EÖTVÖS LORÁND TUDOMÁNYEGYETEM
INFORMATIKAI KAR
TÉRKÉPTUDOMÁNYI ÉS GEOINFORMATIKAI TANSZÉK

Nyílt forráskódú térképes adatbáziskezelő és -megjelenítő a weben

DIPLOMAMUNKA

TÉRKÉPÉSZ MESTERSZAK

Készítette:

Varga Ferenc

térképész szakos hallgató

Témavezető:

Elek István

habilitált docens

ELTE Térképtudományi és Geoinformatikai Tanszék

Budapest, 2018

Tartalom

Ábrák, Forráskódok jegyzéke

1. Bevezetés

Diplomamunkám egy, az interneten is elérhető, térképi adatbáziskezelő, és -megjelenítő alkalmazás megalkotásáról szól. Az adatbázis digitális állományokra fekteti a hangsúlyt. Az egyes térképek - jelen esetben földi referenciával rendelkező raszteres vagy vektoros entitások – megjelenítése. Az adatbázis- és a kezelő is teljesen szabadon elérhető keretrendszerekkel, programnyelvekkel és függvénykönyvtárakkal történt.

Folyamatosan fejlődő világunkban egyre több ember számára elérhető a gyors és megbízható internetkapcsolat, a kisméretű desktop (asztali) alkalmazások egyre inkább háttérbe szorulnak az internet és a mobil platformok rovására. Ugyanígy sok egymástól független egyén számára az elérést a webes „alkalmazás” teszi lényegesen egyszerűbbé, valamint így biztosítható az is, hogy az adatok mindig frissek, naprakészek legyenek.

Nem utolsó sorban a téma választásakor saját szakmai fejlődésem előtérbe helyezése, valamint programozási ismereteim bővítése is nagy szerepet játszott.

A dolgozat értelmezéséhez ajánlott a minimális angol nyelvtudás, és valamely programozási nyelvben való jártasság (de egyik sem feltétel).

2. Előzetes ismeretek

Az alkalmazás *Visual Studio 2017 Community Edition program 15.3*-as verziójában íródott (*Windows 10*-ben). Ebben felhasznált programnyelvek a *C# 7.1*, és a különböző webes nyelvek (*HTML, JavaScript, CSS*) együttese. A projekt típusa *ASP.NET Core 2.0 Web Applikáció*, alkalmazáson belül tárolt identitással. A kiinduló adatbázis *pgAdmin* nevű szoftver 4.0-s változatában készült. Az adatbázis *PostgreSQL 9.6*-os alapokon nyugszik, *PostGIS 2.0*-s bővítménnyel kiegészítve. Kliensoldali *JavaScript* függvénykönyvtárként felhasználtam az *OpenLayers 4.3.3*-as, illetve a *Twitter Bootstrap 3.3.7*-es és a *jQuery 2.2.0*-s kiadásait. Az alkalmazás – egyelőre – nincs *host*-olva.

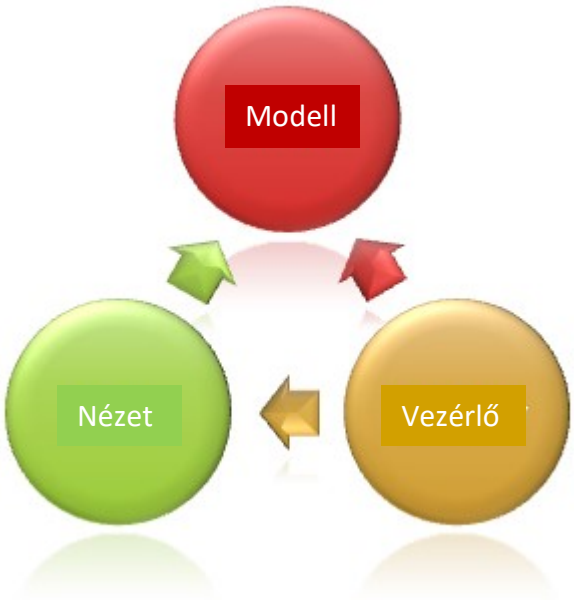
2.1. ASP.NET Core

A Microsoft saját, webre szánt programozási keretrendszereit összefoglalóan *ASP.NET*-nek nevezik. A cég az utóbbi időben egyre nagyobb hangsúlyt fektet a szabadon elérhető megoldásokra. Ennek ékes bizonyítéka a 2016-ban debütáló, a *.NET Framework*-ot felváltó, *.NET Core* (Mag), melynek második változata 2017-től érhető el. Az *ASP.NET Core* egy, a *Microsoft* és a közösség által fejlesztett, több platformon elérhető, nagy teljesítményű, nyílt forráskódú keretrendszer. Segítségével modern felhő- és internetalapú alkalmazásokat lehet készíteni *Windows, Linux* vagy *MacOS* operációs rendszereken. Az *ASP.NET Core* az *ASP.NET* [*Framework*] újratervezett és továbbfejlesztett változata. Az architektúra változtatása egy könnyebb és modulárisabb keretrendszert eredményezett. Előnyei még többek között az *UI* (*User Interface* – Felhasználói felület) és az *API* (*Application Programming Interface* - alkalmazásprogramozási felület) egységesítése, beépített függőségi befecskendezés (*dependency injection*) használata, modern kliens-oldali keretrendszerek alapértelmezetten való integrálása (pl.: *Bower, Bootstrap, AngularJS* stb.), szerver- és kliensoldali *validálás* (érvényesítés) támogatása. (Roth – Anderson – Luttin, 2017)

2.2. MVC

Az *MVC* mozaikszó az angol *Model* (modell), *View* (nézet) és *Controller* (vezérlő) szavak kezdőbetűiből tevődik össze. Az *MVC* minta a „*Separation of Concerns*” (Szerepkörök Szétválasztása) elvből alakult ki, miszerint a különböző típusú funkciókat ellátó feladatokat lehetőség szerint egymástól külön kell kezelni, így – betartva a „*Don’t Repeat Yourself!*” („Ne ismételd magad!”) elvet, ezzel fejlesztésre fordított időt és pénzt

spórolva. Kezdjük a végén: a ***Vezérlő***, a program magja, a szerveroldali feladatokat látja el, és „mondja meg”, hogy mely nézetet mutatjuk a felhasználónak, illetve kérdezi le az adatokat a modellből. A ***Nézetet*** látja a felhasználó, ebben alkotjuk meg a kinézetet, definiáljuk a megjelenő adatokat stb. A Nézet a kliensoldali funkciókért felelős; gyakorlatilag egy *HTML* file – de több annál. A ***Modell*** tartalmazza az összes adatot, amit deklarálunk – ez a háttéradatbázis absztrakciója forráskód szinten. [1. ábra] (Smith, 2016)



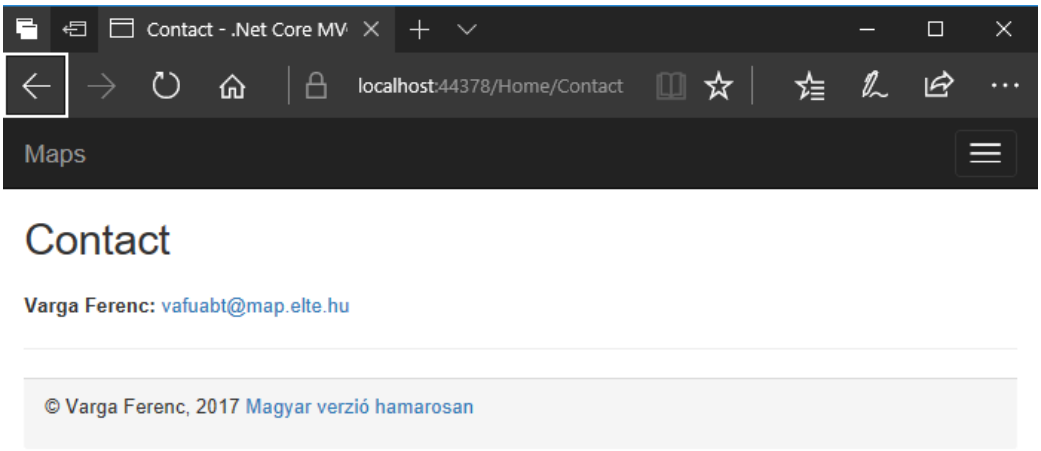
1. ábra: Az MVC minta. A nyilak azt mutatják, hogy melyik elem melyik másik elemre hivatkozik. Forrás: Overview of ASP.NET Core MVC

Létrehozhatunk úgynevezett *ViewModel*-t (nézetmodell), ami szűkítheti vagy bővítheti a modellben definiált változókat. Ez nem közvetlenül a háttéradatbázissal áll kapcsolatban, és tipikusan – nevéből eredően is – egy adott nézetben lévő információk megjelenítéséhez, vagy azok bekéréséhez szükségeltetik, hogy gyorsabban végezzünk el az adatok mozgatását a kliens és a szerver között, vagy még inkább felhasználó-barát nézetet tárjunk a felhasználó elé. Az *ASP.NET Core MVC*-ben a Nézetet a *Razor View Engine* (Razor Nézet Motor) rendereli. Ebben a szokásos *HTML* szabványban ismert funkciók, nyelvek, könyvtárak mellé beépülő C# nyelvet is használhatunk (*Tag*- és *HTML-helper*-ekben), lehetővé téve a modell használatát és az adatok manipulálását, a kliensoldali *JavaScript* mellett, mellyel így dinamikusan változtatható oldalakat hozhatunk létre. [1. forráskód] Az adatbázisok közvetlen kezeléséért az *EntityFrameworkCore* felel, ami a *PostgreSQL* adatbázisrendszerrel az *Npgsql* csomagon keresztül kommunikál. [2.]

```
@{
ViewData["Title"] = "Contact";
}
<h2>@ViewData["Title"]</h2>
<h3>@ViewData["Message"]</h3>

<address>
<strong>Varga Ferenc:</strong>
<a href="mailto:vafuabt@map.elte.hu">vafuabt@map.elte.hu</a>
<br />
</address>
```

1. forráskód: Egy egyszerű nézet (Contacts.cshtml). A „@” jel után a beépülő C# kóddal

A 2. ábrán láthatjuk a böngészőben renderelt weboldalt.



2. ábra: A böngészőben megjelenő nézet (Contacts.cshtml)

2.3. PostgreSQL, PostGIS, PgAdmin

A *PostgreSQL* egy nyílt-forráskódú, objektum-relációs adatbázis rendszer, mely minden ismertebb operációs rendszeren fut (*Linux, UNIX* alapú rendszerek, *Windows*) (The PostgreSQL Global Development Group, 2017). Ennek kiegészítője a *PostGIS* nevű térbeli-adatbázis bővítmény, aminek segítségével földrajzi objektumokra vonatkozó SQL lekérdezéseket lehet futtatni (PostGIS Project Steering Committee, 2017). A *pgAdmin* egy szabadon elérhető szoftver *PostgreSQL*-hez (The pgAdmin Development Team, 2017).

2.4. OpenLayers

Az *OpenLayers* 4.3.3 (valójában a 3-as verzió) egy szabad felhasználású *JavaScript* függvénykönyvtár, mely a térképes megjelenítést a webböngészőben képek segítségével

teszi lehetővé. Képes térkép-csempék, vektoros adatok és marker-ek (pontjelek, jelzők) megjelenítésére. (OpenLayers, 2017)

3. A kezdő adatbázis kialakítása

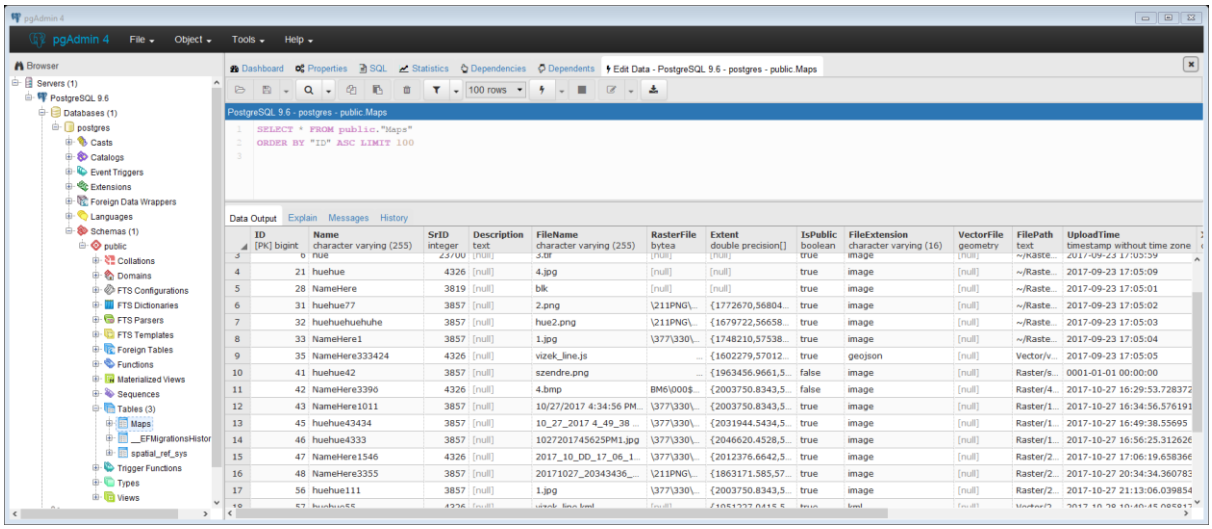
A háttéradatbázis koncepciója az, hogy az alkalmazáson belül lévő filerendszerben tároljuk az állományokat, és ezen file-ok metaadatait az adatbázisban. A filerendszer az alkalmazás gyökér könyvtárában (*wwwroot*) található az *Uploads* (feltöltések) mappában. A képek/raszterek között egyelőre nem teszünk különbséget [a Befejezés című fejezetben kitérek erre bővebben]. Vektoros állományként *GeoJSON* és *kml* file tölthető fel [*ESRI Shapefile* támogatás később]. A kialakított relációs adatbázis *postgres* névre hallgat, melynek *public* nevű sémájában 2 tábla található. A *spatial_ref_sys* tartalmazza az összes *EPSG* által elismert vetületet, *EPSG* kóddal, és *proj4 text*-tel. Elsődleges kulcsa az *srid* mező. A *PostGIS* bővítmény telepítésével került az adatbázisba.

```
-- Table: public.spatial_ref_sys

-- DROP TABLE public.spatial_ref_sys;

CREATE TABLE public.spatial_ref_sys
(
    srid integer NOT NULL,
    auth_name character varying(256) COLLATE pg_catalog."default",
    auth_srid integer,
    srtext character varying(2048) COLLATE pg_catalog."default",
    proj4text character varying(2048) COLLATE pg_catalog."default",
    CONSTRAINT spatial_ref_sys_pkey PRIMARY KEY (srid),
    CONSTRAINT spatial_ref_sys_srid_check CHECK (srid > 0 AND srid <= 998999)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.spatial_ref_sys
    OWNER to postgres;

GRANT ALL ON TABLE public.spatial_ref_sys TO postgres;

GRANT SELECT ON TABLE public.spatial_ref_sys TO PUBLIC;
```

3. ábra: A spatial_ref_sys SQL forráskódja

A *Maps* tábla a vetületi információ (*SrID*), mint a *spatial_ref_sys* tábla idegen kulcsa mellett rendelkezik egy ID nevű elsődleges kulccsal is. A *Title* (cím) és *Description* (leírás) mezők tetszőleges karakterekből, előbbi legfeljebb 255 karakterből álló egyedi azonosító, utóbbi hosszban – elvileg – nem limitált. Az *Uploader* (feltöltő) a feltöltő becenevét (*nickname*), míg az *UploadTime* (feltöltési idő) egy időbélyeget jelent a feltöltés (vagy a

módosítás) időpontjáról. A *Maps* táblába kerülnek a feltöltött képek és vektoros állományok metaadatai is. A file-rendszerbe kerülő állományok közvetlen adataira vonatkoznak a következő oszlopok attribútumai: *FileName* (file-név), *FileExtension* (file-kiterjesztés), *FilePath* (elérési út). A *FileName* változtatás nélkül, a *FileExtension* – egyelőre – csak a kiterjesztés szerint változik a vektoros file-oknál (*geojson, kml*). A *FilePath*-ban az elérési utak következő sémába illeszkednek: az elején *Vector/* vagy *Raster/*, aszerint, hogy milyen típusú állományról beszélünk, majd a feltöltési idő (*éééé*HHnn_óóppmmtt formátumban) egy alátörés jellel folytatva, és végül a feltöltött file nevével. Mindez az *Uploads* mappában lévő mappaszerkezetre utal a file-rendszerben [4. ábra].

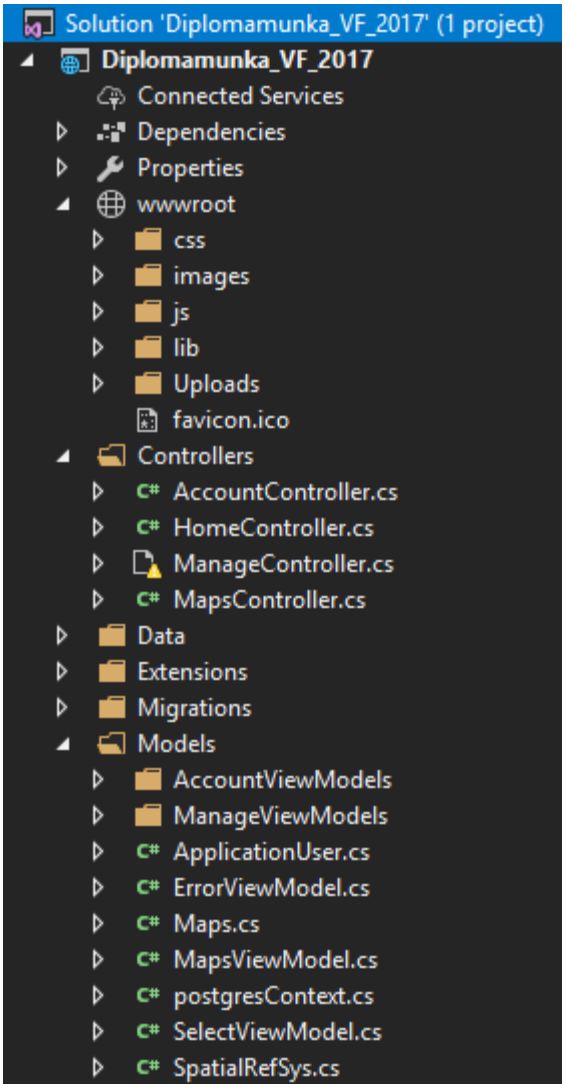


4. ábra: A postgres adatbázis Maps nevű táblája tesztadatokkal pgAdmin 4-ben

Az alkalmazáson belüli, identitásokat kezelő adatbázisról a következő fejezetben ejtek szót.

4. A kezelő megalkotása

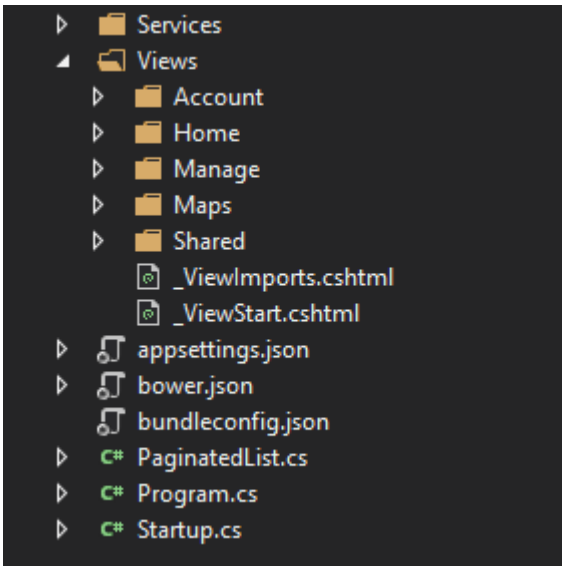
4.1. Egy ASP.NET Core Web Applikáció felépítése MVC mintára, alkalmazáson belül tárolt identitással



5. ábra: Az alkalmazás belső felépítése a Visual Studio Solution Explorer-ében I. (a kép folytatódik)

Az 5. és 6. ábrákon láthatjuk a projekt felépítését az előzőekben már említett *MVC* szerkezettel, és az *Uploads* mappával. A *Dependencies*-ben találjuk az összes referenciát (függőségeket), amire projektünkben hivatkozunk. Ilyen például a *jQuery* validálás vagy a *PostgreSQL* használatát lehetővé tevő *Npgsql NuGet* csomag. A *wwwroot* alapértelmezetten az alkalmazás azon részét jelöli, amelyet a kliens oldalról is közvetlenül el tudunk érni (a Nézetben), ide kerülnek a weblap statikus képei, és ez alatt találjuk a filerendszerünket is. A *Controllers*-ben a vezérlőosztályok találhatóak. A *Data* tartalmazza az identitás kezelésére

szánt adatbázist kezelő osztályt. A bővítmények, kódkiegészítések kerülnek az *Extensions*, „pillanatképek” (először adatbázis módszer esetén) az adatbázis migrálása előttről a *Migrations*, különböző szolgáltatások (pl. Email-küldés) osztályai pedig a *Services* könyvtár alá. A *Models* a modellek (adatbázisok) konténere a programon belül.



6. ábra: Az alkalmazás belső felépítése a Visual Studio Solution Explorer-ében II. (folytatás)

A *Views* a fentebb már említett, *cshtml* kiterjesztésű *Razer Nézeteket* foglalja magában. Végül néhány fontos állomány: az *appsettings.json*-ben vannak az alkalmazás alapdefiníciói, adatbáziskapcsolatokat lehetővé tevő „kapcsolódási szövegek” (*connection string*), és az egyéb lényeges kezdeti paraméterek, amiket nem szeretnénk, hogy a végfelhasználó elérhessen, viszont az applikáció helyes működéséhez elengedhetetlenek. A *Program.cs* az építésért (*build*) felel a *Startup.cs*, mint konfigurációs alosztály, segítségével. Utóbbiban implementáljuk például a szolgáltatásokat, és a *HTTP* kérések elirányítását a programon belül (*routing*).

4.2. A cél megvalósítása I., a Modellek és Nézetmodellek

Ebben a fejezetben az előbbiekben már elkészült *postgres* adatbázist építjük be a programba. Ezt a *Scaffold-DbContext* („adatbázis állványozás”) paranccsal tehetjük meg a *Package Manager Console*-ban (Csomagkezelő konzol) a *Visual Studio*-ban. Paraméterként a *postgres* adatbázishoz való kapcsolódási szöveget, a szolgáltatásellátó szervezetet, esetleg a kimeneti mappát adhatjuk meg. [2. forráskód]

```
Scaffold-DbContext "Host=localhost;Database=postgres;
      Username=postgres;Password=<elrejtve>"
      Npgsql.EntityFrameworkCore.PostgreSQL -OutputDir Models
```

2. forráskód: Adatbázis állványozás

A fenti paranccsal létrejönnek a *postgresContext.cs*, a *Maps.cs* és a *SpatialRefSys.cs* osztályok. Az elsőben az *EntityFrameworkCore* legyártja az adatbázist programkód szintre, minden funkciójával és tulajdonságával. A *Maps.cs* és *SpatialRefSys.cs* osztályok valójában a modellt jelentik esetünkben. Az itt deklarált változók segítségével tudunk adatot kinyerni, rekordot frissíteni, vagy éppen új rekordot létrehozni az adatbázisban.

```
using System;
using System.Collections.Generic;
using NpgsqlTypes;
using System.ComponentModel;

namespace Diplomamunka_VF_2017.Models
{
    public partial class Maps
    {
        public long Id { get; set; }
        public string Title { get; set; }
        [DisplayName("EPSG code")]
        public int SrId { get; set; }
        public string Description { get; set; }
        [DisplayName("Filename")]
        public string FileName { get; set; }
        public byte[] RasterFile { get; set; }
        public double[] Extent { get; set; }
        [DisplayName("Is Public?")]
        public bool IsPublic { get; set; }
        public string FileExtension { get; set; }
        public PostgisGeometry VectorFile { get; set; }
        public string FilePath { get; set; }
        [DisplayName("Last Updated")]
        public DateTime UploadTime { get; set; }
        public string Uploader { get; set; }
        [DisplayName("EPSG code")]
        public SpatialRefSys Sr { get; set; }
    }
}
```

3. forráskód: A Maps.cs osztály

A 3. forráskód a *Maps.cs* osztályt mutatja, melyben felismerhetők az adatbázisban előre megadott mezőnevek. A relációs adatbázis mivolta az utolsó sorban fedezhető fel,

miszerint a *SpatialRefSys.cs* osztályra hivatkozva éri el a *spatial_ref_sys* táblát az adatbázisban az osztály.

Az identitást tároló, *Microsoft SQL Server*-re épülő relációs adatbázist a program a kialakításakor legyártja, melyet később bővíthetünk, kiegészíthetünk az *ApplicationUser.cs* osztály segítségével. Főbb táblái a felhasználókat (*Users*), a szerepköröket (*Roles*) és a felhasználók „szerepkör-követeléseit" (*UserRoles*) tartalmazók. A felhasználókat tartalmazó táblában tároljuk a felhasználóneveket, az e-mail címeket, a jelszavakat (titkosítva), és a telefonszámokat [nincs aktiválva a kétfaktoros azonosítás egyelőre], általam kiegészítve a vezetéknevekkel, a keresztnevekkel és a becenevekkel. A szerepkörök jelenleg az *Admin* (Adminisztrátor), a *Manager* (Kezelő) és a *User* (Felhasználó). A *UserRoles* táblában az egyes felhasználókhoz tartozó szerepköröket tároljuk. Az identitások manipulálása tömegesen kezelve (például regisztrációkor) az *AccountController* vezérlő, az *AccountViewModels* nézetmodellek és az *Account* könyvtár alatt lévő nézetek, egyénileg (például keresztnév megváltoztatásakor) pedig a *ManageController* vezérlő, a *ManageViewModels* nézetmodellek és a *Manage* könyvtár alatt lévő nézetek segítségével történik. Magyarul az *Account* a be- és kijelentkezést, valamint a regisztrációt, a *Manage* a fiókbeállításokat kezeli.

A 4. forráskód a *MapsViewModel.cs* „nézetmodell" osztályt írja le. Az itt deklarált változók nagyban hasonlítanak a *Maps.cs*-ben látottakhoz, ugyanakkor például az *Extent* (Kiterjedés) helyett már az egyes koordinátákhoz tartozó változók (legkisebb és legnagyobb x, illetve y) vannak definiálva. Érdemes még megfigyelni az adatannotációkat (*DataAnnotations* névtér). Ezek segítségével végezhetünk például „valós idejű szerveroldali-validálást" *(Required*: kötelező mező, *Range*: intervallum) vagy határozhatunk meg kiírandó neveket a nézetben (*DisplayName*). Ezt a „nézetmodellt" a Nézetekben használjuk adat-bekérésre, vagyis a felhasználó ezeket a változókat adja meg. Az adatok megadása után a vezérlőben pedig elvégezzük az adatbázisba való leképzést.

```
// modul hivatkozások elrejtve
// névtér, osztálydeklaráció elrejtve
        public long Id { get; set; }
        [Required]
        [DisplayName("Title")]
        public string Title { get; set; }
        [Range(2000, 32766)]
        [Column(name: "SpatialRefSys.Srid")]
        [ForeignKey(name: "SrId")]
        [DisplayName("EPSG code")]
        public int SrId { get; set; }
        [DisplayName("Description")]
        public string Description { get; set; }
        [DisplayName("Filename")]
        public string FileName { get; set; }
        [Required]
        [DataType(DataType.Upload)]
        [DisplayName("File")]
        public ICollection<IFormFile> File { get; set; }
        [Required]
        [DisplayName("Xmin")]
        public double Xmin { get; set; }
        [Required]
        [DisplayName("Ymin")]
        public double Ymin { get; set; }
        [Required]
        [DisplayName("Xmax")]
        public double Xmax { get; set; }
        [Required]
        [DisplayName("Ymax")]
        public double Ymax { get; set; }
        [Required]
        [DisplayName("Is Public?")]
        public bool IsPublic { get; set; }
        [DisplayName("File Extension")]
        public string FileExtension { get; set; }
        public PostgisGeometry VectorFile { get; set; }
        [DisplayName("Filepath")]
        public string FilePath { get; set; }
        [DisplayName("Uploader")]
        public string Uploader { get; set; } //…
```

4. forráskód: A MapsViewModel.cs nézetmodell

4.3. A cél megvalósítása II., a Vezérlők

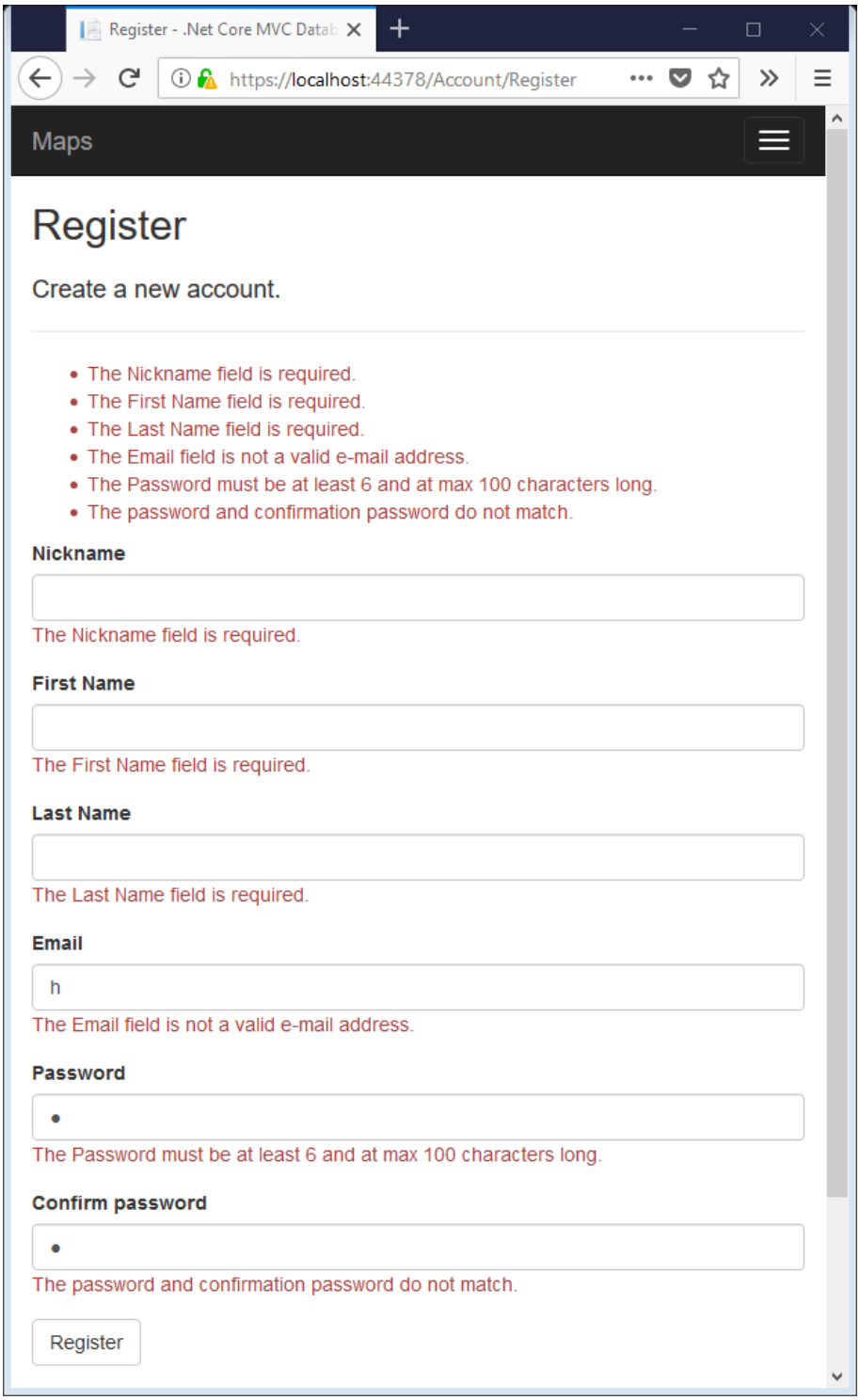
Miután elkészült az adatbázis „lenyomata” a programban, itt az ideje a Vezérlők, és ezeken belül az úgynevezett „Akciók” (*Actions*, Műveletek) kialakításának. Az Akciók

gyakorlatilag egy-egy *HTTP* kérés (pl.: *GET* vagy *POST* – kérés vagy küldés metódusok) bekövetkezésekor meghívott függvényt jelentenek. Az *Account(Controller)* vezérlő a regisztrációért, a be- és kijelentkezésért, a külső szolgáltatóval történő bejelentkezésért, az e-mail konfirmációért, az elfeledett jelszavakért és az ezekhez kapcsolódó ügyekért felelős a szerveren.

```
namespace Diplomamunka_VF_2017.Controllers {
    [Authorize]
    [Route("[controller]/[action]")]
    public class AccountController : Controller {
    //…
        [HttpGet]
        [AllowAnonymous]
        public IActionResult Register(string returnUrl = null) {
            ViewData["ReturnUrl"] = returnUrl;
            return View();
        }
    //…
    }
}
```

5. forráskód: Az Account Vezérlő Register Akciójának HTTP Get kérésekor meghívódó függvénye

Az 5. forráskód a vezérlő *Register* (Regisztrál) műveletét mutatja. Figyeljük meg a függvénydeklaráció előtt lévő *HttpGet* és *AllowAnonymous* attribútumokat. Az első specifikálja, hogy az akció csak *HTTP GET* kérés esetén legyen meghívva, utóbbi az Identitás „kikerülésére” hivatott, magyarul engedélyezi minden (nem regisztrált) felhasználónak a hozzáférést az eljárás invokálásához, anélkül, hogy elvégeznénk az Identitás ellenőrzését (magára a vezérlőre elvégezzük az Identitás-ellenőrzést az *Authorize* attribútummal). Visszatérési értékként egy *View* (Nézet) nevű *ViewResult*-ot („Nézeteredmény”) kapunk, ami (ha nincs másképp definiálva) végeredményben a függvénnyel megegyező nevű nézetet jeleníti meg. A 6. ábra a megjelenő nézetet ábrázolja. Látjuk a szerveroldali validálást felül (*RegisterViewModel* segítségével), pontokba szedve, ez akkor jelenik meg, ha *Register* gombra kattintva valamely mező(k) értéke nem megfelelő. Kliensoldalon is elvégezzük a hasonló validálást (*jQuery*-vel), ezzel jobban felhasználóbaráttá téve alkalmazásunkat. Ezek látszódnak az egyes *input* mezők alatt.



7. ábra: A Regisztációs űrlap (Register.cshtml)

A sikeres regisztráció elvégzését követően a program *Register GET* akció párját, a *POST* eseménykor bekövetkező *Register* műveletet hajtja végre [6. forráskód], a hamisítás elleni védjegy érvényesítése után (*ValidateAntiForgeryToken* attribútum). Paraméterként a függvény már a *RegisterViewModel* nézetmodellt is kéri, ami az űrlap helyes kitöltése után töltődik fel adatokkal. A nézetmodell érvényessége esetén a program a *UserManager*

(Felhasználó-kezelő) segítségével feltölti az *Identitás* adatbázisát a megadott értékekkel, majd a *SingInManager* (Beléptetés-kezelő) belépteti a felhasználót az alkalmazásba.

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Register(RegisterViewModel model,
string returnUrl = null) {
    ViewData["ReturnUrl"] = returnUrl;
    if (ModelState.IsValid) {
        var user = new ApplicationUser { UserName = model.Email,
Email = model.Email, FirstName = model.FirstName, LastName =
model.LastName, NickName = model.NickName };
        var result = await _userManager.CreateAsync(user,
model.Password);
        if (result.Succeeded) {
            _logger.LogInformation("User created a new account with
password.");
            var code = await
_userManager.GenerateEmailConfirmationTokenAsync(user);
            var callbackUrl = Url.EmailConfirmationLink(user.Id,
code, Request.Scheme);
            await
_emailSender.SendEmailConfirmationAsync(model.Email, callbackUrl);
            await _userManager.AddClaimAsync(user, new
Claim("FirstName", user.FirstName));
            await _userManager.AddClaimAsync(user, new
Claim("LastName", user.LastName));
            await _userManager.AddClaimAsync(user, new
Claim("NickName", user.NickName));
            await _signInManager.SignInAsync(user, isPersistent:
false);
            _logger.LogInformation("User created a new account with
password.");
            return RedirectToLocal(returnUrl);
        }
        AddErrors(result);
    }
    return View(model);
}
```

6. forráskód: Az Account Vezérlő Register Akciójának HTTP Post eseményekor meghívódó függvénye

A *Home(Controller)* vezérlő csak a program kezdő- és kontaktoldalának (*Index* és *Contact*) működtetéséért felelős. A *Manage(Controller)* a felhasználó saját adatainak módosítását teszi lehetővé. Akciói többek között a jelszó megváltoztatására vagy egy külső

szolgáltatóval való bejelentkezésre ad lehetőséget. A *Maps(Controller)* vezérlő a már bemutatott *postgres* adatbázis kezelésére írt osztály, melyet regisztrált felhasználók érnek el csak. Az alap CRUD [*Create, Read, Update, Delete* angol szavak kezdőbetűiből] funkciók mellett (Létrehozás, Olvasás, Frissítés, Törlés) rendelkezik egy *ToMap* (Térképre) akcióval is. Az Olvas, esetünkben *Index* néven, kiválasztja az adatbázisból a kért rekordokat [7. forráskód].

```
public async Task<IActionResult> Index(string sortOrder, string
searchString, string currentFilter, int? page) {
    ViewData["CurrentSort"] = sortOrder;
    ViewData["NameSortParm"] = String.IsNullOrEmpty(sortOrder) ?
"name_desc" : "";
    ViewData["DateSortParm"] = sortOrder == "Date" ? "date_desc" :
"Date";
    if (searchString != null) {
         page = 1;
    }
    else {
         searchString = currentFilter;
    }
    ViewData["CurrentFilter"] = searchString;
    var maps = from s in _context.Maps.Include(m => m.Sr) select s;
    if (!String.IsNullOrEmpty(searchString)) {
        maps = maps.Where(s =>
s.Title.ToLower().Contains(searchString.ToLower()));
    }
    switch (sortOrder) {
        case "name_desc":
            maps = maps.OrderByDescending(s => s.Title);
            break;
        case "Date":
            maps = maps.OrderBy(s => s.UploadTime);
            break;
        case "date_desc":
            maps = maps.OrderByDescending(s => s.UploadTime);
            break;
        default:
            maps = maps.OrderBy(s => s.Title);
            break;
    }
    int pageSize = 20;
    return View(await
PaginatedList<Maps>.CreateAsync(maps.AsNoTracking(), page ?? 1,
pageSize));
}
```

7. forráskód: A Maps Vezérlő Index Akciója

Magát a kiválasztást a zölddel kiemelt sor teszi meg. Az akció közvetlenül az adatbázisból olvassa az adatokat, nem támaszkodik egyik (nézet)modellre sem. Ezen kívül figyeljük meg a függvény argumentumait. E paraméterek változtatásával tud a felhasználó rekordokat keresni, szűrni, adatot sorba rendezni és az oldalak között lapozni. Az akció visszatérési értéke egy olyan nézet, ami gyakorlatilag egy lapozható, szűrhető lista. Mindez a *PaginatedList.cs* osztály segítségével jött létre, ami egy kiterjesztése a *List C#* osztálynak. A *Maps* vezérlő *Create* (Létrehoz) művelete szintén szétválasztható *GET* és *POST* metódusok bekövetkezésekor meghívódó műveletpárra. Míg a *GET* csak megjeleníti a *Create* nézetet, addig a *POST Create* akcióban történik az igazi „varázslat”. Paraméterként a *MapsViewModel* nézetmodellt használjuk. Először deklaráljuk a *maps* lokális változónkat (*Maps* osztály) [8. forráskód].

```
[HttpPost("Maps/Create"), ActionName("Create")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create(MapsViewModel mapsViewModel)
{
    Maps maps = new Maps();//…
```

8. forráskód: A Maps Vezérlő Create (Létrehoz) Akciója I.

Ezután a feltöltött file-(oka)t „másoljuk át” a filerendszerbe [illetve képfile esetén magába az adatbázisba is, bytetömbként a *RasterFile* mezőbe – *ESRI Shapefile* is hasonlóképpen lesz megoldva a *VectorFile* mezőbe való betöltéssel, ami egy *PostGIS geometry* típusú oszlop]. A 9. és 10. forráskód mutatja a filefeldolgozást. A művelet úgy lett kialakítva, hogy több file feltöltését is tudja kezelni, viszont a jelenleg támogatott filetípusok mindegyike egy állományból áll, ezért a nézetben és a modellben – és ezzel együtt az adatbázisban – is le van korlátozva a több file feltöltése. Az egész feldolgozás egy *try-catch* utasításpáron belül van, bármilyen probléma esetén hibaüzenetet kapunk. Minden kiválasztott file-ra, aminek a byte-okban mért hosszúsága nagyobb, mint nulla, átvesszük a filenevét a megjelenítéshez, majd kialakítjuk az elérési utat leíró szöveget. Ahhoz, hogy teljesen egyedi legyen az elérési út, a filenév elé beillesztjük a fentebb meghatározott feltöltési időt századmásodperc pontossággal, majd összeállítjuk az elérési utat külön a raszteres, és külön a vektoros állományok esetén is.

```
//…
    try {
        long size = mapsViewModel.File.Sum(f => f.Length);
        var pathR = "";
        var pathV = "";
        foreach (var formFile in mapsViewModel.File) {
            if (formFile.Length > 0) {
                mapsViewModel.FileName = formFile.FileName;
                maps.FileName = mapsViewModel.FileName;
                var parsedContentDisposition =
ContentDispositionHeaderValue.Parse(formFile.ContentDisposition);
                var fileName =
maps.UploadTime.ToString("yyyyMMdd_HHmmssff_") +
Path.GetFileName(formFile.FileName);
                pathR =
Path.Combine(Directory.GetCurrentDirectory(), "wwwroot", "Uploads",
"Raster", fileName);
                        pathV =
Path.Combine(Directory.GetCurrentDirectory(), "wwwroot", "Uploads",
"Vector", fileName);
                if (formFile.ContentType.Contains("image/")) {
                    using (var stream = new FileStream(pathR,
FileMode.Create))  {
                        mapsViewModel.FileExtension = "image";
                        maps.FileExtension =
mapsViewModel.FileExtension;
                        mapsViewModel.FilePath = "Raster/" +
fileName;
                        maps.FilePath = mapsViewModel.FilePath;
                        await formFile.CopyToAsync(stream);
                    }
                    using (var memoryStream = new MemoryStream()) {
                        await formFile.CopyToAsync(memoryStream);
                        maps.RasterFile = memoryStream.ToArray();
                    }
                }//…
```

9. forráskód: A Maps Vezérlő Create (Létrehoz) Akciója II.

Miután megállapítottuk az elérési utat a filerendszerben, elágazások segítségével meghatározzuk a file típusát [Képeknél meg tudjuk mondani, hogy valóban képről van-e szó, vektoros fileformátumok esetén – egyelőre – csak a kiterjesztésre hagyatkozunk, feltételezve, hogy a felhasználó ismeri a feltöltendő file-ok minőségét]. A 9. forráskódban látjuk a képekre vonatkozó elágazást. Figyeljük meg a kettő *using* utasítást, ami egyszer a filerendszerbe való másolását jelenti a file-nak, másodszor pedig az adatbázis *RasterFile* mezőjének feltöltését jelenti a memóriából kiolvasott bytetömbbel. Néhány másik mezőt is feltöltünk, például a kiterjesztésre vonatkozót az *image* szóval, illetve az elérési utat

tartalmazó oszlopot csak a „rövid” elérési úttal [a nézetben kicsit másképp kell elérni a gyökérkönyvtárat].

```
//…
                else if (pathV.ToLower().Contains(".geojson") ||
pathV.ToLower().Contains(".kml")) {
                    using (var stream = new FileStream(pathV,
FileMode.Create)) {
                        if (pathV.ToLower().Contains(".geojson")) {
                            mapsViewModel.FileExtension = "geoJSON";
                            maps.FileExtension =
mapsViewModel.FileExtension;
                            mapsViewModel.FilePath = "Vector/" +
fileName.ToLower().Replace(".geojson", ".js");
                        }
                        if (pathV.ToLower().Contains(".kml")) {
                            mapsViewModel.FileExtension = "kml";
                            maps.FileExtension =
mapsViewModel.FileExtension;
                            mapsViewModel.FilePath = "Vector/" +
fileName.ToLower();
                        }
                        maps.FilePath = mapsViewModel.FilePath;
                        await formFile.CopyToAsync(stream);
                    }
                }
                else if (formFile.ContentType == "") {
                    return Content("Empty file content.");
                }
                else {
                    return Content("Unsupported file Content at the
moment, please contact the page Admin, to ensure support for your
file.");
                }
            }
        }
    }
    catch (Exception e)
    {
        return Content("File processing error. " + e +
maps.FilePath);
    }//…
```

10. forráskód: A Maps Vezérlő Create (Létrehoz) Akciója III.

A 10. forráskód a vektoros fileformátumok (*GeoJSON, kml*) feldolgozását, valamint a különböző kivételek bekövetkezésekor adott visszatérési értékeket ábrázolja. Ezek után, ha minden rendben zajlott, átadjuk a nézetmodell azon attribútumait a *maps* változónak,

amiket nem kívánunk változtatni (pl.: Publikus-e), vagy egyéb metaadatok (pl.: Idő), és elmentjük a változást az adatbázisban [11. forráskód].

```
    double[] ext = { (double)mapsViewModel.Xmin,
(double)mapsViewModel.Ymin, (double)mapsViewModel.Xmax,
(double)mapsViewModel.Ymax };
    if (ModelState.IsValid) {
        try {
            maps.Extent = ext;
            maps.Title = mapsViewModel.Title;
            maps.SrId = mapsViewModel.SrId;
            maps.IsPublic = mapsViewModel.IsPublic;
            maps.Description = mapsViewModel.Description;
            maps.IsPublic = mapsViewModel.IsPublic;
            maps.UploadTime = DateTime.Now;
            maps.Uploader = User.GetNickName();
            _context.Add(maps);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        catch (Exception ex) {
            return NotFound(ex);
        }
    }
    ViewData["SrId"] = new SelectList(_context.SpatialRefSys,
"Srid", "Srid", mapsViewModel.SrId);
    return NotFound(ModelState);
}
```

11. forráskód: A Maps Vezérlő Create (Létrehoz) Akciója IV.

A *Maps* vezérlő *Details* (Részletek) akciója (*GET* kéréskor) csak a kiválasztott rekord részleteit mutatja meg egy nézetben. Az *Edit* (Szerkeszt, *GET* és *POST* metódusoknál) a rekord szerkesztését teszi lehetővé, igaz ez le van korlátozva a nem közvetlenül filetulajdonságokhoz tartozó attribútumokra. A *Delete* (Törlés) szintén *GET* és *POST HTTP* események bekövetkezése esetén érhető el, és értelemszerűen rekordok törlését hivatott elvégezni. Utóbbi kettő, tehát a szerkeszt és töröl, csak Adminisztrátori és Menedzseri jogosultságokkal érhetők el. Végül nézzük meg a vezérlő *ToMap* műveletét. Az *Index* nézetben lévő (egyik) „űrlap kitöltése” (kijelölések) utáni *POST* metódussal érhető el. Paraméterként egy azonosítókat tartalmazó tömböt kér. Ezen a tömbön iterálva feltöltjük az előtte meghatározott, a nézetmodellünket befogadó *List* (Lista) objektumot a megadott azonosítókhoz tartozó rekordokkal. Visszatérési értékként ezt a lista objektumot adjuk át a nézetnek. [12. forráskód]

```
[HttpPost, ActionName("ToMap")]
public async Task<IActionResult> ToMap(long[] ids) {
    if (ids == null) {
        return NotFound();
    }
    List<MapsViewModel> mapsViewModel = new List<MapsViewModel>();
    foreach (var id in ids) {
        var maps = await _context.Maps.SingleOrDefaultAsync(m =>
m.Id == id);
        if (maps.IsPublic == false && !User.IsInRole("Admin")) {
            return Unauthorized();
        }
        var mapsView = new MapsViewModel() {
            Id = maps.Id,
            Description = maps.Description,
            FileExtension = maps.FileExtension,
            FileName = maps.FileName,
            FilePath = maps.FilePath,
            IsPublic = maps.IsPublic,
            Title = maps.Title,
            SrId = maps.SrId,
            Xmax = maps.Extent[2],
            Ymax = maps.Extent[3],
            Xmin = maps.Extent[0],
            Ymin = maps.Extent[1],
            Uploader = maps.Uploader
        };
        mapsViewModel.Add(mapsView);
    }
    return View(mapsViewModel);
}
```

12. forráskód: A Maps vezérlő ToMap művelete

4.4. A cél megvalósítása III., a Nézetek

A Modellek és Vezérlők kialakítása után már „csak” a Nézeteket kell definiálnunk, ezzel elérkezve a leglátványosabb és legváltozatosabb fejezethez. A 2. és 7. ábrákon már láttunk egyszerű nézeteket, ezeknél azonban sokkal jobb és szebb oldalakat fogunk megalkotni az elkövetkezendőkben. Az egységes kinézetet a *Views* könyvtár *Shared* mappájában található nézetek segítik elő. Itt van a *_Layout.cshtml* osztály, ami az elrendezést (*layout*-ot) tartalmazza. Mindig ez a nézet jelenik meg, ha a vezérlőben *View* típusú a visszatérési érték. Lényege csupán annyi, hogy ebben meghatározzuk az egész weboldal kinézetét, majd különböző nézeteket renderelünk a body elemében a *RenderBody() Razor* függvénnyel, valamint elvégezzük a szükséges *CSS* és *JavaScript* importálásokat. A 13.

forráskód ezt a *_Layout.cshtml* osztályt mutatja néhány elemet elhagyva. Minden, az alkalmazásról készült ábrán (képernyőképen) látjuk az itt leírt elemeket, kiegészítve a *RenderBody()* függvény nézeteivel.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
<!-- Title; CSS és Bootstrap hivatkozások elrejtve -->
</head>
<body>
    <nav class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-
toggle="collapse" data-target=".navbar-collapse">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a asp-area="" asp-controller="Maps" asp-action="Index"
class="navbar-brand">Maps</a>
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li><a asp-area="" asp-controller="Home" asp-
action="Index">Home</a></li>
                    <li><a asp-area="" asp-controller="Home" asp-
action="Contact">Contact</a></li>
                </ul>
                @await Html.PartialAsync("_LoginPartial")
            </div>
        </div>
    </nav>
    <div class="container body-content">
        @RenderBody()
        <hr />
<!-- Footer elrejtve -->
    </div>
<!-- Scriptekre való hivatkozások (jQuery, stb.) elrejtve -->
    @RenderSection("Scripts", required: false)
</body>
</html>
```

13. forráskód: A közös nézet (_Layout.cshtml)

Az *Account*, *Manage* és *Home* vezérlőkhöz tartozó nézeteket itt nem tárgyalom, hiszen mindegyik a vezérlő akcióinak kiszolgálására hivatott, és adatbázisügyileg nem érdekesek. A *postgres* adatbázis kezelésére szolgáló, *Maps* vezérlőhöz tartozó nézetek a *Views Maps* nevű könyvtárában találhatók. Az *Index* nézet ezen belül az adatbázis *Maps* táblázatának kezelő kezdőlapja.

```
@model PaginatedList<Diplomamunka_VF_2017.Models.Maps>
@{ ViewData["Title"] = "Index";}
<h2>Index</h2>
<p><a asp-action="Create">Create New</a></p>
<form asp-action="Index" method="get" class="form-inline">
    <div class="form-group no-color">
        <p><label class="control-label">Find by Title:
</label><input type="text" name="SearchString"
value="@ViewData["currentFilter"]" class="form-control"/>
            <input type="submit" value="Search" class="btn btn-
default" /> |
            <a asp-action="Index">Back to Full List</a></p>
    </div>
</form>
@using (Html.BeginForm("ToMap", "Maps")) {
    <table class="table td">
        <thead>
            <tr>
                <th>
                    <a asp-action="Index" asp-route-
sortOrder="@ViewData["NameSortParm"]" asp-route-
currentFilter="@ViewData["CurrentFilter"]">Title</a>
                </th>
                <th>Filename</th>
                <th>EPSG code</th>
                <th>Uploader</th>
                <th>
                    <a asp-action="Index" asp-route-
sortOrder="@ViewData["DateSortParm"]" asp-route-
currentFilter="@ViewData["CurrentFilter"]">Last Updated</a>
                </th>
                <th></th>
            </tr>
        </thead>
```

14. forráskód: A Maps/Index nézet I.

A 14. forráskód az Index nézet első részét mutatja. Felül definiáljuk a függőségi befecskendezést (*dependency injection*) a [nézet]modell eléréséhez. Ez a rész felelős még a létrehozásért (külön linken), a keresésért, és a sorbarendezésért. Ismerjük fel a vezérlőben

már látott objektumokat (pl.: *searchString*, *currentFilter*) [7. forráskód]. A nézet többféleképpen tud a vezérlővel kommunikálni, melyekből itt látunk kettőt (*ViewData*; *name HTML* attribútum *input* mezőkben – ez köti az akció azonos nevű paraméteréhez az *input* értékét). Ebben a kódban látjuk még a táblázat fejlécének definiálását.

```
        <tbody>
           @foreach (var item in Model) {
              <tr>
                 <td style="vertical-align:middle">
                     @Html.DisplayFor(modelItem => item.Title)</td>
                 <td style="vertical-align:middle">
                     @Html.DisplayFor(modelItem =>
item.FileName)</td>
                 <td style="vertical-align:middle">
                     @Html.DisplayFor(modelItem =>
item.Sr.Srid)</td>
                 <td style="vertical-align:middle">
                     @Html.DisplayFor(modelItem =>
item.Uploader)</td>
                 <td style="vertical-align:middle">
                     @Html.DisplayFor(modelItem =>
item.UploadTime)</td>
                 <td style="vertical-align:middle">
                     <a asp-action="Details" asp-route-
id="@item.Id">Details</a>
                     @if (item.Uploader == null ||
User.ToString().ToLower() == item.Uploader.ToLower() ||
User.IsInRole("Admin") == true || User.IsInRole("Manager") == true){
                        <text> | </text><a asp-action="Edit" asp-
route-id="@item.Id">Edit</a> }
                     @if (User.IsInRole("Admin") == true ||
User.IsInRole("Manager") == true) {
                        <text> | </text><a asp-action="Delete" asp-
route-id="@item.Id">Delete</a> }
                     @if (item.IsPublic || item.Uploader == null ||
User.IsInRole("Admin")) {
                        <text> | </text><input id="cb" class="check"
type="checkbox" name="ids" value="@item.Id" /><text> | </text>
                        if (item.FileExtension == "image") {
                            <a href="~/Uploads/@item.FilePath"><img
src="~/Uploads/@item.FilePath" class="img-thumbnail"
alt="@item.FileName" width="64" height="64" style="border:none"
/></a> }
                        else {
                           <a
href="~/Uploads/@item.FilePath">@item.FileName</a>
}}</td></tr>}</tbody></table>
```

15. forráskód: A Maps/Index nézet II.

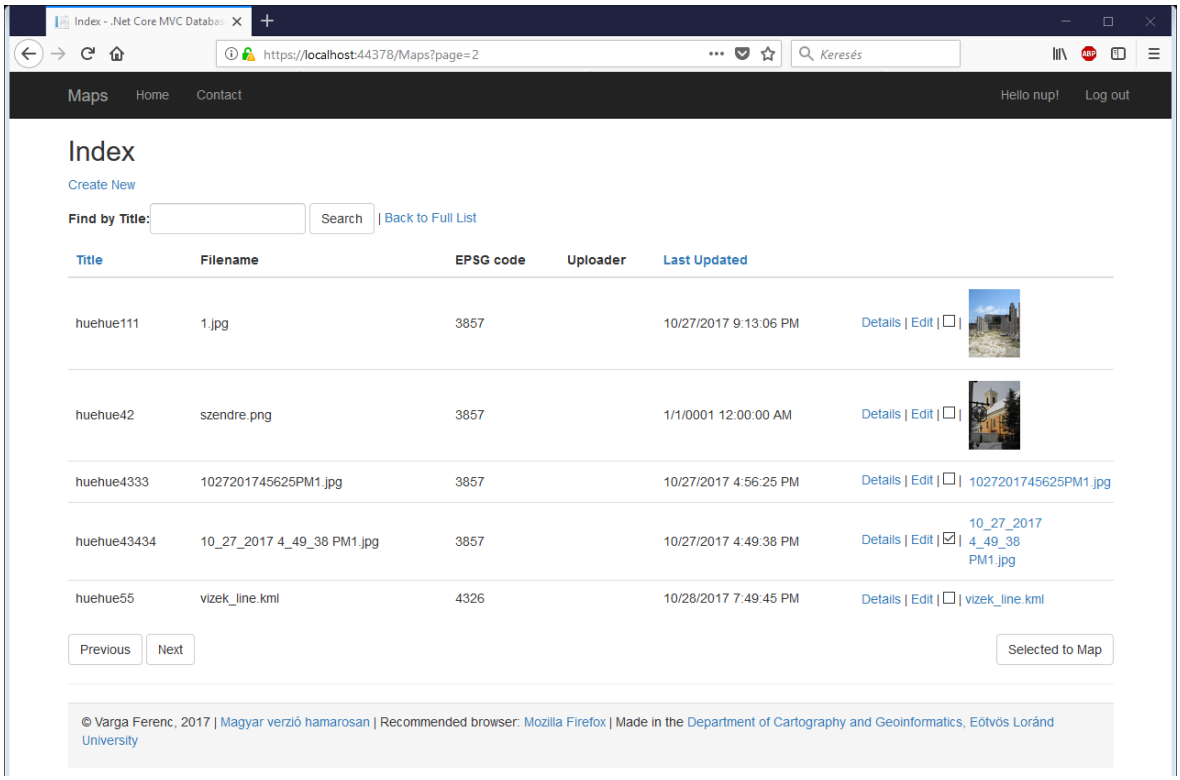
A 15. forráskód az *Index* nézet folytatása, ebben a táblázat törzsét deklaráljuk. Minden vezérlőben visszatérített adatra a modellben létrehozunk a táblázatban egy sort, majd minden mező alá beírjuk a megfelelő attribútumot. Ezeket ún. *HTML*- és *Tag Helper*-ek (HTML- és elemsegítők) segítségével tesszük meg, ezzel dinamikusan kötve az egyes modelladatokat a megfelelő elemekhez. Magyarul nem kell minden rekordot kézzel megírni, elég csak egy ciklust meghatározni. A sor utolsó oszlopába kerülnek a vezérlő többi akciójához vezető linkek (*Details*, *Edit*, *ToMap*) és egy miniatűr kép, ha weben támogatott képről van szó (pl.: *png, jpg*) vagy egy link a file letöltéséhez. Persze mindez a felhasználói fiók jogosultságaitól (szerepkör, követelés) is függ.

```
    <div style="float:right;">
        <input id="SelectedToMapButton" type="submit"
value="Selected to Map" asp-action="ToMap" class="btn btn-default"
disabled="disabled" />
    </div>
}
@{
    var prevDisabled = !Model.HasPreviousPage ? "disabled" : "";
    var nextDisabled = !Model.HasNextPage ? "disabled" : "";
}
<a asp-action="Index"
   asp-route-sortOrder="@ViewData["CurrentSort"]"
   asp-route-page="@(Model.PageIndex - 1)"
   asp-route-currentFilter="@ViewData["CurrentFilter"]"
   class="btn btn-default @prevDisabled">
    Previous
</a>
<a asp-action="Index"
   asp-route-sortOrder="@ViewData["CurrentSort"]"
   asp-route-page="@(Model.PageIndex + 1)"
   asp-route-currentFilter="@ViewData["CurrentFilter"]"
   class="btn btn-default @nextDisabled">
    Next
</a>
<script type="text/javascript"
src="~/lib/jquery/dist/jquery.js"></script>
<script src="~/js/cbClick.js"></script>
```

16. forráskód: A Maps/Index nézet III.

A 16. forráskód az *Index* nézet lezárását ábrázolja, a gombok definiálását és a script-eket. A *SelectedToMap* azonosítójú gomb a *ToMap* akciót hívja meg, a *Previous* és *Next* (előző és következő) gombok az oldalak közötti lapozást teszik lehetővé. A 8. ábrán láhatjuk az *Index* nézetet a webböngészőben néhány tesztadattal. Figyeljük meg a fentebb definiált

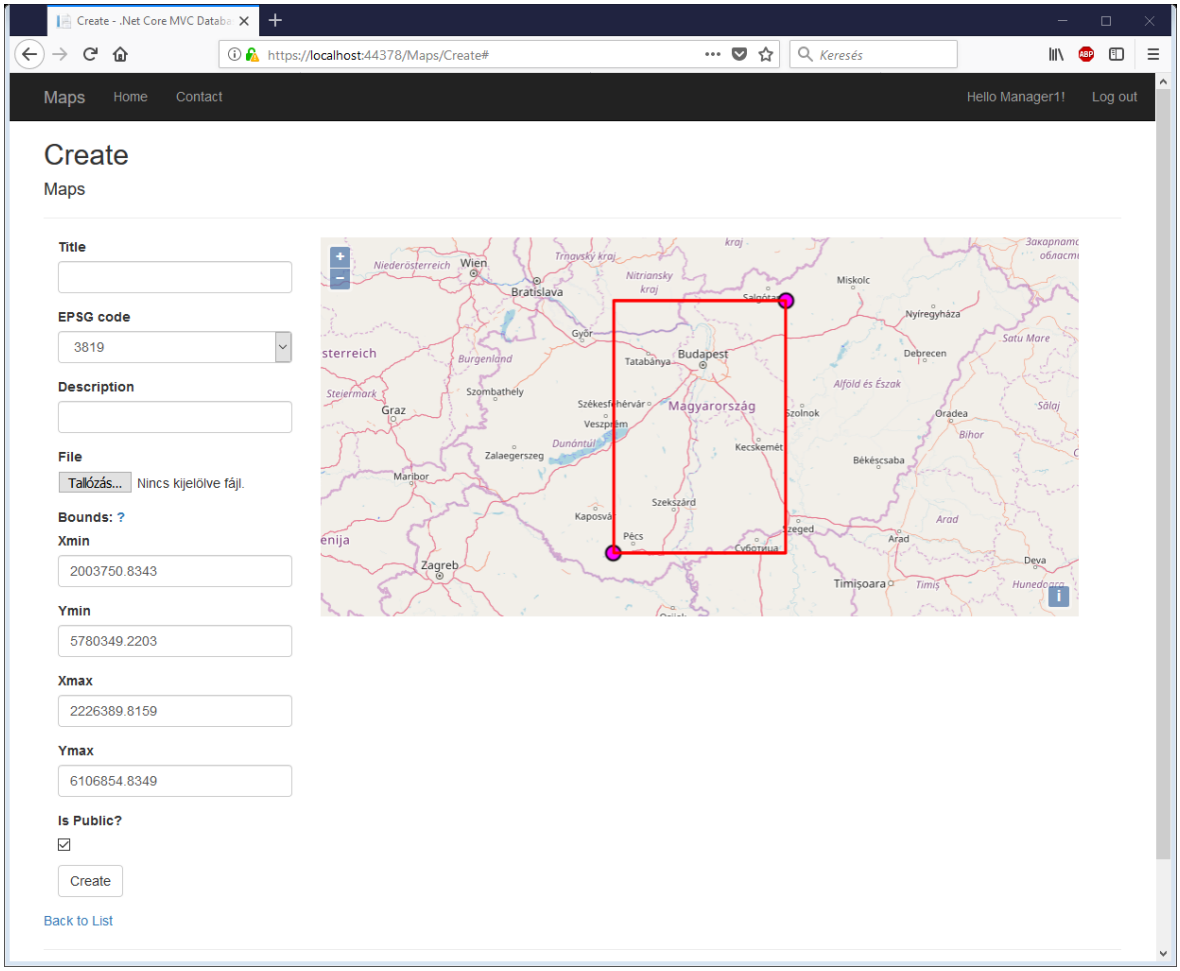
cshtml file renderelt változatát: a Létrehoz (*Create New*), a keresés, a fejléc, a törzs az adatokkal, alul a gombok. Vessünk egy pillantást az elrendezésre (*layout*): felül látjuk a navigációs panelt, alul a *footer*-t (lábjegyzet). A navigációs panel kis felbontás esetén összecsukódik, ebben látjuk a bejelentkezett állapotot is.



8. ábra: A Maps/Index nézet a webböngészőben néhány tesztadattal (Adminisztrátori jogosultsággal)

A *Create* nézettel a felhasználó elé a honlap */Maps/Create URL* címén található weboldalt tárjuk (a vezérlő segítségével) [9. ábra]. Ebben egy rekord hozzáadására van lehetőség, olyan formában, hogy a kötelezően kitöltendő mezők egy részét kérjük be (pl.: Cím, EPSG kód, maga a file stb.), és a többit a vezérlő *Create* akciójában megalkotjuk [11. forráskód]. Ezen kívül van egy „kvázi-georeferálást” segítő térkép, amin a *bounding box*-ot („körülvevő téglalapot” – a képek kiterjedését) módosíthatjuk tetszőlegesen, ezzel az x és y koordináták minimum és maximum értékét beállítva. Ez a funkció képek feltöltése esetén érdekes, a téglalapon (valójában foktrapézon) belül lesz a kép, erre a területre lesz „ráfeszítve”. A jelenleg támogatott vektoros állományokat a megjelenítő dolgozza fel, a koordináták itt csak megjelenítést szolgálják (erre a területre nagyít). Természetesen „kézzel” is be lehet írni a koordinátákat a megfelelő *input* mezőkbe, ekkor változás esetén a térképen lévő téglalap is megváltozik. [Egyelőre csak *Web Mercator* vetületben

(*EPSG:3857*) láthatók a koordináták, azaz méterben, *WGS84*-es dátumban.] Bármilyen regisztrált felhasználó tölthet fel adatot [egyelőre].



9. ábra: A Create nézet a weboldalon

Az *Edit* nézet nagyban hasonlít a *Create* nézethez, gyakorlatilag a file megváltoztatásán kívül mindent megváltoztathatunk. Ugyanakkor az *Edit* csak *Manager* és *Admin* jogosultságokkal érhető el, illetve, ha a felhasználó saját maga töltötte fel a rekordot. A *Delete* nézet a rekord törlését segíti elő, és eggyel szigorúbb, mint az *Edit*, csak *Admin*, és *Manager* státuszban lévő felhasználók számára érhető el. A *Details*-t mindenki, aki regisztrált eléri, de ez csak a metaadatokat mutatja meg, kicsit részletesebben, mint az *Index*, de a file részleteit ennek segítségével nem látjuk. Ezzel elérkeztünk a *ToMap* akció azonos nevű nézetéhez. Ebben a feltöltött file-okat végre megjelenítjük a *Create*-ben már látott térképen, kliensoldali *JavaScript*-et használva. A 17. forráskód mutatja az oldal felépítését a modell, valamint a külső stílusok és script-ek injektálásával. Látjuk, hogy a bal oldalon lesz a térképünk, a jobb oldalon pedig egy lista az *Index* nézetben kiválasztott rekordokkal

egy-egy *div* elemben. A lista segítségével ki- és bekapcsolhatjuk a térképen a fedvényeket, valamint az átlátszóságukat egy-egy csúszka segítségével állíthatjuk.

```
@model List<MapsViewModel>
@{ ViewData["Title"] = "Map";}
<link rel="stylesheet"
href="https://openlayers.org/en/v4.3.3/css/ol.css" type="text/css">
<!-- The line below is only needed for old environments like
Internet Explorer and Android 4.x -->
<script
src="https://cdn.polyfill.io/v2/polyfill.min.js?features=requestAnim
ationFrame,Element.prototype.classList,URL"></script>
<script src="https://openlayers.org/en/v4.4.0/build/ol.js"></script>
<h4>@Model.Count.ToString() selected</h4>
<hr />
<div style="display:table;width:100%;height:100%">
    <div id="map" class="map"
style="width:auto;height:auto;display:table-
cell;position:relative;padding-right:10px"></div>
    <div id="list" style="display:table-
cell;width:auto;height:auto;vertical-align:top">
        @for (var i = 0; i < Model.Count; i++) {
            <input type="checkbox" value="@Model[i].FileName"
checked="checked" id="@i" class="fileName"
onchange="toggleLayer(@i)"/>
            @if (Model[i].IsPublic == false) {
                <label>@Html.DisplayFor(model =>
Model[i].Title)</label>
            }
            else {
                <a
href="~/Uploads/@Model[i].FilePath">@Html.DisplayFor(model =>
Model[i].Title)</a>
            }
            <input id="@i" class="opacRange" type="range" min="0"
max="1" step="0.1" value="1" onchange="setOpac(@i)"/>
        }
    </div>
</div>
```

17. forráskód: A ToMap nézet felépítése

Természetesen attól, hogy definiáljuk a *div* elemeket, nem lesz térképünk, ennek megvalósítását *OpenLayers JavaScript* függvénykönyvtárral oldjuk meg, melyet a következő fejezetben taglalok.

5. A térképes megjelenítő

Ez a fejezet az előbb bemutatott nézetek kiegészítésére szolgáló térképek megalkotásának módjáról szól. A *Create* és *Edit* nézetekben látott térképek a gyökér mappa (*wwwroot*) *js* könyvtárának *OlExtentHelper.js* file-jának segítségével jött létre.

```
var xMin = document.getElementById('Xmin');
var xMax = document.getElementById('Xmax');
var yMin = document.getElementById('Ymin');
var yMax = document.getElementById('Ymax');
var SW = new ol.Feature({geometry: new
ol.geom.Point([parseFloat(xMin.value),parseFloat(yMin.value)])
,});
var NE = new ol.Feature({geometry: new
ol.geom.Point([parseFloat(xMax.value),parseFloat(yMax.value)])
,});
SW.setStyle(new ol.style.Style({
  image: new ol.style.Icon(({color: '#FF00FF',
    crossOrigin: 'anonymous',
    src:
'https://openlayers.org/en/v4.3.3/examples/data/dot.png'}))
}));
NE.setStyle(new ol.style.Style({
  image: new ol.style.Icon(({color: '#FF00FF',
    crossOrigin: 'anonymous',
    src:
'https://openlayers.org/en/v4.3.3/examples/data/dot.png'}))
}));
var polygonFeature = new ol.Feature({id: 'polygonFeature',
  geometry: new ol.geom.Polygon([[[ol.proj.transform([18, 46],
'EPSG:4326', 'EPSG:3857')], [ol.proj.transform([18, 48],
'EPSG:4326', 'EPSG:3857')], [ol.proj.transform([20, 46],
'EPSG:4326', 'EPSG:3857')], [ol.proj.transform([20, 48],
'EPSG:4326', 'EPSG:3857')]]])});
var vectorSource = new ol.source.Vector({features: [SW, NE]});
var vectorLayer = new ol.layer.Vector({source: vectorSource});
var polySource = new ol.source.Vector({features:
[polygonFeature]});
var polyLayer = new ol.layer.Vector({source: polySource});
var view1 = new ol.View({projection: 'EPSG:3857',
  center: [212e4, 602e4],
  zoom: 6});
```

18. forráskód: Az OlExtentHelper.js I.

A 18-20. forráskódok ezen állomány nagy részét mutatják be. Először definiáljuk a *HTML* dokumentum megfelelő elemeit, jelen esetben a nézet koordinátákra vonatkozó *input* objektumokat, majd az *OpenLayers*-hez szükséges változókat. Ezek két pontot és egy

poligont (téglalapot) jelentenek, valamint a konténerüket jelentő „*layer*” típusú objektumokat. Nagyjából Magyarországra nagyítva létrehozunk még egy *View* típusú változót is [Ezen a területen vannak a vektoros rétegek is].

```
var map = new ol.Map({
  layers: [
    new ol.layer.Tile({ source: new ol.source.OSM({}) })],
  overlays: [vectorLayer, polyLayer],
  target: 'map',
  view: view1});
map.getView().fit(vectorSource.getExtent());
var modify = new ol.interaction.Modify({ source: vectorSource
});
map.addInteraction(modify);
var polyMod = function (xmin, ymin, xmax, ymax) {
  var polyCoords = [];
  var coords = ((xmin).toString() + "," + (ymin).toString() +
" " + (xmin).toString() + "," + (ymax).toString() + " " +
(xmax).toString() + "," + (ymax).toString() + " " +
(xmax).toString() + "," + (ymin).toString()).split(' ');
  for (var i in coords) {
    var c = coords[i].split(',');
    polyCoords.push(ol.proj.transform([parseFloat(c[0]),
parseFloat(c[1])], 'EPSG:3857', 'EPSG:3857'));}
  map.removeLayer(polyLayer);
  polySource.removeFeature(polygonFeature);
  polygonFeature.setGeometry(new
ol.geom.Polygon([polyCoords]));
  polySource.addFeature(polygonFeature);
  map.addLayer(polyLayer);}; //…
```

19. forráskód: Az OlExtentHelper.js II.

A 19. forráskód elején létrehozzuk a *map* objektumot, ami megjelenik az azonos nevű *div* elemben a nézetben. Háttértérképként *OpenStreetMap* csempéket jelenítünk meg, majd erre ráhelyezzük a fent deklarált „*layereket*”, és megadjuk neki a *View* objektumot, mint „nézetet”. [Ez a nézet nem ugyanaz a nézet, mint a fenti fejezetekben.] A *map* objektum létrehozását követően definiáljuk a metódusokat, melyek ezt az objektumot – vagy valamely tulajdonságát – manipulálják. Először a két pontra nagyítunk, melyek meghatározzák a téglalapot is, majd „engedélyezzük” ezen elemek „kézzel” való módosítását a felhasználó számára. A *polyMod* nevű függvény a bemeneti 4 koordinátából számítja ki, és rajzolja újra a „körülírt téglalapot”. A bemeneti 4 koordináta a két pont mozgatásakor előidézett esemény hatására érkezik paraméterként (a *vectorLayer onchange* eseményekor). A koordinátákat tartalmazó *input* mezők megváltoztatásakor a megadott értékre változtatjuk a megfelelő pont

koordinátáját, ezzel az előző eseményt újra előidézve, és a téglalapot ismét ráillesztve a két pontra. [20. forráskód]

```
//…
vectorLayer.on('change', function (evt) {
  var SWCoord = SW.getGeometry().getCoordinates();
  var NECoord = NE.getGeometry().getCoordinates();
  var swxy = ol.coordinate.toStringXY(SWCoord, 4);
  var nexy = ol.coordinate.toStringXY(NECoord, 4);
  var xmin, ymin, xmax, ymax;
  xmin = parseFloat(swxy.split(',')[0].trim());
  ymin = parseFloat(swxy.split(',')[1].trim());
  xmax = parseFloat(nexy.split(',')[0].trim());
  ymax = parseFloat(nexy.split(',')[1].trim());
  xMax.value = xmax;
  yMax.value = ymax;
  xMin.value = xmin;
  yMin.value = ymin;
  polyMod(xmin, ymin, xmax, ymax);}); //…
yMax.onchange = function (evt) {
  var ymax = parseFloat(yMax.value.replace(',', '.'));
  var xmax = parseFloat(xMax.value.replace(',', '.'));
  var coords = [xmax, ymax];
  map.removeInteraction(modify);
  NE.getGeometry().setCoordinates([xmax, ymax]);
  map.addInteraction(modify);};//…
```

20. forráskód: Az OlExtentHelper.js III. (részlet)

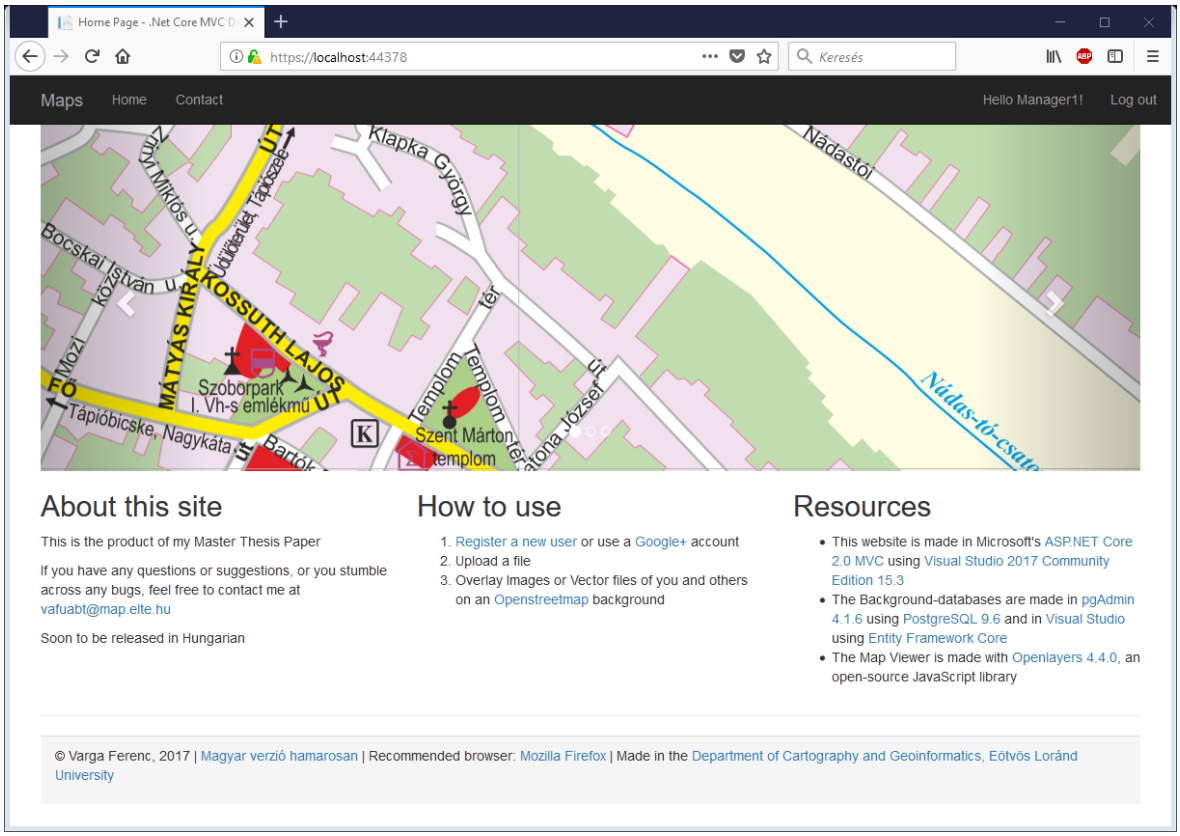
A megjelenítő kicsit másképp néz ki. A *ToMap* nézet felépítését már láttuk a 17. forráskódban, ezt *script*-tel kiegészítve hozzuk létre a térképet a feltöltött file-lal, mint fedvény. A 21. forráskódban látjuk ezt a *script*-et. Az eleje nagyon hasonlít az *OlExtentHelper.js*-ben látottakra, létrehozunk néhány változót, majd a *map* objektumot, szintén *OpenStreetMap* háttértérképpel. Figyeljük meg a modell használatát. Ez a nézet egy *MapsViewModel* modell-listával dolgozik, magyarul több rekordot is átadhatunk a nézetnek (*Maps* vezérlő *ToMap* akciója). Ez azt jelenti, hogy egyszerre több fedvényt helyezhetünk el a térképen. Mindig a modell-lista első elemére nagyítunk (*extent*). Miután deklaráltuk a *map* objektumot, a modell-lista elemein iterálva megadjuk – a filekiterjesztést tartalmazó mező segítségével – a feldolgozási mód típusát az *OpenLayers* függvénykönyvtárnak. Minden így létrejött objektumot egy tömbbe helyezünk, majd a végén egyesével, mint fedvény hozzáadjuk a *map* objektumhoz. Ugyanitt minden modell-lista elemhez kialakítjuk a megfelelő függvényeket a *layer* ki- és bekapcsolásához, valamint átlátszóságuk állításához. [Ez a *script Microsoft Edge* böngészőben nem működik.]

```
var imgLayer = [];
var extent = [];
extent.push(@Model[0].Xmin);
extent.push(@Model[0].Ymin);
extent.push(@Model[0].Xmax);
extent.push(@Model[0].Ymax);
var imageArr = [];
var map = new ol.Map({
  layers: [ new ol.layer.Tile({ source: new ol.source.OSM() }), ],
  target: 'map',
  view: new ol.View({
    projection: 'EPSG:3857',
    center: ol.extent.getCenter(extent),
    zoom: 2,})});
map.getView().fit(extent);
@for (var i = 0; i < Model.Count; i++) {
  <text>imageArr.push("../../Uploads/" + '@Model[i].FilePath');
  if ("@Model[i].FileExtension.ToLower()" === "geojson") {
    imgLayer.push(new ol.layer.Vector({
      source: new ol.source.Vector({
        format: new ol.format.GeoJSON(),
        url: imageArr[imageArr.length -1],})}));}
  else if ("@Model[i].FileExtension.ToLower()" === "image") {
    imgLayer.push(new ol.layer.Image({
      source: new ol.source.ImageStatic({
        url: imageArr[imageArr.length - 1],
        imageExtent: extent}),
      opacity: 1.0}));}
  else if ("@Model[i].FileExtension.ToLower()" === "kml") {
    imgLayer.push(new ol.layer.Vector("KML", {
      projection: 'EPSG:4326',
      source: new ol.source.Vector({
        url: imageArr[imageArr.length - 1],
        format: new ol.format.KML({})})}));}
  var toggleLayer = function (id) {
    var cb = document.getElementById(id);
    if (cb.checked) { map.addOverlay(imgLayer[id]); }
    else if (!cb.checked) { map.removeOverlay(imgLayer[id]);}};
  var setOpac = function (id) {
    var range = document.getElementsByClassName('opacRange')[id].value;
    var ra = document.getElementsByClassName('opacRange')[id].id;
    imgLayer[ra].setOpacity(parseFloat(range));
    map.renderSync();};</text>}
for (var i = 0; i < imgLayer.length; i++) {
  map.addOverlay(imgLayer[i]);}
```

21. forráskód: A ToMap nézet script része, a megjelenítő

6. Kezelési útmutató

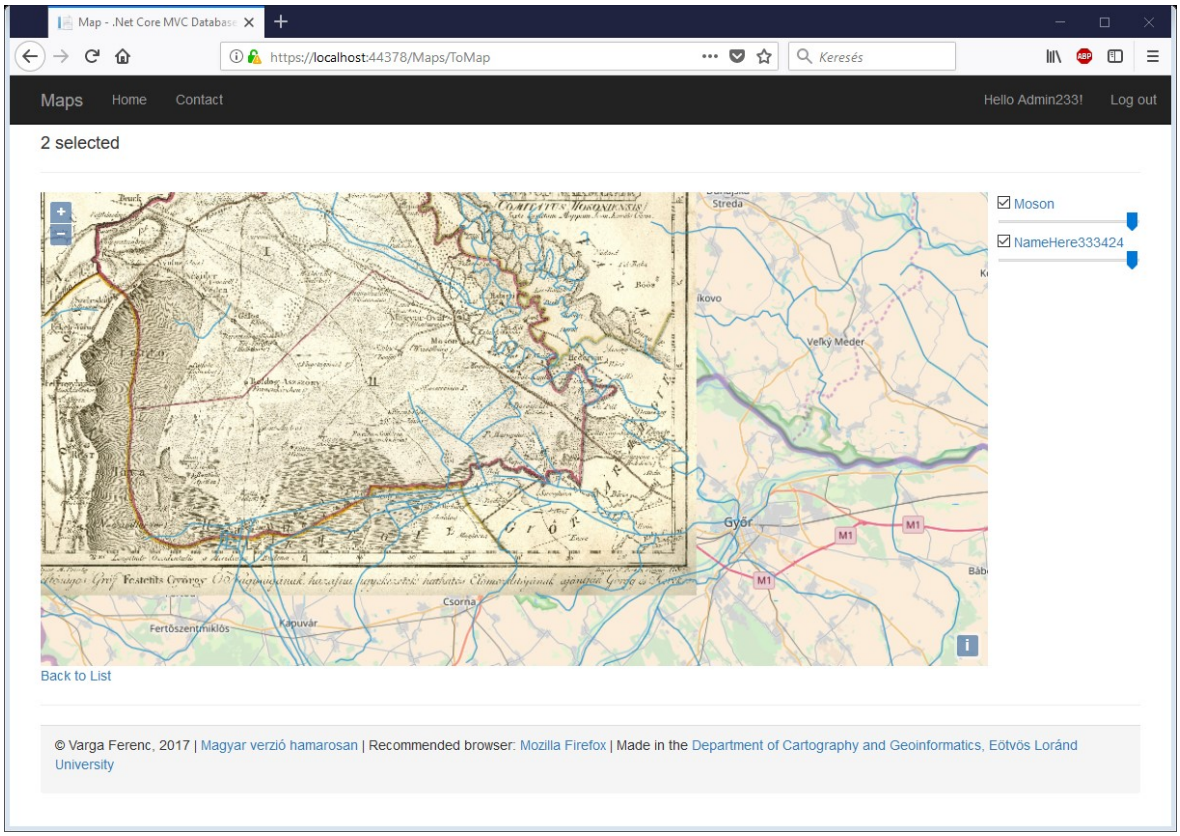
Az előző fejezetekben elmeséltem az alkalmazás kialakítását, és a háttérinformációkat, ebben a fejezetben összefoglalom, és a felhasználói oldalról közelítem meg magát a programot. A kezdőlapra navigálva a kezdőoldal fogad minket (Home/Index). Itt találunk néhány információt a weblappal kapcsolatban, ami [egyelőre] teljes mértékben angol nyelvű. [10. ábra]



10. ábra: A weboldal kezdőlapja

A *Contact* gombra kattintva (koppintva) a 2. ábrán látható oldalt kapjuk, az adminisztrátor e-mail címével. A *Maps* gombra bökve érjük el az adatbázist, de ha nem vagyunk regisztrálva, vagy bejelentkezve, akkor automatikusan a bejelentkezési oldalra irányít a program. Miután bejelentkeztünk/regisztráltunk, illetve, ha be voltunk jelentkezve már el is érjük a kívánt *Maps/Index* oldalt [8.ábra]. Innen a navigálás minimális angoltudással egyértelmű: a *Create New*-ra klikkelve új rekordot adhatunk hozzá, a *Details* a részleteit mutatja, az *Edit*-tel szerkeszthetünk egy-egy rekordot. A *Delete* a törlést teszi lehetővé. A keresőmezőben fent címre tudunk keresni, a táblázat fejléceire kattintva sorbarendezhetjük a fejléc szerint, a rekordokat. A jelölőnégyzetek bepipálásával aktívvá

válik alul a *Selected to Map* gomb. Ennek megnyomásával a *Maps/ToMap* URL címre kerülünk, ahol látjuk az általunk kiválasztott entitásokat. A 11. ábrán látunk 2 kiválasztott entitást a térképen, valamint a jobb oldalon a hozzájuk tartozó jelölőnégyzeteket a ki- és bekapcsoláshoz, és a csúszkákat az átlátszóság állításához.



11. ábra: A térképes megjelenítő (Maps/ToMap)

7. Befejezés

Befejezésül néhány szó a körülményekről és a jövőről. Diplomamunkámban bemutattam egy szabadon elérhető webes adatbáziskezelőt, térképes megjelenítővel. Teljesen nyílt forráskódú projekteken keresztül jutottam el a „végtermékig”, magyarul „csak” a *hardware*-ért és a *domain*-ért kell fizetni az alkalmazás futtatásáért. Ez ugyanakkor nem azt jelenti, hogy a program készen van, csak egy olyan stádiumba érkezett, hogy már egy zárt csoportot rá lehet engedni tesztelni (úgy is mondhatnánk, hogy bétában van). Nagyon sok javítási, bővítési, fejlesztési lehetőség van még. Ilyen például a limitált számú fileformátumok támogatása, vagy az egy-két biztonsági rés, ami még fellelhető. Meg kell alkotni a e-mail-konfirmációt és a két-faktoros azonosítást, valamint a különböző vetületek támogatását is. Ezen kívül szeretném mielőbb magyarra is lefordítani az alkalmazást. Ettől függetlenül úgy gondolom, hogy amit készítettem megfelel egy kiindulási alapnak további fejlesztésekhez.

8. Irodalomjegyzék

[1.] Daniel Roth – Rick Anderson – Shaun Luttin (2017.09.03.): Introduction to ASP.NET Core.

https://docs.microsoft.com/en-us/aspnet/core/

https://github.com/aspnet/Docs/blob/master/aspnetcore/index.md

[2.] Steve Smith (2016.10.14.): Overview of ASP.NET Core MVC

https://docs.microsoft.com/hu-hu/aspnet/core/mvc/overview

[3.] Microsoft (2017.07.25.): ASP.NET and .NET is part of a great open source .NET community

https://www.asp.net/open-source

[4.] Novák I., Velvárt A., Granicz Á., Balássy Gy., Hajdrik A., Sellers, M., Hillar G., Molnár Á., Joydip K.: Visual Studio 2010 and .NET 4 Six-in-One, (2010), Indianapolis, Wrox, ISBN-13: 978-0470499481

[5.] ASP.NET Core, GitHub repository (gyűjtemény)

https://github.com/aspnet/home

[6.] Mario Szpuszta – Matthew MacDonald (2005): Pro ASP.NET 2.0 in C#, Apress. ISBN 1-59059-496-7

[7.] Wikipedia contributors (2017.11.30.): ASP.NET

https://en.wikipedia.org/w/index.php?title=ASP.NET&oldid=812877982

[8.] The PostgreSQL Global Development Group (2017): About.

https://www.postgresql.org/about/

[9.] PostGIS Project Steering Committee (2017): About PostGIS.

http://postgis.net/

[10.] The pgAdmin Development Team (2017): Introduction.

https://www.pgadmin.org/

[11.] OpenLayers (2017): Overwiew.

http://openlayers.org/

[12.] Jeffrey T. Fritz (2016.01.19.): ASP.NET 5 is dead – Introducing ASP.NET Core 1.0 and .NET Core 1.0

https://blogs.msdn.microsoft.com/webdev/2016/01/19/asp-net-5-is-dead-introducing-asp-net-core-1-0-and-net-core-1-0/

A weboldalak utolsó elérése: 2017.12.28.

9. Köszönetnyilvánítás

Jelen diplomamunka megírásáért köszönetet mondok elsősorban témavezetőmnek, Elek Istvánnak, aki bátorításával elérte, hogy belevágjak ebbe a komplex feladatba és a feladat megoldása közben is mindig elérhető volt, ha kérdésem támadt. Továbbá köszönet jár Kovács Bélának a szerver felállításával kapcsolatos kérdések tisztázásában nyújtott elengedhetetlen segítségéhez. Ugyanitt kell megemlítenem kedves menyasszonyomat, Farkas Rékát, aki lelki támogatásával lehetővé tette ezen dolgozat megírását, és az alkalmazás felépítését.

NYILATKOZAT

Alulírott Varga Ferenc (NEPTUN azonosító: LNK3YL), a Nyílt forráskódú térképes adatbáziskezelő és -megjelenítő a weben című diplomamunka szerzője fegyelmi felelősségem tudatában kijelentem, hogy dolgozatom önálló munkám eredménye, saját szellemi termékem, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

A témavezető által elfogadott és elbírált diplomamunka elektronikus közzétételéhez (PDF formátumban a tanszéki honlapon)

HOZZÁJÁRULOK NEM JÁRULOK HOZZÁ

Budapest, 2018.01.08.

Varga Ferenc

Hozzájárulok a szakdolgozat benyújtásához:

Budapest, 2018.01.08.

Elek István

10. Mellékletek

1. Jelen dolgozat PDF/A formátumban.
2. Az alkalmazás minden forráskódja (*.sln*): *Diplomamunka_VF_2017* könyvtár.
3. Egy tömörített állomány (*.zip*) a *host*-oláshoz: *Publish* könyvtár.
4. A *postgres* adatbázis: *postgres.sql*.