allapot +  "</div></td> <td class='pont'>" +

                      adat.tevekenyseg.pontszam +

                      "</td><td class='allapot'> " +  allapot +  "</td></tr>";

        this.kontener.append(elem);

        this.utolsoElem = this.kontener.children(".bejegyzes:last-child");

    }

}

// a bejegyzéslista felépítése tömbadatok alapján

class BejegyzesLista {

    constructor() {}

    bejegyzesListFelepit(tomb) {

        tomb.forEach((elem) => {

            new BejegyzesElem(elem);

        });

    }

}

### Main.js

//oldal betöltődésekor

$(function(){

  new Urlapkezeles();

  new Chart();

});

class Urlapkezeles{

 constructor(){

    const osztalyokApiVegpont='/osztalyok';

    const tevekenysegekApiVegpont ='/tevekenyseg/osszes';

    const bejegyzesekApiVegpont = '/bejegyzes/osszes';

    const ujBejegyzesApiVegpont = "/bejegyzes/beszur";

    const osztalyokTomb = [];

    const tevekenysegekTomb = [];

    const bejegyzesekTomb = [];

    const token = $('meta[name="csrf-token"]').attr("content");

    const myAjax = new MyAjax(token);

    const bejegyzesLista = new BejegyzesLista();

   // this.bejegyzesKontener = $('#bejegyzes-lista'); //nem működött, ezért inkább közvetlenül a selectorra hivatkoztam

   // this.tevekenysegekDropdown = $('#tevekenysegek'); //nem működött

   //  this.osztalyokDropdown = $("#osztalyok"); //nem működött

     myAjax.getAjax(osztalyokApiVegpont,osztalyokTomb,this.osztalyAdatokImport); //osztályadatok betöltése a form dropdown menübe

     myAjax.getAjax(tevekenysegekApiVegpont,tevekenysegekTomb,this.tevekenysegAdatokImport); //tevékenységadatok betöltése a dropdown menübe

     myAjax.getAjax(bejegyzesekApiVegpont,bejegyzesekTomb,bejegyzesLista.bejegyzesListFelepit); //tevékenységadatok betöltése a dropdown menübe

     myAjax.getAjax(osztalyokApiVegpont,osztalyokTomb,this.osztalyAdatokSzureshez); //osztályadatok betöltése a szűrésre szolgáló dropdown menübe

    this.kivalasztBeszurandoAdat()

   //beszúrandó új rekord adatainak elküldése a szerveroldal felé

    $(window).on("ujAdatBeszur", (event) => {

        let ujAdat = event.detail;

        myAjax.postAjax(ujBejegyzesApiVegpont, ujAdat);

        //megjelenítés

        $('#bejegyzes-lista').empty();

        myAjax.getAjax(bejegyzesekApiVegpont,bejegyzesekTomb,bejegyzesLista.bejegyzesListFelepit);

        location.reload(); //újratöltöm az oldalt, hogy a grafikonadatok is frissüljenek

    });

     this.kereses();

     this.getRendezesiAttributum();

     rendez();

     $("#szures-osztalyra").on("change",()=> {

     let aktOsztalyId = $("#szures-osztalyra").val();

     if(!(aktOsztalyId==='0')){

        let apiVegpontSzures = 'bejegyzesek/'+aktOsztalyId;

        $('#bejegyzes-lista').empty();

        myAjax.getAjax(apiVegpontSzures,bejegyzesekTomb,bejegyzesLista.bejegyzesListFelepit);

     }else {

        $('#bejegyzes-lista').empty();

        myAjax.getAjax(bejegyzesekApiVegpont,bejegyzesekTomb,bejegyzesLista.bejegyzesListFelepit);

     }

 });

     function rendez(){

     let asc=true;

     $(window).on("rendezesiAttributum", (event) => {

       let attributum = event.detail;

        myAjax.getAjax(bejegyzesekApiVegpont,bejegyzesekTomb, function(){ //friss adatlista lekérése

            //console.log(bejegyzesekTomb)

            $('#bejegyzes-lista').empty();

             //komplex kulcs esetén

            if(attributum==='tevekenyseg.pontszam' || attributum==='tevekenyseg.nev'){

                asc ? sortByKeyAscComplex(bejegyzesekTomb,attributum) : sortByKeyDescComplex(bejegyzesekTomb,attributum);

            }else{

                //szimpla kulcs esetén

                asc ? sortByKeyAsc(bejegyzesekTomb,attributum) : sortByKeyDesc(bejegyzesekTomb,attributum);

            }

             bejegyzesLista.bejegyzesListFelepit(bejegyzesekTomb);

           })

           asc=!asc;

     });

    }

     //csökkenő sorrendbe történő rendezés - szimpla kulcsra, tehát pl. elem.osztalyId / elem[osztalyId], amelyből a kulcs osztalyId

     function sortByKeyDesc(array, key) {

        return array.sort(function (a, b) {

            return ((a[key]> b[key]) ? -1 : ((a[key] < b[key]) ? 1 : 0));

        });

    }

    //növekvő sorrendbe történő rendezés - szimpla kulcsra, tehát pl. elem.osztalyId / elem[osztalyId], amelyből a kulcs osztalyId

    function sortByKeyAsc(array, key) {

        return array.sort(function (a, b) {

            return ((a[key]< b[key]) ? -1 : ((a[key] > b[key]) ? 1 : 0));

        });

    }

      //csökkenő sorrendbe történő rendezés - komplex kulcsra, tehát pl. elem.tevekenyseg.pontszam / elem[tevekenyseg][pontszam], amelyből a kulcs tevekenyseg.pontszam

      function sortByKeyDescComplex(array, complexKey) {

        let keyOuter = complexKey.split(".")[0];

        let keyInner = complexKey.split(".")[1];

        return array.sort(function (a, b) {

            return ((a[keyOuter][keyInner]> b[keyOuter][keyInner]) ? -1 : ((a[keyOuter][keyInner] < b[keyOuter][keyInner]) ? 1 : 0));

        });

    }

    //növekvő sorrendbe történő rendezés - komplex kulcsra, tehát pl. elem.tevekenyseg.pontszam / elem[tevekenyseg][pontszam], amelyből a kulcs tevekenyseg.pontszam

    function sortByKeyAscComplex(array, complexKey) {

        let keyOuter = complexKey.split(".")[0];

        let keyInner = complexKey.split(".")[1];

        return array.sort(function (a, b) {

            return ((a[keyOuter][keyInner]< b[keyOuter][keyInner]) ? -1 : ((a[keyOuter][keyInner] > b[keyOuter][keyInner]) ? 1 : 0));

        });

    }

 }

 osztalyAdatokImport(tomb) {

    tomb.forEach((element, index) => {

            let option = new Option(element.osztalyId, element.osztalyId); //new Option("option text", "value");

            $("#osztalyok").append(option);

    });//hozzáadunk egy újabb option tag-et

    tomb.splice(0,tomb.length);

}

osztalyAdatokSzureshez(tomb) {

    tomb.forEach((element, index) => {

            let option = new Option(element.osztalyId, element.osztalyId); //new Option("option text", "value");

            $("#szures-osztalyra").append(option);

    });//hozzáadunk egy újabb option tag-et

    tomb.splice(0,tomb.length);

}

tevekenysegAdatokImport(tomb) {

    tomb.forEach((element, index) => {

            let option = new Option(element.nev, element.id); //new Option("option text", "value");

            $("#tevekenysegek").append(option);

    });//hozzáadunk egy újabb option tag-et

    tomb.splice(0,tomb.length);

}

kereses(){

$("#kereses").on("keyup", function() {

    let value = $(this).val().toLowerCase();

    $("table tr").filter(function() {

      $(this).toggle($(this).text().toLowerCase().indexOf(value) > -1)

    });

  });

}

getRendezesiAttributum(){

    $('table').on('click','th',(event)=>{  //amelyik th-ra rákattintottam, annak az id-ját kérem le

        this.esemenyInfoTovábbit('rendezesiAttributum', $(event.target).attr('id')); //a th-kban elrejtettem a tömb kulcsait; ezt kérjük le, és adjuk tovább

    });

}

esemenyInfoTovábbit(esemenyKulcs,adat) {

    let event = new CustomEvent(esemenyKulcs, {

        detail: adat, //ezzel adatokat tudok átadni

    });

    window.dispatchEvent(event);

}

//a beszúrandó adatok kiválasztása a legördülő listákból

kivalasztBeszurandoAdat(){

    $("#kuld").on("click", () => {

        if(!($("#osztalyok").val()==='0' || $("#tevekenysegek").val()==='0')) {

        const beszurandoAdat = {

            //eltárolom a legördülő listák aktuálisan kijelölt értékeit

            osztalyId: $("#osztalyok").val(),

            tevekenysegId: $("#tevekenysegek").val(),

        };

        this.esemenyInfoTovábbit('ujAdatBeszur',beszurandoAdat);

    }else{

        alert('Mindkét legördülő listából válassz értéket!');

    }

    });

}

}

### Chart.js

class Chart{

constructor(){

google.charts.load('current', {packages: ['corechart', 'bar']});

google.charts.setOnLoadCallback(drawStacked);

let apiVegpontStatisztika = '/statisztika';

let adatTomb=[];

const token = $('meta[name="csrf-token"]').attr("content");

const myAjax = new MyAjax(token);

myAjax.getAjax(apiVegpontStatisztika,adatTomb,function(adatTomb){

    const dataArray=new Array();

    dataArray.push(['Osztály','Összpontszám',{ role: 'style' }]);// fejlécek

    adatTomb.forEach(element => {

        dataArray.push([element.osztalyId,Number(element.osszpontszam),'color: green']); //csak az értékeket pakolom át a kulcsok kihagyásával

    });

    drawStacked(dataArray);

});

function drawStacked(dataArray) {

      var chartData = google.visualization.arrayToDataTable(dataArray);

      var options = {

        title: 'Összpontszám osztályonként',

        chartArea: {width: '50%'},

        isStacked: true,

        hAxis: {

          title: 'Pontszám',

          minValue: 0,

        },

        vAxis: {

          title: 'Osztály'

        }

      };

      var chart = new google.visualization.BarChart(document.getElementById('chart\_div'));

      chart.draw(chartData, options);

    }

}

}