

SZEGEDI TUDOMÁNYEGYETEM

Természettudományi és Informatikai Kar

Szoftverfejlesztési Tanszék

Szakdolgozat

Tanterv optimalizáló
(Curriculum optimizer)

Készítette:

Varga Zoltán

Programtervező Informatika szakos hallgató

Témavezető:

Dr. Bilicki Vilmos

egyetemi adjunktus

Szeged

2025

TARTALOMJEGYZÉK

Tartalomjegyzék

TARTALOMJEGYZÉK.....	2
TARTALMI ÖSSZEFOGLALÓ.....	3
FELADATKIÍRÁS.....	4
MOTIVÁCIÓ.....	5
1. TERÜLETI ÁTTEKINTÉS.....	7
1.1 Összehasonlító táblázat [1.1].....	7
1.2 Elemzés és következtetések.....	8
2. AZ ALKALMAZÁS FUNKCIONÁLIS SPECIFIKÁCIÓJA.....	9
2.1 Use-Case.....	9
2.2 funkcionális specifikációk.....	11
2.3 képernyő tervek.....	12
3. FELHASZNÁLT TECHNOLOGIÁK.....	17
3.1 Angular.....	17
3.2 Laravel.....	17
3.3 MySQL.....	18
4. ADATMODELL.....	19
4.1 Relációs vs NoSQL megközelítés összehasonlítása.....	19
4.2 REST vs GraphQL összehasonlítása.....	19
4.3 Részletes adatmodell diagram, normalizálás magyarázata.....	20
5. FONTOSABB KÓDRÉSZEK ISMERTETÉSE.....	24
5.1 Reszponzív design.....	24
5.2 Autentikációs folyamat leírása.....	25
5.3 Jogosultsági modell részletes leírása.....	26
5.4 Frontend optimalizálási technikák leírása.....	27
Nem használt csomagok eltávolítása.....	28
Csomagfrissítések kezelése.....	28
Production build optimalizálás.....	29
5.5 Kód dokumentálása.....	29
5.6 felhasználói dokumentációs.....	30
5.7 Többnyelvűség létrehozása:.....	30
5.8 Értesítési mechanizmusok.....	31
Értesítési architektúra áttekintése:.....	32
5.9 Optimalizáció.....	33
5.10 Statisztikák, adat vizualizációk.....	38
Diákokra vonatkozó statisztikák.....	38
Tantervi háló vizualizáció.....	40
TAPASZTALATOK, TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK.....	41
Tapasztalatok.....	41
Továbbfejlesztési lehetőségek.....	41
IRODALOMJEGYZÉK.....	43
NYILATKOZAT.....	44
KÖSZÖNETNYILVÁNÍTÁS.....	45
ELEKTRONIKUS MELLÉKLET.....	46

TARTALMI ÖSSZEFOGLALÓ

Téma megnevezése:

A szakdolgozatom témája egy átlátható tanterv kezelő program.

Feladat megfogalmazása:

Minden félévben diákok százai vacillálnak milyen kurzusokat kellene felvenniük ebben a félévben, viszont a ehhez biztosított adatok és azoknak megjelenítésük sokszor átláthatatlan és hiányos. Egy olyan alkalmazást szeretnék csinálni ami minden segítséget biztosít a tárgy felvétel eldöntéséhez.

Megoldási mód:

Mivel az egyetemi oktatási szoftverek egy funkcióját akarom elkészíteni/kiegészíteni és azok mind webapp-ok, ezért én is egy webapp-t fogok elkészíteni. A tervezési fázisban elkészítettem a piackutatást a jelenleg használatban lévő hasonló témájú weboldalak alapján. Ez remek támpontot nyújtott abban, hogy pontosan egy ilyen program milyen elemekkel kell rendelkeznie. Majd elkészítettem az UML use-case és adatfolyam diagramokat, a képernyő terveket. Ezek után neki álok a programozásnak.

Alkalmazott eszközök, módszerek:

A frontend-et Angular v18 keretrendszerben készítettem TypeScript, HTML és SCSS nyelveken. A Backend-nek eredetileg a Firebase Firestore NoSQL kollektióit használtam, de később áttértem Laravel 11-re keretrendszerre amit PHP v8.2 nyelven fejlesztem MySQL adatbázis mellett. A kódbázis verzió követését a GitHub verziókövető rendszeren keresztül tároltam. A fejlesztés közben Xampp-val hoztam létre a szükséges lokális MySQL szerveret. A backend-en a manuális tesztek Postman-val hajtottam végre. A host-olást saját domain-val oldottam meg amihez Cpanel-en keresztül fértem hozzá. A kódot pedig Visual Studio Code-ban írtam. A diagramokat pedig draw.io-ban készítettem el.

Elért eredmények:

A szakdolgozatomba implementáltam azokat a funkciókat amíg szerintem megkönnyítenék a kurzusfelvételi időszakot. Név szerint: Tanterv megjelenése és kereshetősége. A teljesített és felvet kurzusok jelölése a tanterven, a tantervből át tudok ugrni a kurzus fórumára ahol megtudhatok mindent ami a kurzus felvételhez kell tudnom mint statisztikák vélemények, tárgyatematika, követelmény. A legfontosabb azonban az hogy programom képes felhasználó igények szerint létrehozni Mohó vagy B&B algoritmussal optimalizált tantervet. A programhoz nyilván elkellet készítenem a szükséges de a cél szempontjából mellékes funkciókat, mint a tanterv és kurzusok létrehozása/módosítása/törlése. A felhasználó regisztrálása és bejelentkezése, illetve a profil.

FELADATKIÍRÁS

Szakdolgozatom célja egy olyan webes alkalmazás fejlesztése volt, amely megkönnyíti az egyetemi kurzusfelvétel folyamatát, valamint lehetőséget nyújt a hallgatók számára tanterveik személyre szabott optimalizálására. A projekt során Angular frontend keretrendszert, Laravel backend keretrendszert, valamint MySQL adatbázist használtam.

A rendszer lehetőséget biztosít a hallgatóknak arra, hogy a megnézhessék rendezhessék a tantervet és a mélyebben megvizsgálóhaság az egyes kurzusok tulajdonságait, illetve igényeiknek megfelelően rendezhessék azokat.

Az alkalmazás ezenkívül biztosítja a tantervekhez és kurzusok hoz tartozó CRUD szolgáltatásokat illetve a diákok autentikációját is.

Összességében a kurzus felvételhez szeretnék támpontokat adni.

MOTIVÁCIÓ

Szakdolgozatom célja, hogy egy olyan webalkalmazást készítssek, amely megkönnyíti a kurzusfelvételi időszakot diáktársaim számára, és lehetőséget biztosít olyan visszajelzések adására az oktatók felé, amelyek segíthetik a tantárgyak fejlesztését.

Egyetemisták százai kerülnek zavarba minden egyes tárgyfelvételi időszak kezdetén, mivel nem érzik magukat elég informátnak ahhoz, hogy milyen szabadon és kötelezően választható tárgyakat kellene felvenniük ahhoz, hogy ambícióiknak és érdeklődési körüknek megfelelő tantervet állíthassanak össze.

Sokan úgy vélik, hogy a Neptun mintatantervei csak a kurzus nevét és az ajánlott félévet tartalmazzák, ami gyakran elégtelen egy ilyen fontos döntés meghozatalához. A tantárgyak követelményeiről és tartalmáról sokszor csak a versenyalapú tárgyfelvétel utolsó heteiben kapnak információt a hallgatók, jellemzően egy másik, CooSpace nevű platformon keresztül – amikor már gyakran túl késő változtatni.

Kevesen tudják, hogy a Neptun rendszer biztosít tantárgytematikákat PDF formátumban, de ezek sokszor hiányosak, vagy olyan szakszavakkal vannak megfogalmazva, amelyeket egy, a kurzust még el nem végzett hallgató nehezen ért meg.

Az oktatási szoftverek sokszor elhanyagolják a statisztikák készítését és közzétételét is. Általában csak a legalapvetőbb gyakorisági megoszlásokat közlik, amelyek nem veszik figyelembe az adatok keletkezési körülményeit.

Problémának tartom továbbá, hogy a diákoknak nincs egy állandó, strukturált platformjuk, ahol értelmes visszajelzést adhatnának az oktatóiknak. Jelenleg csupán egy félév végi, sablonos kérdőív áll rendelkezésre, amelyben olyan általános kérdések szerepelnek, mint például: *„Ön szerint mennyire volt felkészült az oktató az órára?”*. Az ilyen kérdésekre általában lagymatag, semmitmondó válaszok érkeznek, mint például *„Ja”*, amelyek nem segítenek az oktátónak a lehetséges hiányosságok felismerésében.

A felsorolt problémák orvoslására teszek kísérletet a diplomamunkám keretében.

A tantervet olvashatóbbá teszem azáltal, hogy szűrhetővé és rendezhetővé teszem különböző attribútumai alapján. Lehetővé teszem, hogy a hallgató által megadott preferenciák alapján egy személyre szabott, célkitűzéseinek megfelelő optimalizált tantervet állíthasson össze.

Támogatásként minden kurzushoz csatolható lesz egy fórum, ahol a diákok visszajelzéseket írhatnak a tárgyról az oktatók számára. Ugyanezen a felületen elérhetőek lesznek a tantárgyhoz tartozó statisztikák is,:

- Jegyek eloszlását mutató kördiagram
- Bukási arányokat bemutató kördiagram
- Lineáris regresszió (félév és átlag párosítása)
- Box plot
- Diagram a normál eloszlástól való eltérés vizualizálására

Továbbá lehetőséget biztosítok az oktatóknak arra, hogy feltöltsék a tárgytematikát, melynek szerkezete fejezetekre oszlik. Minden fejezet egy olyan kérdésre ad választ, ami a hallgatók számára fontos lehet (pl.: miről szól a kurzus, milyen tudás szükséges, mik a követelmények stb.).

Létrehozok egy oldalt, ahol a diákok megtekinthetik a jegyeiket és leadhatják a frissen felvett kurzusaikat. Emellett lesz egy statisztikai oldal is, ahol a hallgatók személyre szabott elemzéseket láthatnak:

- Lineáris regresszió a TAN (Tanulmányi Átlagszám) és félév viszonylatában
- Vonaldiagram a TAN alakulásáról félévenként
- Vonaldiagram az összes diák TAN-járól
- Egy kimutatás a diák előrehaladásáról

Ebből logikusan következik, hogy több jogosultsági szintet különböztetek meg a webalkalmazásban: admin, tanár és diák.

- Az admin hozzáfér minden funkcióhoz, kivéve a jegyekhez tartozó ellenőrzőhöz és a személye statisztikákhoz (mivel az személyes adatokat tartalmaz). Az admin regisztrálja az összes felhasználót, és ő hozza létre a tantervet.
- A tanárok hozhatnak létre és kezelhetnek kurzusokat, valamint jegyeket írhatnak be a hallgatóknak.
- A diákok saját fiókjukon keresztül kurzusokat vehetnek fel, visszajelzést adhatnak, és megtekinthetik a személyre szabott statisztikáikat.

1. TERÜLETI ÁTTEKINTÉS

Ebben a fejezetben azokat az oktatási célú platformokat tekintem át, amelyek funkcionálisan vagy céljukban hasonlóságot mutatnak az én szakdolgozatom témájával. A célom nem csupán a meglévő rendszerek bemutatása, hanem az is, hogy rávilágítsak azokra a hiányosságokra és piaci résekre, amelyek lehetőséget adnak az új ötletek és megközelítések bevezetésére.

A vizsgált platformok között szerepel a Neptun, a Coospace, a Moodle és a Kréta, melyek mind széles körben elterjedtek az oktatás különböző szintjein, elsősorban felső- és középfokú intézményekben. Ezeket egy funkcionális összehasonlító táblázat segítségével elemeztem, amely az alábbi szempontokat vizsgálja: belépés/regisztráció lehetősége, tanterv megjelenítése és kezelése (szűrés, keresés), kurzusok véleményezhetősége, tárgyatematika elérhetősége, valamint az általános felhasználói élmény (design, többnyelvűség stb.).

1.1 ÖSSZEHASONLÍTÓ TÁBLÁZAT [1.1]

Versenytársak	<u>Coospace</u>	<u>Neptun</u>	<u>Moodle</u>	<u>Kréta</u>
Funkciók	✗	✗	✗	✗
Belépés	✓	✓	✓	✓
Regisztráció	✓	✓	✓	✓
Tanterv megjelenítése	✗	✓	✗	✗
Tanterv rendezhetősége	✗	✗	✗	✗
Tanterv szűrhetősége	✗	✓	✗	✗
Tanterv kershetősége	✗	✓	✗	✗
Tanterv Optimalizálása	✗	✗	✗	✗
Tanterv készítő	✗	✓	✗	✗
Eletronikus ellenőrző	✓	✓	✓	✓
Kurzus Forum	✓	✗	✓	✓
Statistikák generálása	✓	✓	✓	✓
Többnyelvűség	✓	✓	✓	✓
Modern minimalista megjelenés	✓	✓	✓	✓
Kurzusok véleményezése	✓	✗	✗	✗
Tárgyatematika megadása	✗	✓	✗	✗

1.1 Ábra: összehasonlító táblázat

1.2 ELEMZÉS ÉS KÖVETKEZTETÉSEK

A vizsgált rendszerekből jól látható, hogy az alapvető funkciók – mint a belépés, regisztráció, elektronikus ellenőrző és többnyelvűség és a modern letisztult minimalista kinézet minden esetben adottak, ezek tehát iparági alapelvárásnak tekinthetők. Emellett az is megfigyelhető, hogy egyik alkalmazásnál sem lehet regisztrálni közvetlenül. Mindegyiknél regisztrálják a felhasználókat. Ez logikus mivel akárki nem lehet a felhasználó csak az akit vagy diáknak vagy tanárnak felvettek az intézményhez.

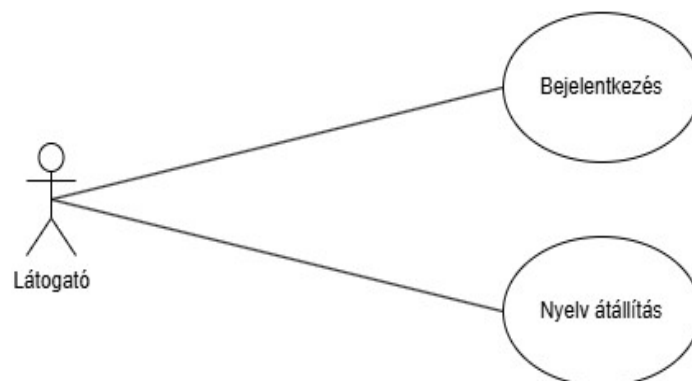
Ugyanakkor szembeűnő, hogy a tanterv kezelésével kapcsolatos funkciók például a rendezhetőség, szűrhetőség, keresetőség, vagy az egyéni tantervkészítés többnyire hiányoznak ezekből a rendszerekből. Sokban még tantervet sem képes jeleníteni közvetlenül csak letölthető állományként teszik elérhetővé. Hasonlóképpen ritka az, hogy a kurzusokról kimutatásokat kapjanak a hallgatók, vagy hogy a tárgyakhoz részletes tematika legyen csatolva.

Ezek az üresen maradt területek megerősítetlen abban, hogy ezek az oktatási szoftverek kis hang sújt fordítanak a kurzusfelvételre és így egy olyan piaci rést célozhatnak meg ami még nincs zsúfolásig megtömve kompetitorokkal.

2. AZ ALKALMAZÁS FUNKCIONÁLIS SPECIFIKÁCIÓJA

2.1 USE-CASE

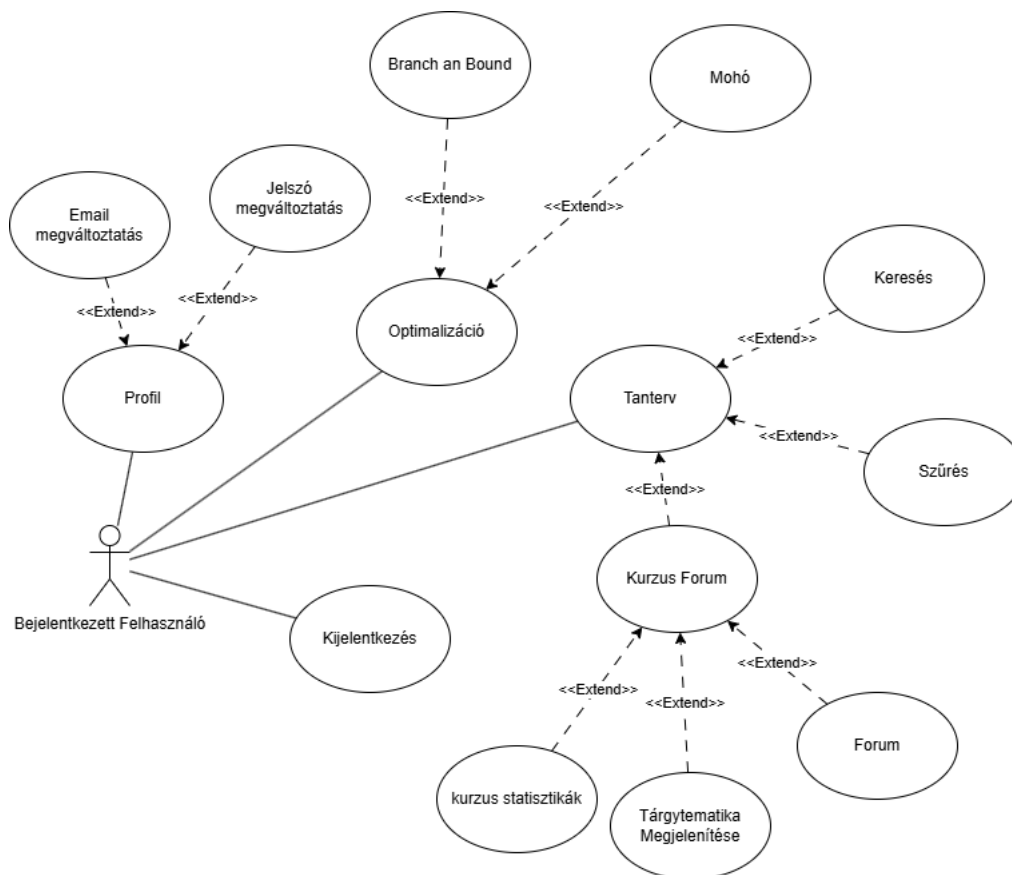
Látogató



2.1.1 Ábra

A látogató [2.1.1] minden olyan felhasználó aki még nem jelentkezett be. Csak bejelentkezés és a nyelv átállítási funkciókhoz fér hozzá.

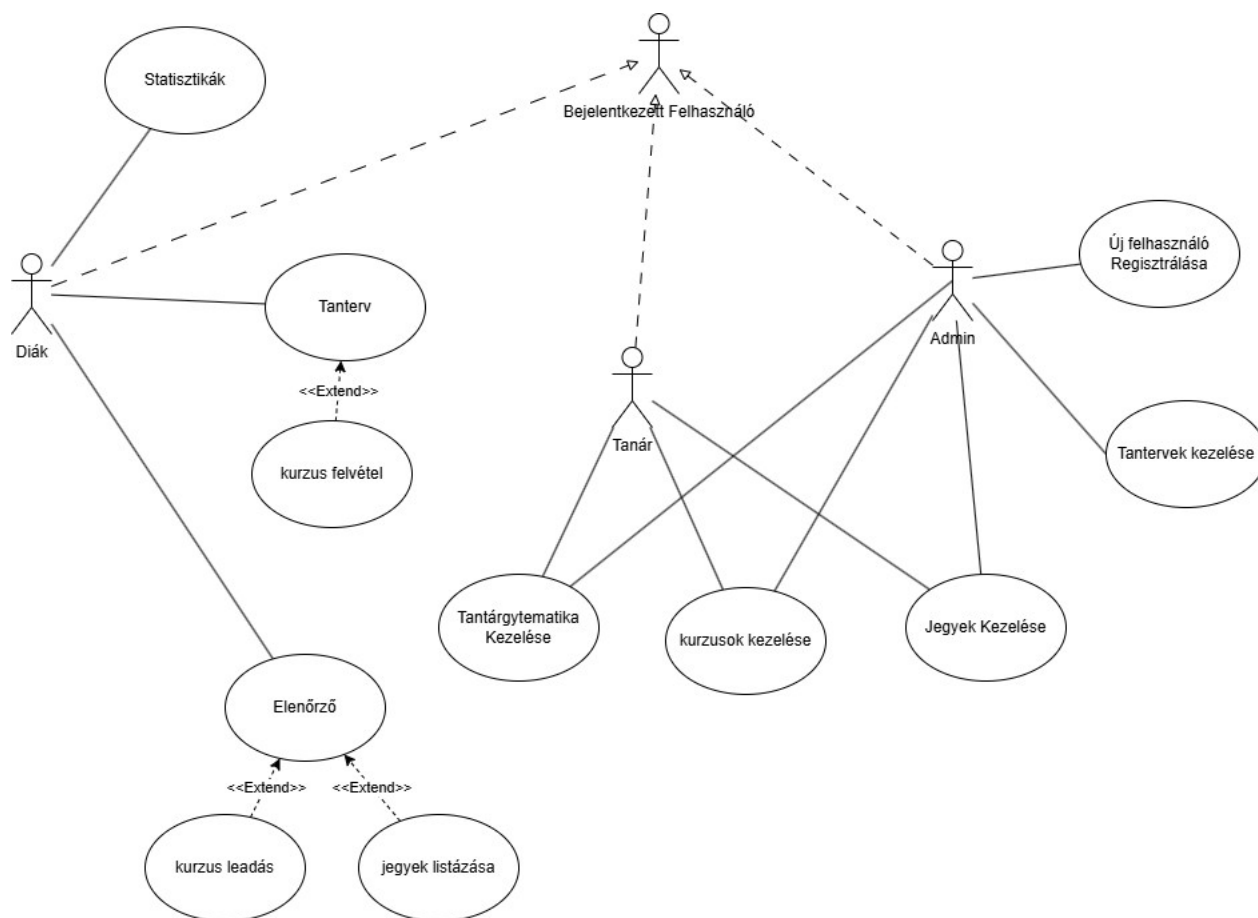
Bejelentkezett felhasználó



2.1.2 Ábra

A bejelentkezett felhasználó [2.1.2] képes megnézni a saját felhasználói adatait megváltoztatni a jelszavát és a email címét a profilban, megnézni a tantervet azt szűrni és keresni adott feltételek alapján, illetve minden a tantervben megjelenített kurzushoz tartozik egy kurzus fórum amit a tantervben rákattintva érhet el. A kurzus fórumban lehetősége van saját szavaival véleményezni/hozzáírni a kurzus forumhoz és megnézni a statisztikákat a tárgyatematikát. Emellett van még egy optimalizációs funkció ahol a tantervet lehet optimalizálni Branch and Bound illetve mohó algoritmusok szerint. Végezetül ki tud jelentkezni.

Diák Tanár és Admin felhasználó



2.1.3 Ábra

Mind a három felhasználó típus a Bejelentkezett felhasználóból öröklődik. Azaz minden funkcióval és jogosultsággal rendelkezik mint az ő.

A diák az örökölt funkciókon túl hozzáfér az elektronikus ellenőrzőjéhez ahol képes megnézni a jegyei alakulását leadni kurzusokat ha még a jelenlegi félévben vette fel. Hozzáfér egyéb diák specifikus statisztikákhoz és a korábban ismertet Tantervben képes felvenni a kurzusokat.

A tanár képes a saját kurzusához tartozó tárgyatematikákat hozzáadni a kurzusfórumhoz, illetve értékelni a diákok félév munkáikat és létrehozni/módosítani/törölni kurzusokat.

Az admin felhasználó képes mindenre amire a tanár képes, és azon felül ő képes kezelni a tanterveket és az új felhasználók regisztrálást.

2.2 FUNKCIONÁLIS SPECIFIKÁCIÓK

- Elektronikus ellenőrző:
 - A diákok képesek megnézni a jegyeiket és leadni az aktuális félévben felvet még nem osztályozót kurzusokat.
- Jegy beírás:
 - A tanárok meg képesek értékelni a diákokat. Erről értesítést kap a diák email-ben
- Diákoknak szóló statisztikák:
 - A diákok képesek megnézni a statisztikáikat.
- Tantervek Kezelése:
 - A tanterveket készíteni/módosítani/törölni lehet.
- Tanterv megjelenítés:
 - Megjeleníti a tantervet. Itt képesek a diákok felvenni a kurzust. Innét át lehet lépni a kurzus fórumra
- Tanterv szűrés:
 - szűrhetővé teszi a tantervet tulajdonsági alapján.
- Optimalizált tanterv generálása:
 - Személyre szóló tanterv generálása a megadót feltételek alapján és cél alapján a Branch and Bound vagy mohó algoritmusokkal.
- Belépés:
 - Belépteti a felhasználókat.
- Regisztrálás:
 - Létrehoz az admin egy új felhasználókat. Erről értesítést kap az új felhasználó email-ben
- Profil:
 - A felhasználói adatok megjelenítése és jelszó email módosítása.
- Kijelentkezés:
 - Kilehet jelentkezni.
- Kurzus Fórum:

- Itt meglehetősen nézni a kurzushoz tartozó statisztikákat adatokat tárgyatematikát és a kurzus fórumát amihez hozzá lehet írni
- Kurzusok Kezelése
 - A kurzusok kezelése (létrehozás/módosítás/törlés)
- Nyelv átváltása:
 - Át lehet átváltani az oldal nyelvét magyarról angolra és vissza.

2.3 KÉPERNYŐ TERVEK

A felhasználó először a belépés oldalával találkozik [2.3.1], ahol bejelentkezik jelszó-azonosító párossal, illetve lehetőségében áll átváltani az oldal nyelvét az alapértelmezett magyarra.

2.3.1 Ábra

Sikeres bejelentkezés után a tanterv oldalára [2.3.2] irányítódnak át, ahol szűrhetik, kereshetik és optimalizálhatják is a tantervet.

2.3.2 Ábra

Ha itt a tanterv oldalán rákattintanak egy kurzusra, akkor átirányítódnak a hozzá tartozó kurzus fórumra [2.3.3], ahol megnézhetik a tárgy tematikát, a követelményeket, a statisztikákat, és a többi felhasználó véleményét a tárggyal kapcsolatosan. Innen a vissza nyíllal lehet visszatérni a tantervre.

2.3.3 Ábra

A menüsávot használva átnavigálhatnak a profil oldalra [2.3.4] és megnézhetik a saját felhasználói adataikat a profil oldalon, és megváltoztathatják a jelszavunkat.

2.3.4 Ábra

A menüsávot használva átnavigálhat a diák az ellenőrző oldalra [2.3.5], ahol megnézheti a saját ellenőrzőjét és hozzáadhat új tárgyakat az aktív félévéhez.

Menü sáv

02. Félév 2024/25

Név	félévtözi értékelés	jegy
Value 1	Value 2	Value 3
Value 4	Value 5	Value 6
Value 7	Value 8	Value 9
Value 10	Value 11	Value 12

kurzus kód

+

01. Félév 2024/25

Név	félévtözi értékelés	jegy
Value 1	Value 2	Value 3
Value 4	Value 5	Value 6
Value 7	Value 8	Value 9
Value 10	Value 11	Value 12

2

2.3.5 Ábra

A menüsávot használva átnavigálhat a tanár és az admin felhasználó a jegyek kezelése oldalra [2.3.6], hogy a kurzusát az aktív félévben hallgató hallgatók tanulmányait értékeljék.

Menüsáv

azonosító	félévtözi értékelés	jegy
Value 1	Value 2	Value 3
Value 4	Value 5	Value 6
Value 7	Value 8	Value 9
Value 10	Value 11	Value 12

kurzus kód

Item 1

Item 2

Item 3

Item 4

Változtatások Mentése

2.3.6 Ábra

A menüsávot használva átnavigálhat a tanár és az admin felhasználó a tárgyatematika kezelése oldalakra [2.3.7], hogy megadhassa ezeket az adatokat.

Menü sáv

Kurzusnév: Tárgytematika

Heading

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Módosít

Mentés

2

2.3.7 Ábra

A menüsávot használva átnavigálhat az admin felhasználó tanterv készítő oldalra [2.3.8], ahol létrehozhatja a tantervet. A plusz gombra kattintva hozzáad, a mínusz gombra kattintva pedig elvesz elemeket.

Menü sáv

Specializáció

kategoria

név	kód	kredit	tárgyfelelős	ajánlott félév	előfeltétel
Value 1	Value 2	Value 3			
Value 4	Value 5	Value 6			
Value 7	Value 8	Value 9			
Value 10	Value 11	Value 12			

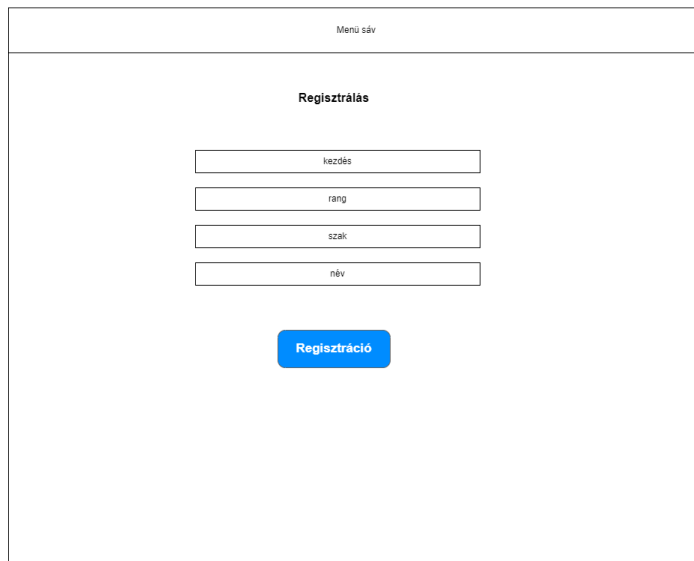
+

+

2.3.8 Ábra

15

A menüsávot használva átnavigálhat az admin felhasználó az új felhasználó kezelése oldalra [2.3.9], ahol új felhasználót tud regisztrálni.



The screenshot shows a web interface for user registration. At the top, there is a header bar labeled "Menü sáv". Below it, the main content area is titled "Regisztrálás". Under this title, there are four input fields stacked vertically, labeled "kezdés", "rang", "szék", and "név". Below these fields is a blue button labeled "Regisztráció".

2.3.9 Ábra

A képernyőterveket a fejlesztés korai stádiumában készítettem el, amiktől később jelentősen eltértem a végleges produktumban. De nyilván a tervezésnek nem az a célja, hogy pontosan meghatározza az eredményt, hanem hogy útmutatást adjon.

3. FELHASZNÁLT TECHNOLOGIÁK

3.1 ANGULAR

Az Angular egy nyílt forráskódú webalkalmazás-fejlesztő keretrendszer, amelyet a Google fejleszt. Főleg egyoldalas alkalmazások (SPA) készítésére használják. Komponensalapú felépítése és TypeScript nyelven történő fejlesztése lehetővé teszi a jól strukturált, gyors és rezponzív webes felületek létrehozását.

Én Angular 18.2-as verzióját használtam a projekthez, ami teljesen meg felelt az elvárásaimnak.

Az Angular-hoz npm-vel (fájl kezelő) rengetek csomagot lehet telepíteni, én felsorolom az általam használtak közül a legfontosabbakat.

- angular material
 - Ez egy nagyon népszerű csomag angular-hoz ami előre elkészített rezponzív UI elemeket tartalmaz (gombok, select-ek ,táblák stb..)
- ngx-translate
 - Ezt használtam az alkalmazás fordítására, amit 18 keretrendszerben lehetővé.
- plotly.js
 - Ez egy diagram készítő könyvtár elsősorban python-hoz én egy javaScriptes korlátozottab verzióját használtam a diagramok elkészítéséhez
- rxjs
 - Az aszinkron lekérdezések kezelésére használtam
- compodoc
 - Dokumentáció generáló eszköz kifejezetten Angular alkalmazásokhoz. Képes automatikusan generálni a projekt dokumentációját a kódban található kommentek alapján.

3.2 LARAVEL

A Laravel egy PHP nyelvű webalkalmazás-keretrendszer, amely egyszerűvé és hatékonyá teszi a szerveroldali fejlesztést. Én a REST API-k elkészítésénél használtam.

A laravel hez a következő könyvtárakat telepitem:

- MATHPHP
 - A mathphp egy matematikai könyvtár a MIT gondozásában a statisztikák mint a lineáris regresszió vagy a gyakoriság számolásánál használtam
- phpdocumentor/shim

- Dokumentáció generáló eszköz PHP alkalmazásokhoz. Képes automatikusan generálni a projekt dokumentációját a kódban található kommentek alapján.

3.3 *MYSQL*

A MySQL egy relációs adatbázis-kezelő rendszer (RDBMS), amelyet gyakran használnak webalkalmazások adatainak tárolására. Gyors, megbízható és támogatja a strukturált lekérdezéseket (SQL).

4. ADATMODELL

4.1 RELÁCIÓS VS NOSQL MEGKÖZELÍTÉS ÖSSZEHASONLÍTÁSA

Az adatmodellezés során két fő megközelítést különböztetünk meg: a relációs és a NoSQL adatbázisokat. A relációs adatbázisok az adatokat táblázatokban tárolják, ahol minden tábla sorokból és oszlopokból áll, és az adatok között szigorú kapcsolatok vannak meghatározva. Ez a megközelítés előre definiált sémát használ, ami azt jelenti, hogy az adatstruktúra előre rögzített, és az adatoknak ehhez kell igazodniuk. A relációs adatbázisok erőssége, hogy támogatják az összetett lekérdezéseket és biztosítják az adatok integritását az ACID tranzakciós tulajdonságok révén, ami különösen fontos olyan rendszerekben, ahol az adatkonzisztencia kritikus.

Ezzel szemben a NoSQL adatbázisok sokkal rugalmasabbak, mivel nem kötődnek szigorú sémához, és különböző típusú adatmodelleket támogatnak, például dokumentumokat, kulcs-érték párokat vagy gráfokat. Ezek az adatbázisok általában jobban skálázhatók, különösen vízszintes irányban, azaz több szerver hozzáadásával képesek kezelni a növekvő adatforgalmat. Noha a NoSQL rendszerek nem mindig garantálják az ACID tulajdonságokat, inkább a teljesítményre és a rendelkezésre állásra helyezik a hangsúlyt, ezért különösen előnyösek olyan alkalmazásoknál, ahol nagy mennyiségű, gyorsan változó vagy strukturálatlan adatot kell kezelni.

Összességében elmondható, hogy a relációs adatbázisok ideálisak olyan helyzetekben, ahol az adatok jól strukturáltak, és fontos az adatintegritás, míg a NoSQL megoldások inkább a rugalmasságot és a skálázhatóságot helyezik a hangsúlyt.

Én elsőnek NoSQL adatbázist választottam, de az a szabadság amit a NoSQL nyújt nekem csak hátráltató tényező volt. A relációs modell egyszerű struktúrájához és szabályaihoz számomra könnyebb alkalmazkodni.

4.2 REST VS GRAPHQL ÖSSZEHASONLÍTÁSA

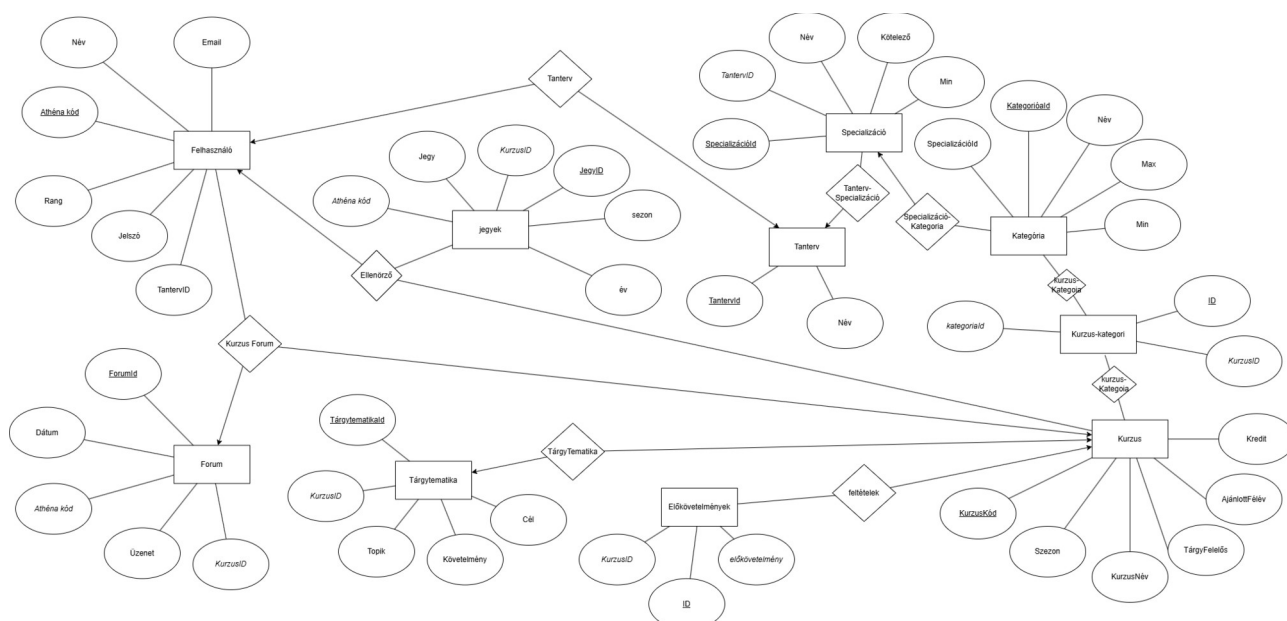
Az API-k területén a REST és a GraphQL két népszerű megközelítés, amelyek különböző módon kezelik az adatok lekérését és kezelését. A REST egy olyan architektúrális stílus, amelyben az adatok erőforrásokként jelennek meg, és minden erőforrásnak saját URL-je (végpontja) van. Az adatokat az HTTP metódusok segítségével lehet lekérdezni vagy módosítani, például GET, POST, PUT vagy DELETE kérésekkel. A REST API-k általában több végpontot használnak, és az adatok előre meghatározott struktúrában érkeznek, ami néha problémát okozhat, ha a kliensnek több vagy kevesebb adatot kellene kapnia, mint amit az adott végpont alapértelmezett válasza tartalmaz.

Ezzel szemben a GraphQL egy lekérdező nyelv, amely lehetővé teszi, hogy a kliens pontosan azt az adatot kérje le, amire szüksége van, egyetlen végponton keresztül. Ez jelentősen csökkenti a hálózati kérések számát, hiszen nem kell külön végpontokat hívni különböző adatokért. A GraphQL

emellett erősen típusos séma alapján működik, ami segít az adatok konzisztenciájának fenntartásában. Továbbá támogatja a valós idejű adatfrissítéseket is, ami előnyt jelenthet dinamikus alkalmazásoknál.

Összefoglalva, a REST egyszerűsége és széles körű támogatottsága miatt még mindig nagyon elterjedt, különösen akkor, ha az adatigények viszonylag egyszerűek és jól definiáltak. A GraphQL viszont nagyobb rugalmasságot és hatékonyságot nyújt, főleg összetettebb vagy változó adatigények esetén, ahol fontos a hálózati forgalom minimalizálása és a kliensoldali kontroll növelése. Én GraphQL minden előnyének ellenében mégis REST API valósítottam meg, mert azzal már több tapasztalom volt.

4.3 RÉSZLETES ADATMODELL DIAGRAM, NORMALIZÁLÁS MAGYARÁZATA



4.3.1 Ábra

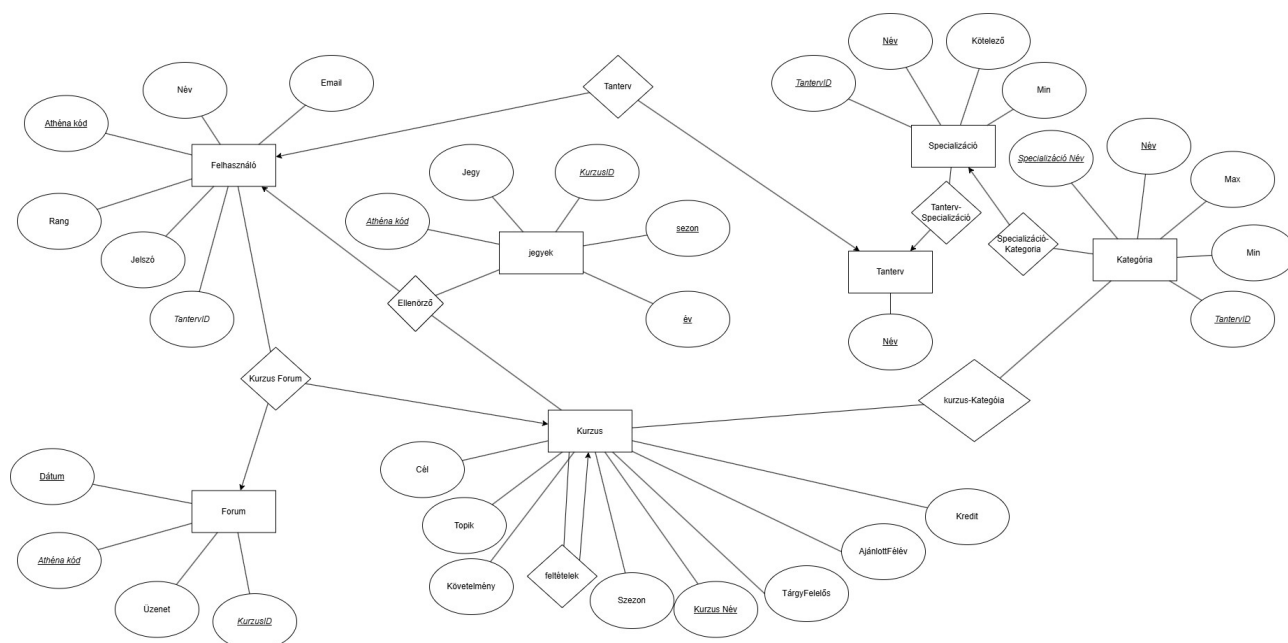
Az alakhamzásom Egyed kapcsolat diagramja [4.3.1]

Ahogy az a [4.3.1]-es ábrán látható, összesen körülbelül 10 egyedem van, amelyek közül kettő kapcsolat tábla. A gyakorlatban ez kiegészül a Laravel saját belső működéséhez generált további 9 táblával, így összesen 19 tábla található az adatbázisban

Az is jól látszik, hogy az adatbázis jelenleg nincs normalizálva. Ennek oka, hogy a projekt felénél tértem át Firebase-ről Laravelre, és mivel a fejlesztés során ismerkedtem meg igazán a Laravel működésével, több technikai akadályba ütköztem, amelyek miatt nem tudtam teljesen normalizált adatbázist kialakítani.

Először is, a Laravel minden tábla létrehozásakor automatikusan hozzáad egy id mezőt elsődleges kulcsként, valamint egy created_at és egy updated_at mezőt (ezek az ábrán nem jelennek meg). Ezt csak a felhasználó egyednél módosítottam, amikor az automatikusan generált id helyett az Athéna kódot használtam elsődleges kulcsként. Ez azonban sok bonyodalmat okozott, mivel a módosításom ellentmondott annak a sémának, amit a Laravel alapértelmezésben feltételez. Hasonló okokból nem tudtam eltávolítani a dátum mezőket sem, mert a Laravel hibát jelez, ha nincsenek jelen a created_at és updated_at mezők. Emellett nem sikerült rájönnöm hogyan kell beállítani több elsődleges kulcsot egy helyett.

Normalizált Adatbázis [4.3.2]



4.3.2 Ábra

Ahogy az ábrán [4.3.2] látszik, a korábbi 10 egyedet 7-re csökkentettem. Ez egyrészt azért lehetséges, mert a korábbi két kapcsoló táblát normalizálás után már kapcsolatként ábrázolom, másrészt mert a Tárgytematika egyedet egyesítem a Kurzus egyeddel. Menjünk végig a folyamaton részletesen:

Felhasználó(Athéna kód, email, jelszó, rang, név, *tantervId*)

A Felhasználó már 3NF-ben volt, ezen nincs mit változtatni. Minden tulajdonsága egyértelműen függ az egyedi Athéna kódtól (az Athéna kód egy általam generált, ötjegyű azonosító, amely véletlenszerű nagybetűkből és 1-4 számból áll). Emellett minden tulajdonsága atomi. Transzitivitás nincs, mivel semmilyen tulajdonságtól nem függ más tulajdonság. Alternatív kulcs lehetne a szintén

egyedi email, de én a rövidebb és gyakrabban használt Athéna kód mellett maradok. (A tantervid-t előzetesen kicserélem névre)

Felhasználó(Athéna kód, email, jelszó, rang, név, *tantervNév*)

Jegy(JegyId, Athéna kód, KuzusId, év, szazon, jegy)

Itt is el lehet mondani, hogy minden tulajdonság atomi, de ezen lehet egyszerűsíteni. Ugyanis egy jegyet egyértelműen meghatároz a diák, kurzus, év, szazon négyes, azaz a JegyId elhagyható, és helyette alkalmazható a korábbi négy tulajdonság összetett kulcsként: (A Kurzusid-t előzetesen kicserélem névre)

Jegy(Athéna kód, KuzusNév, év, szazon, jegy)

Tantern(TanternId, név)

Ha a tantern nevtől megkövetelem az egyediséget – márpedig miért ne lehetne egy tantern neve egyedi, akkor a TernernId elhagyható. Mivel már csak egy mező marad, az biztosan 3NF-ben van:

Tantern(név)

Specializáció(SpecializációId, *TanternId*, név, min, kötelező)

A tulajdonságok itt is atomiak, viszont a korábbi lépésben eltüntettem a TernernId-t, így azt lecserélem a tantern névére. Itt is elmondható, hogy a SpecializációId elhagyható, mert bár itt nem tudom megkövetelni, hogy a specializáció neve egyedi legyen (mivel több tantervben lehet ugyanolyan nevű specializáció), de egy tanterven belül egy specializáció név már csak egyszer fordulhat elő, így ők ketten már használhatóak összetett kulcsként:

Specializáció(*TanternNév*, név, min, kötelező)

Kategória(KategóriaId, *SpecializációId*, név, min, max)

Itt ugyanezt kell megtennünk, mint korábban: elhagyom a KategóriaId-t, a SpecializációId-t lecserélem az új összetett kulcsra, és egy új összetett kulcsot alkotok a kategória névével.

Kategória(*TanternNév*, *SpecializációNév*, név, min, max)

Tárgyteamatika(TárgytematikaId, KurzusId, Cél, topik, követelmény)

A kurzus és a tárgytematika között 1:1 kapcsolat van, ebből következik, hogy a kurzus, azaz a KurzusId egyértelműen meghatározza a tárgytematika tulajdonságait, így ez összevonható a kurzussal. Ezt meg is teszem, így ez az egyed megszűnik.

Kurzus(KurzusId, név, szazon, *Tárgyfelelős*, ajánlott félév, kredit)

Már rögtön azzal kezdem, hogy hozzá kell adni a tárgytematika tulajdonságait. Emellett a KurzusId elhagyható, ha a kurzusId lecserélhető a kurzus névére, amennyiben biztosítom, hogy a név egyedi. (A kurzus szazonja nem ugyanaz, mint a jegy szazonja, mert a jegy szazonja boolean, a kurzusé pedig boolean|null. A jegy szazonja a jegy keletkezési szazonja, a kurzus szazonja pedig az, amikor elindítható a kurzus, ami a keresztféléves kurzusok miatt nem elhagyható.)

Kurzus(név, *szezon*, *Tárgyfelelős*, *ajnlott félév*, *kredit*, *cél*, *topik*, *követelmény*)

Forum(ForumId, *Athéna kód*, *KurzusId*, *üzenet*, *dátum*)

A fórum bejegyzést egyértelműen meghatározza a kurzus, amire írják, az író személye és a keletkezés dátuma. Így a ForumId elhagyásával az Athéna kód, KurzusId, dátum hármását teszem összetett elsődleges kulccsá, és lecserélem a KurzusId-t kurzusnévre:

Forum(Athéna kód, KurzusNév, *üzenet*, dátum)

Kurzus-Katedória(KategóriaId, KurzusId, Id)

Előkövetelmény(KurzusId, előkövetelmény, Id)

Mindkettőnél elhagyom az Id-t, amire már nincs szükségem, és frissítem a KategóriaId-t és a KurzusId-t a normalizált egyedben látható új kulcsaikra (az előkövetelmény is KurzusId, így ezt nyilván ott is végrehajtom).

Kurzus-Katedória(KurzusNév, TantervNév, SpecializációNév, KategóriaNév)

Előkövetelmény(Kurzusnév, előkövetelményNév)

Ezzel az adatbázist 3NF-be hoztam, ami bár kevésbé redundáns, de a hosszú összetett kulcsok miatt nem állítanám, hogy optimálisabb lekérdezéseket eredményezne. A normalizált adatokra nincs értelme indexeket létrehoznom, de a valódi adatmodellemre létrehoztam többet is. Egyet a tanterv nevére és egyet a kurzus nevére, ugyanis ezeket szoktam sokszor kilistázni a választómenükben, másra nem láttam értelmét.

5. FONTOSABB KÓDRÉSZEK ISMERTETÉSE

5.1 RESZPONZÍV DESIGN

A reszponzív designt telefonra, tabletre és asztali gépre többféleképpen is meg lehet valósítani. Az egyik lehetőség, hogy minden képernyőmérethez külön layoutot készítünk, de ez rengeteg munkával jár, és nehezen karbantartható. Alternatív megoldás, hogy minden elem méretét százalékos értékekkel adjuk meg, így az elrendezés igazodik a képernyő méretéhez, de ez önmagában nem akadályozza meg, hogy az oldal szétessen, vagy az elemek túl kicsik legyenek, ami használhatatlanná teheti a felületet.

Én nem használtam több különálló layoutot, hanem a reszponzivitást többféle módszer ötvözésével értem el. Először is, ahol csak lehetett, Angular Material UI elemeket használtam, amelyek eleve a reszponzivitás figyelembevételével készültek, így sok esetben önmagukban is elegendőek a skálázhatósághoz. Emellett elsősorban olyan mértékegységeket alkalmaztam, mint a százalék (%) vagy az em, amelyek az elemek méretét a képernyőhöz képest relatívan határozzák meg, így azok automatikusan igazodnak a képernyő méretéhez.

Annak érdekében, hogy kis képernyőn se essen szét az oldal, gyakran használtam a SCSS overflow-y:auto ami a az oldal vertikális görgetését teszi lehetővé túlsordulásakor, és display:flex tulajdonságokat, amelyek automatikusan rendezik az elemeket. Előfordult azonban, hogy ezek sem voltak elegendőek; ilyen esetekben a SCSS @media query-ket és az RxJS BreakPointObserver-ét alkalmaztam. Az előbbivel lehetőségem volt olyan stílusokat megadni, amelyek csak bizonyos képernyőszélesség esetén lépnek életbe, utóbbival pedig figyelni tudtam a képernyő méretét, és ha túl kicsi volt, akkor ennek megfelelően módosítottam a komponenseim állapotát.

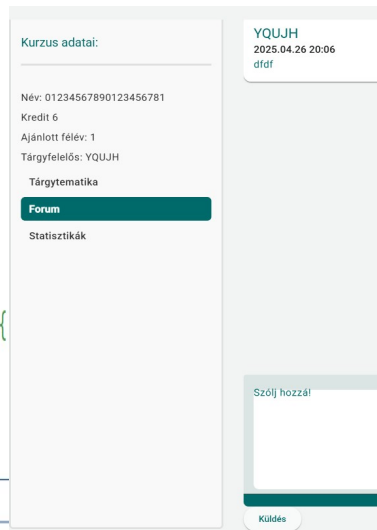
Nem mutatok be minden oldalt részletesen, de két példát kiemelek:

Az egyik esetben a képernyő szélességétől függően @media query-vel állítottam át a display:flex flex-direction értékét sorról oszlopra, hogy a beviteli mezők kis képernyőn is jól láthatóak és könnyen kezelhetőek legyenek [5.1.1] [5.1.2].[5.1.3]

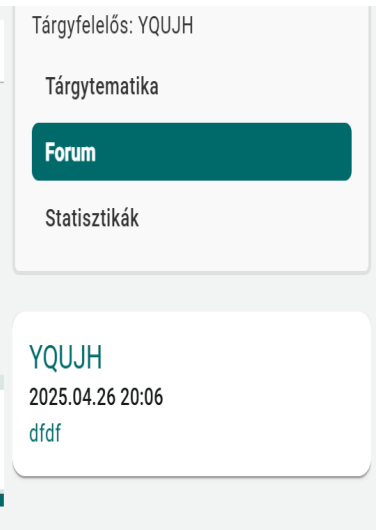

```
#container {
  flex-grow: 1;
  display: flex;
  flex-direction: column;
  overflow-y: auto;
  gap: 1.5rem;

  @media (min-width: $breakpoint-medium) {
    flex-direction: row;
    gap: 2rem;
  }
}
```

5.1.1 Ábra



5.1.2 Ábra



5.1.3 Ábra

Egy másik példában a BreakPointObserver segítségével értem el, hogy a táblázat ne 5, hanem csak 3 oszlopot jelenítsen meg kis képernyőn, mert csak ennyi fér el olvashatóan. Hasonló módon a menü is egy breakpoint hatására alakul át kis képernyőn egy kihúzható (drawer) oldalsó menüvé. [5.1.4][5.1.5][5.1.6]

Név	Módorítás	Töröl	Ellenőrző	Forum
01234567890123456781				
01234567890123456789				
Abt ill. prapertium.				

5.1.4 Ábra

Név	Ellenőrző	Forum
01234567890123456781		
01234567890123456789		

5.1.5 Ábra

```
Breakpoints.XSmall,
Breakpoints.Small
)).pipe(
  takeUntil(this.destroy$)
),subscribe(result => {
  this.isNarrowScreen.set(result.matches);
  if (result.matches) {
    this.displayedColumns = ['name', 'controller', 'view'];
  } else {
    this.displayedColumns = ['name', 'update', 'delete', 'controller', 'view'];
  }
});
```

5.1.6 Ábra

5.2 AUTENTIKÁCIÓS FOLYAMAT LEÍRÁSA

A projektben a Laravel 11 Sanctum csomagját használtam az autentikáció megvalósítására. A Laravel Sanctum egy egyszerű, tokeneken alapuló API autentikációs rendszer, amely kifejezetten

SPA (Single Page Application) alkalmazásokhoz készült. A Sanctum két fő autentikációs módszert támogat:

1. SPA autentikáció Cookie-alapú session-ökkel
2. API Token autentikáció

Én az API Token autentikációs megoldást használtam, ami következőképpen működik:

1. A felhasználó az Athéna kód-jelszó párossal jelentkezik be.
2. A Laravel backend ellenőrzi hogy az Athéna kódhoz tartozó hash-elt jelszó egyezik-e a bejelentkezéskor megadott jelszóval. Ha létezik felhasználó az Athéna kóddal és a hozzá tartozó jelszó egyezik akkor generál egy token, amit vissza küld a kliensnek, és eltárolja a `personal_access_tokens` táblába a Athéna kóddal.
3. A frontend-en elmentem a válaszban kapott token a `localStorage`-ben.
4. Minden további API kéréshez az alkalmazás csatololom a token a kérés fejlécéhez (`Authorization: Bearer {token}`).

A kijelentkezéskor a rendszer érvényteleníti a token a szerveren, a kliens pedig törli a `localStorage`-ből.

5.3 JOGOSULTSÁGI MODELL RÉSZLETES LEÍRÁSA

A szakdolgozatomban 4 szerepkört különböztetem meg. Az admin, tanárt, diákot és a látogatót (be nem jelentkezett felhasználó). A szerepkörök jogosultságait az alábbi jogosultsági mátrix ábrázolja[5.3.1].

	Admin	Tanár	Diák	Látogató
Elektronikus ellenőrző			X	
Jegy beírás	X	X		
Diákoknak szóló statisztikák			X	
Tantervek Kezelése	X			
Tanterv megjelenítés	X	X	X	
Tanterv szűrés	X	X	X	
Optimalizált tanterv generálása	X	X	X	
Belépés				X
Regisztrálás	X			
Profil	X	X	X	
Kijelentkezés	X	X	X	
Kurzus Fórum	X	X	X	
Nyelv átállítása	X	X	X	X
Tanterv választás	X	X		
Kurzusok Kezelése	X	X		

5.3.1 Ábra

A látogatót a többi bejelentkezett felhasználótól backenden a végpontok middleware-rel való levédésével, a frontend oldalon pedig Angular guard-val különböztetem meg.

A már bejelentkezett felhasználók szerepköreit a felhasználó role mezője alapján kezeltem, amit autentikációkor kértem le sikeres bejelentkezés után, majd tároltam el egy BehaviorSubject-ben. A frontenden való jogosultságkezelést a lekért role tulajdonságra épülő feltételekkel és az előbb említett guard segítségével végeztem, ami szintén a role-t veszi alapul.

A backenden kicsit más a helyzet. A Laravel a kliens által küldött érvényes token alapján képes betölteni a felhasználó minden adatát, így a controllerben elsősorban az Illuminate\Support\Facades\Auth segítségével ellenőriztem, hogy a felhasználó be van-e jelentkezve és hogy megvan-e a kellő jogosultsága. Az extra biztonságért létrehoztam néhány Gate policy-t is ami ugyan ezt ellenőrzi, de az idő hiányában ezeket nem alkalmaztam általánosan.

5.4 FRONTEND OPTIMALIZÁLÁSI TECHNIKÁK LEÍRÁSA.

Az Angular keretrendszer számos lehetőséget kínál a frontend alkalmazások teljesítményének optimalizálására. A fejlesztés során az alábbi technikákat alkalmaztam a projekt hatékonyságának növelésére:

Lazy loading használata

A lazy loading (lusta betöltés) segítségével a komponensek csak akkor töltődnek be, amikor ténylegesen szükség van rájuk. Ez jelentősen csökkenti az alkalmazás indulási idejét, mivel a kezdeti betöltéshez csak az alapvető komponensekre van szükség. A routing konfiguráció során a loadComponent opció használatával külön töltöttem be őket aszinkron módon.[5.4.1]

```
export const routes: Routes = [
  {
    path: 'registration',
    loadComponent: () =>
      import('./pages/registration/registration.component').then(
        (c) => c.RegistrationComponent
      ),
    canActivate: [roleGuard],
    data: { roles: ['admin'] }
  },
  {
    path: 'profile',
    loadComponent: () =>
      import('./pages/profile/profile.component').then(
        (c) => c.ProfileComponent
      )
  }
]
```

5.4.1 Ábra

Nem használt csomagok eltávolítása

A fejlesztés során sok felesleges függőségeket telepitem amik az alkalmazás méretét és karbantarthatóságát is negatívan befolyásolják. Az `npm-check` (output:[5.4.2]) parancs segítségével azonosítottam a nem használt npm csomagokat, amelyeket később eltávolítottam a projektből. Ezáltal csökkent a `node_modules` mappa mérete, valamint az alkalmazás build-ideje és mérete is optimalizálódott.

```
@angular/fire MAJOR UP Major update available. https://github.com/angular/angularfire#readme
npm install @angular/fire@19.1.0 --save to go from 18.0.1 to 19.1.0
NOTUSED? Still using @angular/fire?
Depcheck did not find code similar to require('@angular/fire') or import from '@angular/fire'
Check your code before removing as depcheck isn't able to foresee all ways dependencies can
Use rc file options to remove unused check, but still monitor for outdated version:
.npmcheckrc {"depcheck": {"ignoreMatches": ["@angular/fire"]}}
Use --skip-unused to skip this check.
To remove this package: npm uninstall @angular/fire --save

@angular/forms MAJOR UP Major update available. https://github.com/angular/angular#readme
npm install @angular/forms@19.2.9 --save to go from 18.2.13 to 19.2.9

@angular/material MAJOR UP Major update available. https://github.com/angular/components#readme
npm install @angular/material@19.2.14 --save to go from 18.2.14 to 19.2.14

@angular/platform-browser MAJOR UP Major update available. https://github.com/angular/angular#readme
npm install @angular/platform-browser@19.2.9 --save to go from 18.2.13 to 19.2.9

@angular/platform-browser-dynamic MAJOR UP Major update available. https://github.com/angular/angular#readme
npm install @angular/platform-browser-dynamic@19.2.9 --save to go from 18.2.13 to 19.2.9
NOTUSED? Still using @angular/platform-browser-dynamic?
Depcheck did not find code similar to require('@angular/platform-browser-dynamic') or import
from '@angular/platform-browser-dynamic'.
Check your code before removing as depcheck isn't able to foresee all ways dependencies can
```

5.4.2 Ábra

Csomagfrissítések kezelése

Az `npm-check -u` parancs egy interaktív felületen keresztül ([5.4.3]) lehetőséget biztosított az elavult csomagok egyszerű frissítésére. Annak érdekében, hogy a frissítések ne okozzanak kompatibilitási problémát vagy a program összeomlását, kizárólag minor verziófrissítéseket hajtottam végre. Ennek ellenére előfordult hiba, például amikor a `@angular/compiler` 18.2-es verziója már nem támogatta a TypeScript 5.5.3-ról 5.8.3-as verziójára való frissítést hiába minor update-ről volt szó. Ebben az esetben vissza kellett térnem egy korábbi, kompatibilis TypeScript verzióhoz a megfelelő működés érdekében.

```

Non-Semver Versions less than 1.0.0, caution.
( ) zone.js 0.14.10 > 0.15.0 https://github.com/angular

Space to select. Enter to start upgrading. Control-C to car

Minor Update New backwards-compatible features.
(*) @types/node devDep 22.14.1 > 22.15.3 https://github.com
(*) jasmine-core devDep 5.2.0 > 5.7.1 https://jasmine
>(*) typescript devDep 5.5.4 > 5.8.3 https://www.t

Major Update Potentially breaking API changes. Use caution.
( ) @angular/animations 18.2.13 > 19.2.
( ) @angular/cdk 18.2.14 > 19.2.
( ) @angular/common 18.2.13 > 19.2.
( ) @angular/compiler 18.2.13 > 19.2.
( ) @angular/core 18.2.13 > 19.2.
( ) @angular/forms 18.2.13 > 19.2.
( ) @angular/material 18.2.14 > 19.2.
( ) @angular/platform-browser 18.2.13 > 19.2.
(Move up and down to reveal more choices)

```

5.4.3 Ábra

Production build optimalizálás

Az alkalmazás végleges buildjének optimalizálását az Angular CLI beépített eszközeivel végeztem. A `ng build --configuration production` parancs használatával a következő optimalizálások történnek automatikusan:

- A kód minifikálása
- A nem használt változók eltávolítása (dead code elimination)
- Tree-shaking (nem használt exportok kiszűrése)
- A build fájlok méretének csökkentése gzip/uglify használatával

Ez a lépés elengedhetetlen a gyors betöltési idő és az alacsony hálózati erőforrás-felhasználás biztosításához a végfelhasználók számára.

5.5 KÓD DOKUMENTÁLÁSA

A kódomat fejlesztés közben igyekeztem speciális, `/** */` típusú kommentekkel ellátni. Ezek a fejlesztői kommentek rövid leírást adnak a program egyes elemeinek működéséről, céljáról, valamint a paramétereikről. Segítségükkel egyértelműsíthető a változók típusa is. Megpróbáltam minden általam létrehozott függvény, service, interface és osztály működését dokumentálni ezekkel a kommentekkel.

Bár ezek a kommentek nem befolyásolják a program futását, támpontot adhatnak az IDE-nek és a fordítónak a változók típusára vonatkozóan, különösen, ha azokat explicit módon adjuk meg. Ez

azonban veszélyes is lehet, mert elkerülhetjük vele a típusellenőrzést, ami helytelen típusmegadás esetén a program összeomlását is okozhatja.

A fejlesztői kommentek alapján bizonyos könyvtárak képesek interaktív HTML dokumentációt generálni. A frontendhez egy kifejezetten Angular specifikus könyvtárat használtam, a Compodoc-ot. Ezt a projekt gyökerében futtatva sikeresen legenerálta a kód dokumentációját.

A backendhez a PHP egyik legismertebb dokumentációgeneráló eszközét, a phpDocumentor/shim-et használtam. A generálás azonban nem ment zökkenőmentesen: egy ponton egy meg nem nevezett tömbváltozó stringként való olvasása hibát okozott, ami miatt a folyamat végtelen ciklusba került. Emiatt manuálisan le kellett állítanom a generálást, majd szűkítettem a feldolgozási tartományt az app mappa tartalmára. Ennek köszönhetően a generálás már sikeresen lefutott.

A generált dokumentációk megtalálhatók a GitHub repozitóriumomban, mind frontend, mind backend esetén (backend: /Dokumentáció/api/ frontend:/Dokumentáció/documentation/). A GitHub azonban nem tudja ezeket közvetlenül megjeleníteni, de letöltés után az index.html fájl böngészőben való megnyitásával a dokumentációk megtekinthetők.

5.6 FELHASZNÁLÓI DOKUMENTÁCIÓS

A könnyebb használhatóság érdekében létrehoztam egy végfelhasználói dokumentációt is. Az ilyen útmutatók elkészítéséhez több formátum közül választhatunk, mint például a széles körben elterjedt és könnyen szerkeszthető Word, a statikus PDF, vagy az interaktív, de nehezebben elkészíthető és karbantartható HTML.

Én a Markdown formátumot választottam, mivel ez egyszerűen szerkeszthető, jól karbantartható, és a GitHub ahol ez az útmutató elérhető képes stílusosan megjeleníteni azt (/Dokumentáció/README.md).

Az útmutatóban végigmentem az összes oldal funkcióján, és részletesen leírtam a weboldal használatát. A megértést képernyőképekkel is segítettem.

5.7 TÖBBNYELVŰSÉG LÉTREHOZÁSA:

Az applikáció lokalizálásánál (mint például a többnyelvűség biztosítása) gyakran felmerül az i18n (internacionalizáció) és az l10n (lokalizáció) közötti különbség. Míg az i18n elsősorban arra fókuszál, hogy a felhasználók több nyelvet beszélhetnek, és ennek megfelelően a felhasználói felület szövegeit fordíthatóvá teszi, addig az l10n ezen túlmenően figyelembe veszi a helyspecifikus kulturális szokásokat is. Nemcsak a nyelvet fordítja le, hanem alkalmazkodik a kultúrához kötődő szóhasználatához, írásmódhoz, szám- és pénznemformátumhoz, valamint akár az adott ország

ünnepeihez, vallási szokásaihoz is. Bizonyos értelemben tehát az i18n az l10n előkészítő lépése, annak egy belső halmazának is tekinthető.

A weboldal felhasználóbarátabbá tétele érdekében támogattam a többnyelvű megjelenítést is. Az Angular frontendben az ngx-translate nevű könyvtárat használtam, amely az i18n elvet követi.

A HTML-sablonokon belül a translate pipe-ot alkalmaztam a szövegek fordítására, például így: `{{ 'profil.TITLE' | translate }}`. Azokban az esetekben, amikor nem lehetett pipe-ot használni – például a MatSnackBar üzeneteknél vagy programból generált szövegeknél –, a TypeScript kódban a `translate.instant('KEY')` függvényt használtam a fordításra.

A fordítandó szövegek tárolására az assets mappában két külön JSON fájlt hoztam létre:

- hu.json a magyar nyelvű fordításokhoz
- en.json az angol nyelvű fordításokhoz

Mindkét fájlban kulcs-érték párokat használtam. Minden megjelenítendő szöveghez egyedi azonosító (kulcs) tartozik, amit a program mindenhol ezen keresztül hivatkozik. Ez a megközelítés lehetővé teszi a könnyű bővíthetőséget, valamint a fordítások karbantartását.

5.8 ÉRTESÍTÉSI MECHANIZMUSOK

Egy modern webalkalmazás esetében alapvető fontosságú, hogy a rendszer a fontos eseményekről értesítse a felhasználót. Számos értesítési mód létezik, ezek közül a legelterjedtebbek:

- Email értesítés: megbízható és univerzálisan elérhető.
- Push értesítések: gyorsak és közvetlenek, de mobil vagy böngésző-engedély szükséges.
- In-app (alkalmazáson belüli) értesítések: az alkalmazás felületén jelennek meg, de csak akkor hasznosak, ha a felhasználó épp aktív.
- SMS értesítések – azonnaliak, de drágák.

Jelen alkalmazásban a legkézenfekvőbb és legmegbízhatóbb megoldást választottam: email alapú értesítést. Ez minden felhasználó számára elérhető, platformfüggetlen és archiválható, ami a többi opcióról nem biztosan mondható el.

Az értesítési rendszer fő célja, hogy a felhasználókat időben és biztonságosan tájékoztassa a rendszer által generált eseményekről. Az alkalmazásomban jelenleg két eseménytípus esetén történik automatikus értesítés:

1. Regisztráció[5.8.1]

A felhasználó egy megerősítő emailt kap, amely tartalmazza az egyedi Athéna kódját és a jelszavát. Ez azért fontos mert az új felhasználót az admin regisztrálja be és így mind a jelszó mint az Athéna kód generált (tehát ha nem kap emailt senki nem tudja beléptetni)

2. Jegy[5.8.2]

módosításakor:

Amikor egy tanár módosítja egy diák jegyét, a diák emailt kap az új jegyről, így naprakész információval rendelkezik a teljesítményéről.



5.8.2 Ábra

Persze mivel a projektben az összes felhasználót seed-eltem (generáltam) így senkinek sincs valódi e-mail-címe, azért hogy használni lehessen az alkalmazást minden email-t elküldök a saját e-mail-címemre is.

A backend-en az értesítések kezeléséhez a beépített Mail osztályokat használom. Az egyes levéltípusokhoz külön-külön mail osztályokat hoztam létre, amelyek a tartalom formázását Blade-alapú Markdown sablonokon keresztül végzik. Ez lehetővé teszi, hogy az üzenetek strukturáltak, letisztultak és jól olvashatók legyenek.

Az emailküldéshez egy bérelt webtárhelyen található domain (12re.hu) segítségével kapcsolódom az SMTP szerverhez, SSL titkosítással. Az email feladója az athena@12re.hu, amely hivatalos és biztonságos kommunikációs csatornát biztosít.

Értesítési architektúra áttekintése:

- Esemény kiváltása:
Regisztráció vagy jegymódosítás történik az alkalmazásban.

- **Mail osztály meghívása:**
A backend meghívja a megfelelő Mailable osztályt.
- **Tartalom generálás:**
A levél tartalma egy Markdown alapú Blade sablonból készül.
- **SMTP küldés:**
A Laravel a konfigurált SMTP kapcsolaton keresztül (SSL titkosítással) elküldi a levelet a címzettnek.

Ez a megoldás stabil, könnyen bővíthető, valamint külön modulba van szervezve, így a jövőben egyszerűen bővíthető fejleszthető.

5.9 OPTIMALIZÁCIÓ

A Cél

Az optimalizáció funkció célja, hogy segítsen a felhasználónak optimalizálni a választott tanterv teljesítését (a tanterv kiválasztása diák felhasználónál nem opcionális mivel minden diákhoz hozzá van rendelve egy tanterv). Az optimalizálás arra törekszik, hogy a lehető legkevesebb kurzus felvételével, állítson össze egy érvényes tantervi hálót, figyelembe véve a felhasználó egyéni igényeit és a tantervi szabályokat.

A Tanterv Működése és Szabályai

- **Specializációk:** Egy tantervhez több specializáció tartozhat. Ezek közül egy kötelező (ezt mindenképpen teljesíteni kell) és választhatók. A felhasználó kiválaszthatja, mely specializációkat szeretné teljesíteni (a kötelezőn felül).
- **Kategóriák:** Egy specializáció több kategóriából áll (pl. egy Szoftverfejlesztési specializációhoz tartozhat egy matematika és informatika kategóriák). Egy specializáció akkor tekinthető teljesítettnek, ha minden hozzá tartozó kategória követelményei teljesülnek.
- **Kategória Teljesítése:** Egy kategória akkor teljesül, ha a felhasználó a kategóriához tartozó kurzusokból összegyűjti a kategóriára előírt minimális kreditmennyiséget.
- **Kurzusok Felvétele:** Egy kurzust a tervbe csak akkor lehet beilleszteni egy adott félévbe, ha:
 - Minden előfeltételként megjelölt kurzus már szerepel a terv korábbi féléveiben (vagy a felhasználó előzményeiben).
 - A kurzus meghirdetésre kerül az adott félév szezonjában (ősz vagy tavasz).
 - A kurzus még nem szerepel a tervben vagy az előzményekben.
 - A félévre megadott maximális kreditkeretet nem lépi túl a kurzus hozzáadásával.

- Ajánlott félév figyelembe vétele (opcionális): Az aktuális félév sorszáma eléri vagy meghaladja a kurzushoz ajánlott félévét.
- Kurzus Teljesítése (Valóságban vs. Tervben): A valóságban egy kurzus akkor teljesített, ha a hallgató érvényes (nem elégtelen) jegyet szerez rá. Az optimalizálás során tervet készítünk: a tervben szereplő kurzusok azt jelzik, hogy mit és mikor kellene teljesíteni az optimális előrehaladás érdekében. Maga az optimalizáló nem foglalkozik a jegyekkel, csak a kurzusok sikeres felvételével és a kreditek gyűjtésével.

Felhasználói Beállítások és Előzmények

Az optimalizálás indításakor a felhasználó megadja:

- A választott tantervet.
- A teljesíteni kívánt specializáció(ka)t (a kötelező automatikusan hozzáadódik).
- A félévenként felvenni kívánt maximális kredit számát.
- Azokat a kurzusokat, amelyeket mindenképpen szeretne beilleszteni a tervbe (pozitív lista).
- Azokat a kurzusokat, amelyeket semmiképpen nem szeretne felvenni (negatív lista).
- Egy kapcsolót, hogy az algoritmus vegye-e figyelembe a kurzusokhoz tartozó ajánlott félévet.
- Ha a felhasználó már rendelkezik korábban teljesített kurzusokkal (előzmények), a rendszer ezeket automatikusan figyelembe veszi. Azaz a teljesített félévek részét képezik a tervnek. (mivel kurzust csak diák vehet fel ezért ez csak a diáknál lehetséges)

Az Optimalizálás Eredménye

Az optimalizálás végén a felhasználó kap egy optimalizált tantervi hálót ez félévekre bontva mutatja, hogy mely kurzusokat mikor javasolt felvenni, egy kimutatást ez specializációnként és kategóriánként összegzi a megszerzett krediteket és a teljesítettséget, vagy, ha nem sikerült érvényes vagy optimális tervet találni a megadott feltételekkel, akkor hibaüzeneteket kap, amelyek jelzik a problémát.

Az Optimalizáló Algoritmusok

Az optimalizáláshoz két különböző algoritmus áll rendelkezésre, amelyek közül a felhasználó választhat egy Mohó algoritmus és egy Elágazás és Korlátozás (Branch and Bound) algoritmus között.

A Mohó Algoritmus Működése

A mohó algoritmus célja, hogy gyorsan találjon egy jó, de nem feltétlenül a legjobb megoldást.

Lépései a következők:

- Inicializálás: Az algoritmus feldolgozza a tanterv adatait és a felhasználói beállításokat.
- Előzmények Feldolgozása: Ha vannak előzmények, azokat beépíti a terv kezdeti állapotába (teljesített kurzusok, megszerzett kreditek, utolsó félév meghatározása).
- Alapvető Ellenőrzések: Mielőtt nekikezdene, ellenőrzi a nyilvánvaló ellentmondásokat: Van-e olyan kurzus, ami egyszerre kötelező (pozitív) és tiltott (negatív)? Van-e olyan kötelező kurzus, amelynek valamelyik előfeltétele tiltott? A tiltott kurzusok eltávolítása után a maradék kurzusok kreditjei elméletileg elegendőek-e a kategóriák minimum követelményeihez? Ha itt probléma van, azonnal hibával leáll.
- Kurzusválasztás:
 - Megnézi, minden féléven belül hogy van-e olyan kötelező (pozitív) kurzus, amelynek minden feltétele teljesül az aktuális félévben, és még belefér a kreditlimitbe. Ha igen, hozzáadja a félévhez. Ezt addig ismétli, amíg van ilyen felvehető kötelező kurzus.
 - Ezután megvizsgálja a többi, még nem teljesített, releváns kurzust. Kiszámol egy "hatékonysági" mutatót minden kurzushoz (a kurzus kreditje szorozva azzal, hogy hány, még nem teljesített kategóriához járul hozzá). A kurzusokat e mutató szerint csökkenő sorrendbe rendezi.
 - Végigmegy a rendezett listán, és minden kurzust megpróbál hozzáadni az aktuális félévhez, ha annak minden felvételi feltétele teljesül, és a kurzus még releváns (hozzájárul legalább egy, még nem teljesített kategóriához), akkor hozzáadásra jelöli.
 - Amikor már nem tud több kurzust hozzáadni az aktuális félévhez (mert elérte a kreditlimitet, vagy elfogytak a felvehető kurzusok), lezárja a félévet, rögzíti a felvett kurzusokat, és a következő félévre lép.
- A ciklus addig fut, amíg minden kiválasztott specializáció minden kategóriája teljesül, és minden kötelező (pozitív) kurzus bekerült a tervbe, vagy eléri a beállított időkorlátot (59 másodperc), vagy két egymást követő félévben sem sikerül egyetlen kurzust sem hozzáadni. Ez azt jelzi, hogy se ősszel se tavasszal nincsen felvehető kurzus ezért nincs értelme tovább folytatni. Különleges eset, ha az ajánlott félévet figyelembe vesszük: ilyenkor az algoritmus "várhat", ha látja, hogy van olyan kurzus, ami csak egy későbbi ajánlott félév miatt nem felvehető most, ekkor addig szabadon lépek a következő félévbe az üres félévek számának figyelmen kívül hagyása mellett amíg nem marad olyan kötelező vagy releváns kurzus aminek az ajánlott félévét már meg nem haladtam.

Az Elágazás és Korlátozás (Branch and Bound) Algoritmus Működése

Ez az algoritmus arra törekszik, hogy megtalálja a biztosan optimális megoldást (a lehető legkevesebb kurzussal teljesíthető tervet), de ez általában több időt vesz igénybe, mint a mohó megközelítés.

- **Inicializálás:** Hasonlóan a mohóhoz, feldolgozza a tantervet és a felhasználói beállításokat. Létrehozza a szükséges belső adatstruktúrákat (kurzusok listája, előfeltételek, kategóriákhoz tartozás). Kiszámolja a kurzusok hatékonysági mutatóját is (bár itt nem ez alapján történik az optimalizálás). Külön kezeli a kötelező (pozitív) és tiltott (negatív) kurzusokat. Fontos lépés itt, hogy a negatív listát kiterjeszti: nemcsak a direkt megadott kurzusok lesznek tiltottak, hanem rekurzívan azok is, amelyeknek egy tiltott kurzus az előfeltétele.(ez a kiegészítés nem szükséges mert nem tudná felvenni a kurzusokat egyébként se ezért sincs benne a mohóban, de az idő korlát miatt gyorsítanom kellett és így könnyebb felismerni leheleten eseteket)
- **Megoldhatóság Ellenőrzése:** Az optimalizálás megkezdése előtt egy ellenőrzést végez, hogy a probléma egyáltalán megoldható-e:
 - Ütközés a pozitív és (kiterjesztett) negatív listák között.
 - Pozitív kurzus előfeltétele szerepel-e a negatív listán.
 - A releváns, nem tiltott kurzusok kreditjei elegendőek-e kategóriánként a minimum követelmények teljesítéséhez. Ha bármelyik feltétel sérül, az algoritmus hibával leáll, mert a probléma a megadott feltételekkel biztosan megoldhatatlan.
- **Kezdő Állapot:** Definiál egy kiindulási állapotot a kereséshez. Ez tartalmazza, hogy mely kurzusok vannak eddig kiválasztva (kezdetben üres, vagy az előzményekkel feltöltött), melyek teljesítve (szintén üres vagy előzmények), mennyi kredit gyűlt össze kategóriánként (kezdetben nulla vagy előzmények alapján), hanyadik félévénél tartunk (1. vagy az előzmények utáni), és milyen szezon van (ősz vagy tavaszi).
- **Előzmények Feldolgozása:** Ha vannak előzmények, azokat beépíti a kezdő állapotba, frissítve a teljesített kurzusokat, a kategória krediteket, és beállítva a kezdő félévet és szezont.
- **Rekurzív Keresés:** Az algoritmus egy rekurzív függvény segítségével építi fel a lehetséges megoldásokat, mint egy döntési fát. Minden lépésben az aktuális állapotból indul ki, hogy ne járja be ugyanazt az állapotot (ugyanazok a kurzusok felvéve, ugyanabban a félévben) többször, és elkerülje a végtelen ciklusokat, egyedi azonosítót (hash) generál minden állapotból, és eltárolja a már meglátogatottakat. Ha egy már látott állapotba ér, azon az ágon nem megy tovább. Minden lépésben ellenőrzi, hogy nem lépte-e túl az időkorlátot.

Ha igen, az adott keresési ágot leállítja. Megvizsgálja, hogy az aktuális állapot egy teljes értékű, érvényes megoldás-e (minden kategória követelmény teljesül, minden pozitív kurzus felvéve). Ha igen, összehasonlítja a megoldás "költségét" (a felvett kurzusok számát) az eddig talált legjobb megoldás költségével. Ha ez a mostani megoldás jobb (kevesebb kurzusból áll), akkor eltárolja ezt mint az új legjobb megoldást, és frissíti a legjobb költséget. Ezen az ágon nem kell tovább keresni, mert itt már találtunk egy (potenciálisan) optimális végeredményt.

- **Korlátozás (Bounding / Metszés):** Ez az algoritmus kulcslépése. Az algoritmus kiszámol egy alsó becslést arra, hogy az aktuális állapotból kiindulva legalább hány további kurzust kell még felvenni a cél eléréséhez. Ez a becslés figyelembe veszi a még hiányzó kötelező kurzusokat és a kategóriákban hiányzó krediteket (utóbbihoz a leghatékonyabb kurzusokat használva becsül). Ha az aktuális állapotban lévő kurzusok száma plusz ez a becsült minimális további kurzusszám már most eléri vagy meghaladja az eddig talált legjobb megoldás kurzusszámát, akkor felesleges tovább vizsgálni ezt az ágot. Innen már biztosan nem fogunk jobb megoldást találni. Ilyenkor az algoritmus "levágja" ezt az ágot, és nem megy tovább rajta.
- **Elágazás (Branching):** Ha az ágot nem vágtuk le, az algoritmus új állapotokat hoz létre a lehetséges következő lépésekkel:
 - **Kurzus Felvétele:** Megvizsgálja az aktuális félévben felvehető kurzusokat. Minden ilyen kurzus felvételével létrehoz egy új állapotot, és rekurzívan meghívja önmagát erre az új állapotra.
 - **Következő Félévre Lépés:** Létrehoz egy új állapotot, amely a következő félévre lépést reprezentálja (lezárja az aktuális félévet, a benne felvett kurzusokat "teljesítettnek" tekinti a további előfeltételekhez, nullázza a félév kreditjeit, lépteti a félévszámlálót, vált szezont). Erre az új állapotra is rekurzívan meghívja önmagát.
- **Eredmény:** Miután a rekurzív keresés befejeződött, az algoritmus a legjobbként eltárolt megoldást adja vissza, formázva (félévekre bontva, üres félévekkel kiegészítve, ha szükséges), vagy hibaüzenetet, ha nem talált megoldást.

Összegzés

Mindkét algoritmus célja a tanterv optimalizálása, de más stratégiával:

- A Mohó gyorsan ad egy működőképes tervet, ami általában jó, de nem garantáltan a legkevesebb kurzust tartalmazza.

- Az Elágazás és Korlátozás módszeresen keresi a bizonyítottan legjobb megoldást de ez időigényesebb lehet, és összetett esetekben elérheti az időkorlátot, mielőtt megtalálná a tökéletes optimumot.

5.10 STATISZTIKÁK, ADAT VIZUALIZÁCIÓK

Az alkalmazás fontos célja, hogy ne csak adatokat tároljon, hanem azokból hasznos következtetéseket is lehessen levonni. Ennek érdekében különféle statisztikai megjelenítéseket építettem be, amelyek segítik a felhasználókat (különösen a tanárokat) az oktatási eredmények elemzésében. A diagramokat a Plotly.js könyvtár segítségével készítettem, amely interaktív ábrázolást tesz lehetővé.

A statisztikák két fő csoportba sorolhatók: diákokra vonatkozó és kurzusokra vonatkozó elemzésekre.

Diákokra vonatkozó statisztikák

1. Vonaldiagram

Ez a diagram megmutatja az egyes diákok tanulmányi átlagának időbeli változását. Célja, hogy a felhasználó könnyen felismerje a tanulmányi előmenetelének alakulását.

2. Összesített vonaldiagram

Az összes diák tanulmányi átlagát ábrázolja félévenként. Ez a grafikon elsősorban arra szolgál, hogy a diákok össze tudják hasonlítani az eredményüket a középmezőnnyel.

3. Lineáris regresszió

A lineáris regresszió egy olyan statisztikai módszer, amely egyenes vonallal próbálja megközelíteni az adatok közti kapcsolatot. Jelen esetben azt vizsgálom vele, hogy van-e összefüggés a félévek előrehaladása és a tanulmányi átlag változása között. Ha az egyenes emelkedő, akkor az átlagos teljesítmény javul, ha csökkenő, akkor romló tendenciát mutat.

4. Kártyák

A diák tantervéhez tartozó összes specializációt a hozzá tartozó kategóriákkal ábrázolom kis kártyákkal, amiken folyamjelző sávok jelölik hol tart a felhasználó az egyes kategória vagy specializáció teljesítésében. Így egy átlátható formában követheti nyomon az előrehaladását.

Meg jegyném, hogy ezeket a kártyákat nem Plotly.js-vel állítom elő mert a könyvtár erre nem adott lehetőséget hanem SCSS-vel formázott div-vel

Kurzusokra vonatkozó statisztikák:

1. Lineáris regresszió

Hasonlóan az előzőhöz, itt azt vizsgálom, hogy az adott kurzuson félévről félévre hogyan alakulnak de itt a jegyek átlagát veszem és nem a tanulmányi átlagot. Ez segíthet annak megértésében, hogy a kurzus könnyebbé vagy nehezebbé vált-e az idő során.

2. Boxplot

A boxplot (magyarul dobozdiagram) az adatok eloszlását, mediánját, kvartiliseit és szélsőértékeit jeleníti meg egyetlen ábrán. Segít megmutatni, hogy a jegyek hogyan szóródnak, vannak-e kiugró (szokatlanul magas vagy alacsony) értékek, és mennyire „sűrűn” helyezkednek el az adatok egy adott tartományban. Én minden félévhez külön Boxplot-t generálok

3. Kördiagramok

A kördiagramot azt hiszem nem kel bemutatni az adatok százalékos arányát mutatja egy kör formájában. Én két kördiagramot használok egyet a jegyek megoszlásának, és egyet a bukási ráta ábrázolásra

4. Összetett statisztikai diagram

Ez a speciális diagram három elemet kombinál:

- Oszlopdiagram: a kurzuson kiosztott jegyek gyakoriságát mutatja.
- Számított normál eloszlás görbe: a valós adatokra illesztett eloszlásgörbe, amely megmutatja, hogyan oszlanak el ténylegesen az értékek.
- Ideális normál eloszlás görbe: szimmetrikus haranggörbe, amelynél a középérték a várható érték azaz a 3, és a szórás az adatpontok átlagos eltérése 1. Ez a görbét azért tartom ideálisnak mert feltételezem, hogy a diákok teljesítménye természetes adat lévén normál eloszlást követ és ha csak nem maga a kurzus nehéz vagy könnyű a jegyek megoszlásának ennek kéne legyen.

A normál eloszlás egy klasszikus statisztikai modell, amely sok természetes jelenség (pl. testmagasság, IQ) eloszlását jellemzi. Ideálisan a középérték körül szimmetrikusan

helyezkednek el az adatok. Ha az adatgörbe balra tolódik (pl. sok egyes), akkor valószínűleg nehéz a kurzus; ha jobbra tolódik (sok ötös), akkor könnyű. A diagram így vizuálisan segít megállapítani, hogy a kurzus eredményei mennyire „normálisak”, és mennyire térnek el az ideális eloszlástól.

Tantervi háló vizualizáció

A tanterv optimalizációját is ábrázolnom kell valahogy, ehhez egy saját, SVG-alapú tantervhálót generálok. Az SVG HTML tagekből, így könnyen beágyazható az alkalmazásba.

A tantervhálóban a félévenként felveendő kurzusokat helyezem el, maximum négy kurzus magas oszlopokba. Minden kurzust egy ovális alak jelöl, amelyben szerepel a kurzus neve és kreditértéke. A kurzusok felett egy számegyenes jelzi, hogy melyik félévhez tartoznak, illetve azt is, hogy az adott félévhez összesen hány kredit tartozik. A háló jobb szélén egy plusz oszlopban egy összegző doboz is található, amelyben látható, hogy összesen hány kurzust és hány kreditet kell elvégezni, valamint van egy letöltőgomb is, amellyel le lehet tölteni az SVG-t PNG formátumban.

Az SVG összeállítása elég nehéz volt tulajdonképpen nem is sikerült elsőre, ezért az első tantervhálót a draw.io-val rajzoltam meg, és SVG formátumban töltöttem le. A Visual Studio Code egyik bővítményét használva bele tudtam nézni a kép SVG-kódjába, ami első ránézésre egy több oldalas, értelmezhetetlen, hash-elt szövegnek tűnt. Tudtam azonban, hogy a böngésző nem értelmezhet hash-elt szöveget SVG kódként, így biztos voltam benne, hogy csak kiegészítették az SVG-kódot felesleges, véletlenszerű karakterekkel, hogy összezavarjanak. Ezért eltávolítottam a felesleges "hash"-t.

Ezután már csak HTML-tagek maradtak, viszont a fájl tele volt irreleváns elemekkel, például `<image>` tagekkel, és sok SVG-tag többszörözve is szerepelt. Ezeket letisztítottam, így végül sikerült visszanyernem egy olyan SVG-t, amely megfelelt a dokumentációkban olvasott példáknak. Innen már próbálkozással sikerült a frontend oldalon ciklusokkal és függvényekkel dinamikusan generálnom az optimalizált tanterv alapján az egész ábrát.

Ezen felül a korábban említett kártyás ábrákat is használom annak szemléltetésére, hogy az optimalizált tanterv hány kreditet ér el specializációként.

Ezek a statisztikai megjelenítések nemcsak a felhasználói élményt növelik, hanem valódi döntéstámogató szerepet is betöltenek. Lehetővé teszik a teljesítmények, előmenetel, nehézségi szint és haladási mintázatok objektív és átlátható értékelését.

TAPASZTALATOK, TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

Tapasztalatok

Ez volt az első alkalom, hogy a teljes fejlesztési folyamatot az elejétől a végéig önállóan vittem végig, és ennek során rendkívül sokat tanultam. A projekt keretein belül mélyebben megismertem a Laravel keretrendszer működését, a dokumentációkészítés jelentőségét és gyakorlatát, valamint a RxJS által biztosított aszinkron hívások kezelését is. Az optimalizálási algoritmusok tervezése és alkalmazása, valamint az SVG-elemek és diagramok generálása szintén új kihívásokat jelentett számomra.

A unit tesztek írása során különösen sokat fejlődtem, mivel korábban kevés tapasztalatom volt ezen a területen. Az egész projekt során folyamatosan tanultam új módszereket, technológiákat, amiket remélem hasznosítani fogok a közel jövőben.

Továbbfejlesztési lehetőségek

A projekt fejlesztése során számos olyan funkciót terveztem, amelyeket az időkorlátok miatt végül ki kellett hagynom. Az egyik legfontosabb ilyen lehetőség a kurzusokhoz kapcsolódó követelmények rögzítése lett volna. Az elképzelésem szerint a tanár megadhatná, hogy az adott kurzushoz milyen dolgozatokat, kell teljesíteni, ezek hány pontot érnek, illetve milyen ponthatárok vagy speciális feltételek (pl. egy minimum pontszám elérése egy adott ZH-ban) szükségesek a sikeres teljesítéshez.

A diák ezeket az információkat az ellenőrzőben követhette volna, így pontos képet kaphatott volna arról, hogy a kurzushoz tartozó követelményekből hol tart, és mik az esélyei a félév végi sikeres teljesítésre. A tanár számára a félév végén az alkalmazás a beállított ponthatárok és feltételek alapján automatikusan javasolhatta volna az érdemjegyet. Ehhez természetesen statisztikai kimutatások is társultak volna.

Szintén kihagyásra került a kurzusok nehézségi szint szerinti osztályozása, amit beépítettem volna a tanterv optimalizációjába is. A kurzus nehézségét harang görbétől való eltéréssel határoztam volna meg, hogy a kurzus átlagának és szórásának veszem a 3-mas átlagtól és 1-es szórástól való különbséget

A tesztelés tekintetében sem tudtam minden tervezett lépést végrehajtani. Bár elkezdtem frontend komponensekhez unit tesztek írást, a célul kitűzött 70%-os kódfedettséget nem sikerült elérnem. A backend oldalon, Laravel környezetben a tesztelést el sem tudtam kezdeni az idő szűkössége miatt.

Szerettem volna emellett statikus elemzést is alkalmazni, valamint kipróbálni az automatizált end-to-end (E2E) tesztelést is.

Egy másik fontos terület, amit későbbi fejlesztés során érdemes lenne megvalósítani, az a CI/CD pipeline kiépítése, amely automatikus buildelést és unit tesztelést is tartalmazna. Emellett nem foglalkoztam a projekt konténerizálásával sem, pedig a fejlesztés skálázhatósága és egyszerű telepíthetősége érdekében ez hasznos lenne.

A felhasználói visszajelzések jelenleg alapvető szintűek; ezek részletesebbé és kontextusérzékenyebbé tétele a felhasználói élményt jelentősen javítaná. Továbbá az adatbázis is lehetne normalizálalni, amivel csökkenthető lenne a redundancia és növelhető az adatkezelés hatékonysága.

IRODALOMJEGYZÉK

- Angular dokumentáció(<https://angular.dev/overview>)
- Egyed kapcsolat diagram alkotás(https://inf.u-szeged.hu/~gnemeth/adatbgyak/exe/EK_diagram/az_egyedkapcsolat_diagram_elemei.html)
- Adatbázis normalizálás (https://www.inf.u-szeged.hu/~gnemeth/adatbgyak/exe/Normalizalas_web/)
- Angular material dokumentáció(<https://material.angular.io/components/categories>)
- Angular tutorial(<https://www.youtube.com/@MonsterlessonsAcademy>)
- B&B php implementációs példa(<https://www.srimax.com/travelling-salesman-problem-using-branch-bound-approach-php/>)
- B&B(https://hu.wikipedia.org/wiki/Elágazás_és_korlátozás)
- laravel tutorial(https://www.youtube.com/watch?v=LmMJB3STuU4&list=PL38wFHH4qYZUXLba1gx1l5r_qqMoVZmKM)
- laravel tutorial(<https://www.youtube.com/watch?v=WumgBzENYYk&t=133s>)
- laravel+Angular autentikáció(<https://www.youtube.com/watch?v=gUihLkfrbvI&t=143s>)
- laravel 11 dokumentáció(<https://laravel.com/docs/11.x>)
- Angular routing(<https://www.youtube.com/watch?v=dT3f0KTdNyA>)
- npm-check(<https://dev.to/michelebitetto/delete-unused-node-modules-and-improves-performance-in-a-minute-3hon>)
- athena icon (https://pngtree.com/freepng/athena-greek-goddess-from-ancient-mythology-female_5143126.html)
- png to svg (<https://www.freeconvert.com/png-to-svg/download>)
- Laravel mailable(<https://mailtrap.io/blog/send-email-in-laravel/>)
- Jelszó generálás(<https://dev.to/yasserelegammal/generate-random-password-in-laravel-4003>)
- I18N és L10N (<https://www.szabilinux.hu/forditasok/gnome-faq/i18n.html>)
- Svg(https://www.w3schools.com/Html/html5_svg.asp)
- PHP(<https://www.w3schools.com/php/default.asp>)

NYILATKOZAT

Alulírott Varga Zoltán programtervező informatikus Bsc szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Intézet Szoftverfejlesztés Tanszékén készítettem, programtervező informatikus Bsc diploma megszerzése érdekében.

Kijelentem, hogy a dolgozatot más szakon korábban nem védtem meg, saját munkám eredménye, és csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy szakdolgozatomat a Szegedi Tudományegyetem Diplomamunka Repozitóriumban tárolja.

Dátum: 2025.05.20

Aláírás

KÖSZÖNETNYILVÁNÍTÁS

Ezúton szeretnék köszönetet mondani azoknak az embereknek, akik segítsége nélkül nem készülhetett volna el a szakdolgozatom.

Köszönöm Dr. Bánhelyi Balázsnak, hogy válaszolt a leveleimre és segített meg találnia megfelelő algoritmust amivel képes vagyok elvégezni az optimalizálást, illetve köszönettel tartozom Varga Zoltánnak, aki szerveret biztosított nekem a weboldalam host-olásához.

Továbbá köszönettel tartozom Kiss Zsuzsanna és Uhrin Beáta végzős szegedi hallgatóknak ugyanis az ő szakdolgozat dokumentációikat használtam a sajátom elkészítésében, bár az Én témám és fejezeteim drasztikusan eltértének az övékétől, de a példájuk segített elindulni az úton

ELEKTRONIKUS MELLÉKLET

A szakdolgozatom github linkje: <https://github.com/urambatyam/Szakdolgozat>

A hostol weboldal: <https://12re.hu/athena/login>

admin felhasználó: Athéna kód: YQUJH jelszó: password

A szakdolgozat működését bemutató videó:

https://drive.google.com/file/d/1hrTC_b5cwFj4OJSmQjMjc4QAeXdzRwV-/view?usp=drive_link

