



EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

PROGRAMOZÁSI NYELVEK ÉS FORDÍTÓPROGRAMOK

TANSZÉK

Beadandó kezelő rendszer megvalósítása

Témavezető:

Poór Artúr

egyetemi tanársegéd

Szerző:

Csiki Erik Gergely

programtervező informatikus BSc

Budapest, 2021

Az eredeti szakdolgozati / diplomamunka témabejelentő helye.

Tartalomjegyzék

1. Bevezetés	4
2. Felhasználói dokumentáció	5
2.1. Bejelentkezés és nyelvválasztás	5
2.1.1. Bejelentkezés	5
2.1.2. Nyelvválasztás	6
2.2. Szerepkörök	7
2.2.1. Rendszergazda	7
2.2.2. Tárgyfelelős	10
2.2.3. Gyakorlatvezető	13
2.2.4. Hallgató	17
2.2.5. Mindenki számára elérhető oldalak	19
3. Fejlesztői dokumentáció	22
3.1. Keretrendszerek és az alkalmazás felépítése	22
3.1.1. Keretrendszerek	22
3.1.2. Az alkalmazás felépítése	22
3.2. Naplózás	24
3.3. Adatbázis	25
3.3.1. Technológiák	25
3.3.2. Adatbázis <i>code first</i> objektumai	26
3.3.3. Az adatbázis táblái	27
3.4. Model réteg	32
3.4.1. Üzleti logika	32
3.4.2. Adatok megjelenítésére szolgáló modellek	39
3.4.3. Adatok bevitelére szolgáló modellek	40

3.4.4.	Egyéb segédosztályok	41
3.5.	Vezérlő réteg	44
3.5.1.	<i>Home</i> vezérlő	44
3.5.2.	<i>Base</i> vezérlő	45
3.5.3.	<i>Admin</i> vezérlő	46
3.5.4.	<i>Instructor</i> vezérlő	48
3.5.5.	<i>Teacher</i> vezérlő	49
3.5.6.	<i>Student</i> vezérlő	50
3.6.	Nézet réteg	52
3.6.1.	<i>Home</i> vezérlő nézetei	52
3.6.2.	<i>Admin</i> vezérlő nézetei	52
3.6.3.	<i>Instructor</i> vezérlő nézetei	53
3.6.4.	<i>Teacher</i> vezérlő nézetei	53
3.6.5.	<i>Student</i> vezérlő nézetei	53
3.6.6.	Megosztott nézetek	54
3.7.	Lokalizáció	54
4.	Tesztelés	57
4.1.	Futtatott teszt esetek	58
4.1.1.	ASS_AdminTests.cs	58
4.1.2.	ASS_OtherTests.cs	58
4.1.3.	ASS_InstructorTests.cs	59
4.1.4.	ASS_StudentTests.cs	59
4.1.5.	ASS_TeacherTests.cs	59
5.	Összegzés	61
5.1.	Használt fejlesztői eszközök	61
5.2.	Telepítés	61
5.3.	Továbbfejlesztési lehetőségek	62
6.	Köszönetnyilvánítás	63
	Irodalomjegyzék	64

Ábrajegyzék	66
Táblázatjegyzék	68
Forráskódjegyzék	69

1. fejezet

Bevezetés

A szakdolgozatom témája egy beadandó kezelő rendszer megvalósítása webes alkalmazásként. Az elkészült rendszer az Assignment Supervisor System nevet kapta (röviden ASS), a továbbiakban így hivatkozom rá.

Az alkalmazás célja, hogy segítse az egyetemi munkát, mind a hallgatók, mind az oktatók részére. Lehetőségünk van tantárgyak létrehozására, amelyekhez létrehozhatunk csoportokat. A hallgatók ezekbe a csoportba tudnak jelentkezni, amit a gyakorlatvezető hagyhat jóvá. A hallgatók számára lehetőségünk van határidővel ellátott feladatokat kiírni. A feladatkiírásánál támogatott a **Markdown** és a **L^AT_EX** kifejezések. A beküldött megoldásokat a gyakorlatvezetők értékelhetik. Az alkalmazás támogatja a külföldi hallgatók munkáját, számukra elérhető az alkalmazás angol nyelvű változatban.

2. fejezet

Felhasználói dokumentáció

2.1. Bejelentkezés és nyelvválasztás

Az oldalra érkezve a kezdőoldalt láthatjuk, ahol egy üdvözlő üzenet fogad minket. Majd lehetőségünk nyílik bejelentkezni a rendszerbe, vagy a rendszer által támogatott lokalizációt tudjuk kiválasztani (2.1 ábra). ¹

Köszöntjük!

Az oldalt az INF-es (Pandorás) azonosítóval tudjuk használni.

INF-es felhasználónév

INF-es jelszó

Bejelentkezés

Magyar English

2.1. ábra. Főoldal

2.1.1. Bejelentkezés

A rendszerbe bejelentkezni az INF-es felhasználónkkal tudunk. Ha a bejelentkezés sikertelen volt, azt a rendszer hibaüzenetekkel jelzi a számunkra (2.2 ábra).

¹Erre majd még a bejelentkezés után is lehetőségünk nyílik lásd később, 2.2.5: „Mindenki számára elérhető oldalak”

Köszöntjük!

Az oldalt az INF-es (Pandorás) azonosítóval tudjuk használni.

A INF-es felhasználónév megadása kötelező!

A INF-es jelszó megadása kötelező!


Bejelentkezés

Magyar

English

2.2. ábra. Bejelentkezési hiba

Amennyiben a bejelentkezés sikeres volt, a *szerepkörnek* megfelelő kezdőoldalon találjuk magunkat (2.3 ábra).




 Főoldal

 Teszt Elek / student Csoport jelentkezés Kijelentkezés

Feladatok

Csoport

▲ Gyak #1

Feladat	Határidő	Értékelés	
1. Házi feladat	2021.05.30. 01:00		 Feladat megtekintése
Elso	2021.05.07. 00:00		 Feladat megtekintése
2. Házi feladat	2021.05.30. 01:00		 Feladat megtekintése

1

1-1 a(z) 1 elemből

2.3. ábra. Sikeres bejelentkezés

2.1.2. Nyelvválasztás

A rendszer kilistázza a támogatott lokalizációkat (jelenleg magyar és angol). Alapértelmezett beállítás a magyar. Ezt felültudjuk írni, ha valamelyik gombra rákattintunk. (2.4 ábra)

Welcome!

This site can be only used with an INF (Pandora) account.

INF username	
INF password	
Login	
Magyar	English

2.4. ábra. Nyelvváltás

2.2. Szerepkörök

A felhasználók négy csoportba tartozhatnak:

- Rendszergazda
- Tárgyfelelős
- Gyakorlatvezető
- Hallgató

Egy felhasználó tartozhat több szerepkörbe. Ha egy felhasználó több szerepkörbe is tartozik, akkor a felület menüsorán megjelenik egy "Szerepkör váltás" lenyitható menü, ahol a felhasználóhoz rendelt szerepköröket találjuk, a kiválasztott linkre kattintva a csoporthoz tartozó kezdőoldalra navigáljuk magunkat. A felhasználóhoz a szerepköröket a felhasználó létrehozásakor is megadhatjuk, valamint a létrehozást követően tudjuk módosítani.

2.2.1. Rendszergazda

A rendszergazda a következő funkciókat érheti el:

- Tantárgy létrehozása, módosítása, törlése, tárgyi információk megtekintése
- Felhasználó létrehozása, módosítása, a felhasználók adatainak a megtekintése

Ha rendszergazdaként jelentkezőnk be az alábbi két táblázat fogad minket a kezdőoldalon (2.5 ábra).

Főoldal		Teszt Elek / admin		Tárgy létrehozás	Felhasználó létrehozás	Kijelentkezés
---------	--	--------------------	--	------------------	------------------------	---------------

Tantárgyak információi

Keresés...		
Tantárgy neve	Tárgyfelelős(ök)	
Funkcionális programozás EA+GY	Bozó István (bozo_i), Poór Artúr (poora)	Szerkesztés Törölés
Funkcionális nyelvek EA+GY	Poór Artúr (poora)	Szerkesztés Törölés

1-2 a(z) 2 elemből

(a) Tantárgyak táblázata

Felhasználók listája

Keresés...				
Név	Neptun kód	Email	Szerepkörök	
Teszt Elek	student	student@inf.elte.hu	Hallgató	Szerkesztés
Poór Artúr	poora	poora@inf.elte.hu	Tárgyfelelős, Gyakorlatvezető	Szerkesztés
Csikó Erik Gergely	csikie	csikie@inf.elte.hu	Gyakorlatvezető	Szerkesztés
Bajkó János Róbert	bajkoj	bajkoj@inf.elte.hu	Gyakorlatvezető	Szerkesztés
Bozó István	bozo_i	bozo_i@inf.elte.hu	Tárgyfelelős	Szerkesztés

1-5 a(z) 5 elemből

(b) Felhasználók táblázata

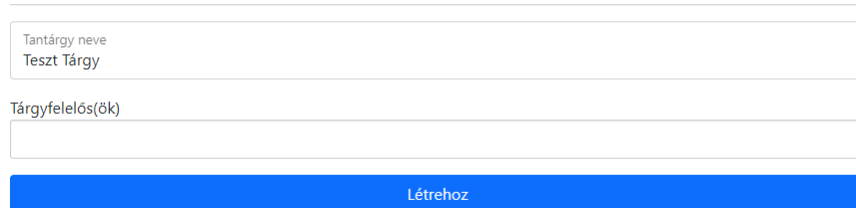
2.5. ábra. Rendszergazdai szerepkör kezdőoldala

Az első táblázatban a rendszerben létrehozott tantárgyak és a hozzájuk tartozó információk olvashatóak le. A táblázatban az egyes tantárgyakhoz tartozó adatok módosíthatóak, illetve az egész tárgyat lehet törölni. A módosítás során validálásra kerül, hogy a módosított név létezik-e már a rendszerben, ha igen, akkor ezt a rendszer jelzi számunkra. A második táblázatban a rendszerben létrehozott felhasználókat és a hozzájuk tartozó információkat láthatjuk. A rendszergazda a felhasználók adatait és szerepköreit tudja módosítani.

Tantárgy létrehozása

A „Tárgy létrehozás” linkre kattintva az alkalmazás átnavigál minket egy űrlapra, ahol az új tantárgy szükséges adatait tudjuk kitölteni (2.6(a) ábra). Ha az adatok validálása és feldolgozása sikeres, akkor visszanavigálódunk a kezdőoldalra. Az esetleges validálási hibákat a rendszer jelzi számukra (2.6(b) ábra).

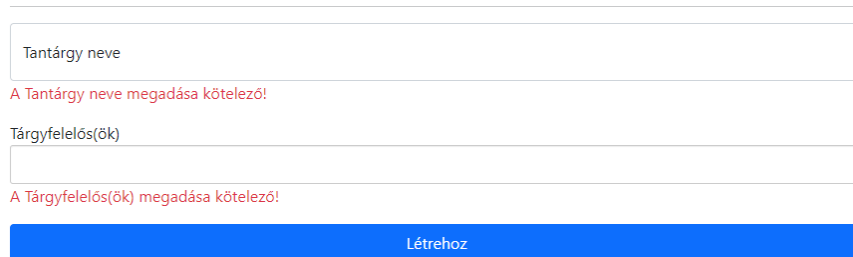
Tantárgy létrehozás



The screenshot shows a web form titled "Tantárgy létrehozás". It has two input fields: "Tantárgy neve" with the value "Teszt Tárgy" and "Tárgyfelelős(ök)". Below the fields is a blue button labeled "Létrehoz".

(a) Űrlap

Tantárgy létrehozás



The screenshot shows the same web form as in (a), but with validation errors. The "Tantárgy neve" field has a red error message below it: "A Tantárgy neve megadása kötelező!". The "Tárgyfelelős(ök)" field also has a red error message below it: "A Tárgyfelelős(ök) megadása kötelező!". The blue "Létrehoz" button is still visible at the bottom.

(b) Adatok validálása

2.6. ábra. Tantárgy létrehozás

Felhasználó létrehozása

Felhasználót létrehozni a „Felhasználó létrehozás” linkre kattintva tudjuk megtenni, ami továbbnavigál minket egy űrlapra, ahol az új felhasználónak az adatait tudjuk megadni (2.7(a) ábra). Ha az adatok validálása sikeres, akkor a felhasználó elkészült és visszanavigálódunk a kezdőoldalra, ha nem volt sikeres, akkor a rendszer ezt hibaüzenetekkel jelzi nekünk (2.7(b) ábra).

Felhasználó létrehozás

Neptun kód

Név

Email

Jelszó

Jelszó megerősítése

Szerepkör(ök)

Létrehoz

(a) Űrlap

Felhasználó létrehozás

Neptun kód

A Neptun kód megadása kötelező!

Név

A Név megadása kötelező!

Email

A Email megadása kötelező!

Jelszó

A Jelszó megadása kötelező!

Jelszó megerősítése

A Jelszó megerősítése megadása kötelező!

Szerepkör(ök)

A Szerepkör(ök) megadása kötelező!

Létrehoz

(b) Adatok validálása

2.7. ábra. Felhasználó létrehozás

2.2.2. Tárgyfelelős

Ha tárgyfelelősként jelentkezünk be a rendszerbe, akkor az alábbi kezdőoldal fogad minket (2.8 ábra). A kezdőoldalon egy táblázat található, amiben látjuk azo-

kat a tantárgyakat, és a tantárgyakhoz tartozó csoportokat, amelyeknek a felelősei vagyunk. A tárgyfelelős a következő funkciókat használhatja:

- Csoport létrehozása egy tantárgyhoz
- Csoport módosítása

Tantárgyak

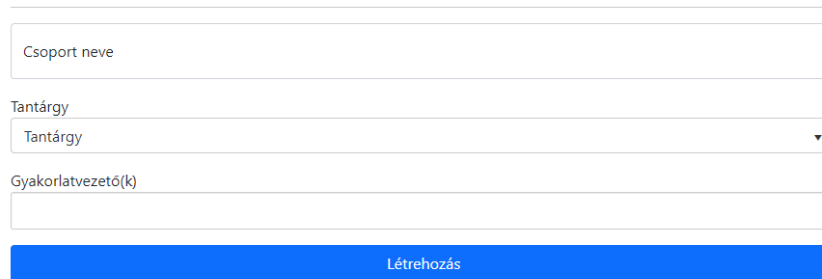
Keresés...		
Tantárgy		
▼ Funkcionális programozás EA+GY		
Csoport	Gyakorlatvezető(k)	
Gyak #1	Bajkó János Róbert (bajkoj), Csiki Erik Gergely (csikie)	Szerkesztés
▼ Funkcionális nyelvek EA+GY		
Csoport	Gyakorlatvezető(k)	
Gyak #2	Csiki Erik Gergely (csikie)	Szerkesztés
<div> 1 </div> <div>1-2 a(z) 2 elemből</div>		

2.8. ábra. Tárgyfelelős kezdőoldala

Csoport létrehozása

A menüsoron a „Csoport létrehozás” linkre kattintva a rendszer átirányít minket egy űrlapra, ahol létre tudunk hozni egy csoportot (2.9 ábra). Az adatokat a rendszer validálja, és az esetleges hibákat jelzi számunkra.

Csoport létrehozás



Csoport neve

Tantárgy

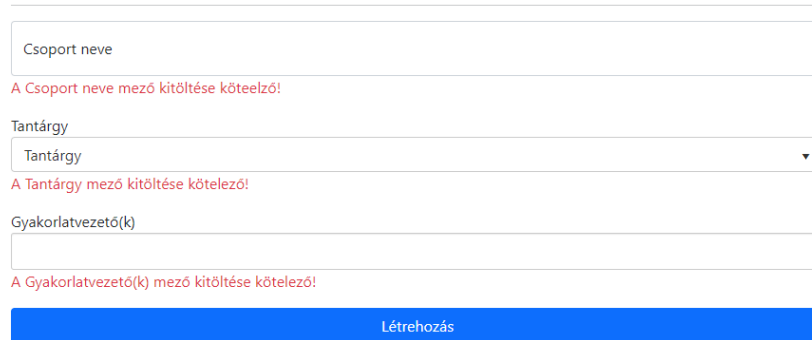
Tantárgy

Gyakorlatvezető(k)

Létrehozás

(a) Űrlap

Csoport létrehozás



Csoport neve

A Csoport neve mező kitöltése kötelező!

Tantárgy

Tantárgy

A Tantárgy mező kitöltése kötelező!

Gyakorlatvezető(k)

A Gyakorlatvezető(k) mező kitöltése kötelező!

Létrehozás

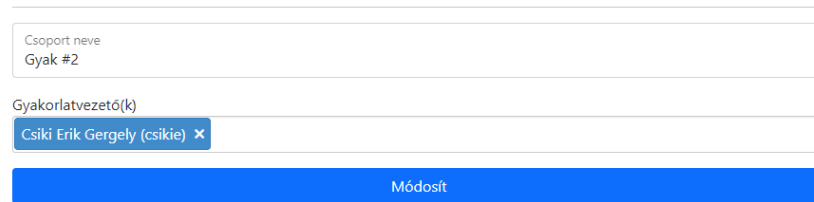
(b) Adatok validálása

2.9. ábra. Csoport létrehozás

Csoport módosítása

Csoportokat szerkeszteni a kezdőoldalon található táblázat segítségével tudunk. A táblázatban lenyitható minden kilistázott tantárgy. Itt találjuk a tantárgyakhoz már létrehozott csoportokat. Minden tantárgy mellett találunk egy „Szerkesztés” gombot, melyre kattintva elérhetővé válik a csoport szerkesztése. Az adatok validálásra kerülnek, az esetleges hibákat a rendszer jelzi számunkra (2.10 ábra).

Csoport szerkesztése



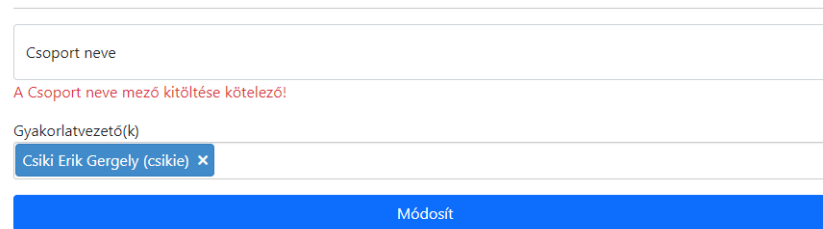
Csoport neve
Gyak #2

Gyakorlatvezető(k)
Csiki Erik Gergely (csikie) ✕

Módosít

(a) Űrlap

Csoport szerkesztése



Csoport neve

A Csoport neve mező kitöltése kötelező!

Gyakorlatvezető(k)
Csiki Erik Gergely (csikie) ✕

Módosít

(b) Adatok validálása

2.10. ábra. Csoport módosítása

2.2.3. Gyakorlatvezető

Gyakorlatvezetőként bejelentkezve a rendszerbe a 2.11 ábrán látható kezdőoldal fogad minket. Az oldalon az „Értékelendő beadandók” cím alatt, a hozzánk rendelt csoportok hallgatóit láthatjuk egy-egy táblázatban, ahol láthatjuk, hogy egy hallgató az adott feladatra adott-e be megoldást. A „Hallgatói várólista” cím alatt szintén egy táblázatot találunk (2.11(b) ábra), ahol tantárgyanként csoportosítva a következő információkat olvashatjuk le:

- Hallgató neve
- Hallgató neptun kódja
- Csoport neve

1. Házi feladat

- Mely csoporthoz legyen létrehozva²

Feladat létrehozása

Feladat neve

Kezdés dátuma: 2021.05.07 00:00

Befejezés dátuma: 2021.05.07 00:00

Leírás

Ennek a szövegnek markdown formátumban kell lennie. Íme egy rövid áttekintés:

- Ez egy lista egyik eleme, *dólt betűvel*.
- Ez pedig egy másik elem, **félkövérrel**. Ügyeljünk arra, hogy a szöveg többi része igazítva maradjon.

Néha előre megformázott szöveget akarunk írni:

```
~~~~~
előre megformázott
szöveg
~~~~~
```

Habár `szavakat` bármikor írhatunk a backtick (```) szimbólum segítségével.

Csoport(ok)

Létrehozás Előnézet Törlés

2.12. ábra. Feladat létrehozás

A rendszer támogatja, hogy a feladatnak leírása ne csak egyszerű szöveg legyen. A szövegdobozban megadhatunk **Markdown** és **L^AT_EX** kifejezéseket is. Mielőtt létrehoznánk a feladatot, meg tudjuk tekinteni, hogy a hallgató milyen formában fogja látni a kiírva a feladatot. Így le tudjuk ellenőrizni kényelmesen a feladat leírását, valamint azt is tudjuk ellenőrizni, hogy a **Markdown** és **L^AT_EX** kifejezéseinket helyesen írtuk-e meg. Ha végeztünk, a „Létrehozás” gombbal tudjuk elküldeni a rendszernek az adatokat. Az adatokat a rendszer leellenőrzi, az esetleges hibákat jelzi számunkra (2.13).

²A lenyíló kiválasztó menüben lehetőségünk van több csoportot is kiválasztani

Feladat létrehozása

Feladat neve

A Feladat neve megadása kötelező!

Kezdés dátuma

2021.05.07 00:00

A Kezdés dátuma megadása kötelező!

Befejezés dátuma

2021.05.07 00:00

A Befejezés dátuma megadása kötelező!

Leírás

A Leírás megadása kötelező!

Csoport(ok)

A Csoport(ok) megadása kötelező!

2.13. ábra. Adatok validálása

Jelentkezések bírálata

Egy hallgatónak a csoportba való jelentkezését a „Hallgatói várólista” táblázatában tudjuk megtenni (2.11(b) ábra), az utolsó oszlopban lévő gombok segítségével. Miután elvégeztük a bírálatot, a táblázatból törlődik a jelentkezett hallgató, és az oldal frissítése után, a megfelelő táblázatban látjuk, hogy a hallgatót a rendszer felvette a jelentkezett csoportba.

Beadott munka értékelése

Egy feladatra beadott megoldás értékeléséhez a kívánt feladat oszlopában kattintsunk a *szürke négyzetre*. Ilyenkor a rendszer átirányít minket az értékelő felületre (2.14). A rendszer a felületre a „Megoldás(ok)” alatt felsorolja a hallgatónak az összes beadott megoldását a beküldés ideje szerint csökkenően rendezve. Elég egy megoldást értékelnünk, de értékelhetjük az összeset is. Viszont a rendszer a legutolsó értékelést veszi számításba. Ezt fogja a hallgató is látni. A beadott megoldások között a kívánt sorra kattintva tudjuk kiválasztani, hogy melyik megoldást szeretnénk változtatni. Ilyenkor a „Beadott megoldás” alatt látjuk, mi a hallgatónak a beadott megoldása. Az értékelésünket az oldal alján található űrlapon tudjuk megtenni. Az

értékelés után a rendszer a kezdőoldalra navigál minket. Az értékelt feladatnál a *szürke négyzet egy körben lévő pipára* cserélődik.

Feladat értékelése

Információk

Hallgató neve	Teszt Elek (student)
Feladat neve	2. Házi feladat
Kezdés dátuma	2021. 05. 07. 0:00:00
Befejezés dátuma	2021. 05. 30. 1:00:00

Beadott megoldás

Ez az én megoldásom.

Megoldás(ok)

Beadási idő: 2021. 05. 07. 16:30:28

Értékelés

Mentés

2.14. ábra. Feladat értékelése

2.2.4. Hallgató

Hallgatóként bejelentkezve a 2.15 ábrán látható kezdőoldal fogad minket. A „Feladatok” cím alatti táblázatban a hallgató számára listázásra kerül az összes olyan csoportja, ahova elfogadták a jelentkezését. A táblázatban csoportokra lebontva jelennek meg a hallgató számára a kiírt feladatok. A táblázatban egy feladatról a következő információkat láthatjuk: neve, határideje, kapott értékelés.

Feladatok

Keresés...

Csoport

Gyak #1

Feladat	Határidő	Értékelés	
1. Házi feladat	2021.05.30. 01:00		Feladat megtekintése
Elso	2021.05.07. 00:00		Feladat megtekintése
2. Házi feladat	2021.05.30. 01:00		Feladat megtekintése

1-1 a(z) 1 elemből

2.15. ábra. Hallgató kezdőoldala

A hallgató az alábbi funkciókat használhatja:

- Csoportba jelentkezés
- Megoldás beadása

Csoportba jelentkezés

Csoportba jelentkezni a „Csoport jelentkezés” menüpontra kattintva tudunk. A rendszer egy űrlapot biztosít számunkra (2.16 ábra), ahol listázásra kerülnek a rendszerben található csoportok, amelyekre még nem jelentkezünk. A rendszer lehetőséget biztosít számunkra, hogy akár egyszerre több csoportra is leadjuk a jelentkezésünket. Jelentkezésünket a „Jelentkezés” gombbal tudjuk továbbítani a rendszer számára. Az űrlap validálásra kerül, hogy üresen ne tudjuk beküldeni azt. Sikeres jelentkezés esetén a rendszer a kezdőoldalra navigál minket.

Csoport jelentkezés

Csoport

Csoport(ok)

Jelentkezés

2.16. ábra. Csoportba jelentkezése

Megoldás beadása

Megoldás beküldéséhez válasszuk ki a kívánt feladatot, amire megoldást szeretnénk beküldeni, majd kattintsunk a „Feladat megtekintése” gombra. Ekkor a rendszer egy új ablakban megnyitja a feladatot (2.17 ábra). Az oldalon a következő információkat látjuk:

- Határidő visszaszámláló
- Feladat neve, leírása
- Beküldött megoldások
- Űrlap a megoldás beküldéséhez

Határidő: 22 nap 8:12:53

2. Házi feladat

Ennek a szövegnek markdown formátumban kell lennie. Íme egy rövid áttekintés:

- Ez egy lista egyik eleme, *dolt betuvel*.
- Ez pedig egy másik elem, **félkövérrel**. Ügyeljünk arra, hogy a szöveg többi része igazítva maradjon.

Néha előre megformázott szöveget akarunk írni:

```
elore megformázott
szöveg
```

Habár *szavakat* bármikor írhatunk a backtick (‘) szimbólum segítségével.

Bizonyosan kódrészletet is szeretnénk mutatni:

```
main :: IO ()
main = putStrLn "I'm highlighted"
```

Számos nyelv támogatott, többek között *ada, agda, c, cpp, erlang, java, lex, python* és *yacc*.

Képletek és egyenletek pont úgy működnek, mint *LaTeX*-ben. Használható szövegközi és kiemelt mód a matematikai tartalom szép szedésére. A szövegközi módnál, például $\sum_{j=0}^n (a + bj)$ esetében, két dollár szimbólum közé írjuk a képletet vagy egyenletet. A nyitó dollár szimbólum után és a záró előtt nincs szóköz. A kiemelt mód nagyobb hangsúlyt ad:

$$\sum_{j=0}^n (a + bj) = a + (a + b) + \dots + (a + nb)$$

A támogatott *LaTeX* makrók [ezen az oldalon](#) vannak felsorolva.

Emellett még linkek is [illeszthetők](#) a szövegbe.

Beküldési idő: 2021. 05. 07. 16:30:28

✓

2.17. ábra. Feladat megtekintése

2.2.5. Mindenki számára elérhető oldalak

A rendszerben jelenleg három olyan oldal található, amelyet minden szerepkörben elérhetünk. Az egyik oldal a felhasználónk adatainak megtekintésére szolgál. Ezt a funkciót a menüsoron a nevünkre kattintva tudjuk elérni. Ezen a felületen (2.18 ábra) a következőket tekinthetjük meg a „Személyes adatok” cím alatt: név, neptun kód, e-mail cím és a felhasználónkhoz rendelt szerepkörök. Ezen felületen továbbá be

tudjuk állítani, hogy a rendszer milyen lokalizációval működjön (magyar és angol). Ezt a megfelelő gombra kattintva tudjuk változtatni.



2.18. ábra. Profil oldal

A másik két oldal az esetleges nem várt hibákról tájékoztat minket. Ezen hibák két kategóriába oszthatók: jogosulatlan kérés a rendszer felé, egyéb nem várt hiba. Jogosulatlan kérés akkor lép fel, ha megpróbálunk a szerepkörünkhöz nem tartozó funkciót elérni a rendszerben. Például: csak hallgatói szerepkörrel rendelkező felhasználóval vagyunk bejelentkezve a rendszerbe és a webcím végén a „Student”-et lecseréljük „Admin”-ra. Ezzel olyan kérést indítunk a rendszernek, hogy navigáljon minket a rendszergazdai szerepkörhöz tartozó kezdőoldalra, amihez nincs jogosultságunk, ezért a rendszer megtadja a hozzáférést a kért oldalhoz. Ilyenkor a rendszer az alábbi 2.19 ábrán látható oldalra navigál minket, ahonnan lehetőségünk van visszatérni a szerepkörünkhöz tartozó kezdőoldalra.

Hozzáférés megtagadva.

A viharba, nincs hozzáféréseid ehhez a kéréshez! 😞

[Vissza a kezdőoldalra](#)

2.19. ábra. Jogosulatlan kérés

Egyéb nem várt hiba lehet például, hogy a rendszer nem tud csatlakozni a hozzá tartozó adatbázishoz. Ilyenkor az alábbi 2.20 ábrán látható oldalra navigál minket.

Error.

Hiba lépett fel a folyamat során! Kérlek beszélj a rendszergazdával!

2.20. ábra. Egyéb nem várt hiba

3. fejezet

Fejlesztői dokumentáció

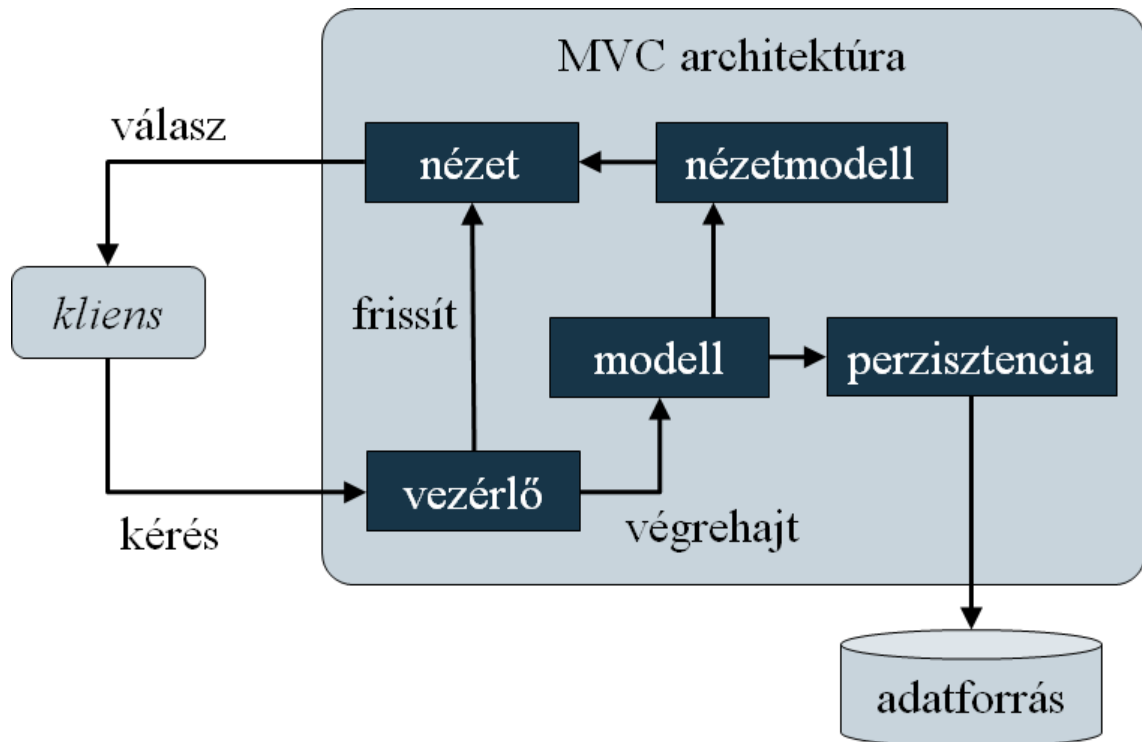
3.1. Keretrendszerek és az alkalmazás felépítése

3.1.1. Keretrendszerek

Az alkalmazás ASP.NET core 3.1 keretrendszerben készült [1], ami egy nyílt forráskódú, webes alkalmazások készítésére szolgáló programkönyvtár, melyet a *Microsoft* fejleszt. A keretrendszer lehetővé teszi, hogy az alkalmazás több platformon is tudjon futni (*Linux*, *macOS* és *Windows*). Továbbá a *Kendo UI Core for jQuery*[2] keretrendszer biztosítja számunkra a felületen található felhasználóbarát táblázatokat, űrlap elemeket. A saját *HTML* elemek stílusait a *Bootstrap*[3] keretrendszer biztosítja.

3.1.2. Az alkalmazás felépítése

Az alkalmazás az *MVC* architektúrára épül (3.1 ábra)[4]. Tehát három rétegre bontható a felépítése, Modell-Nézet-Vezérlő. A Modell (angolul *Model*) réteg tartalmazza az üzleti logikát, amely az adatokat kezeli és kapcsolatban van az adatbázissal. A nézet réteg (angolul *View*) felelős a megjelenítésért. A vezérlő réteg (angolul *Controller*) fogadja a kliens kéréseit és válaszol azokra. Az *MVC* architektúra fő előnye, hogy jól elkülöníthetők a rétegek, így a nézet független marad a modelltől. Ezáltal, ha szükséges, könnyedén le tudjuk cserélni az egész alkalmazás nézetét, vagy fordítva újra implementálhatjuk a modell réteg működését, anélkül hogy ez a nézeten bármi gondot okozna.



3.1. ábra. A Modell-Nézet-Vezérlő architektúra

Az alkalmazásban a könnyebb és egyszerűbb fejleszthetőség miatt a *Model* réteget több komponensre bontjuk. Így az alábbi komponensekből áll össze a *Model* réteg:

```

ASS
├── ASS.BLL/
│   ├── Interfaces/
│   └── Services/
├── ASS.DAL/
│   ├── Models/
│   ├── ASSContext.cs
│   └── DbInitializer.cs
├── ASS.WEB/
│   ├── Models/
│   │   ├── DTOs/
│   │   └── ViewModels/

```

ASS.BLL: az üzleti logikai réteget megvalósító komponens (angolul *Business Logic Layer*).

ASS.DAL: az adatelérési réteget megvalósító komponens (angolul *Data Access Layer*).

ASS.WEB.Models: ebben a komponensben tároljuk az adatok bevitelére és az adatok megjelenítésére szolgáló osztályokat.

ASSContext.cs: az adatbázist leíró osztály.

DbInitializer.cs: az adatbázist létrehozó statikus osztály.

3.2. Naplózás

Az alkalmazás fájl szintű naplózást tartalmaz, amit a *Serilog.Extensions.Logging.File* nyílt forráskódú programkönyvtár használatával valósítjuk meg [5]. Az alkalmazás automatikusan naplózza a futás közbeni eseményeket és az esetleges kivételeket. Természetesen támogatott a saját bejegyzések létrehozása is. A naplózás beállításait az *appsettings.json* (?? ábra) fájlban tudjuk személyre szabni. Az alábbi négy értéket szabjuk személyre az alkalmazáshoz:

- **PathFormat:** itt tudjuk megadni az alkalmazás naplófájljainak a mentési helyét, és egy sablont a fájlok nevére. A *{Date}* paraméter helyére az aktuális dátum kerül beillesztésre (pl.: 20210513). Ha az elérési útban található mappa nem létezik azt a programkönyvtár automatikusan létrehozza a számunkra.
- **OutputTemplate:** itt adható meg a bejegyzések sablonja, hogy hogyan nézzenek ki a bejegyzés³. Az alkalmazás a következő sablont használja a bejegyzésekre: *[Időbélyeg] - [Esemény súlyossági szintje] - [Üzenet] Új sor [Kivétel (ha van)]*.
- **LogLevel:** itt állíthatjuk be, hogy milyen minimum szintű események kerüljenek naplózásra [6]. A jelenlegi beállítással az alkalmazás minden legalább *Information* szinttel rendelkező eseményt naplóz.

```
1 ...  
2 "Logging": {
```

³Ezen a linken részletes leírást olvashatunk az *OutputTemplate*-ben használható paraméterekről.

```
3 "PathFormat": "../Logs/log-{Date}.log",
4 "OutputTemplate": "[{Timestamp:yyyy.MM.dd HH:mm:ss}] - [{Level:u
    ↪ }]] - {Message}{NewLine}{Exception}",
5 "LogLevel": {
6     "Default": "Debug",
7     "Microsoft": "Information"
8 }
9 },
10 ...
```

3.1. forráskód. Naplózás beállításai

3.3. Adatbázis

3.3.1. Technológiák

Az alkalmazáshoz szükséges telepítünk egy *MySQL Community Server*-re, ajánlott a 8.0.25-ös verzió.⁴ Az autentikáció és autorizáció megvalósításához a Microsoft által készített *Microsoft.AspNetCore.Identity.EntityFrameworkCore* nyílt forráskódú programkönyvtárat használja rendszer. A programkönyvtár tartalmaz meglévő adatbázis táblákat, melyeknek a tartalma és működése elolvasható a Microsoft hivatalos honlapján [7]. A programkönyvtár gondoskodik a jelszavak biztonságos tárolásáról, melyet időfüggő szózással és a jelszó hashelésével valósít meg.

Az adatbázis *code first* módszerrel van megvalósítva, tehát nem az adatbázis szerveren *SQL* kódot futattva hozzuk létre az adatbázis táblákat, hanem modell osztályokkal definiáljuk az adatbázis táblákat [8]. Ezen modelleket az *ASS.DAL.Models* névtérben tároljuk.

Az adatelérést az *Entity Framework Core ORM* keretrendszer biztosítja [9]. Az objektum-relációs leképezés (angolul *Object-Relational Mapping*), egy technika az adatok konvertálására nem kompatibilis típusos rendszerek és objektumorientált programozási nyelvek között. Így az alkalmazás forráskódjában nincsenek beégetett *SQL* kódok. Ezek helyett a *CRUD* (Create,Read,Update,Delete műveleteknek a rövidítése) műveleteket a *.NET* nyújtotta és az *Entity Framework Core* által is

⁴Az alkalmazás működik régebbi verzióval is. Viszont az alkalmazás nincs felkészítve az esetleges verziók közötti különbségekre.

támogatott *LINQ* (Language Integrated Queries) metódushívásokkal valósul meg [10]. Továbbá a keretrendszer védelmet biztosít az *SQL Injection* támadások ellen [11], ugyanis a műveletek a *C#* és *LINQ* metódusokból kerülnek előállításra paraméterezetten.

Az adatbázis elérését az alkalmazás konfigurációs fájljában (*appsettings.json*) tudjuk megadni illetve módosítani.

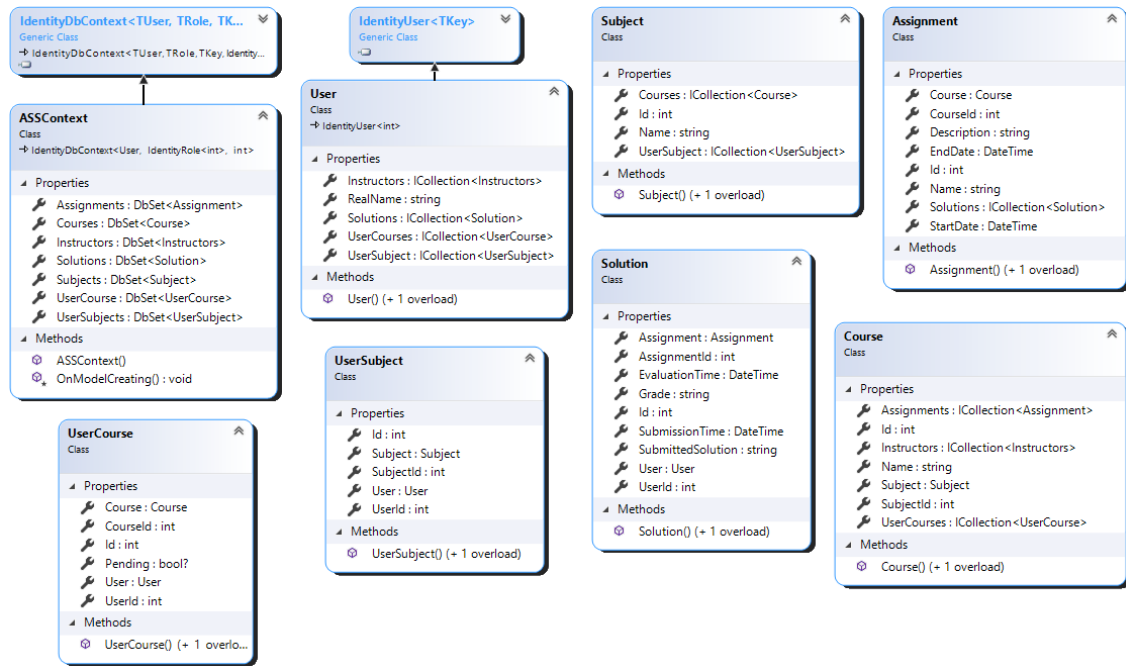
```
1 ...  
2 "ConnectionStrings": {  
3   "DefaultConnection": "server=localhost;database=ASS;uid=username;  
   ↪ password=fooBarraBoof"  
4 },  
5 ...
```

3.2. forráskód. Adatbázis elérése

3.3.2. Adatbázis *code first* objektumai

A *C#* objektumok amelyekből az adatbázis képződik a 3.2 ábrán tekinthetjük meg. Maga az adatbázis az *ASSContext* osztályból képződik. Minden egyes *DbSet<T>*⁵ típusú tulajdonság (angolul Property), egy adatbázis táblát jelent. Az osztály *OnModelCreating* metódusában számos adatbázisra vonatkozó beállítást van lehetőségünk beállítani (pl.: táblák elsődleges kulcsai, külső kulcsai). A *DbInitializer* osztály egy nyilvános *Initialize* metódussal rendelkezik, mely létrehozza az adatbázis szerveren az adatbázist, ha még nem létezik, illetve a szükséges konstans adatokkal tölti fel az adatbázist (szerepkörök felvétele és rendszergazdai felhasználó létrehozása).

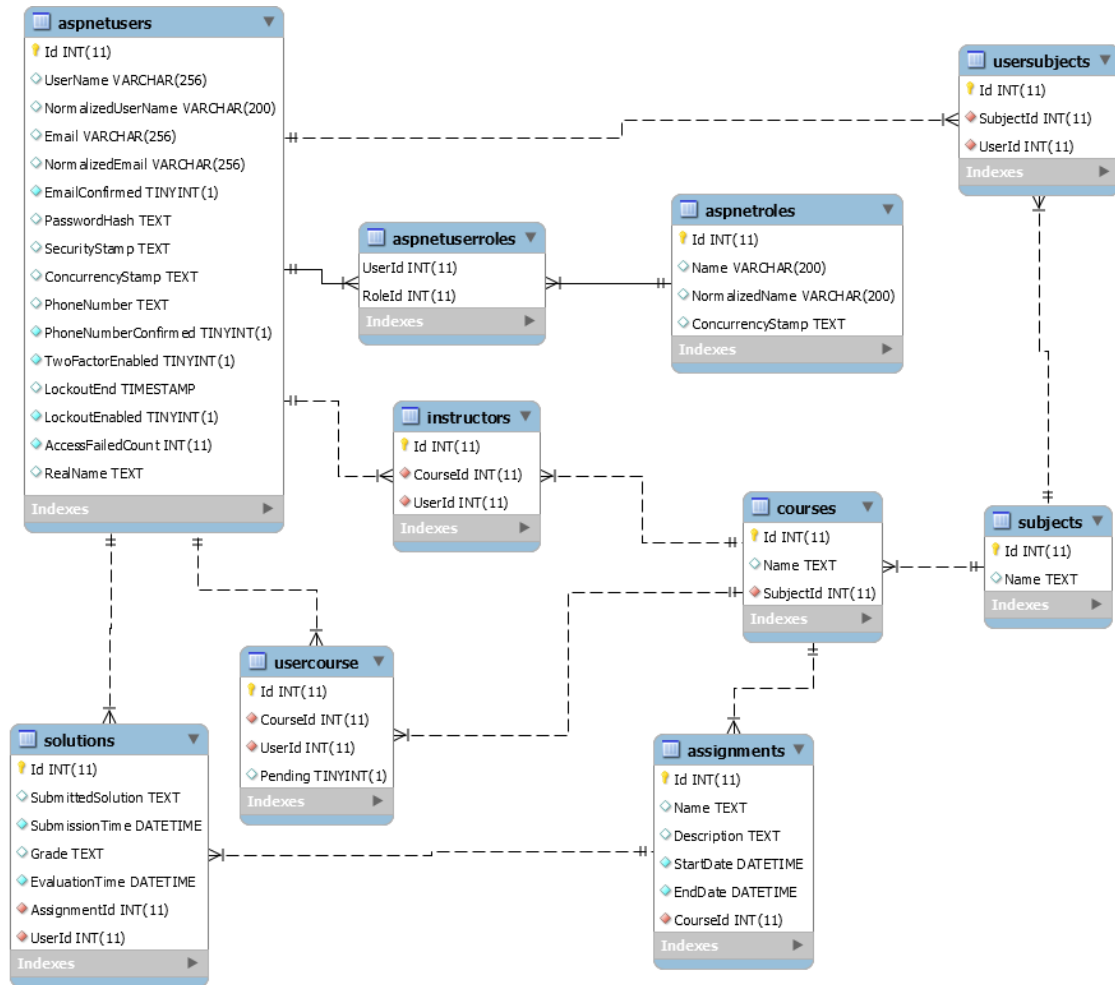
⁵Ahol a *T* egy generikus típusparaméter.



3.2. ábra. Az adatbázist leképző objektumok

3.3.3. Az adatbázis táblái

Az alkalmazás adatbázis diagramját a 3.3 ábrán tekinthetjük meg. A *Microsoft.AspNetCore.Identity.EntityFrameworkCore* keretrendszer által létrehozott táblákból csak azon a táblák és mezők kerülnek részletezésre, melyeket a rendszer aktívan használ. A táblák, amik nem kerülnek részletezésre a Microsoft hivatalos honlapján meg lehet tekinteni [7].



3.3. ábra. Az adatbázis táblái

aspnetusers

A *Microsoft.AspNetCore.Identity.EntityFrameworkCore* programkönyvtár által automatikusan létrehozott tábla. A felhasználók adatait tárolja.

Mező neve	Típus	Leírás
Id	egész	Elsődleges kulcs
UserName	szöveg	Felhasználónév (neptun kód)
Email	szöveg	Felhasználó e-mail címe
RealName	szöveg	Felhasználó neve
PasswordHash	szöveg	Felhasználó hashelt jelszava

3.1. táblázat. Adatbázis: felhasználók táblája

aspnetroles

A *Microsoft.AspNetCore.Identity.EntityFrameworkCore* programkönyvtár által automatikusan létrehozott tábla. A rendszerben használt szerepköröket tárolja.

Mező neve	Típus	Leírás
Id	egész	Elsődleges kulcs
Name	szöveg	Szerepkör megnevezése

3.2. táblázat. Adatbázis: szerepkörök táblája

aspnetuserroles

A *Microsoft.AspNetCore.Identity.EntityFrameworkCore* programkönyvtár által automatikusan létrehozott tábla. Egy kapcsolótábla, mely tárolja a felhasználóhoz rendelt szerepköröket.

Mező neve	Típus	Leírás
UserId	egész	Elsődleges kulcs
RoleId	egész	Elsődleges kulcs

3.3. táblázat. Adatbázis: felhasználók és szerepkörök kapcsolótáblája

assignments

Az *assignments* tábla a csoportokhoz létrehozott beadandó feladatok adatainak tárolására szolgál.

Mező neve	Típus	Leírás
Id	egész	Elsődleges kulcs
Name	szöveg	A feladat neve
Description	szöveg	A feladat leírása
StartDate	dátum	A feladat elérésének dátuma
EndDate	dátum	A feladat határidejének dátuma
CourseId	egész	Arra vonatkozó kulcs, hogy a feladat melyik csoporthoz tartozik

3.4. táblázat. Adatbázis: feladatok táblája

courses

A *courses* tábla a tantárgyakhoz létrehozott csoportok adatait tárolja.

Mező neve	Típus	Leírás
Id	egész	Elsődleges kulcs
Name	szöveg	A csoport neve
SubjectId	egész	Arra vonatkozó kulcs, hogy a csoport melyik tantárgyhoz tartozik

3.5. táblázat. Adatbázis: csoportok táblája

instructors

A *instructors* tábla egy kapcsolótábla, melyben a *Gyakorlatvezető* szerepkörrel rendelkező felhasználókat kapcsoljuk a hozzájuk tartozó csoportokhoz.

Mező neve	Típus	Leírás
Id	egész	Elsődleges kulcs
CourseId	egész	A csoportra vonatkozó kulcs
UserId	egész	A felhasználóra vonatkozó kulcs

3.6. táblázat. Adatbázis: gyakorlatvezetők táblája

solutions

A *solutions* tábla a feladatokra beadott megoldásokat tárolja.

Mező neve	Típus	Leírás
Id	egész	Elsődleges kulcs
SubmittedSolution	szöveg	A feladatra beadott megoldás
SubmissionTime	dátum	A megoldás beküldésének az időpontja
Grade	szöveg	A feladatra adott értékelése
EvaluationTime	dátum	A feladat értékelésének időpontja
AssignmentId	egész	Arra vonatkozó kulcs, hogy a megoldás melyik feladathoz tartozik
UserId	egész	Arra vonatkozó kulcs, hogy melyik felhasználó adta be a megoldást

3.7. táblázat. Adatbázis: megoldások táblája

subjects

A *subjects* tábla a rendszerben létrehozott tantárgyak adatait tárolja.

Mező neve	Típus	Leírás
Id	egész	Elsődleges kulcs
Name	szöveg	A tantárgy neve

3.8. táblázat. Adatbázis: megoldások táblája

usercourse

A *usercourse* tábla egy kapcsoló tábla, melyben a hallgatókat és a csoportok összerendelése valósul meg.

Mező neve	Típus	Leírás
Id	egész	Elsődleges kulcs
CourseId	egész	A csoportra vonatkozó kulcs
UserId	egész	A felhasználóra vonatkozó kulcs
Pending	igaz/hamis	Annak az értéke, hogy a felhasználónak a jelentkezése elfogadásra, vagy elutasításra került

3.9. táblázat. Adatbázis: hallgatók és csoportok kapcsolótáblája

usersubjects

A *usersubjects* tábla egy kapcsoló tábla, melyben a tárgyfelelősök és a tantárgyak összerendelése valósul meg.

Mező neve	Típus	Leírás
Id	egész	Elsődleges kulcs
SubjectId	egész	A tantárgyra vonatkozó kulcs
UserId	egész	A felhasználóra vonatkozó kulcs

3.10. táblázat. Adatbázis: tárgyfelelősök és tantárgyak kapcsolótáblája

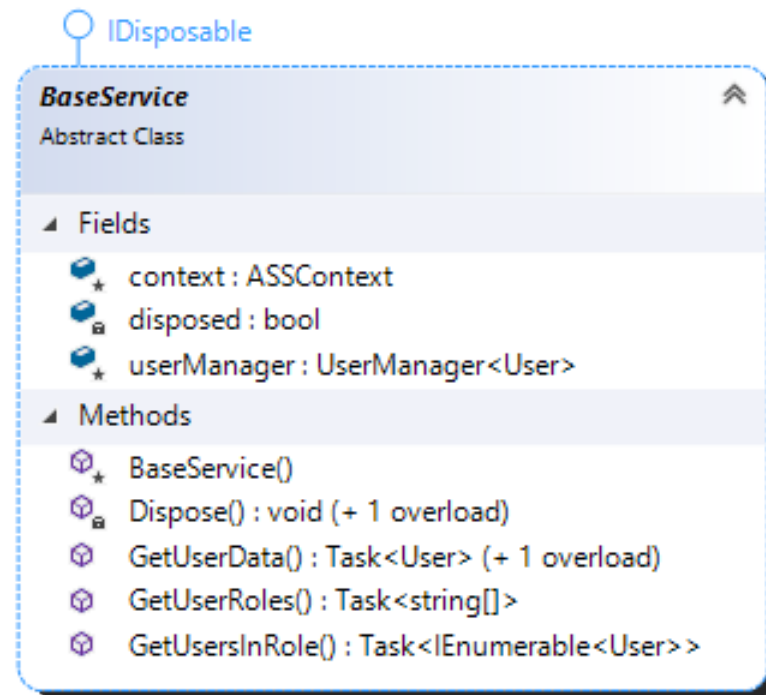
3.4. Model réteg

3.4.1. Üzleti logika

Az üzleti logikát megvalósító objektumokat az *ASS.BLL.Interfaces* és az *ASS.BLL.Services* névtérben tároljuk. Az üzleti logikát szerepkörökre bontva valósítjuk meg. Minden szerepkörhöz tartozik egy *interface*, mely leírja a szerepkörhöz tartozó funkciók metódusait, valamint egy osztály, ami implementálja az adott *interface*-t. Az *interface*-ket megvalósító osztályok a *BaseService* osztályból származnak le⁶ (3.4 ábra), melyben azok a funkcionalitások kerültek implementálásra, amiket minden egyes szerepkörhöz tartozó *service* osztálynak meg kell valósítania.

⁶Kivéve a *LoginService* osztályt.

Ezeket a *service* osztályokat az alkalmazás *IoC* (*Inversion of Control*) konténerébe [12] regisztráljuk. Ezáltal a vezérlő osztályok rendelkeznek a hozzájuk tartozó *service* osztály egy példányával, melyet konstruktoron keresztüli függőségi befecskendezéssel kapnak meg. A vezérlő osztályok ezen *service* osztályok metódusainak segítségével dolgozzák fel a kliens kéréseit és állítják elő a megfelelő válaszokat.



3.4. ábra. Service osztályok őse

BaseService osztály

GetUserData(ClaimsPrincipal): paraméterül a bejelentkezett felhasználót⁷ kapja a metódus, majd eredményül a paraméterül kapott felhasználónak az adataival tér vissza.

GetUserData(int): az előbbi metódus túlterhelése, itt a keresendő felhasználó egyedi kulcsát kapja a metódus paraméterül.

GetUserRoles(ClaimsPrincipal): paraméterül a bejelentkezett felhasználót kapja, majd a felhasználó szerepköreivel tér vissza.

⁷Ezen a linken elolvashatjuk a *ClaimsPrincipal* osztály dokumentációját.

GetUsersInRole(Role): paraméterül egy *Role enum* értéket kap, majd a paraméterül kapott szerepkörrel rendelkező felhasználókkal tér vissza.

Dispose(): az *IDisposable interface* metódusa, mely gondoskodik a külső erőforrások felszabadításáról.

LoginService osztály

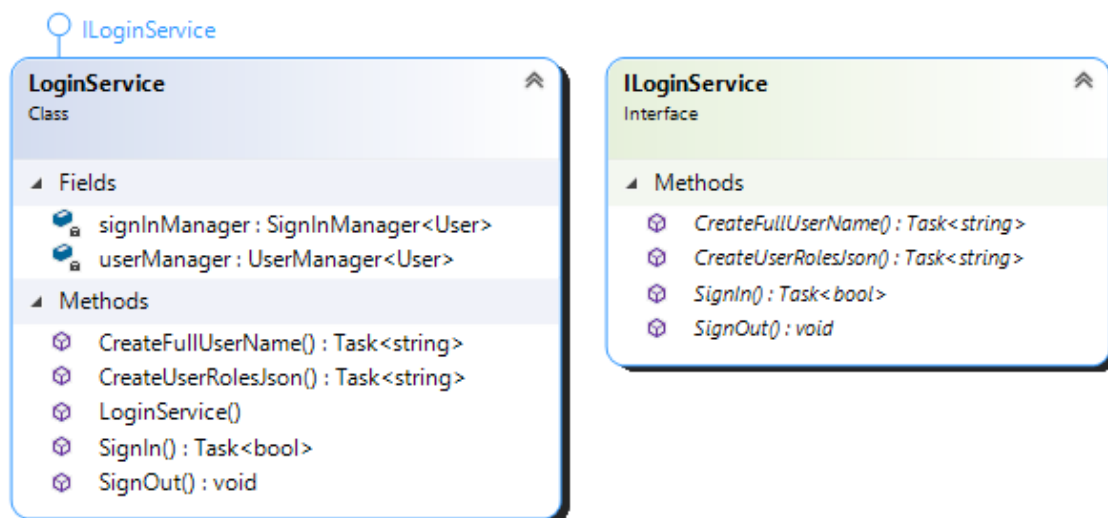
Ez az osztály felelős az alkalmazásba való bejelentkeztetésért, kijelentkeztetésért, valamint a bejelentkeztetett felhasználó fontos adatainak (felhasználónév, név, szerepkörök) lekérdezéséért, hogy a felhasználó munkamenetében (angolul *session*) tudja tárolni a rendszer.

CreateFullName(ClaimsPrincipal): paraméterül a bejelentkezett felhasználót kapja, majd a felhasználó polgári nevéből és felhasználónevéből képzett *string*-el tér vissza.

CreateUserRolesJson(ClaimsPrincipal): paraméterül a bejelentkezett felhasználót kapja, majd a felhasználó szerepköreivel tér vissza.

SignIn(string,string,bool,bool): paraméterül a bejelentkezési adatokat kapja, majd egy igaz/hamis értékkel tér vissza, ami a bejelentkezés sikerességét jelzi.

SignOut(): kijelentkezteti a felhasználót az alkalmazásból.



3.5. ábra. LoginService osztály és interface

AdminService osztály

CreateSubject(string[],string): paraméterül *tárgyfelelősi* szerepkörrel rendelkező felhasználók felhasználóneveit és a létrehozandó tantárgy nevét kapja. A metódus leellenőrzi, hogy a paraméterül kapott tantárgy név létezik-e már a rendszerben, ha nem, akkor a rendszer létrehozza a tantárgyat, egyébként kivétel váltódik ki.

GetSubjects(): a metódus visszatérési értéke a rendszerben létrehozott összes tantárgy.

UpdateSubject(int,string,string[]): paraméterül egy tantárgy egyedi azonosítóját, tantárgy nevet és *tárgyfelelősi* szerepkörrel rendelkező felhasználók felhasználóneveit kapja. A metódus módosítja a kapott paraméterek alapján a tantárgy adatait, ha az új tantárgynév még nem foglalt, egyébként kivétel váltódik ki.

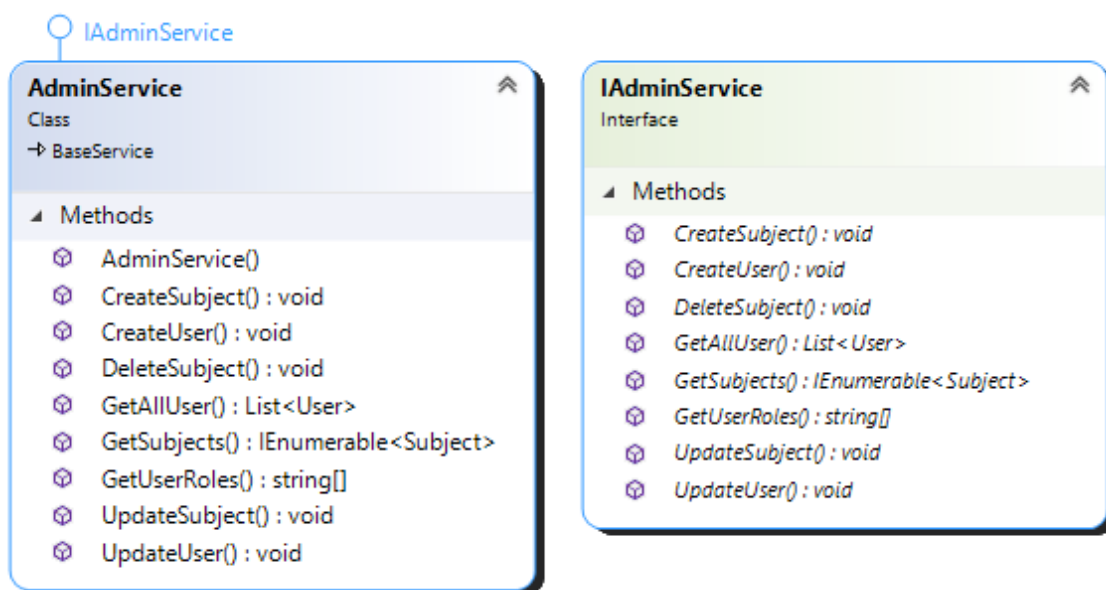
DeleteSubject(int): paraméterül egy tantárgy egyedi azonosítóját kapja, majd a megfelelő tantárgyat a metódus törli a rendszerből.

GetAllUser(): a metódus listázza a rendszerben tárolt összes felhasználót.

GetUserRoles(int): paraméterül egy felhasználó egyedi azonosítóját kapja, majd a megfelelő felhasználó szerepköreivel tér vissza.

CreateUser(string,string,string,string,string[]): a metódus egy új felhasználót hoz létre a rendszerben a paraméterül kapott adatok alapján.

UpdateUser(int,string,string,string,string[]): paraméterül egy felhasználó adatait kapja (egyedi azonosító, felhasználónév, polgári név, e-mail cím, szerepkörök). A metódus a megfelelő felhasználó adatait módosítja.



3.6. ábra. AdminService osztály és interface

TeacherService osztály

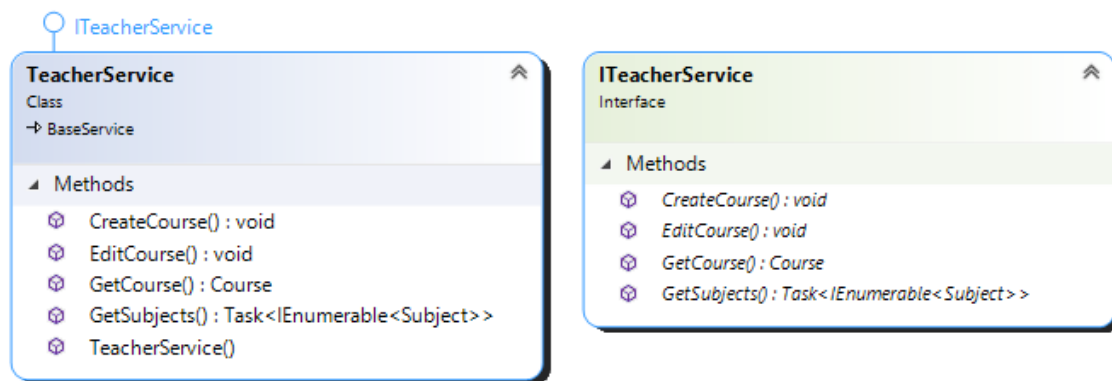
Ez az osztály implementálja a *tárgyfelelősi* szerepkörhöz tartozó funkciókat.

CreateCourse(string[],int,string): paraméterül felhasználónevek tömbjét, egy tantárgynak az egyedi azonosítóját illetve egy csoportnevet kap. A metódus létrehozza a paraméterül kapott tantárgyhoz az új csoportot, amennyiben ez lehetséges. Ha sikeres volt a csoport létrehozása, akkor a paraméterül kapott felhasználókat hozzárendeli a csoporthoz.

GetSubjects(ClaimsPrincipal): paraméterül kap egy bejelentkezett felhasználót, majd a hozzárendelt tantárgyakkal tér vissza egy listában.

GetCourse(int): paraméterül egy csoportnak az egyedi azonosítóját kapja, majd visszatér ezen csoport adataival.

EditCourse(int,string,string[]): paraméterül egy csoportnak az egyedi azonosítóját, a csoport nevét, és gyakorlatvezetők felhasználóneveit kapja. A metódus a paraméterül kapott adatokkal módosítja a megfelelő csoportot.



3.7. ábra. TeacherService osztály és interface

InstructorService osztály

GetPendingList(ClaimsPrincipal): paraméterül a bejelentkezett felhasználót kapja, visszatérési értéke a felhasználóhoz tartozó csoportba jelentkezett hallgatók listája.

ProcessPendingStatus(int,bool): paraméterül a jelentkezéseket tároló kapcsolótábla (*UserCourses*) egyedi azonosítóját és egy igaz/hamis érték kap. A metódus az igaz/hamis érték alapján frissíti a megfelelő jelentkezési státuszt. Az igaz érték a jelentkezés elfogadását jelenti, a hamis pedig az elutasítást.

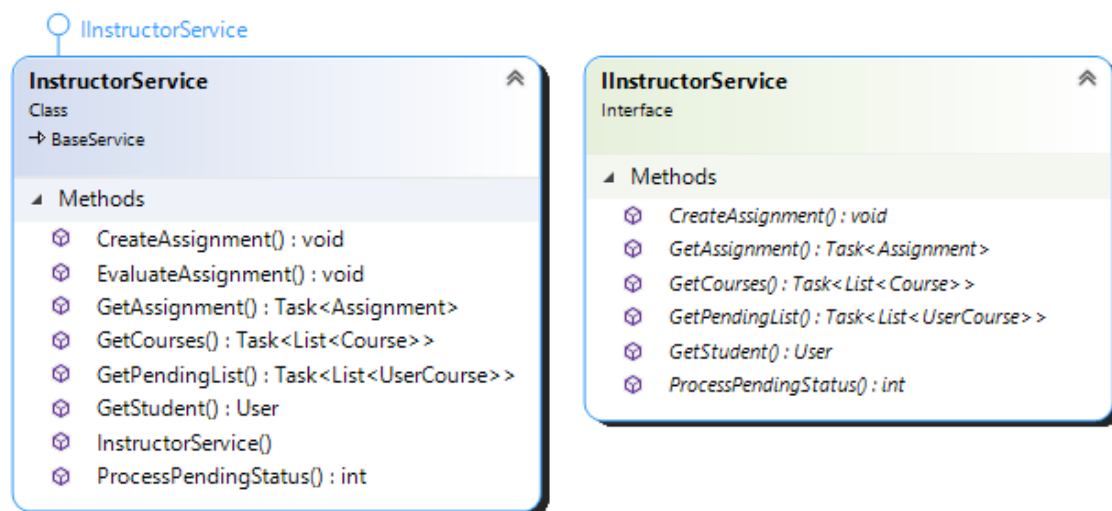
GetCourses(ClaimsPrincipal): paraméterül a bejelentkezett felhasználót kapja, eredményül pedig a felhasználóhoz tartozó csoportokkal tér vissza.

CreateAssignment(string,string,DateTime,DateTime,int[]): paraméterül egy feladatnak az adatait kapja. A metódus leellenőrzi, hogy a feladat elérésének a dátuma korábban van-e mint a beadási határidő, ha igen akkor elmenti a feladatot, ha nem teljesül a feltétel, akkor kivétel váltódik ki.

GetAssignment(int,int,ClaimsPrincipal): paraméterül egy csoport és egy feladat egyedi azonosítóját valamint a bejelentkezett felhasználót kapja. A metódus a paraméterül kapott feladat adataival tér vissza, amennyiben a felhasználó gyakorlatvezetője a paraméterül kapott csoportnak, egyébként kivétel váltódik ki.

GetStudent(int): paraméterül egy hallgató egyedi azonosítóját kapja, visszatérési értéke a megfelelő felhasználó adatai.

EvaluateAssignment(int,string,DateTime,ClaimsPrincipal): paraméterül egy feladat egyedi azonosítóját, a feladatra beadott megoldás értékelését, az értékelésnek az időpontját és a bejelentkezett felhasználót kapja. A metódus ellenőrzi, hogy az a felhasználó, aki az értékelést végrehajtja, a kiiírt feladat csoportjának a gyakorlatvezetője-e. Ha igen, elmentődik az értékelés, egyébként kivétel váltódik ki.



3.8. ábra. InstructorService osztály és interface

StudentService osztály

Ez az osztály implementálja a *hallgatói* szerepkörnek a funkcióit.

GetCourses(ClaimsPrincipal): paraméterül a bejelentkezett felhasználót kapja meg, majd a felhasználóhoz tartozó csoportokkal tér vissza.

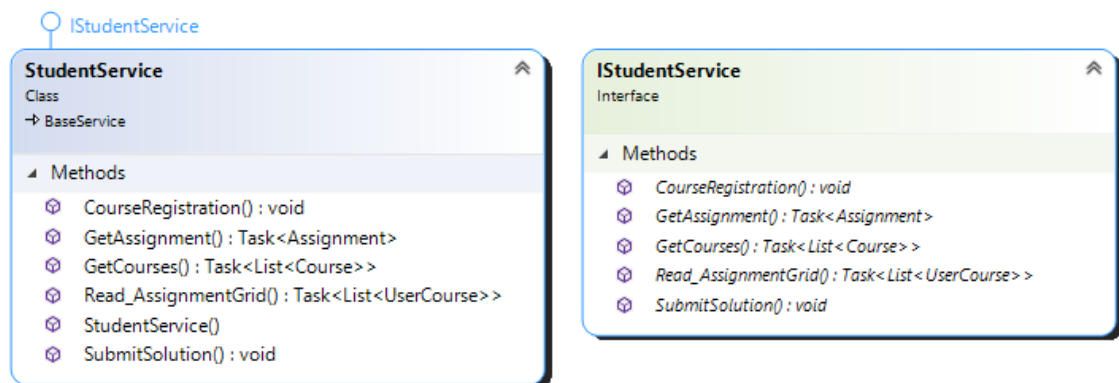
CourseRegistration(int[],ClaimsPrincipal): paraméterül csoportok egyedi azonosítójának a tömbjét és a bejelentkezett felhasználót kapja, majd a felhasználót felveszi a paraméterül kapott csoportokba⁸.

Read_AssignmentGrid(ClaimsPrincipal): paraméterül a bejelentkezett felhasználót kapja, majd a hozzá tartozó csoportok listájával tér vissza.

⁸Ezen a ponton még a hallgató csak jelentkezést adott le az adott csoport(ok)ba.

GetAssignment(int,ClaimsPrincipal): paraméterül egy feladatnak az egyedi azonosítóját és a bejelentkezett felhasználót kapja, majd visszatér a paraméterül kapott feladat adataival, ha a felhasználó tagja annak a csoportnak, amelyiket lekérdeztük.

SubmitSolution(int,ClaimsPrincipal,string,DateTime): paraméterül egy feladat egyedi azonosítóját, a bejelentkezett felhasználót, a feladat megoldását és a beadás időpontját kapja. A metódus ellenőrzi, hogy a beadás időpontja korábbi-e mint a feladat beadási határideje. Amennyiben helyes a beadási idő, elmenti a beadott megoldást, ellenkező esetben kivétel váltódik ki.



3.9. ábra. StudentService osztály és interface

3.4.2. Adatok megjelenítésére szolgáló modellek

Az adatok megjelenítésére adatátviteli objektumok⁹ (angolul *Data transfer object DTO*) kerülnek definiálásra. Feladatuk a folyamatok között közvetíteni a szükséges adatokat. Jelen esetben a vezérlő meghívja az üzleti logika megfelelő metódusát a kérés során, majd egy ilyen *DTO* osztályba csomagolja az üzleti logika által visszaadott *entitás*¹⁰ osztályokban tárolt adatokat és ezt az objektumot kapja meg a nézet, hogy megtudja jeleníteni a kliens számára az adatokat. Ezeket az osztályokat a *ASS.WEB.Models.DTOs* névtérben tároljuk.

⁹https://en.wikipedia.org/wiki/Data_transfer_object

¹⁰Azokat az osztályokat nevezzük *entitás* osztálynak, melyekből az adatbázis táblák képződnek le.

3.4.3. Adatok bevitelére szolgáló modellek

Az adatok bevitele nézetmodellek (angolul *viewmodel*) segítségével valósul meg. A nézetmodelleket a *ASS.WEB.Models.ViewModels* névtérben tároljuk. A nézetmodellek tulajdonságaira (angolul *property*) megszabhatunk (egy vagy több) attribútumot, melyet a *System.ComponentModel.DataAnnotations* névtérből érünk el. Az attribútumok használatával egyszerűen tudjuk validálni nézetmodelljeinket, vagy a validációs hibaüzenetek testreszabni. Ezt a *.NET* keretrendszer biztosítja. Ugyanis fontos a felhasználó által kitöltött űrlapok ellenőrzése, hogy hibás adatok ne kerülhessenek a rendszerbe. A nézetmodelleket a rendszer az űrlapoknál használja fel, tehát a nézetmodellek egy-egy kitöltött űrlap adatait képes tárolni. Az attribútumok leírását a Microsoft hivatalos honlapján részletesen el lehet olvasni [13].

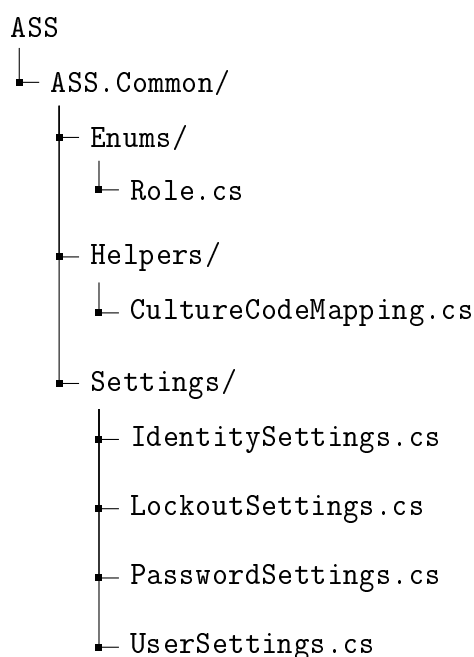
```
1 using System.ComponentModel.DataAnnotations;
2
3 namespace ASS.WEB.Models.ViewModels
4 {
5     public class LoginViewModel
6     {
7         [Required(ErrorMessageResourceType = typeof(Resources.Models.
8             ViewModels.LoginViewModel), ErrorMessageResourceName = "
9             UsernameRequired")]
10        public string Username { get; set; }
11
12        [Required(ErrorMessageResourceType = typeof(Resources.Models.
13            ViewModels.LoginViewModel), ErrorMessageResourceName = "
14            PasswordRequired")]
15        public string Password { get; set; }
16
17        [StringLength(maximumLength: 30, ErrorMessageResourceType =
18            typeof(Resources.Models.ViewModels.LoginViewModel),
19            ErrorMessageResourceName = "PasswordLength")]
20        public string PasswordConfirm { get; set; }
21
22        [Display(ResourceType = typeof(Resources.Models.ViewModels.
23            LoginViewModel), Name = "Username")]
24        public string UsernameLabel { get; set; }
25
26        [Display(ResourceType = typeof(Resources.Models.ViewModels.
27            LoginViewModel), Name = "Password")]
28        public string PasswordLabel { get; set; }
29
30        [Display(ResourceType = typeof(Resources.Models.ViewModels.
31            LoginViewModel), Name = "PasswordConfirm")]
32        public string PasswordConfirmLabel { get; set; }
33    }
34 }
```

```
15     public string Password { get; set; }
16 }
17 }
```

3.3. forráskód. Példa az attribútumok használatára

3.4.4. Egyéb segédosztályok

Az alkalmazásban definiálásra kerülnek egyéb segédosztályok és egy felsorolási típus (angolul *enum*), melyeket az alább ábrán látható helyen találunk.



Role.cs

A *Role* felsorolási típus segítségével definiáljuk a rendszerben tárolt szerepköröket. Ugyanis így a forráskódban nem szükséges beégetett szövegeket használnunk a szerepkörökre¹¹.

```
1 namespace ASS.Common.Enums
2 {
3     public enum Role
4     {
5         Admin,
```

¹¹Ez alól kivétel az *Authorize* attribútum, mivel az attribútumokban a keretrendszer csak konstans értékeket enged használni.

```
6     Teacher ,
7     Instructor ,
8     Student
9 }
10 }
```

3.4. forráskód. Szerepkörök felsorolási típusa

Továbbá a *Role enum* segítségével egy ciklussal könnyedén tudjuk az adatbázisba perzisztálni az *enum* értékeit.

```
1 ...
2 foreach (Role item in Enum.GetValues(typeof(Role)))
3 {
4     context.Roles.Add(new IdentityRole<int>() { Name = item.ToString
5         (), NormalizedName = item.ToString() });
6 }
7 ...
```

3.5. forráskód. Szerepkörök tárolása az adatbázisba (DbInitializer.cs)

CultureCodeMapping.cs

A *CultureCodeMapping* egy statikus segédosztály, amely egy nyelvi kódból a nyelvet adja vissza. Ezt a segédosztályt a lokalizációnál használjuk, hogy a felületen ne a nyelvi kód (pl.: *hu-HU*) jelenjen meg, hanem az adott nyelv neve.

```
1 namespace ASS.Common.Helpers
2 {
3     public static class CultureCodeMapping
4     {
5         public static string CultureCodeToCountryName(string
6             cultureCode)
7         {
8             switch (cultureCode)
9             {
10                 case "hu-HU":
11                     return "Magyar";
12                 case "en-US":
13                     return "English";
14                 default:
```

```
14         return "Ismeretlen";
15     }
16 }
17 }
18 }
```

3.6. forráskód. CultureCodeMapping osztály

Settings osztályok

A *Microsoft.AspNetCore.Identity* [14] lehetővé tesz különböző konfigurációk beállítását a felhasználói fiókokra. Például jelszóra vonatkozó konfigurációkat (minimum hossz, kötelező számot tartalmaznia stb), felhasználóra vonatkozó megszorításokat (minden felhasználó egyedi e-mail címmel rendelkezzen) és hibás bejelentkezés esetén konfigurálhatjuk a felhasználó kizárását az alkalmazásból (a kizárás időtartama). Ezen konfigurációk könnyű állíthatósága érdekében a konfigurációs értékek az alkalmazás konfigurációs fájljába (*appsettings.json*) kiszervezésre kerültek.

```
1 ...
2 "User": {
3     "RequireUniqueEmail": false
4 },
5 "Password": {
6     "RequiredLength": 1,
7     "RequireLowercase": false,
8     "RequireUppercase": false,
9     "RequireDigit": false,
10    "RequireNonAlphanumeric": false,
11    "RequiredUniqueChars" : 1
12 },
13 "Lockout": {
14     "AllowedForNewUsers": false,
15     "DefaultLockoutTimeSpanInMins": 30,
16     "MaxFailedAccessAttempts": 10
17 }
18 ...
```

3.7. forráskód. Felhasználói fiók konfigurációs beállításai

Ugyanis így a kód módosítása nélkül tudjuk változtatni ezen konfigurációs beállításokat és nem kell a változások után újra fordítani az alkalmazást, hanem elengedő csak újraindítani, hiszen a forráskód nem változott. Ezeket a beállításokat az alkalmazás indításakor kerül kiolvasásra az *appsettings.json* fájlból, majd szerializálja a megfelelő objektumba (*IdentitySettings* osztály) az adatokat. Továbbá egyszerre több fajta konfigurációs beállítást megadhatunk, annak függvényében, hogy az alkalmazás milyen módban fut¹² (3.8 forráskód).

```
1 "Mode": "Development",  
2 ...
```

3.8. forráskód. Alkalmazás futási módja

Ennek használatával elég csak a módot változtatni, nem kell az összes *IdentitySettings*-hez tartozó értéket módosítani.

3.5. Vezérlő réteg

3.5.1. *Home* vezérlő

A *Home* vezérlő (3.10 ábra) az alkalmazás alap funkcionálisait implementálja, mint például a bejelentkezés vagy a kijelentkezés. Ez a vezérlő mindenki számára elérhető, nincs szerepkörhöz kötve.

[HttpGet] Index(): visszatér az alkalmazás főoldalával.

[HttpPost] Login(LoginViewModel): paraméterül a bejelentkezési adatokat kapja (felhasználónév, jelszó), majd a kapott adatokkal megpróbálja bejelentkeztetni az alkalmazásba a felhasználót, sikeres bejelentkezés esetén a szerepkörének megfelelő kezdőoldalra irányítja, egyébként jelzi a hibát a felhasználónak.

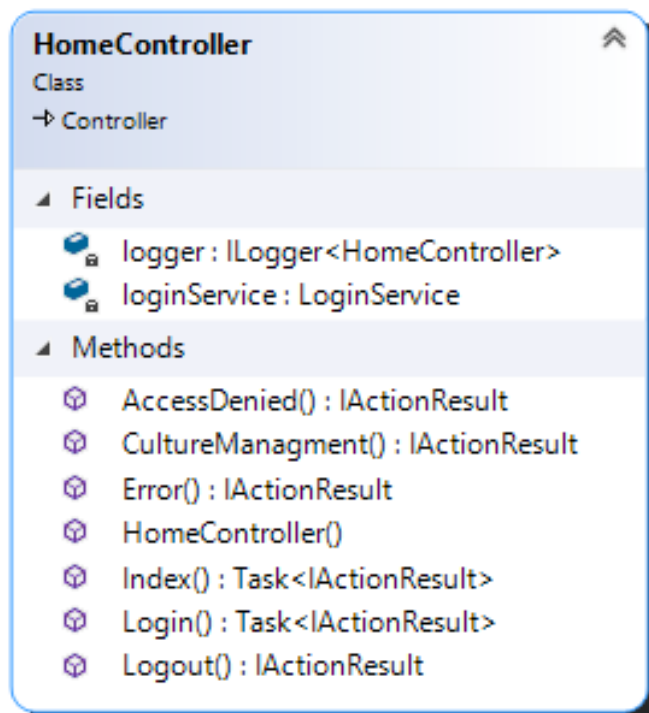
[HttpPost] CultureManagment(string): paraméterül egy nyelvi kódot kap (pl.: hu-HU), majd *sütibe* menti a kapott nyelvi kódot, és visszatér az alkalmazás főoldalával.

¹²Ezt a beállítást is az *appsettings.json* fájlban állíthatjuk

[HttpGet] Error(string): paraméterül a keletkezett hibának az azonosítóját kapja, majd visszatér az alkalmazás hibaoldallával.

[HttpGet] AccessDenied(): az alkalmazás *jogosulatlan kérés* oldalával tér vissza.

[HttpGet] Logout(): kijelentkezteti a felhasználót, majd visszatér az alkalmazás főoldalával.



3.10. ábra. *Home* vezérlő

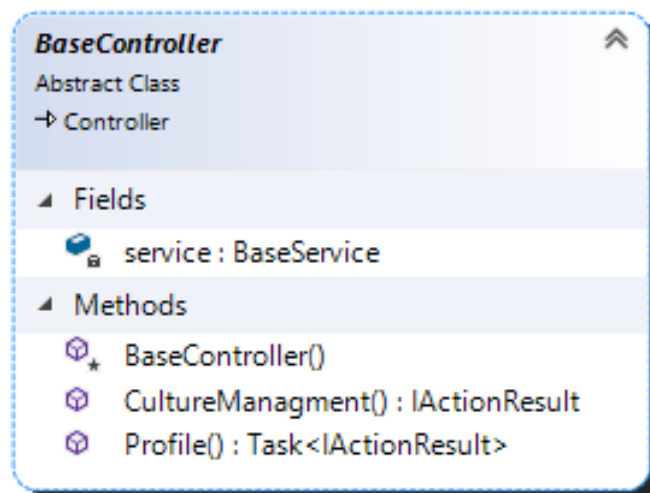
3.5.2. *Base* vezérlő

A vezérlők őse¹³ (3.11 ábra), mely azokat a funkcionalitásokat valósítja meg, amellyel minden szerepkörhöz tartozó vezérlőnek tudnia kell.

[HttpGet] Profile(): lekéri a bejelentkezett felhasználó adatait, majd visszatér személyes adatokat megjelenítő oldallal.

[HttpPost] CultureManagment(string): paraméterül egy nyelvi kódot kap (pl.: hu-HU), majd *sütibe* menti a kapott nyelvi kódot, és visszatér a személyes adatokat megjelenítő oldallal.

¹³Leszámítva a *Home* vezérlőt.

3.11. ábra. *Base* vezérlő

3.5.3. *Admin* vezérlő

Az *Admin* vezérlőt csak rendszergazdai szerepkörrel rendelkező felhasználók érhetik el.

[HttpGet] Index(): a rendszergazdai szerepkör főoldalával tér vissza.

[HttpGet] GetTeachers(): lekéri a rendszerben szereplő tárgyfelelősi szerepkörrel rendelkező felhasználókat és ennek eredményével tér vissza.

[HttpGet] CreateSubject(): a tantárgyak létrehozására alkalmas felületet adja vissza.

[HttpPost] CreateSubject(CreateSubjectViewModel): a tantárgy létrehozásának fogadása, ellenőrzi az adatok helyességét. Sikeres adatrögzítés után átirányítás történik a szerepkör főoldalára, egyébként jelzi a hibát a felhasználónak.

[HttpGet] GetSubjects(): lekéri a rendszerben szereplő tantárgyakat és ennek eredményével tér vissza.

[HttpPost] DeleteSubject(string): paraméterül egy *Json string*-et¹⁴ kap melyben a törölni kívánt tantárgy adatai vannak tárolva. Ezt a tárgyat törli a rendszerből.

¹⁴ Javascript Object Notation

[HttpPost] UpdateSubject(string): paraméterül egy *Json string*-et kap melyben a módosítani kívánt tantárgy adatai vannak, sikertelen módosítás esetén átirányítás történik a hibaoldalra.

[HttpGet] Read_UserGrid(): lekéri a rendszerben tárolt felhasználókat és ennek eredményével tér vissza.

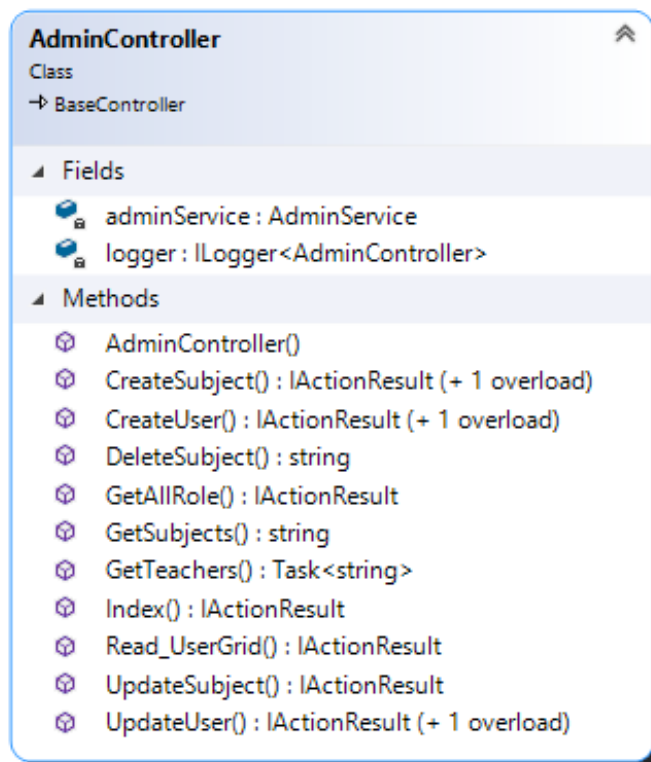
[HttpGet] CreateUser(): a felhasználók létrehozására alkalmas felületet adja vissza.

[HttpPost] CreateUser(CreateUserViewModel): a felhasználó létrehozásának fogadása, ellenőrzi az adatok helyességét. Sikeres rögzítés esetén átirányítás történik a szerepkör főoldalára, egyébként jelzi a hibát a felhasználónak.

[HttpGet] GetAllRole(): lekéri a rendszerben található szerepköröket és ennek eredményével tér vissza.

[HttpGet] UpdateUser(int): paraméterül egy felhasználó egyedi azonosítóját kapja, majd lekéri a paraméterül kapott felhasználó adatait. Ezután a felhasználó módosítására alkalmas felületet adja vissza, a szükséges adatokkal (a felhasználó adatai).

[HttpPost] UpdateUser(UpdateUserViewModel): a felhasználó módosításának fogadása, ellenőrzi az adatok helyességét. Sikeres rögzítés esetén átirányítás történik a szerepkör főoldalára, egyébként jelzi a hibát a felhasználónak.

3.12. ábra. *Admin* vezérlő

3.5.4. *Instructor* vezérlő

Az *Instructor* vezérlőt csak gyakorlatvezetői szerepkörrel rendelkező felhasználók érhetik el.

[HttpGet] Index(): a gyakorlatvezetői szerepkör főoldalával tér vissza.

[HttpGet] CreateAssignment(): feladat létrehozására alkalmas felületet adja vissza.

[HttpPost] CreateAssignment(CreateAssignmentViewModel): a feladat létrehozásának fogadása és ellenőrzi az adatok helyességét. Sikeres rögzítés esetén átirányítás történik a szerepkör főoldalára, egyébként jelzi a hibát a felhasználónak.

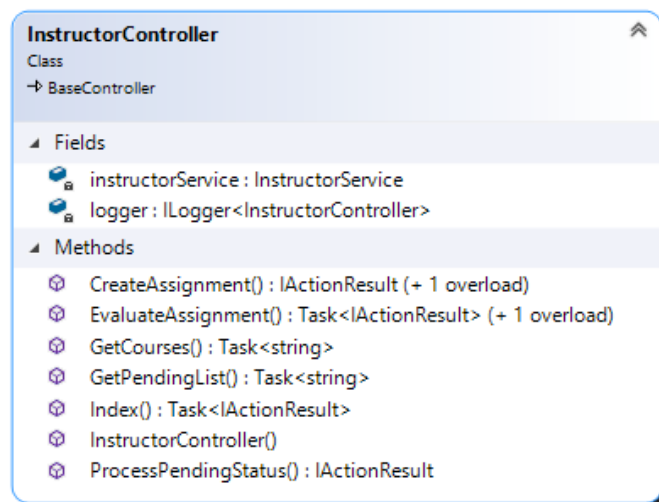
[HttpGet] GetCourses(): lekéri a bejelentkezett felhasználóhoz tartozó csoportokat és ennek eredményével tér vissza.

[HttpGet] GetPendingList(): lekéri a bejelentkezett felhasználóhoz tartozó csoportokra jelentkezett felhasználókat és ennek eredményével tér vissza.

[HttpGet] EvaluateAssignment(int, int, int): a paraméterül kapott egyedi azonosítók alapján (csoport,feladat,hallgató) lekéri a megfelelő adatokat és a feladat értékelésére alkalmas felületet adja vissza ezen adatokkal.

[HttpPost] EvaluateAssignment(int, string): a feladat értékelésének fogadása, ellenőrzi az adatok helyességét. Sikeres rögzítés esetén átirányítás történik a szerepkör főoldalára, egyébként jelzi a hibát a felhasználónak.

[HttpPost] ProcessPendingStatus(int, bool): a csoportba való jelentkezés bírálatának fogadása.



3.13. ábra. *Instructor* vezérlő

3.5.5. *Teacher* vezérlő

A *Teacher* vezérlőt csak tárgyfelelősi szerepkörrel rendelkező felhasználók érhetik el.

[HttpGet] Index(): a tárgyfelelősi szerepkör főoldalával tér vissza.

[HttpGet] CreateCourse(): csoport létrehozására alkalmas felületet adja vissza.

[HttpPost] CreateCourse(CreateCourseViewModel): csoport létrehozásának fogadása, ellenőrzi az adatok helyességét. Sikeres rögzítés esetén átirányítás történik a szerepkör főoldalára, egyébként jelzi a hibát a felhasználónak.

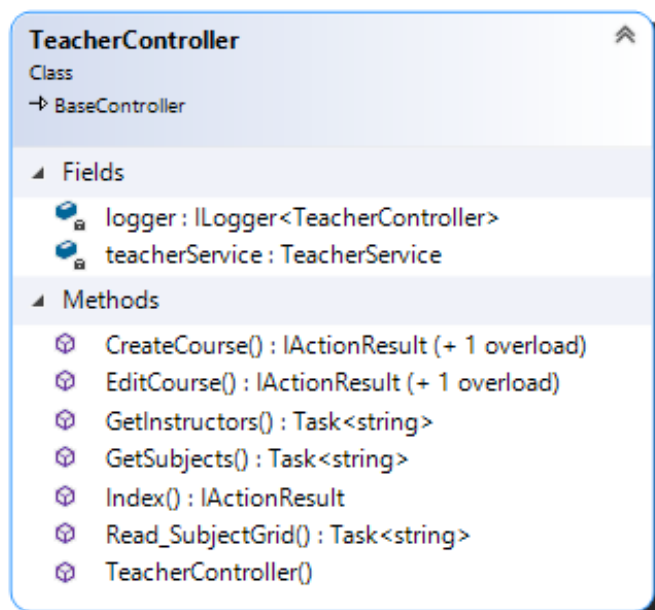
[HttpGet] Read_SubjectGrid(): a bejelentkezett felhasználóhoz tartozó tantárgyakat és a hozzá tartozó adatokat kéri le és ennek eredményével tér vissza.

[HttpGet] GetSubjects(): a bejelentkezett felhasználóhoz tartozó tantárgyak neveit és egyedi azonosítóit kéri le és ennek eredményével tér vissza.

[HttpGet] GetInstructors(): a rendszerben tárolt gyakorlatvezetői szerepkörrel rendelkező felhasználókat kéri le és ennek eredményével tér vissza.

[HttpGet] EditCourse(int): paraméterül egy csoport egyedi azonosítóját kapja, majd a megfelelő csoport adatait kéri le és a csoport módosítására alkalmas felületet adja vissza ezen adatokkal.

[HttpPost] EditCourse(EditCourseViewModel): a csoport módosításának fogadása, ellenőrzi az adatok helyességét. Sikeres rögzítés esetén átirányítás történik a szerepkör főoldalára, egyébként jelzi a hibát a felhasználónak.



3.14. ábra. *Teacher* vezérlő

3.5.6. *Student* vezérlő

A *Student* vezérlőt csak hallgatói szerepkörrel rendelkező felhasználók érhetik el.

[HttpGet] Index(): a hallgatói szerepkör főoldalával tér vissza.

[HttpGet] GetCourses(): lekérdezi a bejelentkezett felhasználóhoz tartozó csoportokat és ennek eredményével tér vissza.

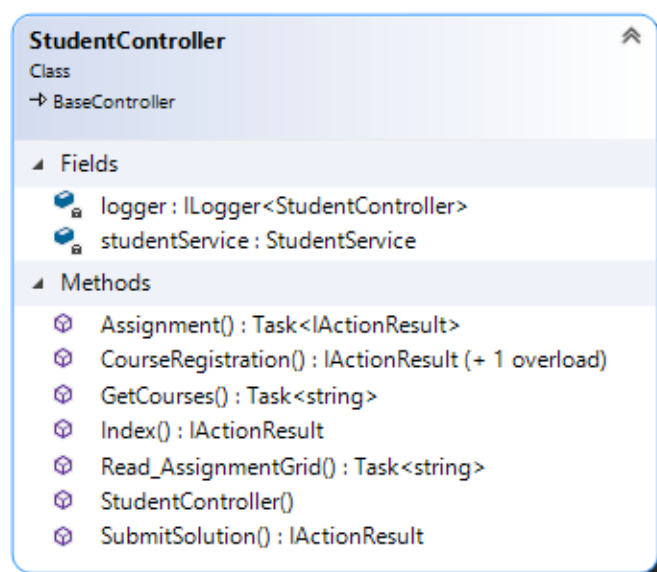
[HttpGet] CourseRegistration(): a csoportba való jelentkezésre alkalmas felületet adja vissza.

[HttpPost] CourseRegistration(CourseRegistrationViewModel): a csoportba való jelentkezés fogadása, ellenőrzi az adatok helyességét. Sikeres rögzítés esetén átirányítás történik a szerepkör főoldalára, egyébként jelzi a hibát a felhasználónak.

[HttpGet] Read_AssignmentGrid(): lekérdezi a bejelentkezett felhasználóhoz tartozó csoportokat és annak adatait, majd ennek eredményével tér vissza.

[HttpGet] Assignment(int): paraméterül egy feladat egyedi azonosítóját kapja, majd lekéri a feladat adatait, és ezeket az adatokat, valamint a feladat megjelenítésére és a megoldás beküldésére alkalmas felületet adja vissza.

[HttpPost] SubmitSolution(SolutionSubmissionViewModel): a feladatra való megoldásának fogadása, ellenőrzi az adatok helyességét. Sikeres rögzítés esetén átirányítás történik a szerepkör főoldalára, egyébként jelzi a hibát a felhasználónak.



3.15. ábra. *Student* vezérlő

3.6. Nézet réteg

Ez a réteg felelős az adatok megjelenítéséért, illetve az egész alkalmazás kinézetéért. A nézeteket a *Views* mappában találjuk vezérlők szerinti almappákba csoportosítva, valamint a megosztott nézeteket a *Shared* mappában. A nézetek megvalósításakor *Razor* [15] szintaxist is használnunk, mellyel *C#* forráskódot illeszthetünk a *HTML* alapú nézetekbe, ezzel megvalósítva a nézetek dinamikus működését.

```
Views/  
├─ Admin/  
├─ Home/  
├─ Instructor/  
├─ Shared/  
├─ Student/  
└─ Teacher/
```

3.6.1. *Home* vezérlő nézetei

Index: az alkalmazás főoldala, továbbá ezen a nézeten lehet bejelentkezni és az alkalmazás nyelvét módosítani.

3.6.2. *Admin* vezérlő nézetei

Index: A rendszergazdai szerepkör főoldalának nézete.

CreateSubject: tantárgyak létrehozására szolgáló nézet, amely egy űrlapot tartalmaz, ami a megfelelő vezérlőnek továbbítja a bevitt adatokat.

CreateUser: felhasználók létrehozására szolgáló nézet, amely egy űrlapot tartalmaz, ami a megfelelő vezérlőnek továbbítja a bevitt adatokat.

UpdateUser: felhasználók adatainak módosítására szolgáló nézet, amely egy űrlapba tölti a felhasználó aktuális adatait, majd a megfelelő vezérlőnek továbbítja az adatokat.

3.6.3. *Instructor* vezérlő nézetei

Index: a gyakorlatvezetői szerepkör főoldalának nézete.

CreateAssignment: feladat létrehozására szolgáló nézet, amely egy űrlapot tartalmaz, ami a megfelelő vezérlőnek továbbítja a bevitt adatokat.

EvaluateAssignment: feladat értékelésére szolgáló nézet, mely megjeleníti a hallgató beadott munkáit, valamint tartalmaz egy űrlapot, amivel az értékelést tudjuk elküldeni a megfelelő vezérlőnek.

3.6.4. *Teacher* vezérlő nézetei

Index: a tárgyfelelősi szerepkör főoldalának nézete.

CreateCourse: csoport létrehozására szolgáló nézet, amely egy űrlapot tartalmaz, ami a megfelelő vezérlőnek továbbítja a bevitt adatokat.

EditCourse: csoport módosítására szolgáló nézet, amely egy űrlapba tölti a csoport aktuális adatait, majd a megfelelő vezérlőnek továbbítja az adatokat.

3.6.5. *Student* vezérlő nézetei

Index: a hallgatói szerepkör főoldalának nézete.

Assignment: egy feladat megjelenítésére szolgáló nézet, amely tartalmazza a feladat adatait, valamint egy űrlapot, amin keresztül a feladatra megoldást tudunk beküldeni.

CourseRegistration: csoportba való jelentkezésre szolgáló nézet, mely egy űrlapot tartalmaz, ami a megfelelő vezérlőnek továbbítja a bevitt adatokat.

SubmitSolutionForm: az *Assignment* nézeten használt parciális nézet¹⁵ (angolul *Partial View*), ami magát a feladat beküldésére szolgáló űrlapot tartalmazza.

¹⁵<https://docs.microsoft.com/en-us/aspnet/core/mvc/views/partial?view=aspnetcore-3.1>

3.6.6. Megosztott nézetek

_Layout: az alkalmazás főoldalának és az *AccessDenied* nézetnek az általános kinézetét és menüsorát tartalmazó nézet.

_MainLayout: az összes többi nézetnek az általános kinézetét és menüsorát tartalmazó nézet.

AccessDenied: a jogosulatlan kérés esetén megjelenítendő nézet.

Error: az egyéb hibák esetén megjelenítendő nézet.

Profile: a felhasználó személyes adatait megjelenítő nézet, ami egy csak olvasható űrlapba tölti a felhasználó adatait, illetve tartalmaz két gombot, melyekkel változtatni tudjuk az alkalmazás nyelvét.

3.7. Lokalizáció

Az *ASP.NET Core* keretrendszer támogatja az alkalmazások lokalizációját [16]. A lokalizáció megvalósításához pár beállítást kell elvégeznünk az alkalmazásban. Első lépésként be kell állítani, hogy az alkalmazás lokalizálható legyen, valamint meg kell adnunk a támogatott nyelveket, az alkalmazás alapértelmezett nyelvét (3.9 forráskód) és meg kell adnunk, hogy az alkalmazás mely mappában keresse a lokalizációs fájlokat.

```
1 services.AddLocalization(option => option.ResourcesPath = "  
    Resources");  
2 services.AddMvc().AddViewLocalization();  
3 services.Configure<RequestLocalizationOptions>(options =>  
4 {  
5     var supportedCultures = new List<CultureInfo>()  
6     {  
7         new CultureInfo("hu-HU"),  
8         new CultureInfo("en-US"),  
9     };  
10    options.DefaultRequestCulture = new RequestCulture(  
        supportedCultures[0]);  
11    options.SupportedCultures = supportedCultures;
```



```
12 | options.SupportedUICultures = supportedCultures;  
13 | });  
14 | ...
```

3.9. forráskód. Lokalizáció beállítása

Ezzel az összes szükséges beállítást elvégeztük. Második lépésként a befecskendezzük a lokalizációhoz szükséges osztályokat a `_ViewImports.cshtml` fájl segítségével, hogy az összes nézetten elérjük ezeket az osztályokat. Az `IViewLocalizer` objektum tárolja a fordítandó kulcs-érték párokat, az `IOptions<RequestLocalizationOptions>` objektum segítségével az alkalmazás által támogatott nyelveket tudjuk lekérdezni, hogy ezeket a szükséges nézeteken meg tudjuk jeleníteni, lehetővé téve a nyelv kiválasztását.

```
1 | ...  
2 | @inject IViewLocalizer localizer  
3 | @inject IOptions<RequestLocalizationOptions> localizationOption
```

3.10. forráskód. Lokalizációhoz szükséges objektumok befecskendezése

Utolsó lépésként pedig a fájlnev és mappa struktúra konvenciókat kell követnünk. A megadott `ResourcesPath` mappában létrehozuk a lokalizálni kívánt nézetek, nézetmodellek és `DTO` mappáit, majd ezekhez létrehozuk a megfelelően elnevezett `resource` fájlokat¹⁶ (pl.: `Index.hu-HU.resx`).

¹⁶A `viewmodelleknél` és a `DTO`-knál csak a nem alapértelmezett nyelvi `resource` fájlokhoz kell kitenni a nyeli kódot (en-US)

```
Resources/  
├── Models/  
│   ├── ViewModels/  
│   │   ├── ...  
│   │   ├── LoginViewModel.resx  
│   │   ├── LoginViewModel.en-US.resx  
│   │   └── ...  
│   └── DTOs/  
└── Views/  
    ├── ...  
    └── Teacher/  
        ├── ...  
        ├── Index.hu-HU.resx  
        ├── Index.en-US.resx  
        └── ...
```

4. fejezet

Tesztelés

Az alkalmazáshoz automatizált felületi tesztek tartoznak, melyet a *Selenium* [17] programkönyvtár segítségével valósítunk meg. Továbbá szükségünk lesz a számítógépünk található *Microsoft Edge* verziójával azonos *Microsoft Edge Driver*-re¹⁷ a *Selenium*-hoz, melyet a `../UITest` mappába kell másolnunk és *Visual Studio*-ban ennek a fájlnak a tulajdonságainál be kell állítani a *Copy to Output Directory*-nak a *Copy always* értéket. Valamint szükséges telepítenünk az alkalmazást a *dotnet publish*-ban olvasható módon. Ezek után nyissuk meg a *testhost.dll.config* fájlt és adjuk meg az *Url* és *DriverPath* kulcsok értékeit¹⁸. Majd mielőtt a tesztet futtatnánk, még indítsuk el az *Edge* böngészőnk és a szerverünk (*"dotnet publishban megadott mappa"/ASS.WEB.exe*) és látogassunk el a *localhost:5001* címre, hogy be tudjuk állítani az alkalmazást biztonságos webhelyként. Ezek után a `../UITest` mappán állva a *dotnet test* vagy *Visual Studio*-ban a *Test explorer*-en a *Run* gombbal tudjuk lefuttatni.

¹⁷<https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/>

#downloads

¹⁸Az *Url* kulcs értéke a kitelepített alkalmazás elérési címe (alapértelmezetten *localhost:5001*). A *DriverPath* kulcs értéke pedig *.."Path"../UITest/bin/Debug/netcoreapp3.1*.

4.1. Futtatott teszt esetek

4.1.1. ASS_AdminTests.cs

Admin_CreateSubject_EmptyFields: annak ellenőrzése, hogy hiányos űrlapot nem lehet beküldeni.

Admin_CreateSubject_AlreadyUsedSubjectName: annak ellenőrzése, hogy foglalt tantárgynévvel nem enged az alkalmazás tantárgyat létrehozni.

Admin_CreateSubject_Ok: tantárgy létrehozás sikerességének a tesztelése.

Admin_CreateUser_Ok: felhasználó létrehozás sikerességének a tesztelése.

Admin_CreateUser_EmptyFields: annak ellenőrzése, hogy hiányos űrlapot nem lehet beküldeni.

Admin_CreateUser_PasswordsNotMatch: annak ellenőrzése, hogy az űrlapon beírt két jelszónak meg kell egyeznie.

Admin_CreateUser_ErrorWhileCreateUser: annak tesztelése, hogy létező felhasználónévvel nem lehet új felhasználót létrehozni.

Admin_UpdateSubject_Ok: a tantárgy módosításnak a sikerességének tesztelése.

Admin_UpdateSubject_Error: meglévő tantárgynévre az alkalmazás nem enged módosítani egy másik tantárgynak a nevét.

Admin_UpdateUser_Ok: a felhasználó módosításának a sikerességének tesztelése.

4.1.2. ASS_OtherTests.cs

ChangeLanguageTest: a nyelvválasztás tesztelése.

EmptyLoginFieldsTest: hiányos űrlappal nem lehet bejelentkezni.

LengthUsernameFieldsTest: nem megfelelő felhasználónév hosszúsággal nem lehet bejelentkezni.

LoginWrongUserTest: rossz vagy nem létező felhasználóval való bejelentkezés.

LoginTest: bejelentkezés tesztelése.

LogoutTest: kijelentkezés tesztelése.

4.1.3. ASS_InstructorTests.cs

Instructor_ApproveRegistration: a hallgató csoportba jelentkezésének az elfogadásának a tesztelése.

Instructor_CreateAssignment_Emptyfields: hiányos űrlappal való feladat létrehozásának a tesztelése.

Instructor_CreateAssignment_WrongRange: rossz időintervallummal való feladat létrehozásnak a tesztelése.

Instructor_CreateAssignment_Ok: sikeres feladat létrehozásnak a tesztelése.

Instructor_EvaluateAssignment: feladat értékelésének a tesztelése.

4.1.4. ASS_StudentTests.cs

Student_CourseRegistration_EmptyField: hiányos űrlappal csoportba való jelentkezésnek a tesztelése.

Student_CourseRegistration_Ok: sikeres csoportba jelentkezésnek a tesztelése.

Student_SubmitSolution_Ok: feladatra való megoldás beküldésének a tesztelése.

4.1.5. ASS_TeacherTests.cs

Teacher_EditCourse_Emptyfields: csoport módosításának tesztelése hiányos űrlappal.

Teacher_EditCourse_Ok: sikeres csoportmódosításnak a tesztelése.

Teacher_CreateCourse_EmptyFields: csoport létrehozásának a tesztelése hiányos űrlappal.

Teacher_CreateCourse_Ok: sikeres csoport létrehozásnak a tesztelése.

5. fejezet

Összegzés

5.1. Használt fejlesztői eszközök

Az alkalmazás fejlesztése során az alábbi fejlesztői eszközök voltak használva¹⁹:

- Microsoft Visual Studio Enterprise 2019
- Visual Studio for Mac
- MySql Workbench

5.2. Telepítés

Előkövetelmények: Microsoft .Net core 3.1 SDK, MySql 8.0 adatbázis szerver.

1. **Forráskód:** töltsük le *GitHub*-ról az alkalmazás forráskódját²⁰
2. **Telepítés:**

Microsoft Azure kihelyezés: [18]

***dotnet publish*:** a másik lehetőségünk az alkalmazás telepítésére, hogy az `../ASS.WEB` mappában állva a parancssorban kiadjuk a `dotnet publish -o "Path"` parancs lefordítja és összeszedi a szükséges fájlokat a kért `"Path"` mappába. Majd az `ASS.WEB.exe` futtatásával lehet elindítani az alkalmazást. Alapértelmezetten a `localhost:5001`-es címen lehet elérni.

¹⁹Ezek használata nem kötelező, az `../ASS.WEB` könyvtárból tudjuk fordítani és futtatni is a `dotnet build` és `dotnet run` parancsokkal.

²⁰<https://github.com/csikie/ASS>

5.3. Továbbfejlesztési lehetőségek

Automata tesztelő: a beadott megoldásokat legyen lehetőség automatán tesztelni *Docker* segítségével.

Szerkeszthető feladatok: legyen lehetőség a kiírt feladatokat módosítani.

Értesítések: a felhasználók kapjanak értesítést a rendszerben ha például egy új feladat került kiírásra egy csoportjukban.

Vizsga mód: ha be van kapcsolva egy feladatnál, akkor a feladat ideje alatt a csoporthoz tartozó megoldott feladatok nem látszanak a felületen.

Tantárgy szintű feladatok: legyen lehetőség egy tantárgy összes csoportjához ugyan azt a feladatot kiírni.

6. fejezet

Köszönetnyilvánítás

Szeretném megköszönni Poór Artúr félévést munkáját, észrevételeit és tanácsait az alkalmazással kapcsolatban, valamint az alkalmazás manuális tesztelésében való segítségét.

Szeretném megköszönni Czuczor Eszternek, hogy megrajzolta az alkalmazás logóját.

Irodalomjegyzék

- [1] Microsoft. *Introduction to ASP.NET Core*. URL: <https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-3.1>.
- [2] Kendo. *Welcome to Kendo UI for jQuery*. URL: https://docs.telerik.com/kendo-ui/introduction?_ga=2.88439835.1270843564.1621007125-198120074.1592317525.
- [3] Bootstrap. *Introduction*. URL: <https://getbootstrap.com/docs/5.0/getting-started/introduction/>.
- [4] Cserép Máté. *Webfejlesztés MVC architektúrában (ASP.NET Core)*. URL: https://mcserep.web.elte.hu/data/education/2019-2020-2_WAF/elte_waf_ea02.pdf.
- [5] Serilog. *Serilog.Extensions.Logging.File*. URL: <https://github.com/serilog/serilog-extensions-logging-file>.
- [6] Nicholas Blumhardt. *Logging “levels” in a structured world*. URL: <https://nblumhardt.com/2014/03/logging-levels-in-a-structured-world/>.
- [7] Microsoft. *Introduction to Identity on ASP.NET Core*. URL: <https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-3.1&tabs=visual-studio>.
- [8] Entity Framework Tutorial. *What is Code-First?* URL: <https://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx>.
- [9] Microsoft. *Entity Framework Core*. URL: <https://docs.microsoft.com/en-us/ef/core/>.

- [10] Microsoft. *Language Integrated Query (LINQ)*. URL: <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/>.
- [11] OWASP. *SQL Injection*. URL: https://owasp.org/www-community/attacks/SQL_Injection.
- [12] Microsoft. *Dependency injection in ASP.NET Core*. URL: <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-3.1>.
- [13] Microsoft. *System.ComponentModel.DataAnnotations Namespace*. URL: <https://docs.microsoft.com/en-us/dotnet/api/system.componentmodel.dataannotations?view=netcore-3.1>.
- [14] Microsoft. *Configure ASP.NET Core Identity*. URL: <https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity-configuration?view=aspnetcore-3.1>.
- [15] Microsoft. *Razor syntax reference for ASP.NET Core*. URL: <https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-3.1>.
- [16] Microsoft. *Globalization and localization in ASP.NET Core*. URL: <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/localization?view=aspnetcore-3.1>.
- [17] Selenium. *Getting started*. URL: https://www.selenium.dev/documentation/en/getting_started/.
- [18] Cserép Máté. “Webszolgáltatások kihelyezése”. (), 12–. old. URL: https://mcserep.web.elte.hu/data/education/2019-2020-2_WAF/elte_waf_ea11.pdf.

Ábrák jegyzéke

2.1. Főoldal	5
2.2. Bejelentkezési hiba	6
2.3. Sikeres bejelentkezés	6
2.4. Nyelvváltás	7
2.5. Rendszergazdai szerepkör kezdőoldala	8
2.6. Tantárgy létrehozás	9
2.7. Felhasználó létrehozás	10
2.8. Tárgyfelelős kezdőoldala	11
2.9. Csoport létrehozás	12
2.10. Csoport módosítása	13
2.11. Gyakorlatvezető kezdőoldala	14
2.12. Feladat létrehozás	15
2.13. Adatok validálása	16
2.14. Feladat értékelése	17
2.15. Hallgató kezdőoldala	18
2.16. Csoportba jelentkezése	18
2.17. Feladat megtekintése	19
2.18. Profil oldal	20
2.19. Jogosulatlan kérés	21
2.20. Egyéb nem várt hiba	21
3.1. A Modell-Nézet-Vezérlő architektúra	23
3.2. Az adatbázist leképző objektumok	27
3.3. Az adatbázis táblái	28
3.4. Service osztályok őse	33
3.5. LoginService osztály és interface	34
3.6. AdminService osztály és interface	36

3.7. TeacherService osztály és interface	37
3.8. InstructorService osztály és interface	38
3.9. StudentService osztály és interface	39
3.10. <i>Home</i> vezérlő	45
3.11. <i>Base</i> vezérlő	46
3.12. <i>Admin</i> vezérlő	48
3.13. <i>Instructor</i> vezérlő	49
3.14. <i>Teacher</i> vezérlő	50
3.15. <i>Student</i> vezérlő	51

Táblázatok jegyzéke

3.1. Adatbázis: felhasználók táblája	28
3.2. Adatbázis: szerepkörök táblája	29
3.3. Adatbázis: felhasználók és szerepkörök kapcsolótáblája	29
3.4. Adatbázis: feladatok táblája	30
3.5. Adatbázis: csoportok táblája	30
3.6. Adatbázis: gyakorlatvezetők táblája	30
3.7. Adatbázis: megoldások táblája	31
3.8. Adatbázis: megoldások táblája	31
3.9. Adatbázis: hallgatók és csoportok kapcsolótáblája	32
3.10. Adatbázis: tárgyfelelősök és tantárgyak kapcsolótáblája	32

Forráskódjegyzék

3.1. Naplózás beállításai	24
3.2. Adatbázis elérése	26
3.3. Példa az attribútumok használatára	40
3.4. Szerepkörök felsorolási típusa	41
3.5. Szerepkörök tárolása az adatbázisba (DbInitializer.cs)	42
3.6. CultureCodeMapping osztály	42
3.7. Felhasználói fiók konfigurációs beállításai	43
3.8. Alkalmazás futási módja	44
3.9. Lokalizáció beállítása	54
3.10. Lokalizációhoz szükséges objektumok befecskendezése	55