

---

**UNIVERSITATEA „SAPIENTIA” DIN CLUJ-NAPOCA**  
**FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,**  
**TÎRGU-MUREȘ**  
**SPECIALIZAREA CALCULATOARE**

**APLICAȚIE WEB PENTRU**  
**GESTIONAREA**  
**ACTIVITĂȚILOR DE PRACTICĂ**  
**ȘI INTERNSHIP**  
**PROIECT DE DIPLOMĂ**

**Coordonator științific:**

**Dr. Szabó László Zsolt**

**Absolvent:**

**Nüszl Richárd-István**

**2023**

UNIVERSITATEA „SAPIENTIA” din CLUJ-NAPOCA

Viza facultății:

Facultatea de Științe Tehnice și Umaniste din Târgu Mureș

Specializarea: Calculatoare



## LUCRARE DE DIPLOMĂ

Coordonator științific:

ș.l. dr. ing. Szabó László Zsolt

Candidat: Nüszl Richárd-István

Anul absolvirii: 2023

**a) Tema lucrării de licență:**

Aplicație web pentru gestionarea activităților de practică și internship

**b) Problemele principale tratate:**

- Studiu bibliografic privind sistemele similare
- Studiu bibliografic și implementarea unui algoritm de recomandare adecvat temei
- Realizarea unei aplicații web pentru gestionarea activităților de practică și internship

**c) Desene obligatorii:**

- Schema bloc al aplicației
- Diagrame UML privind software-ul realizat.

**d) Softuri obligatorii:**

- Aplicație web pentru gestionarea activităților de practică și internship
- Implementarea unui algoritm de recomandare

**e) Bibliografia recomandată:**

1. Kim Falk: Practical Recommender Systems, Manning 2019.
2. Lucas da Costa: Testing JavaScript Applications, Manning 2021.
3. V. Seiji: Full-stack web development with Spring Boot and Angular: A practical guide to building your full-stack web application. Packt, 2022.

**f) Termene obligatorii de consultații:** săptămânal

**g) Locul și durata practicii:** Universitatea Sapientia,  
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș

**Primit tema la data de:** 30.04.2022

**Termen de predare:** 27.06.2023

Semnătura Director Departament

Semnătura responsabilului  
programului de studiu

Semnătura coordonatorului

Semnătura candidatului

---

## Declarație

Subsemnatul Nüszl Richárd absolvent(ă) al/a specializării Calculatoare, promoția 2023 cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea,

Data: 27.06.2023.

Semnătura.......... Absolvent

---

# APLICAȚIE WEB PENTRU GESTIONAREA ACTIVITĂȚILOR DE PRACTICĂ ȘI INTERNSHIP

## Extras

Tema proiectului este dezvoltarea unei aplicații web care să conecteze studenții cu companiile și universitățile în timpul stagiilor lor de practică. Scopul este de a dezvolta o platformă ușor de utilizat, prietenoasă cu utilizatorul, capabilă să gestioneze întregul proces legat de stagiile de practică.

Unul dintre principalele obiective ale proiectului este de a funcționa ca o platformă comunitară orientată către profesionalism pentru studenți și companii, promovând astfel dezvoltarea de relații profesionale și menținerea parteneriatelor dintre companii și universități.

Un alt obiectiv important al proiectului este ca platforma să ofere recomandări studenților în funcție de profilul lor și domeniul lor de studiu. Aceasta le permite să găsească oportunități relevante în domeniul stagiilor de practică. Algoritmii sistemului meu de recomandare utilizează metoda de similaritate cosinus pentru a lua în considerare abilitățile, interesele studenților și cerințele stabilite de companii. Astfel, platforma ajută studenții să găsească cele mai potrivite locuri de practică, optimizând procesele și sporind eficiența stagiilor lor.

Pentru dezvoltarea proiectului, folosesc tehnologii web moderne, precum Angular, Java Spring Boot, PostgreSQL și AWS. Astfel am reușit să creez o platformă robustă, scalabilă, sigură și rapidă.

**Cuvinte cheie:** Internship, Sistem de recomandare, Similaritate Cosinus, Angular, Spring Boot

**SAPIENTIA ERDÉLYI MAGYAR  
TUDOMÁNYEGYETEM  
MAROSVÁSÁRHELYI KAR  
SZÁMÍTÁSTECHNIKA SZAK**

**SZAKMAI GYAKORLAT ÉS  
GYAKORNOKI ÁLLÁSOK  
KEZELÉSÉT MEGVALÓSÍTÓ WEB  
ALKALMAZÁS**

**DIPLOMADOLGOZAT**

**Témavezető:**

Dr. Szabó László Zsolt

**Végzős hallgató:**

Nüszl Richárd-István

**2023**

---

# Kivonat

A dolgozat témája egy webalkalmazás fejlesztése, amely a hallgatókat, a cégeket és az egyetemeket a szakmai gyakorlat tevékenysége során köti össze. A cél az, hogy egy könnyen használható, felhasználóbarát platformot fejlesszek, amely képes a szakmai gyakorlatokkal kapcsolatos teljes folyamat lebonyolítására.

A projekt egyik célja, hogy a hallgatók és a cégek számára egy szakmai közösségi platformként működjön az alkalmazás, ezáltal elősegítve a szakmai kapcsolatok kialakulását, valamint a cégek és az egyetemek közötti partneri kapcsolatok ápolását.

A projekt másik fontos célja, hogy a platformon a hallgatók a saját profiljuknak és szakjuknak megfelelően kapjanak ajánlásokat. Ez lehetővé teszi számukra, hogy releváns lehetőségeket találjanak a szakmai gyakorlatok terén. Az ajánlórendszerem algoritmusai, például a Cosinus hasonlóságot felhasználva figyelembe veszik a hallgatók készségeit, érdeklődését és a cégek által meghatározott követelményeket. Így a platform segít a hallgatóknak a legmegfelelőbb gyakorlati helyek megtalálásában, optimalizálva ezzel a kapcsolódó folyamatokat és eredményessé téve a gyakorlatokat.

A projekt elkészítéséhez modern webes technológiákat használok, Angular-t, Java Spring Bootot, PostgreSQL-t és AWS-t. Ezek segítenek a platformomat robusztussá, skálázhatóvá biztonságossá és gyorsá tenni.

**Kulcsszavak:** Internship, Ajánlórendszer, Cosinus hasonlóság, Angular, Spring Boot

---

# WEB APPLICATION FOR MANAGING INTERNSHIP AND TRAINEESHIP ACTIVITIES

## Abstract

The topic of the project is to develop a web application that connects students, companies, and universities during their internships. The goal is to develop a user-friendly platform capable of managing the entire process related to internships.

One of the major objectives of the project is for the application to function as a professional-oriented social platform for students and companies, thereby facilitating the development of professional relationships and nurturing partnerships between companies and universities.

Another important objective of the project is to provide students with recommendations on the platform based on their profiles and fields of study. This allows them to find relevant opportunities in the field of internships. The algorithms of my recommendation system, using techniques like Cosine Similarity, consider the students' skills, interests, and the requirements set by the companies. Thus, the platform helps students find the most suitable internship positions, optimizing the related processes and ensuring the effectiveness of the internships.

To develop the project, I am using modern web technologies such as Angular, Java Spring Boot, PostgreSQL, and AWS. These technologies help make the platform robust, scalable, secure, and fast.

**Keywords:** Internship, Recommendation System, Cosine Similarity, Angular, Spring Boot

## Tartalomjegyzék

1. Bevezető.....	12
1.1. Célkitűzések .....	14
2. Elméleti megalapozás és szakirodalmi tanulmány .....	15
2.1. Szakirodalmi tanulmány .....	15
2.2. Ajánló rendszerekkel kapcsolatos elméleti alapok.....	16
2.2.1. Cosinus hasonlóság.....	16
2.2.2. Jaccard index.....	18
2.3. Ismert hasonló alkalmazások.....	19
2.4. Felhasznált technológiák .....	19
2.4.1. Webes frontend keretrendszerek.....	20
2.4.2. Tesztelési keretrendszer .....	24
2.4.3. Backend.....	24
3. A rendszer specifikációi és architektúrája.....	29
3.1. Felhasználói követelmények .....	29
3.2. Rendszerkövetelmények.....	33
3.2.1. Funkcionális követelmények .....	33
3.2.2. Nem funkcionális követelmények.....	35
4. Részletes tervezés.....	36
4.1. Az architektúra és a felhasznált technológiák .....	36
4.2. Az adatbázis.....	37
4.2.1. Tervezés .....	37
4.2.2. A PostgreSQL adatbázis létrehozása és kezelése .....	37
4.2.3. Az adatbázis szerkezete .....	38
4.3. A kommunikáció .....	39
4.4. Backend .....	40
4.4.1. Konfigurálás.....	40
4.4.2. A felhasználói fiókok biztonsága.....	42
4.4.3. Egy CRUD funkcionalitás megírása .....	43
4.4.4. Ajánló rendszer implementáció .....	46
4.5. Frontend.....	49
4.5.1. Konfigurálás.....	49
4.5.2. Felépítés .....	49
4.5.3. A felhasználói felület .....	50
4.6. Verziókövetés és projektmenedzsment és egyéb fejlesztői eszközök.....	52
4.6.1. A GitHub.....	52
4.6.2. Trello.....	53
4.6.3. Excalidraw .....	54
4.6.4. Canva és Storyset.....	55
4.7. Tesztelés .....	56
5. Üzembe helyezés és kísérleti eredmények .....	57
5.1. Üzembe helyezési lépések .....	57
5.1.1. A frontend .....	57
5.1.2. Backend.....	58
5.2. Felmerült problémák és megoldásaik.....	58
5.3. Az ajánlórendszer tesztelési eredményei.....	59
6. A rendszer felhasználása .....	62



7. Következtetések .....	63
7.1. A rendszer állapota .....	63
7.2. Hasonló rendszerekkel való összehasonlítás .....	63
7.3. Továbbfejlesztési lehetőségek .....	64
7.3.1. A platform.....	64
7.3.2. Az ajánlórendszer.....	64
8. Irodalomjegyzék.....	65
9. Függelék.....	67
9.1. Ajánlórendszer kódja.....	67
9.2. Felhasználói felület.....	69

## Ábrák jegyzéke

1.1. ábra Szakmai nyárigyakorlat illusztráció .....	13
2.1. ábra Adatkötési architektúrák összehasonlítása (saját átdolgozás) .....	23
2.2. ábra Cypress logó .....	24
2.3. ábra Bootstrap logó .....	24
2.4. ábra React logó .....	24
2.5. ábra Angular logó .....	24
2.6. ábra Visual Studio Code logó .....	24
2.7. ábra Spring Boot alkalmazás rétegei közötti kapcsolat (saját átdolgozás) .....	26
2.8. ábra Spring Boot alkalmazás flow architektúrája (saját átdolgozás) .....	27
2.9. ábra IntelliJ logó .....	28
2.10. ábra Liquibase logó .....	28
2.11. ábra PostgreSQL logó .....	28
2.12. ábra Java Spring Boot logó .....	28
3.1. ábra Az applikáció use case diagrammja .....	30
4.1. ábra A rendszer architektúrája .....	36
4.2. ábra Az adatbázis szerkezete a fontosabb kulcsokkal .....	38
4.3. ábra Spring initializr .....	40
4.4. ábra Az alkalmazás backendjének konfigurációs filejai .....	41
4.5. ábra Az alkalmazás backendjének fontosabb könyvtárai .....	43
4.6. ábra Ajánló rendszer működési diagrammja .....	47
4.7. ábra Az ajánlórendszer folyamatábrája .....	48
4.8. ábra A backend repositoryjának dashboardja .....	52
4.9. ábra A frontend repositoryjának dashboardja .....	53
4.10. ábra Trello board és az általam használt címkézések .....	54
4.11. ábra Excalidraw felület .....	54
4.12. ábra Példa az oldalon szereplő egyik kép elkészítésére Canvaban .....	55
4.13. ábra Storyet kereső .....	55
4.14. ábra A szcenárió tesztelés egyik állapota .....	56
5.1. ábra A dist könyvtár a buildelés folyamatát követően .....	57
5.2. ábra TestFirstName TestLastName hallgató képességei .....	59
5.3. ábra Informatika szakhoz akkreditált cégek .....	60

5.4. ábra Az ajánlómotor eredményességének mérése.....	61
9.1. ábra Céges profil .....	69
9.2. ábra Hallgató profiljának beállításai .....	70
9.3. ábra Új pozíció létrehozása .....	70
9.4. ábra Hallgató profilja .....	71

## **Táblázatok jegyzéke**

3.1. Táblázat Funkcionális követelmények .....	35
4.1. Táblázat Állapot jelző kódok .....	39

## **Szómagyarázat**

- A szövegben az angol “skill” szó helyett a magyar megfelelő “képesség” szót fogom használni.
- AWS: Amazon Web Services
- CosSim: Cosine Similarity
- BOW: Bag of Words

# 1. Bevezető

A téma, amit választottam a szakmai nyári gyakorlatok kezelésével foglalkozik, és pedig egy webalkalmazás megtervezésével és fejlesztésével, amely digitalizálja a szakmai nyári gyakorlatok keresésének és elvégzésének folyamatát. Ennek segítségével a diákok könnyedén találhatnak és jelentkezhetnek nyári gyakorlatokra. A platformot Angular keretrendszerrel, Java Spring Boot technológiával és AWS felhőszolgáltatásokkal megvalósítottam meg.

A nyári gyakorlatok a hallgatók számára rendkívül fontosok azért, mert lehetővé teszik számukra, hogy elsajátítsák azokat az ismereteket és készségeket, amelyekre az elhelyezkedésükhöz szükségük van. Azonban az egyetemi vagy főiskolai diákok számára nehéz lehet megtalálni a megfelelő gyakorlatokat. A meglévő platformok is sok esetben túl bonyolultak és nem eléggé felhasználóbarátak, vagy éppen nem biztosítják a diákok számára az összes szükséges információt. Emiatt az a célom, hogy olyan platformot tervezek és fejlesszek, amely könnyen használható és minden szükséges információt tartalmaz a diákok számára.

A platform tervezése során Angular keretrendszert használok, amely egy nyílt forráskódú, egyszerű és hatékony keretrendszer. Az Angular segítségével könnyedén tudom kezelni az oldalak dinamikus változásait, és lehetővé teszi számomra az egyszerű és gyors felhasználói felület tervezését. A Java Spring Boot technológiával lehetőségem lesz egy nagyon hatékony, skálázható és biztonságos rendszer létrehozására. Az AWS szolgáltatások használatával pedig a platformom rendelkezni fog az automatikus skálázással, az adatbázis és tároló rendszerek, valamint az infrastruktúra biztonságának biztosításával.

Összességében tehát, a célom egy olyan platform létrehozása, amely lehetővé teszi a diákok számára a szakmai gyakorlatok egyszerű megtalálását és jelentkezését. Az Angular, Java Spring Boot és AWS használata azért fontos, mert lehetővé teszik számomra, hogy egy biztonságos, hatékony és skálázható platformot hozzak létre.

A platformom a diákok és a cégek számára is előnyös lehet, mivel lehetővé teszi a hatékonyabb kommunikációt és együttműködést. A diákok például könnyen értesülhetnek az új gyakorlati lehetőségekről, és egyszerűen jelentkezhetnek rájuk, míg a cégek könnyedén kereshetnek és kiválaszthatnak a számukra megfelelő jelentkezőket.

A platform használatának másik nagy nyertesei a partner egyetemek. Az egyetemek ugyanis egyre nagyobb hangsúlyt fektetnek a hallgatóik gyakorlatorientált képzésére és a szakmai tapasztalat megszerzésére.

A platform segítségével az egyetemeken könnyen megtalálhatják azokat a cégeket, amelyek a hallgatóik számára releváns gyakorlati lehetőségek állnak rendelkezésre, emellett a platform lehetővé teszi az egyetemeken számára a diákok szakmai fejlődésének követését és értékelését is.

További előny az egyetemeken számára, hogy az összes gyakorlathoz tartozó papírmunka könnyedén elintézhető és átláthatóvá válik számukra, ezáltal pénzt és időt megtakarítva.

A platform az egyetemeken számára, segít javítani a kapcsolatot a vállalatokkal. Az egyetemeken és a cégek közötti együttműködési lehetőségek bővítése hosszú távon elősegítheti a diákok elhelyezkedési esélyeit és az egyetemeken hírnevét is.

Összefoglalva, a szakmai nyári gyakorlatok kezelő platform megtervezése és fejlesztése fontos és hasznos lehet a diákok és a cégek számára is. Az Angular, Java Spring Boot és AWS használata lehetővé teszi a hatékony, biztonságos és skálázható platform kialakítását.

Remélem, hogy a platformom segítségével a diákok könnyedén megtalálják majd a számukra megfelelő gyakorlatokat, és ezzel hozzájárulok a szakmai fejlődésükhöz és elhelyezkedési esélyeik növeléséhez.



*1.1. ábra Szakmai nyárigyakorlat illusztráció*

## 1.1. Célkitűzések

A munka kezdetén az alábbi célkitűzéseket fogalmaztam meg:

1. A hatékony és biztonságos platform megtervezése és fejlesztése Angular, Java Spring Boot használatával.
2. A diákok és a cégek számára egy könnyen használható, felhasználóbarát platform létrehozása, amely lehetővé teszi a hatékonyabb együttműködést.
3. A diákok számára lehetővé tenni, hogy könnyen megtalálják a számukra megfelelő gyakorlati lehetőségeket, és egyszerűen jelentkezzenek rájuk.
4. A cégek számára lehetővé tenni, hogy könnyedén keressenek és kiválasszanak a számukra megfelelő diákokat a szervezett nyári gyakorlatokra.
5. Az egyetemek számára lehetővé tenni, hogy könnyen megtalálják azokat a cégeket, amelyek a hallgatóik számára releváns gyakorlati lehetőségeket kínálnak, ezáltal is javítani az egyetemek kapcsolatát a vállalatokkal.
6. Azonosítani és kiszűrni a platformon belüli esetleges biztonsági kockázatokat, és megfelelő intézkedéseket tenni azok megelőzése érdekében.
7. Folyamatosan frissíteni és fejleszteni a platformot, hogy az a diákok, cégek és egyetemek igényeinek megfeleljen.
8. A platform feltöltése és futtatása AWS-re

## 2. Elméleti megalapozás és szakirodalmi tanulmány

Az applikációmmal az egyik legfontosabb célkitűzésem az, hogy a hallgatók minél pontosabb képet kapjanak arról, hogy milyen állások érhetőek el, szakmai nyári gyakorlatra és hogy azok közül minél fókuszáltabban kapja ajánlásba azokat, amelyek valóban érdeklik vagy valóban látja benne esetleg a jövőjét. Ezért volt számomra fontos, hogy a hallgató a profiljában legyen egy opció, ahol személyre szabott ajánlatokat kaphat állásokról, amik hozzá közel állhatnak.

Ezért körvonalazódott ki számomra az, már az ötlet csiszolása közben, hogy szükség lesz, a rendszerbe integrálni egy ajánlórendszerre.

Tanulmányaim során arra voltam elsősorban kíváncsi, hogy milyen megoldások léteznek egy preferenciákra épülő ajánló rendszer megvalósítására.

Az állásajánlórendszerek a modern munkaerőpiacon kiemelkedő jelentőséggel bírnak, mivel segítségükkel a munkaadók és a munkavállalók hatékonyabban találhatnak egymásra. Ezek az algoritmusok olyan adatelemzési technikákat alkalmaznak, amelyek alapján a rendszer képes összehasonlítani a munkaadók által meghirdetett állásokat a munkavállalók profiljaival, és így ajánlásokat tehet a legmegfelelőbb munkalehetőségekről.

### 2.1. Szakirodalmi tanulmány

Az első könyv, amit olvastam az a **Recommender Systems Handbook** volt, amelyet **Francesco Ricci, Lior Rokach és Bracha Shapira** szerkesztett. A könyv az ajánlórendszerekkel foglalkozik, amelyek olyan technológiák és algoritmusok, amelyek személyre szabott ajánlatokat készítenek a felhasználók számára. A könyv az ajánlórendszerek különböző aspektusait és alkalmazásait tárgyalja, és átfogó képet nyújt a témáról [1].

A második könyv, amelyből tájékozódtam az a **Recommender Systems An Introduction**, amely négy szerző munkája, **Dietmar Jannaché, Markus Zankeré, Alexander Felfernigé és Gerhard Friedriché**. A könyv az ajánlórendszerek alapelveit, módszereit és technikáit tárgyalja. Bemutatja a legfontosabb fogalmakat és technikákat, példákat és esettanulmányokat használva a valós alkalmazásokból. A könyv célja, hogy egy átfogó áttekintést nyújtson az ajánlórendszerek terén, kezdve az alapoktól egészen a fejlettebb módszerekig [2].

A harmadik forrás, amelyből tájékozódtem az nem más, mint a **Combinations of Jaccard with Numerical Measures for Collaborative Filtering Enhancement: Current Work and Future Proposal**, ami **Ali A. Amer, Loc Nguyen** dolgozata. A dolgozat lényege, hogy a Jaccard Indexet és annak kombinálását vizsgálja más, hasonlósági mutatókkal, például a cosinus hasonlósági mutatóval. [3]

A negyedik forrás, amiből ki is indultam később az egy fejtegetés volt az ismertebb állás kereső platformok, mint például a LinkedIn által használt algoritmusokról. Itt is szó esett természetesen a cosinus hasonlósági algoritmusról, de más, például neuronhálós megoldásokról is. A tartalom címe: **Building a Job Recommendation Strategy for LinkedIn and XING - Part1** [4].

Az ötödik forrás a negyedik forrásom egyik folytatása, amely inkább azt elemzi, hogy egy ilyen platform, amelynek a tervezésével foglalkozok mást is szem előtt kell tartson egy adott pozíció ajánlásakor, például a felhasználó által posztolt és kedvelt tartalmakat. A tartalom címe: **Strategizing a Recommendation System on a Jobs Platform | Personalising Jobs Feed for LinkedIn - Part 2** [5].

A legtöbb tanulmány abba az irányba mutatott, hogy egy ilyen típusú platformhoz, amit én is tervezek, a legjobb megoldás egy tartalom alapú ajánló rendszer kiépítése, lehetőleg valami hasonlósági algoritmusra alapozva, esetemben például a hallgató képességei vagy elvárásai és a pozíció elvárásai között.

A hasonlósági vizsgálatok az állásajánlórendszerek alapját képezik. Ezek a vizsgálatok arra törekednek, hogy meghatározzák, mennyire hasonlóak egy adott munkavállaló profilja és egy adott álláshirdetés követelményei. Ezen vizsgálatok során a munkavállaló és az álláshirdetés közötti kapcsolatot matematikai mérőszámok segítségével értékelik.[2]

## **2.2. Ajánló rendszerekkel kapcsolatos elméleti alapok**

### **2.2.1. Cosinus hasonlóság**

A cosinus hasonlóság egy olyan algoritmus, amely a dokumentumok (például álláshirdetések és munkavállalói profilok) vektoros reprezentációját használja fel a hasonlóság mértékének meghatározására. Ez a módszer alapvetően arra épül, hogy a dokumentumokat vektorokként ábrázolja, és kiszámítja azok közötti szöget a vektorok skalárszorzata alapján.



A cosinus hasonlóság algoritmus előnye, hogy hatékonyan működik nagy adathalmazokon is, és a dimenziók számától függetlenül eredményesen alkalmazható. Továbbá, a cosinus hasonlóság értéke 0 és 1 között mozog, így könnyen értelmezhető és összehasonlítható.

A cosinus hasonlóság algoritmus egy matematikai módszer a vektorok közötti hasonlóság mérésére. Az állásajánlórendszerekben a cosinus hasonlóság alapú algoritmusok az álláshirdetések és a munkavállalói profilok vektoros reprezentációját alkalmazzák.

A cosinus hasonlóság algoritmusát a következő képlet segítségével számítják ki [2]:

$$\text{CosSim}(\theta) = (\mathbf{A} \cdot \mathbf{B}) / (\|\mathbf{A}\| \|\mathbf{B}\|)$$

Ahol:

$\mathbf{A}$  és  $\mathbf{B}$  az vektorok

$\mathbf{A} \cdot \mathbf{B}$  a két vektor skaláris szorzata

$\|\mathbf{A}\|$  és  $\|\mathbf{B}\|$  a két vektor hossza (euklideszi normája)

**Példa** cosinus hasonlóság számítására:

legyen  $x = \{3, 2, 0, 5\}$

$y = \{1, 0, 0, 0\}$

$x \cdot y = 3 \cdot 1 + 2 \cdot 0 + 0 \cdot 0 + 5 \cdot 0 = 3$

$\|x\| = \sqrt{(3)^2 + (2)^2 + (0)^2 + (5)^2} = 6.16$

$\|y\| = \sqrt{(1)^2 + (0)^2 + (0)^2 + (0)^2} = 1$

$\text{CosSim}(x, y) = 3 / (6.16 \cdot 1) = 0.49$

### Valós példa egy állásajánló rendszer esetében

Adottak a következő szövegek:

- Hallgató képességei: "programozás", "adatbázis-kezelés", "projektmenedzsment", "kommunikáció"
- Munka 1 elvárásai: "programozás", "adatbázis-kezelés", "proaktív hozzáállás"
- Munka 2 elvárásai: "projektmenedzsment", "kommunikáció", "önálló munkavégzés"
- Munka 3 elvárásai: "kreativitás", "kommunikáció", "proaktív hozzáállás"

Első lépésben elő kell készítenünk a szövegeket, hogy alkalmasak legyenek a cosinus hasonlósági algoritmus használatára. Ehhez vektorokká kell alakítani őket. Egy lehetséges módszer a BOW módszer használata[6], ahol a szövegeket vektorokká alakítjuk, amelyek az egyes szavak jelenlétét/távollétét jelzik.

A hallgató képességei és a munkák elvárásainak  
leképzése vektoros formába

- Hallgató vektorja: [1, 1, 1, 1, 0, 0, 0]
- Munka 1 vektorja: [1, 1, 0, 0, 1, 0, 0]
- Munka 2 vektorja: [0, 0, 1, 1, 0, 1, 0]
- Munka 3 vektorja: [0, 0, 0, 1, 1, 0, 1]

Cosinus hasonlósági eredmények

- Hallgató és Munka1: Cosinus hasonlóság = 0.67
- Hallgató és Munka2: Cosinus hasonlóság = 0.33
- Hallgató és Munka3: Cosinus hasonlóság = 0.33

### 2.2.2. Jaccard index

A másik algoritmus, amit gyakran alkalmaznak a tartalom alapú ajánlórendszerek az a Jaccard index, amelyet a dokumentumok vagy halmazok közötti hasonlóság kimutatására és mérésére használnak [3].

A Jaccard hasonlósági index egy olyan statisztikai mérték, amely a két halmaz közötti hasonlóságot jellemzi. Az index értéke a két halmaz közös elemeinek arányát mutatja a halmazok összes eleméhez viszonyítva. Az index kiszámítása a következőképpen történik:

$$J(A, B) = |A \cap B| / |A \cup B|$$

Ahol:

**A** és **B** két halmaz

**|A|** jelöli A halmaz elemszámát (**cardinalis**)

**∩** jelöli a halmazok metszetét (azokat az elemeket, amelyek mindkét halmazban megtalálhatóak)

**∪** jelöli a halmazok unióját (azokat az elemeket, amelyek legalább az egyik halmazban megtalálhatóak)

A Jaccard-index értéke 0 és 1 között változik. Minél közelebb van az érték 1-hez, annál nagyobb a hasonlóság a két halmaz között. Az érték 0, ha a két halmaz nincs közös elem, vagyis a halmazok teljesen eltérőek.

Az állásajánlórendszerekben a Jaccard hasonlósági indexet alkalmazzák a munkakeresők profiljainak és az álláslehetőségek leírásainak összehasonlítására.

## 2.3. Ismert hasonló alkalmazások

A piacon akad néhány olyan alkalmazás, amely egyhelyre gyűjti a szakmai nyári gyakorlatokat. Példának okáért tudom említeni a Glassdoort, vagy épp az Internshalát. Ezek az oldalak azonban csak munkák és szakmai nyári gyakorlatok összegyűjtésével foglalkozik, keresni lehet közöttük.

Az én platformomhoz talán a LinkedIn áll a legközelebb, hiszen a LinkedIn is a munkakeresés és az egész szakmaiság kérdést, közösségi oldalas szemmel fogta meg, ami az én platformom esetében is hasonlóan néz ki. Éppen ezért is, sok olyan tényező van, amely a platform esetén is megvalósítható, például a közösségi platform mivoltának erősebb és jobb kihasználása, illetve egyéb olyan platformok és szolgáltatások integrálása, amelyek a platformot egy valóban jól használható szakmába bevezető közösségi térré teszi.

Kutatásaim alapján nincs még egy olyan webes platform sem, ahol a szakmai nyári gyakorlatozás folyamatát, mind a három résztvevő (egyetem, hallgató, cég) számára digitalizálta volna.

## 2.4. Felhasznált technológiák

A projekt célom az volt, hogy egy skálázható, felhasználóbarát, precíz és gyors webalkalmazást hozzak létre. Ehhez két részre osztottam a projektet: frontend és backend.

A frontend fejlesztéséhez az Angular keretrendszert választottam, mivel ez a keretrendszer lehetővé teszi a gyors fejlesztést, a modularitást és a skálázhatóságot. Az Angularhoz a Bootstrap stílus keretrendszert kapcsoltam, amely segít a felhasználóbarát és esztétikus felhasználói felület kialakításában. A teszteléshez pedig a Cypress tesztelési keretrendszert használom, amely lehetővé teszi a megbízható és hatékony tesztek futtatását.

A backend fejlesztéséhez a Java Spring Boot keretrendszert választottam. Ez egy népszerű és megbízható keretrendszer a Java alapú alkalmazásokhoz. Az adatbázis kezeléséhez PostgreSQL-t és Liquibase-t használok, amelyek bevált technológiák az iparban, és biztosítják a tartósságot, a skálázhatóságot és a biztonságot.

Az alkalmazás telepítéséhez AWS felhőszolgáltatást fogok használni. Az AWS lehetőséget nyújt a webalkalmazások folyamatos üzemeltetésére, skálázására és biztonságára. Az AWS számos különböző szolgáltatást kínál, amelyeket használhatok az alkalmazás telepítéséhez

és üzemeltetéséhez, például az EC2 (Elastic Compute Cloud) az alkalmazás futtatásához, vagy az RDS (Relational Database Service) a PostgreSQL adatbázis kezeléséhez.

A választott technológiai kapcsolódások és felhőszolgáltatások lehetővé teszik, hogy a projekt tartós, skálázható és biztonságos legyen, valamint lehetővé teszik a hatékony fejlesztést és tesztelést.

#### **2.4.1. Webes frontend keretrendszerek**

A webes keretrendszerek olyan fejlesztői eszközök és platformok, amelyek segítségével könnyebbé válik a webalkalmazások és weboldalak fejlesztése. Ezek a keretrendszerek előre meghatározott szerkezetet, sablonokat és eszközöket biztosítanak a fejlesztők számára, hogy gyorsabban és hatékonyabban tudjanak dolgozni. Az alábbiakban néhány népszerű webes keretrendszert és azok főbb jellemzőit mutatom be.

Az Angular egy olyan fejlesztői platform és alkalmazástervező keretrendszer, amely a TypeScript nyelven épül. A célja, hogy segítsen a fejlesztőknek méretezhető webalkalmazásokat készíteni komponens alapú architektúrával. Az Angular számos beépített könyvtárral és szolgáltatással rendelkezik, amelyek megkönnyítik a front-end fejlesztést, például a kliens-szerver kommunikációt és az útválasztást. Emellett a keretrendszer tartalmaz egy teljes fejlesztői eszközkészletet, amely segít a projektfejlesztésben és skálázásban, legyen szó kisebb projektekről vagy vállalati szintű alkalmazásokról [7], [8].

Az Angular, fejlesztés közben a TypeScript nyelvet használja alapértelmezetten ám mivel a Google is a fejlesztésében van, ezért az programozási nyelvük, a Dart is helyet kapott a támogatott nyelvek listáján, a JavaScript mellett, amelyre végül a buildelést követően a teljes kód átfordítódik. Ez lehetővé teszi a statikus típusellenőrzést, a fejlettebb fejlesztői eszközöket és a jobb kód érthetőséget.

Az Angular komponens-alapú architektúrával rendelkezik [7], [8], ahol a fejlesztési egység a komponens. A komponensek önálló, újrahasznosítható egységek, amelyek saját kódot, sablonokat (template) és stílusokat (style) tartalmaznak. Ez a kódot modulárisrá és a könnyű karbantarthatóvá teszi, hiszen úgy viselkednek, mint valami LEGO kockák.

Erős adatkötési mechanizmus jellemzi az Angulart, amely automatikusan frissíti az adatokat és a velük kapcsolatos UI elemeket, amikor az adatok megváltoznak. Ez lehetővé teszi a gyors és hatékony fejlesztést, valamint a felhasználói felület dinamikus és interaktív jellegét. Az adatkötés lehet két vagy egyirányú, attól függően, hogy az adatok hogyan áramlanak a komponensek és a felhasználói felület között [9].

Az Angular úgynevezett modulokat használ az alkalmazás komponenseinek és szolgáltatásainak szervezésére és elkülönítésére.

Az említett szolgáltatások szintén központi elemeit jelentik az Angularnak, hiszen ezek olyan osztályok, amelyek központi funkcionalitást nyújtanak az alkalmazásban, úgy, hogy megoszthatók és újrahasznosíthatók a komponensek között. Például ilyen szolgáltatásokon keresztül kommunikálhatunk az alkalmazásunk backendjével vagy egyéb API-okkal [8], [9].

Az Angular stílusnyelve alapvetően lehet a CSS de akár a modernebb SCSS is. Emellett olyan keretrendszerek és külső könyvtárak is kapcsolhatók hozzá, mint a Bootstrap vagy a Google által fejlesztett Material Design. Ezek segítségével az alkalmazásunk kinézete egyedi mégis könnyedén egységesé és ami ennél is fontosabb, reszponzívvá varázsolható, hiszen a mai világban egy webalkalmazás egyszerre kell használható legyen mobilról, tabletről, laptopról és asztali számítógépről is.

Angular fejlesztéshez szinte elengedhetetlen az úgynevezett Angular CLI (Command Line Interface) ami egy parancssoros eszköz, amely automatizálja a fejlesztési feladatokat, például a projekt létrehozását, a komponensek és szolgáltatások generálását, a tesztelést és a futtatást.

További nagy előnyt jelent az Angular esetében, hogy rengeteg fejlesztői eszköz kompatibilis vele, vagy pontosan azért van elkészítve, hogy a lehető legjobb legyen az Angular alkalmazások fejlesztése. A fejlesztést az iparban általában Visual Studio Code-al szokták végezni, ami rengeteg kiegészítőt kínál kifejezetten az Angular keretrendszerhez. Én ezek közül a legfontosabbnak talán a **Prettier - Code formatter** [10], kód formátáló kiegészítőt tartom, hiszen különböző verziókövető rendszerek használata során fontos az, hogy a kód egységesen legyen tördelve, erre pedig a Prettier egy nagyszerű eszköz, mivel beállítható úgy, hogy mentéskor automatikusan elvégezze a formázást, ami ezáltal is megkönnyíti a fejlesztő munkáját.

Napjaink másik ismert webes keretrendszere a React. A React egy JavaScript könyvtár, amely nyílt forráskódú és célja a front-end fejlesztés támogatása. Az UI komponensek építésére szolgál, és könnyedén lehet vele interaktív és összetett felhasználói felületeket létrehozni. A React komponens alapú architektúrája és deklaratív megközelítése lehetővé teszi a fejlesztők számára, hogy egyszer tanuljanak meg vele dolgozni, majd bármely platformon alkalmazható alkalmazásokat fejlesszenek. A React projektet a Facebook React csapata és egy nagy fejlesztői közösség aktívan támogatja.[7]

## Összehasonlítás:

### 1. Szintaxis és tanulási görbe:

- Angular: Az Angular egy teljes körű keretrendszer, amelyet a TypeScript nyelven írnak. A szintaxisa egyedülálló és bonyolultabb lehet az új kezdők számára. A komponensek kifejezése és a felépítése erőteljes, de megtanulása időigényes lehet.
- React: A React könnyebb tanulási görbével rendelkezik, mivel a JSX használata hasonlít az HTML-hez. A JSX-t JavaScript nyelven írják, ami ismerős lehet azoknak, akik már jártasak a JavaScriptben. A React komponensek modulárisabbak és egyszerűbben kezelhetők [7].

### 2. Méret:

- Angular: Az Angular alkalmazások általában nagyobbak lehetnek, mivel az Angular maga is egy nagyobb keretrendszer. Az Angular projekt jelentős méretű fájlokat generál, amelyeknek a betöltése több időt vehet igénybe.
- React: A React kisebb méretű és könnyebb alkalmazásokat hoz létre. Az alapvető React könyvtár kisebb, és a fejlesztők csak azokat a kiegészítő könyvtárakat használják, amelyekre szükségük van. Ennek eredményeként a React alkalmazások általában gyorsabban töltenek be és kevesebb erőforrást használnak [7].

### 3. Ökoszisztéma és közösség támogatása:

- Angular: Az Angular egy hivatalosan támogatott keretrendszer, amelyet a Google fejleszt. Erős közösség és dokumentáció áll rendelkezésre, és a Google által nyújtott támogatás is nagy segítség lehet. Az Angular rendelkezik számos beépített funkcióval és komponenssel.
- React: A Reactet a Facebook fejleszti, és szintén rendelkezik egy aktív fejlesztői közösséggel. Rengeteg nyílt forráskódú könyvtár és csomag érhető el a React projektekhez, és számos segédanyag és oktatóanyag áll rendelkezésre [7].

### 4. Használati esetek:

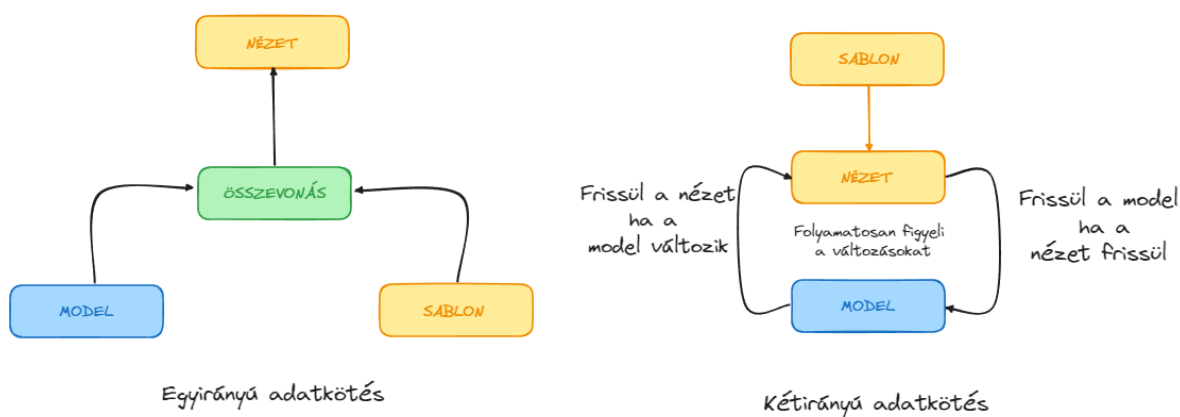
- Angular: Az Angular alkalmazkodik a komplexebb alkalmazásokhoz, amelyek sok oldalt és nagy adatforgalmat tartalmaznak. Az Angular a vállalati szektorban is népszerű, és nagyvállalatok által használt projektekben is megtalálható.
- React: A React rugalmasabb és könnyebben skálázható, ezért jól működik a kisebb és közepes méretű projektekben. A Reactet gyakran választják a startupok és az új fejlesztők, valamint az olyan projektek esetén, amelyekben a gyors prototípuskészítés és a rendszeres frissítések kulcsfontosságúak [7].

## 5. Teljesítmény:

- Angular: Az Angular keretrendszer a komplexebb alkalmazásokhoz lett tervezve, és a nagyobb méretű projektekhez is jól alkalmazkodik. Jelentős számítási feladatokat is el tud látni, ami erős teljesítményt biztosít. Az Angular az AOT (Ahead-of-Time) kompilációval nagyobb teljesítményt érhet el.
- React: A React büszkélkedhet a virtuális DOM-jával, ami az egyik legkeresettebb tulajdonsága az alkalmazás teljesítménye szempontjából. Ez lehetővé teszi a front-end fejlesztők számára, hogy változtatásokat hajtsanak végre anélkül, hogy teljes mértékben újraírják a HTML dokumentumot. Ez gyorsabb teljesítményt eredményez a frissítések gyorsabb renderelése és az adatok gyorsabb frissítése révén a weboldalakon. A komponensek újra felhasználhatósága egy másik aspektusa a Reactnek, amely versenyelőnyt biztosít neki. Ez kritikus kérdés lehet, amikor a fejlesztők különböző projekteken dolgoznak és komplex logikával foglalkoznak, amit nem tudnak újra felhasználni a projekteken [7].

## 6. Adatkötés

Az Angular és a React közötti legnagyobb különbség az adatkezelés megközelítésében rejlik. Az Angular beépített adatkötéssel rendelkezik, ami lehetővé teszi a kétirányú adatáramlást. A React viszont inkább a külső könyvtárakra támaszkodik, például a Reduxra, hogy egyirányú adatáramlást és változtathatatlan adatokat biztosítson. Ez a választás az adott projekt igényeitől és a fejlesztők preferenciáitól függ, így nincs egyértelműen "jobb" megközelítés [7].



2.1. ábra Adatkötési architektúrák összehasonlítása (saját átdolgozás)

### 2.4.2. Tesztelési keretrendszer

Az Angularral készített frontendek tesztelésére számos keretrendszer áll rendelkezésre. Néhány közülük a Jasmine vagy a Karma. Azonban én a Cypress nevű keretrendszert használtam, amely egy modern, nyílt forráskódú tesztelő keretrendszer a front-end webalkalmazásokhoz. A Cypress segítségével a fejlesztők automatizált teszteket hozhatnak létre, amelyek biztosítják az alkalmazás minőségét és stabilitását.

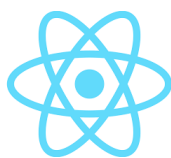
A Cypress egy end-to-end (E2E) tesztelő keretrendszer, amely lehetővé teszi a tesztelési folyamat teljes körű automatizálását. A Cypress számos eszközt és funkciót kínál, amelyek megkönnyítik a tesztek létrehozását és karbantartását. A tesztek futtatása közben egy interaktív felhasználói felületet is biztosítva van, amelyen keresztül a tesztelők vizuálisan ellenőrizhetik a tesztek állapotát és az alkalmazás működését [11].



2.6. ábra Visual Studio Code logó



2.5. ábra Angular logó



2.4. ábra React logó



2.3. ábra Bootstrap logó



2.2. ábra Cypress logó

### 2.4.3. Backend

A Java Spring Boot egy nyílt forráskódú eszköz, amely a Java programozási nyelvet és keretrendszereket használja a mikroszolgáltatások és webalkalmazások könnyebb létrehozásához. A Java a fejlesztési nyelvek között az egyik legnépszerűbb és legelterjedtebb, és széles körben használják az alkalmazásfejlesztés területén. A Java rugalmas és felhasználóbarát, így sok fejlesztő kedveli, legyen szó közösségi médiáról, webes alkalmazásokról, játékokról, hálózatkezelésről vagy nagyvállalati alkalmazásokról. A Spring Boot pedig segíti a fejlesztőket a Java-alapú keretrendszerek könnyebb használatában, így lehetővé téve a hatékonyabb és gyorsabb alkalmazásfejlesztést [12]. Ez a keretrendszer lehetővé teszi a fejlesztők számára, hogy Java virtuális gépeken (JVM) önálló alkalmazásokat hozzanak létre. Fő célja az, hogy lerövidítse a kódot, ami könnyebbé teszi az alkalmazás futtatását.



A Spring Boot kibővíti a Spring keretrendszert, amely kényelmesebb módot biztosít az alkalmazások konfigurálásához. Rendelkezik beépített automatikus konfigurációs képességekkel, amelyek a Spring Frameworköt és a harmadik féltől származó csomagokat a felhasználó beállításai alapján konfigurálják. A Spring Boot felhasználja ezt a tudást a kódhibák elkerülésére a konfiguráció során, csökkentve az olyan kódrészletek mennyiségét, amelyek több helyen ismétlődnek, csekély változtatás nélkül [13].

A Java Spring Boot projektek Maven-t szoktak használni az alkalmazás függőségeinek kezelésére, a forráskód fordítására, a csomagolásra is.

A Spring Boot alkalmazásokban a Maven pom.xml fájlja definiálja a projekt szerkezetét és a függőségeket, amelyekre az alkalmazásnak szüksége van. A pom.xml fájlban megadjuk a Spring Boot függőségeket, például a spring-boot-starter-web-et, amely a Spring Boot alkalmazás alapvető webes funkcióit tartalmazza. A Maven automatikusan letölti ezeket a függőségeket, és biztosítja, hogy a szükséges verziók és függőségek konzisztensek legyenek. Ez megkönnyíti a fejlesztők számára, hogy gyorsan és egyszerűen konfigurálják a Spring Boot projekteket, illetve naprakészen tudják azokat tartani.

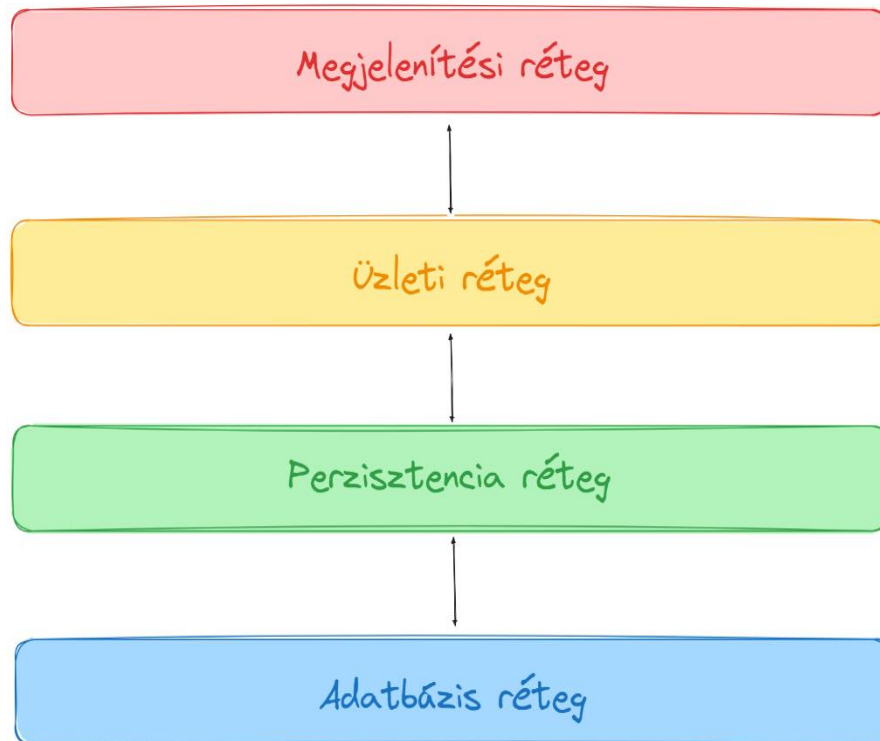
Emellett a Maven lehetővé teszi a Spring Boot alkalmazások build folyamatának automatizálását is. A Maven segítségével könnyedén létrehozhatunk futtatható JAR-fájlokat vagy WAR-csomagokat a Spring Boot alkalmazásokból.

A Java Spring Boot nagy előnye, hogy használhatjuk a Lombok keretrendszert, amely segítségével annotációkat (@) használhatunk mint @Data, amely az adat struktúráinkhoz automatikusan képes kigenerálni a getereket, setereket, de akár az equals(), hashCode() és toString() metódusokat is.

Egy Spring Boot alkalmazás jól definiált rétegekre bontható

- Megjelenítési réteg (presentation layer)
  - hitelesítés
  - JSON objektumok fordítása
- Üzleti réteg (business layer)
  - üzleti logika
  - érvényesítés
  - jogok ellenőrzése
- Perzisztencia réteg (persistence layer)
  - tárolási logika
- Adatbázis réteg (database layer)
  - maga az adatbázis

A rétegek közötti kapcsolatot a 2.7 ábrán szereplő diagrammal lehet szemléltetni:



2.7. ábra Spring Boot alkalmazás rétegei közötti kapcsolat (saját átdolgozás)

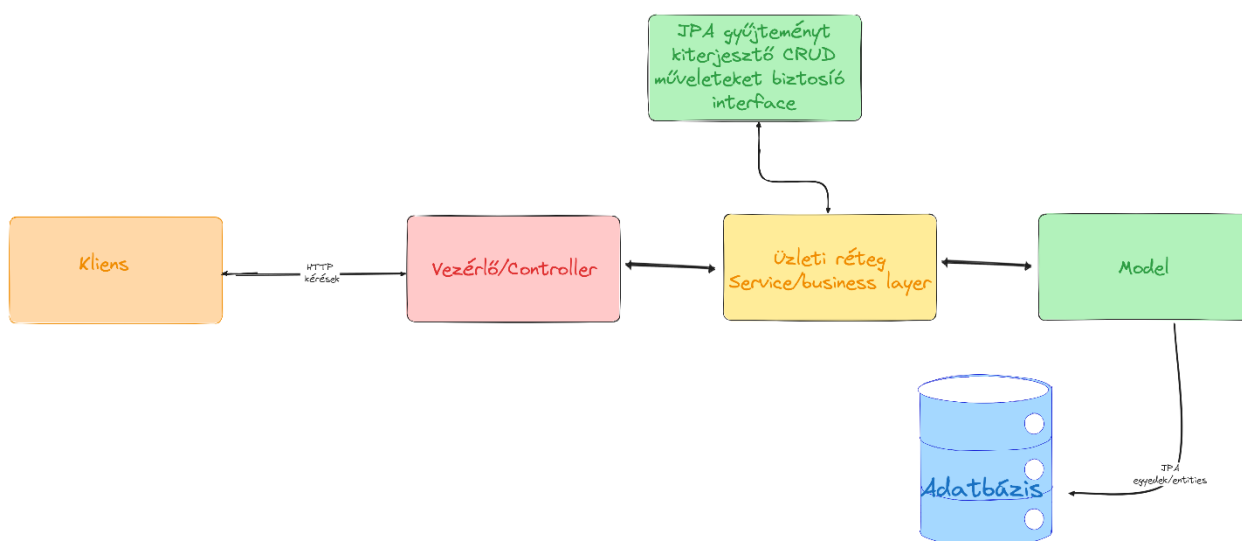
A Java Spring Boot alkalmazások másik nagy előnye, hogy használhatjuk a JPA-t is. A JPA (Java Persistence API) egy specifikáció a Java nyelvhez, amely lehetővé teszi az objektum-relációs leképezést (ORM) a Java alkalmazások számára. Az ORM segítségével az alkalmazás objektumokat kezelhet és tárolhat adatbázisokban, miközben automatikusan gondoskodik az objektumok és az adatbázis táblák közötti leképezésről [14].

A JPA absztrakciós réteget nyújt az adatbázisok kezeléséhez, így a fejlesztőknek nem kell közvetlenül SQL lekérdezéseket írniuk az adatok lekérdezéséhez vagy módosításához. Ehelyett az alkalmazás a JPA-t használja az objektumok adatbázisba való mentésére és az adatok visszanyerésére az adatbázisból.

A JPA lehetővé teszi az objektumok közötti kapcsolatokat, például az egyszerű és összetett kapcsolatokat (1 az 1-hez, 1 a sokhoz, sok a sokhoz) leképezését az adatbázis táblák között. A relációk leírásához annotációkat használ, amelyek meghatározzák a kapcsolat típusát és a kapcsolódó oszlopokat. A JPA segítségével létrejövő CRUD függvényeket meg lehet írni kézzel

is, de szerencsére léteznek külső eszközök, amelyeket a fejlesztői környezetbe integrálva ezeket a függvényeket könnyedén ki tudjuk generálni, és testre is tudjuk szabni. Az eszköz, amit erre a célra használtam az a **JPA buddy**[15] amely az általam használt **IntelliJ IDEA**[16] fejlesztői környezetébe egyszerűen és ingyenesen telepíthető.

A Spring Boot alkalmazások flow architektúrája a következő folyamat diagrammal szemléltethető:



2.8. ábra Spring Boot alkalmazás flow architektúrája (saját átdolgozás)

A projektem adatbázisaként PostgreSQL adatbázist választottam, amely az iparban gyakori technológia kapcsolás a Spring Boothoz. A PostgreSQL egy nyílt forráskódú, relációs adatbázis-kezelő rendszer, amely kiterjedt funkciókészlettel rendelkezik és a SQL nyelvet használja adatmanipulációra és adat lekérdezésre. Nagyon fontos, hogy skálázható, ami azt jelenti, hogy támogatja a replikációt és a részleges indexelést, valamint lehetőséget nyújt a fejlett konfigurációra és a teljesítményoptimalizálásra [17].

Ahhoz, hogy a fejlesztés a lehető legbiztonságosabb legyen, a backendet és az adatbázist egy verziókövető és adatbázis migráló eszköz segítségével kötjük össze. Erre azért van szükség, mert az éles rendszerekben nincs olyan, hogy „végleges adatbázis séma”. Hasonlóan az alkalmazás kiadásokhoz, az adatbázis is folyamatosan fejlődik és változik. Új táblák és oszlopok jönnek létre, megszűnnek, vagy átnevezésre kerülnek, esetleg az adatok más táblába kerülnek. Ez a folyamat gyorsan rendetlenséget okozhat egy fejlesztői csapat esetében. Ezért általában bevezetnek egy adatbázis migrációs rendszert, amely segít kezelni az adatbázis változásait.

Esetemben ez a rendszer a Liquibase [18].

Az adatbázis migrációs rendszerek általában négy alapvető komponenst tartalmaznak:

- **Migráció/revízió/változás:** Az adatbázis migrációs rendszerekben a változtatásokat migrációk vagy revíziók formájában rögzítik. Ezek a migrációk tartalmazzák az adatbázis változtatásait, például új táblák létrehozását, oszlopok hozzáadását vagy meglévők módosítását.
- **Visszavonás:** Az adatbázis migrációs rendszerek lehetővé teszik a visszavonást, vagyis az adatbázis visszaállítását egy korábbi verzióra. Ez hasznos lehet, ha hibás változtatásokat kell javítani vagy vissza kell állítani az adatbázis állapotát.
- **Ismételhető migráció:** Az ismételhető migrációk olyan változtatásokat tartalmaznak, amelyeket minden adatbázis futtatáskor újra és újra végrehajtanak. Ezek általában beállításokat vagy adatokat tartalmaznak, amelyek állandóak és nem változnak az adatbázis állapotától függetlenül.
- **Verziókövetés:** Az adatbázis migrációs rendszerek nyomon követik az adatbázis változásait és verzióit. Ez lehetővé teszi a fejlesztők számára, hogy kövessék és kezeljék az adatbázis változásait idővel. Ez segít a csapatmunkában és biztosítja az adatbázis konzisztenciáját a különböző fejlesztői környezetekben [19].



2.9. ábra IntelliJ  
logó



2.12. ábra Java Spring Boot logó



2.10. ábra  
Liquibase logó



2.11. ábra  
PostgreSQL logó

### 3. A rendszer specifikációi és architektúrája

A rendszer ötletének megszületésekor további információk gyűjtése és konzultációk megtartása segített a rendszer specifikációinak kidolgozásában. Az alábbiakban részletezem, hogy milyen lépéseket tettem a rendszer kidolgozása érdekében:

- Konzultáció hallgatókkal: Találkoztam és konzultáltam olyan hallgatókkal, akik potenciális felhasználói lehetnek a rendszernek. Megtudtam, hogy milyen problémákkal szembesülnek, milyen funkciókat tartanának hasznosnak, és milyen elvárásaik vannak egy ilyen rendszerrel szemben.
- Konzultáció cégvezetőkkel: Leültem olyan cégvezetőkkel, akik az általam tervezett rendszert hasznosnak láthatják a vállalkozásuk szempontjából. Kikértem véleményüket, megismertem az üzleti igényeiket és a rendszerrel szemben támasztott követelményeiket. Ez segített abban, hogy a rendszer megfeleljen az üzleti céloknak és igényeknek.
- Konzultációja HR-esekkel: A HR-szakemberekkel való konzultáció során megismertem a munkaerő-gazdálkodással és az emberi erőforrás menedzsmenttel kapcsolatos kihívásokat. Megtudtam, hogy milyen funkciókat és jellemzőket tartanának hasznosnak a rendszerben az alkalmazottak kezelése, a toborzás és kiválasztás folyamatai, az adatkezelés területén.
- Egyetemi tanárok konzultációja: A konzultációk során tanárokkal és szakértőkkel beszéltem, akik széleskörű tapasztalattal rendelkeznek az adott területen. Ők segítettek a rendszer azon funkciók specifikációinak kidolgozásában, amelyek őket érintik.

#### 3.1. Felhasználói követelmények

A rendszer fő célja a szakmai nyári gyakorlatok folyamatainak a digitalizálása, így a követelményeket e-köré a téma köré kellett felépíteni.

Mint ahogy az majd a lenti use case diagrammból is látszik, 5 különböző típusú felhasználó lehet a platformon. Ez az 5 típus felhasználó más-más jogkörrel rendelkezik, aminek hála a feladatok szépen, könnyen és világosan választhatóak el egymástól. Erre azért is van szükség mert ezáltal is nő a rendszer biztonsága és a személyes adatok védelme.

Ez az 5 szerepkör a következő:

1. Admin
2. Cég menedzser / Company manager
3. Egyetem menedzser / University manager
4. Szakkoordinátor / Instructor
5. Hallgató / Student

Az, hogy erre az 5 típusú szereplőre van szükség a platform megtervezésekor vált nyilvánvalóvá, és a kivitelezés során mindez igazolódott is.



3.1. ábra Az applikáció use case diagrammja

Ha valaki megnyitja az alkalmazást, akkor kezdetben bármi féle szerep nélkül tudja görgetni a kezdőlapot. Ezután, ha a valaki a Login gombra kattint, akkor az átdobja az login felületre, ahol három opció közül választhat. Ha hallgató és még nincs fiókja, akkor egyszerűen be tud regisztrálni az egyetemi email címével (a regisztrációt el kell fogadjon majd az egyetem menedzsere is). Ha van már profilja, de elfelejtett a jelszavát, akkor azt is meg tudja változtatni. Hogyha van fiókja és tudja a jelszavát, akkor bárki beléphet. A login logika eldönteni azt, hogy a felhasználó szerepköre alapján milyen felületre kell kerüljön és tovább is irányítja oda.

Az **Admin** tudja kezelni az egyetemeket és a cégeket. A platform üzleti modellje alapján a cégek és az egyetemek pénzért férhetnek hozzá a platform szolgáltatásaihoz, így aztán fontos az Admin számára, hogy minden egyetemet és céget egyénileg tudjon kezelni, és látja a fizetési határidőket, illetve a lejárat dátumokat.

A **cég menedzser** az a személy vagy személyek, aki a platformon az adott céget képviseli. Így aztán lehetősége van a saját profiljának kezelése mellett az cégének profilját is. Ez azt jelenti, hogy tudja állítani a cégének az adatait, mint például az elérhetőségek vagy akár a cég logója, de ő tudja meghirdetni a cég nyitott pozícióit is. Ezeket a pozíciókat, illetve a rájuk érkező jelentkezéseket tudja kezelni. A jelentkezőket a nyárigyakorlat lejártakor tudja értékelni, amely értékelés alapján később a szakkoordinátor tudja a diákot értékelni.

Az **egyetemi menedzser** az a személy, aki a platformon az egész egyetemet képviseli. Ő az a személy, aki a szakokat feltölti, azokhoz szakkoordinátorokat köt, illetve a hallgatók regisztrációit kezeli.

A **szakkoordinátorok** azok a felhasználók, akik egy egyetem egy adott szakjáért és azok hallgatóiért felelősek. Minden szakkoordinátornak joga van hozzá, hogy a platformra regisztrált cégeket akkreditálja az adott szakhoz. Ez azért fontos, mert szeretnénk elkerülni, hogy a hallgató, ha tegyük fel mondjuk számítástechnika szakra jár akkor hozzáférhessen olyan pozíciókhoz, amiket egy fordítóiroda ad ki. További fontos érv, az akkreditálás szakkoordinátorra való bízásában az is, hogy így lehetőség van az olyan cégek kiszűrésére a platformról, amire a hallgatói visszajelzések nem megfelelőek, például a szervezetlenség vagy egyéb más okok miatt.

A szakkoordinátorok másik fontos feladata az is, hogy a szakhoz tartozó regisztrált hallgatókat osztályozza. A platformon keresztül erre is lehetőség van. A diákok évfolyamonként vannak kilistázva az szakkoordinátorok számára, akik láthatják azt, hogy a diák milyen gyakorlaton vett részt, melyik cégnél és hogy milyen képességeit erősítette, valamint azt is, hogy az adott cégtől milyen értékelést kapott, de azt is, hogy a hallgató milyen értékelést adott. Ezen információk alapján, a szakkoordinátor egy osztályzatot adhat a diák számára, amely 1-től 10-ig terjedhet. Később ezt a jegyet már nem tudja módosítani.

A szakkoordinátornak lehetősége van minden diákot és a neki adott osztályzatot kilistázni és azt exportálni.

A **hallgató** tudja úgy használni a platformot, mint egy fél-profi szakmai közösségi platformot. Ez azt jelenti, hogy posztokat tud kitenni az elvégzett kurzusairól vagy versenyeken elért sikereiről tud beszámolni, fel tudja tölteni a profilját a tanulmányaival a személyes adataival, valamint elérhetőségeivel, de ami a legfontosabb, hogy feltudja tölteni a képességeit, amelyeket priorizálni is tud. A platform algoritmusa ezen képességek alapján fogja a hallgató számára a legrelevánsabb pozíciókat ajánlani. A hallgatók platformja, a cégekéhez hasonlóan nyilvános, így egyetemi társai vagy a cégek azt láthatják. A cégek a látott információk alapján hozhatnak arról döntést, hogy a hallgatók jelentkezését elfogadják-e az adott pozícióra.

A hallgatók az ajánlott pozíciókra, vagy a szakjához akkreditált cégek pozícióira tud jelentkezni. A szakmai nyári gyakorlat lejárta után a hallgató értékelni tudja az adott céget vagy pozíciót, amelye értékelések alapján a szakkoordinátorok át tudják értékelni az adott céggel való partneri kapcsolatot.

A feltöltött információkból a diák kigenerálhatja magának az önéletrajzát, amely tartalmazni fogja a platformon tartózkodása alatt elvégzett gyakorlatokat, valamint a képességeit, tanulmányait és elérhetőségeit.



## 3.2. Rendszerkövetelmények

### 3.2.1. Funkcionális követelmények

A funkcionális követelmények az alábbi 3.1 táblázatban vannak felsorolva.

FK1	Az alkalmazás lehetőséget kell biztosítson arra, hogy a platform főbb funkcionálisait bemutassa a nem regisztrált felhasználók számára is. A kezdőlap három nyelven kell elérhető legyen: magyar, román és angol.
FK2	Az alkalmazás lehetőséget kell biztosítson arra, hogy egy felhasználó belépjen a hitelesítési adataival. A hitelesítési adatok a következők: email cím és jelszó.
FK3	Az alkalmazás lehetőséget kell biztosítson arra, hogy a hallgatók be tudjanak regisztrálni és kitudják választani a saját szakjukat, valamint évfolyamukat.
FK4	Az alkalmazás lehetőséget kell biztosítson az adminisztrátornak az egyetemek kezelésére. Egy egyetem beregisztálásához szükséges az egyetem neve, email címe, telefonszáma, valamint elhelyezkedése, és az egyetem diákok által használt domainje. Az adminisztrátor kell tudja kezelni az egyetem menedzsereit is, be kell tudja őket regisztrálni, egy email cím, és név alapján.
FK5	Az alkalmazás lehetőséget kell biztosítson az adminisztrátornak a cégek kezelésére. Egy cég beregisztálásához szükséges a cég neve email címe, telefonszáma, valamint elhelyezkedése. Az adminisztrátor kell tudja kezelni a cég menedzsereit is, be kell tudja őket regisztrálni, egy email cím, és név alapján.
FK6	Az alkalmazás lehetőséget kell biztosítson az adminisztrátornak a képességek kezelésére, valamint azok kategorizálására.
FK7	Az alkalmazás lehetőséget kell biztosítson az adminisztrátornak a tanulmányi típusok kezelésére. Például: líceum, gimnázium, egyetem stb..
FK8	Az alkalmazás lehetőséget kell biztosítson a cégek megbízottjait számára, hogy a saját profilját testre szabja. Kell tudja változtatni a profilképét, nevét, valamint jelszavát.
FK9	Az alkalmazás lehetőséget kell biztosítson arra, hogy a cégek megbízottjai a cég profilját tudják testre szabni, valamint a megfelelő információkat feltölteni. Kell tudja módosítani a cég arculati elemeit, mint a logó vagy a cég színei, a cég helyét, valamint elérhetőségeit és bio adatait, mint például a cég neve, leírása céljai, alapítási dátuma stb..

FK10	Az alkalmazás lehetőséget kell biztosítson arra, hogy a cégek megbízottjai állásokat tudjanak meghirdetni. Meg kell tudják adni a pozíció címét, leírását, helyszínét, esetleges fizetését, valamint az elvárt képességeket, amelyeket priorizálni lehet.
FK11	Az alkalmazás lehetőséget kell biztosítson arra, hogy a cégek megbízottjai a beérkező jelentkezéseket meg tudják vizsgálni és el tudja bírálni. El kell tudják utasítani, valamint el kell tudják fogadni azt.
FK12	Az alkalmazás lehetőséget kell biztosítson arra, hogy a cégek megbízottjai pozíciókat betöltő hallgatókat értékelje egy pontszámmal 1-től 5-ig, valamint emellé szöveges érvelést és véleményt adjon.
FK13	Az alkalmazás lehetőséget kell biztosítson arra, hogy a cégek megbízottjai újabb megbízottakat adjanak hozzá.
FK14	Az alkalmazás lehetőséget kell biztosítson arra, hogy az egyetem megbízottjai feltöltsék az adott egyetem szakjait. Egy szak feltöltéséhez szükség van a szak nevére, illetve arra, hogy melyik évfolyamoknak kell gyakorlatozniuk.
FK15	Az alkalmazás lehetőséget kell biztosítson arra, hogy az egyetem megbízottjai egy adott szakhoz, szakkoordinátort adjanak. Egy szakkoordinátor hozzáadásához szükség van egy email címre és egy névre.
FK16	Az alkalmazás lehetőséget kell biztosítson arra, hogy az egyetem megbízottjai megtekintsék az összes hallgatót szakokra és évfolyamokra bontva.
FK17	Az alkalmazás lehetőséget kell biztosítson arra, hogy a szakkoordinátorok megtekintsék az összes hallgatót évfolyamokra bontva.
FK18	Az alkalmazás lehetőséget kell biztosítson arra, hogy a szakkoordinátorok jegyet adjanak a szak diákjainak a szakmai nyári gyakorlaton nyújtott teljesítménye alapján.
FK19	Az alkalmazás lehetőséget kell biztosítson arra, hogy a szakkoordinátorok kiexportálhassák a szak hallgatóinak névsorát a kapott jegyeket valamilyen táblázat kezelő rendszer formátumába.
FK20	Az alkalmazás lehetőséget kell biztosítson arra, hogy a szakkoordinátorok a regisztrált cégeket akkreditálja szakjához, vagy visszavonja az akkreditációt.
FK21	Az alkalmazás lehetőséget kell biztosítson arra, hogy egy hallgató felépítse a saját profilját. Megadhatja a tanulmányait, a képességeit, amelyeket priorizálni is tud, a bio adatait, valamint az elérhetőségeit is.

FK22	Az alkalmazás lehetőséget kell biztosítson arra, hogy egy hallgató posztokat tehessen ki a profiljára, amelyek létrehozásához szükség van egy cím és a tartalom megadására.
FK23	Az alkalmazás lehetőséget kell biztosítson arra, hogy egy hallgató a megadott és priorizált képességei alapján kapjon állásajánlatokat, bármelyik olyan cégtől, amit a szakkoordinátora akkreditált.
FK24	Az alkalmazás lehetőséget kell biztosítson arra, hogy a hallgatók láthassák azokat a céges profilokat, amiket a szakkoordinátora akkreditált.
FK25	Az alkalmazás lehetőséget kell biztosítson arra, hogy a hallgatók lássák a társaik posztjait és reagálni tudjanak rá.
FK26	Az alkalmazás lehetőséget kell biztosítson arra, hogy a hallgatók ki tudják generálni a CV-jüket a feltöltött adataik és az elvégzett gyakorlatok alapján.
FK27	Az alkalmazás lehetőséget kell biztosítson arra, hogy a hallgatók értékeljék azt a céget és ahol a gyakorlatot végezték egy pontszámmal 1-től 5-ig, valamint emellé szöveges érvelést és véleményt adjon.

3.1. Táblázat Funkcionális követelmények

### 3.2.2. Nem funkcionális követelmények

A funkcionális követelmények az alábbi 3.2 táblázatban vannak felsorolva.

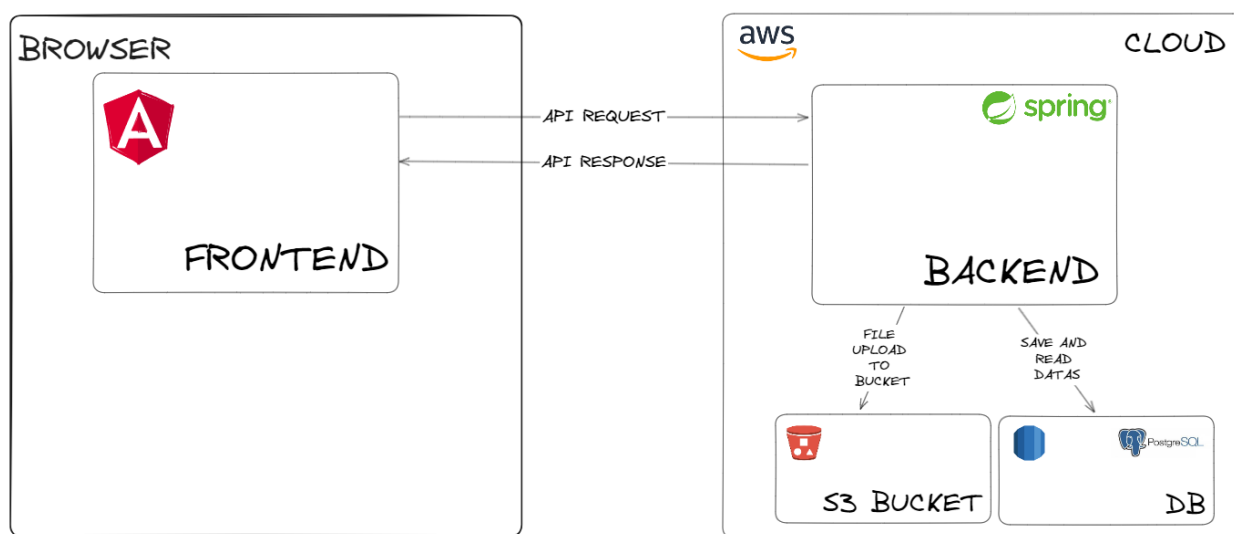
NFK1	A felhasználók nem korlátozódnak egy specifikus operációs rendszerhez, hanem szabadon használhatják az alkalmazást bármely olyan operációs rendszeren, amely támogatja egy webböngésző futtatását.
NFK2	A regisztrált felhasználók adatai egy PostgreSQL adatbázisban vannak tárolva, amely nem közvetlenül a felhasználó eszközén helyezkedik el.
NFK3	Az azonosítás a Spring Boot beépített hitelesítési moduljával történik.
NFK4	A felhasználói felületen visszajelzést kell kapjon a felhasználó a sikeres és a sikertelen műveletekről.
NFK5	A felhasználó minden visszafordíthatatlan művelet előtt figyelmeztetést kell kapjon a művelet súlyáról.
NFK6	A felhasználó érzékeny adatai titkosítva legyenek eltárolva
NFK7	A rendszer által használható táblázat kezelő formátum a .csv

## 4. Részletes tervezés

### 4.1. Az architektúra és a felhasznált technológiák

A rendszerem 2+1 nagy alrészre osztható. Van egy Angular-ban megírt frontendem, illetve egye Java Spring Bootban megírt backendem. A backendhez hozzátartozik az adatbázis is, lévén, hogy használtam liquibaset is, aminek segítségével az adatbázis verziókövetését végeztem.

A rendszer architektúrája a következőképpen néz ki:



4.1. ábra A rendszer architektúrája

Az alkalmazásom kész frontendje a kliens oldalán a böngészőben fut. Ez azt jelenti, hogy az alkalmazás már JavaScript nyelvre van fordítva, vagyis az alkalmazás ki van már buildelve és telepítve van a szerverre, ahonnan a böngésző elérí, betölti és futtatja.

Az alkalmazásom backendje, a felhőben, az Amazon Web Services egyik szolgáltatásában az EC2-ben (Elastic Compute Cloud). Ehhez kapcsolódik az adatbázisom RDS szolgáltatásába, illetve ami egy S3 (Simple Storage Service), ami a fileok, például fotók vagy dokumentumok tárolására szolgál.

Az API (Application Programming Interface) egy olyan szabályrendszer, amely lehetővé teszi a különböző alkalmazások közötti kommunikációt. Az API-k az adatátvitelt feldolgozó köztes réteggént működnek a rendszerek között, lehetővé téve a vállalatok számára, hogy megnyissák alkalmazásaik adatait és funkcióit külső harmadik felek, üzleti partnerek és belső osztályok számára [3].

## 4.2. Az adatbázis

### 4.2.1. Tervezés

A rendszer elméleti megtervezését követően az adatbázis megtervezését kezdtem meg elsőként. Mivel relációs adatbázist használok, ezért első sorban egy jó tervező szoftvert kellett keressek, mely segítségével a főbb funkcionálisokat kiszolgáló adattáblákat és a köztük levő kapcsolatot ábrázolni tudtam. Ekkor találtam rá az **sqldbd** [20] nevű adatbázis tervező online eszközre, amely mind erre lehetőséget ad, sőt még azt is be tudom állítani, hogy melyik adatbázis kezelő rendszerben szeretném megvalósítani az adott adatbázist, és ha előfizetnék a szolgáltatásra, akkor még ki is generálja nekem a megfelelő SQL kódra. Mivel azonban a backendemhez szerettem volna adatbázis migráló eszközt, Liquibaset használni, ezt a funkciót nem használtam ki.

### 4.2.2. A PostgreSQL adatbázis létrehozása és kezelése

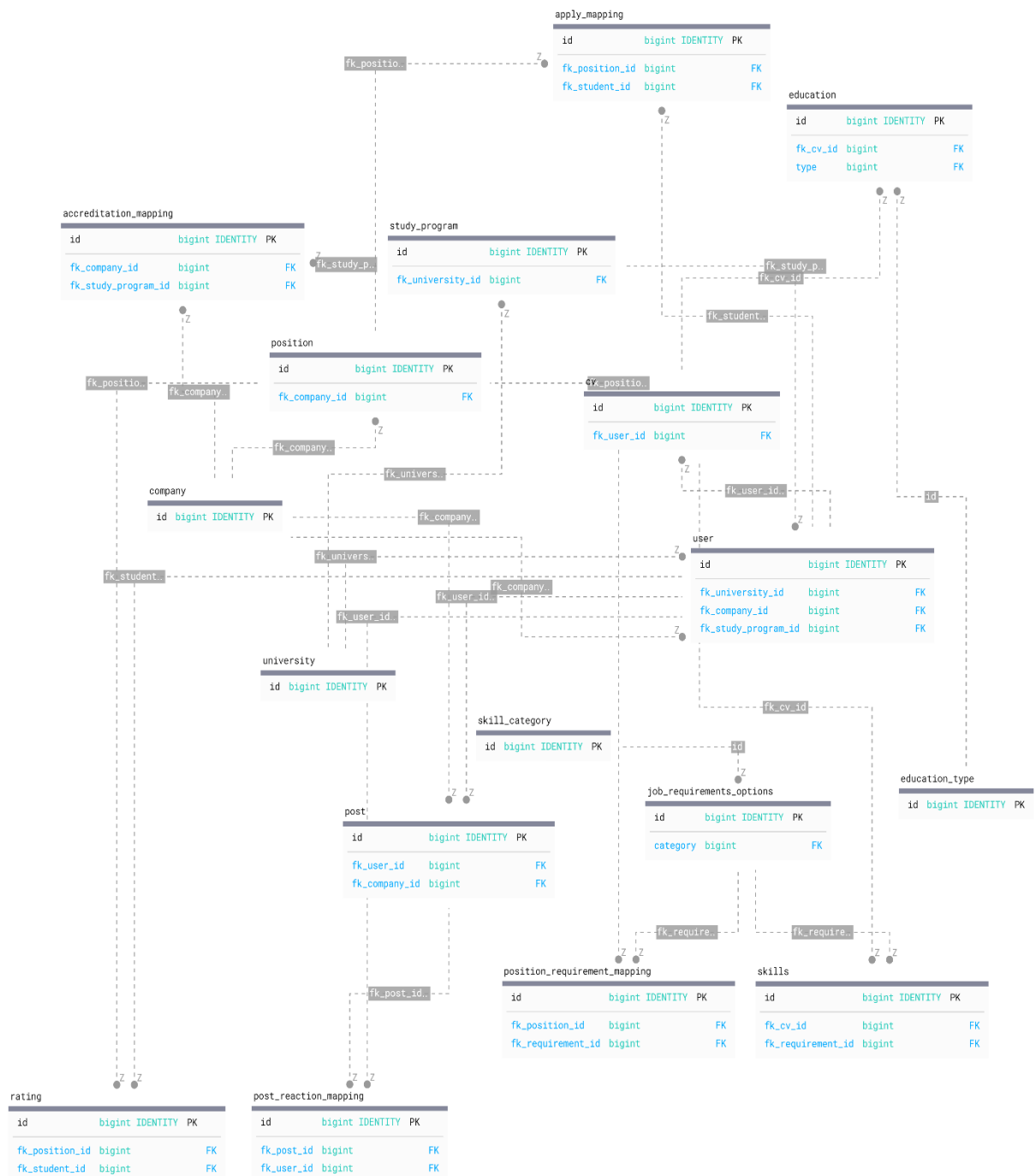
A PostgreSQL adatbázis létrehozása és kezelése a pgAdmin nevű grafikus felhasználói felületen keresztül is lehetséges. Először telepítenünk kell a PostgreSQL szervert és a pgAdmin alkalmazást a számítógépünkre.

Miután a szerver fut, a pgAdmin segítségével kapcsolódhatunk hozzá. Létrehozhatunk új adatbázist az "Adatbázisok" mappában a jobb gombbal történő kattintással, majd az "Új adatbázis" lehetőség kiválasztásával. Itt meg kell adnunk az adatbázis nevét, a tulajdonosát és más beállításokat, mint például a karakterkódolást.

A pgAdmin felhasználói felületének segítségével könnyedén hozhatunk létre és kezelhetünk PostgreSQL adatbázisokat, és különféle műveleteket végezhetünk rajtuk a vizuális eszközökkel és parancsokkal. Az adatbázisom végső, sematikus ábrázolását is a pgAdmin beépített eszközével generáltam ki.

Mivel az adatbázis létrehozása, frissítése, illetve adatokkal való feltöltése már az applikációval történik, így nagyjából csak az adatbázis létrehozás és konfigurálás, illetve a beérkező adatok és elkészült táblák kilistázása funkciókat használtam, de természetesen nagyon sok más lehetőséget is biztosít ez az adatbázis kezelő rendszer, ami által a fejlesztést könnyedebbé teszi.

### 4.2.3. Az adatbázis szerkezete



4.2. ábra Az adatbázis szerkezete a fontosabb kulcsokkal

### 4.3. A kommunikáció

A frontend és a backend REST API-kon keresztül kommunikál egymással, kérések és válaszok formájában. A kérések négy metódust használnak az alkalmazásom esetében, melyek a következők: GET, POST, PATCH, DELETE. A POST és PATCH metódusok törzsében DTO-t (Data Transfer Object, amely igazából egy JSON típusú objektum) küldök, míg a GET és a DELETE metódusok esetében az API elérési útjában (a továbbiakban path) definiált ID változót küldök.

A kérések különböző eredményekkel térnek vissza, általában egy DTO és egy állapot jelző kód formájában. A kód a válasz fejlécében míg a DTO a válasz törzsében jön vissza (DELETE metódus esetén általában nincs visszatérített objektum csak az állapotot jelző kód).

Az állapot jelző kódok sok féleké lehetnek, de alapvetően 5 nagy csoportba sorolhatók.

Állapot jelző kód	Leírás
1xx	Tájékoztató (Informational)
2xx	Siker (Success)
3xx	Átirányítás (Redirection)
4xx	Kliens oldali hiba (Client Side Error)
5xx	Szerver oldali hiba (Server Side Error)

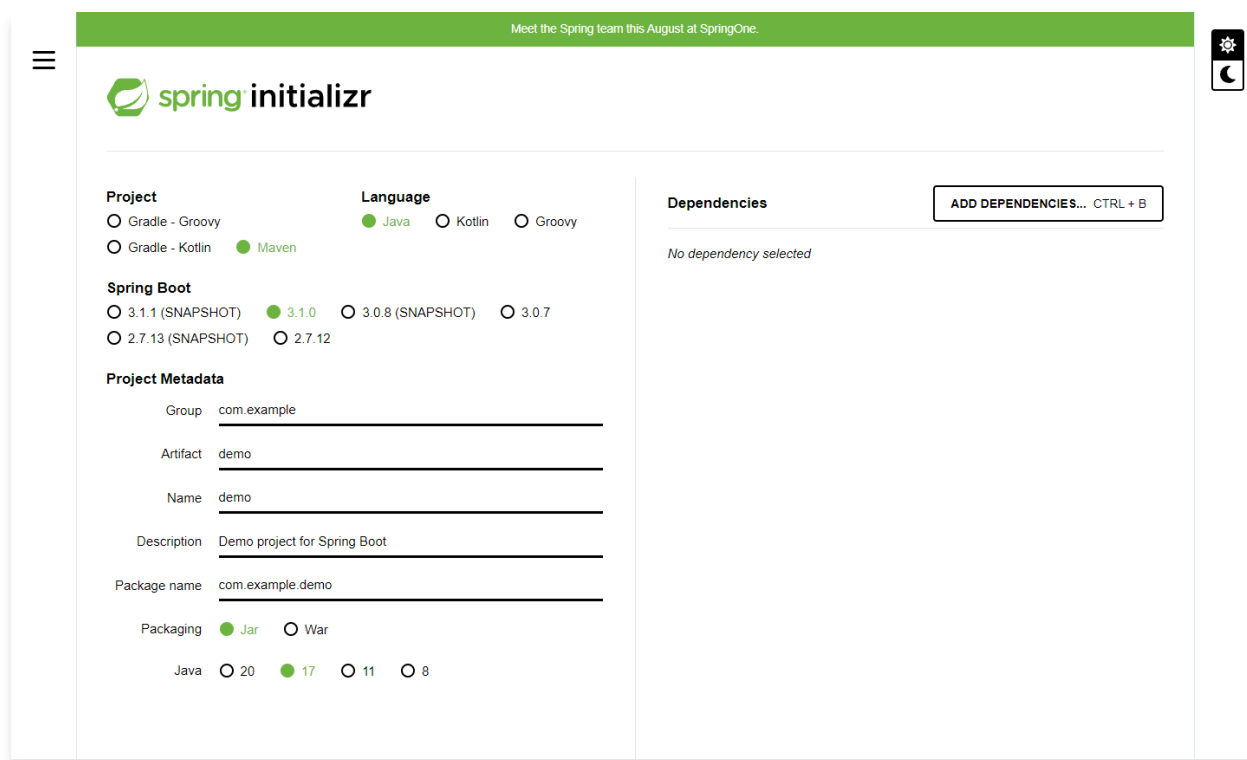
4.1. Táblázat Állapot jelző kódok

A kliens oldali hibák közül számomra az egyik legfontosabb kód a 401 és a 403, amelyek azt jelzik, hogy a felhasználónak nincs meg a jogosultsága a kért függvény futtatásához. Mivel ez biztonsági kockázattal jár ezért a frontenden implementálni kell egy függvényt, amely észleli, ha a backend ilyen hibakóddal tér vissza, és végrehajt egy eseményt. Az én esetemben ez az esemény nem más, mint hogy a felhasználót kijelentkezteti (logout) és visszadobja őt a kezdőlapra. Mivel a backendem jelezte a hibát így értelem szerűen onnan sem érkezik más, biztonsági kockázatot jelentő információ, így a probléma a frontend és a backend részéről is meg van akadályozva.

## 4.4. Backend

### 4.4.1. Konfigurálás

A backend részét Java nyelven implementáltam, és a Spring Boot keretrendszert használtam. A Spring Boot előnye, hogy nem szükséges külön foglalkozni a Maven fájlok és a Tomcat alkalmazás-szerver szinkronizálásával. A Spring Boot fejlesztői létrehoztak egy weboldalt, ahol Spring Boot típusú projekteket lehet létrehozni. Ez az oldal nem csak a projekt struktúrájának létrehozásában segít, hanem a különböző függőségek beállításában is egyszerűen. Ez az oldal nem más, mint a Spring Initializr [4].



The screenshot shows the Spring Initializr web application. At the top, there is a green banner with the text "Meet the Spring team this August at SpringOne." Below the banner is the Spring Initializr logo. The main content area is divided into several sections:

- Project:** Radio buttons for "Gradle - Groovy", "Gradle - Kotlin", and "Maven" (selected).
- Language:** Radio buttons for "Java" (selected), "Kotlin", and "Groovy".
- Spring Boot:** Radio buttons for "3.1.1 (SNAPSHOT)", "3.1.0" (selected), "3.0.8 (SNAPSHOT)", "3.0.7", "2.7.13 (SNAPSHOT)", and "2.7.12".
- Project Metadata:** Text input fields for "Group" (com.example), "Artifact" (demo), "Name" (demo), "Description" (Demo project for Spring Boot), and "Package name" (com.example.demo).
- Packaging:** Radio buttons for "Jar" (selected) and "War".
- Java:** Radio buttons for "20", "17" (selected), "11", and "8".
- Dependencies:** A section with a button "ADD DEPENDENCIES... CTRL + B" and the text "No dependency selected".

4.3. ábra Spring initializr

Ahogy az a 15-ös ábrán is látható, az initializr segítségével be tudjuk állítani azt, hogy mit használjon a projektünk alkalmazás függőségeinek kezelésére (Maven vagy Gradle), milyen programozási nyelvet használjunk (Java vagy Kotlin), a keretrendszer verzióját, a csomagolás kiterjesztését, a nyelv verzióját, illetve a kezdetben csatolt függőségeket. Ha kész vagyunk egy

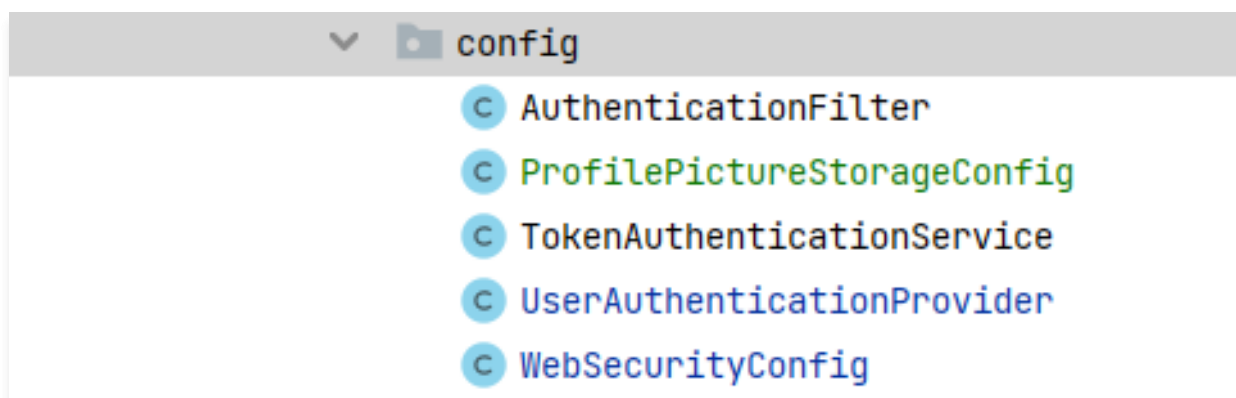


gombnyomásra tudjuk a projektet kigenerálni, majd letölteni. Így egy kész struktúrát kapunk, amivel elkezdhetjük a fejlesztést.

A függőségek közt itt be lehet integrálni a JPA-t, a liquibaset, JWT token generálást és spring boot beépített hitelesítési modulját is.

Innentől kezdve még el kell végezni a konfigurációs beállításokat, amelyek elősegítik az alkalmazás fejlesztésének gördülékenységet és a biztonságot, így például be kell állítani azt, hogy az alkalmazás honnan fogadjon el kéréseket, és azt, hogy mely endpointok nem igényelnek hitelesítést. Erre azért van szükség, mert a hitelesítő modul, amit telepítünk az alapból minden endpointhoz hitelesítést vár, így például egy login nem történhetne meg, hiszen a JWT token csak a login sikerességét követően generálódik ki. A kigenerálódott token a login sikeres kimenetelekor a válasz fejlécében kerül vissza a frontendre, a válasz törzsében pedig DTO formájában a felhasználó. Ezt a DTO-t én UserResponseDto-nak neveztem el. Ez megszűri a felhasználó adatait, hogy például a jelszava ne érkezhessen vissza a frontendre, ezáltal is növelve a biztonságot.

Az alkalmazásomban az is konfigurálva van, hogy ha egy felhasználó hitelesítve is van, akkor is el lehessen tiltani egy adott endpointtól, a szerepe (ROLE) alapján. Erre azért van szükség, mert meg kell akadályozni az olyan helyzeteket, hogy illetéktelenek férjenek hozzá, érzékeny adatokhoz. Például egy STUDENT nem férhet hozzá olyan endpointokhoz, amelyek mondjuk egy cég személyes adatait módosítja vagy törli.



4.4. ábra Az alkalmazás backendjének konfigurációs filejai

#### 4.4.2. A felhasználói fiókok biztonsága

A felhasználó a fiókját jelszóval védi a rendszer, amely jelszó biztonságos módon, titkosítva tároljuk az adatbázisban.

A titkosításért az alkalmazás backendje felelős és egy jól ismert, nagyon biztonságos algoritmust használ, amelyet egy külső függőséggel érünk el. Ez az algoritmus nem más, mint a bcrypt.

A Bcrypt egy jelszóhashelési eljárás, amely a Blowfish blokk titkosító eljárásán alapul. Célja az ellenállás a brute force támadásokkal szemben, és az, hogy biztonságos maradjon még a hardverfejlesztések ellenére is [21].

A Bcrypt algoritmus megbízhatósága a következő tulajdonságokon alapul:

Lassúság: Az algoritmus lassú végrehajtása azzal a céllal történik, hogy megnehezítse a támadók számára a brute force támadásokat. A lassúság azt jelenti, hogy a jelszó hashelése hosszabb ideig tart, ami visszatartja a támadókat a nagy sebességű támadásoktól.[21]

Salt használata: A salt egy véletlenszerűen generált érték, amelyet hozzáadnak a jelszóhoz a hashelés előtt. A salt használata megakadályozza, hogy az azonos jelszók ugyanazt a hash értéket kapják, ami megnehezíti a támadók számára a hash értékek visszafejtését. [21]

Iteráció: Az iteráció a Bcrypt algoritmusban azt jelenti, hogy többször alkalmazzák a hash függvényt a jelszóra. Ez növeli a támadásokkal szembeni ellenállást, mivel megnöveli a végrehajtási időt. [21]

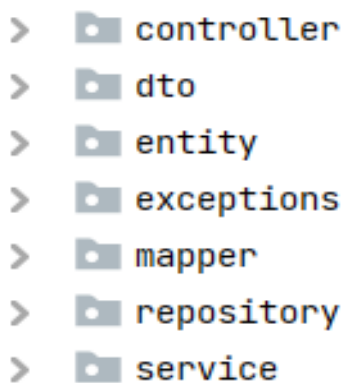
A felhasználó regisztrációjakor, illetve létrehozásakor a jelszó titkosítva kerül eltárolásra az adatbázisba. Ebből következik, hogy azt, hogy mi a felhasználó jelszava, akkor sem kerül nyilvánosságra, ha magát az adatbázist törnék fel. A felhasználó beléptetése úgy történik, a belépéskor megadott jelszavát a felhasználónak szintén hasheljük, és a kapott hash értéket hasonlítjuk össze az adatbázisban tárolttal. Mivel az algoritmus erőforrás igényes, ezért a brute force feltörési módszereknek is ellenál.

```
if (user.isEmpty() || !bcryptPasswordEncoder.matches(password,
user.get().getPassword())) {
    throw new BadCredentialsException("Invalid login credentials");
}
```

#### 4.4.3. Egy CRUD funkcionalitás megírása

A projektem úgy van felépítve, hogy ahhoz, hogy egy CRUD funkcionalitást (Create, Read, Update, Delete) megírjunk 6 komponensre van szükségünk:

1. Adatmodell (Entity): A CRUD műveletek elvégzéséhez először definiálni kell az adatmodellt vagy entitást, amelyet tárolni és manipulálni szeretnénk az adatbázisban. Az adatmodell egy Java osztály, amely annotációkkal (pl. `@Entity`, `@Table`) jelzi, hogy hogyan kell mappelni a táblákat az adatbázisban.
2. Adatelérési réteg (Repository): Az adatelérési réteg felelős az adatok lekérdezéséért és módosításáért az adatbázisban. A Spring Boot keretrendszerben a JPA (Java Persistence API) használatával könnyedén készíthetsz adatelérési réteget. A Repository interfészben kiterjeszthetjük a `JpaRepository`-t így sok beépített metódust használhatunk, mint például a `findById` vagy a `save`. Emellett a JPA buddy segítségével generálhatunk újakat. Ezen felül, ha valamilyen egyedi adatbázis lekérésre lenne szükség az is megírható SQL segítségével
3. Szolgáltatás réteg (Service): A szolgáltatás réteg magasabb szintű üzleti logikát valósít meg, és összekapcsolja az adatelérési réteget a vezérlő réteggel. A szolgáltatások tartalmazzák a CRUD műveletek implementációját, valamint további üzleti logikát, amelyet az alkalmazás igényel.
4. Vezérlő réteg (Controller): A vezérlő réteg felelős az HTTP kérések fogadásáért és a megfelelő válaszok elkészítéséért. Itt vannak definiálva az útvonalak és az HTTP metódusok (GET, POST, PATCH, DELETE), amelyek az egyes CRUD műveletekhez tartoznak. A vezérlők használják a szolgáltatás réteget a műveletek végrehajtásához és a válasz elkészítéséhez.
5. DTO-k (Data Transfer Object): DTO-kat használunk az adatok közvetítésére az egyes rétegek között, illetve backend és a frontend között is. Ezek elkülönülnek az entitiktől, de hasonló hozzájuk. Ugyanúgy adatmodellként funkcionálnak, de teljesen egyedi módon lehet őket létrehozni annak függvényében mit akarunk benne közvetíteni.
6. Leképezések (Mapperek): Mappereket használunk ahhoz, hogy az entitásokat át tudjuk alakítani DTO-kra és fordítva.



```
> controller
> dto
> entity
> exceptions
> mapper
> repository
> service
```

4.5. ábra Az alkalmazás backendjének fontosabb könyvtárai

```

@Entity @Data @Table(name = "position", schema = "public" )
public class PositionEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Long id;
    @Column(name = "title")
    private String title;
    @Column(name = "lead")
    private String lead;
    @Column(name = "description")
    private String description;
    @Column(name = "active")
    private Boolean active;
    @Column(name = "salary")
    private String salary;
    @Column(name = "create_date")
    private Date createDate;
    @Column(name = "start_date")
    private Date startDate;
    @Column(name = "end_date")
    private Date endDate;
    @ManyToOne(cascade = CascadeType.DETACH)
    @JoinColumn(name = "FK_COMPANY_ID")
    private CompanyEntity company;
}

```

*Entity osztály*

```

@Repository
public interface PositionRepository extends JpaRepository<PositionEntity,
Long> {
    List<PositionEntity> findAll();
    List<PositionEntity> findByCompany_Id(Long companyId);
    @Query(value = "SELECT * FROM public.position p " +
        "JOIN public.company c ON p.fk_company_id = c.id " +
        "JOIN public.accreditation_mapping a ON a.fk_company_id = c.id "
+
        "JOIN public.study_program sp ON a.fk_study_program_id = sp.id "
+
        "WHERE sp.id = :studyProgramId", nativeQuery = true)
    List<PositionEntity> findAllByAccreditedStudyProgram(Long
studyProgramId);
}

```

*Repository mely kiterjeszti a JPA repositoryt*

```

public interface PositionService {
    /**
     * Creates Position
     * @param positionDto
     */
    PositionDto create(PositionDto positionDto);
}

```

*Service interface*

```

@Override
    public PositionDto create(PositionDto positionDto) {
        if(positionDto.getTitle() == null){
            throw new IllegalArgumentException("Title cannot be null at
position create !");
        }
        /* Check all required field */
        List<JobSkillsDto> jobSkillsDtos = positionDto.getJobSkillsDtos();
        PositionEntity newPosition = new PositionEntity();
        if(positionDto.getCompany() != null){
            Optional<CompanyEntity> companyOptional =
companyRepository.findById(positionDto.getCompany().getId());
            if(companyOptional.isEmpty())
                throw new EntityNotFoundException("Company was not found with
the provided id");
            CompanyEntity selectedCompany = companyOptional.get();
            Calendar cal = Calendar.getInstance();
            Date createDate = cal.getTime();
            newPosition.setTitle(positionDto.getTitle());
            /* Check all required field */
        }
        PositionEntity positionEntity = positionRepository.save(newPosition);
        List<JobSkillsDto> jobSkillsDtoList = new ArrayList<JobSkillsDto>();
        for(JobSkillsDto jobSkillDto: jobSkillsDtos){
            if(jobSkillDto.getJobRequirementOptionDto().getId()==null){
jobSkillDto.setJobRequirementOptionDto(jobRequirementOptionService.create(job
SkillDto.getJobRequirementOptionDto()));
                JobSkillEntity positionRequirementMappingEntity = new
JobSkillEntity();
                positionRequirementMappingEntity.setPosition(positionEntity);
                positionRequirementMappingEntity.setJobRequirementOption(jobRequirementOption
Mapper.toEntity(jobSkillDto.getJobRequirementOptionDto()));
                positionRequirementMappingEntity.setPriority(jobSkillDto.getPriority());
                positionRequirementMappingRepository.save(positionRequirementMappingEntity);
                jobSkillsDtoList.add(jobSkillDto);
            }
            else{
                JobSkillEntity positionRequirementMappingEntity = new
JobSkillEntity();
                positionRequirementMappingEntity.setPosition(positionEntity);
                positionRequirementMappingEntity.setJobRequirementOption(jobRequirementOption
Mapper.toEntity(jobSkillDto.getJobRequirementOptionDto()));
                positionRequirementMappingEntity.setPriority(jobSkillDto.getPriority());
                positionRequirementMappingRepository.save(positionRequirementMappingEntity);
                jobSkillsDtoList.add(jobSkillDto);
            }
        }
        PositionDto newPositionDto;
        newPositionDto = positionMapper.toDto(positionEntity);
        newPositionDto.setJobSkillsDtos(jobSkillsDtoList);
        return newPositionDto;
    }

```

*Create függvény a Service implementációban*

```

@Controller
@RequestMapping("/position")
@CrossOrigin
@AllArgsConstructor
public class PositionController {
    private final PositionService iPositionService;
    @PostMapping("/create")
    public ResponseEntity<?> create(@RequestBody PositionDto positionDto) {
        try {
            return
ResponseEntity.ok().body(iPositionService.create(positionDto));
        }
        catch (Exception e) {
            return ResponseEntity.badRequest().body(e.getMessage());
        }
    }
}

```

*Controller*

#### 4.4.4. Ajánló rendszer implementáció

A backendnél fontos megemlíteni az általam implementált ajánló rendszert is.

Az ajánló rendszeremet a szakirodalmi tanulmány alapján terveztem meg és a Cosinus hasonlósági algoritmust, illetve a Jaccard indexet használja a pozíciók ajánlására a hallgatók számára.

Mivel mind a kettő algoritmusnak meglehetnek a saját hátrányai, így a legjobb megoldásnak azt láttam, hogy a két algoritmust ötvözzem. Ebből adódóan tehát implementáltam a rendszerembe úgy a Jaccard-indexet, mint a CosSim algoritmust.

Ha a hallgató a neki ajánlott pozíciókat kéri, az algoritmus először is lekéri az adatbázisból az összes olyan meghirdetett pozíciót, amelyet valamelyik a hallgató szakja által akkreditált cég tett közzé. Ezáltal már egy előzetes szűrést végzünk a pozíciókon. Ezt követően lekérjük a hallgató által beállított elvárásokat és képességeket. A hallgató elvárásait és képességeit ezután vektorizáljuk és elmentjük egy változóba.

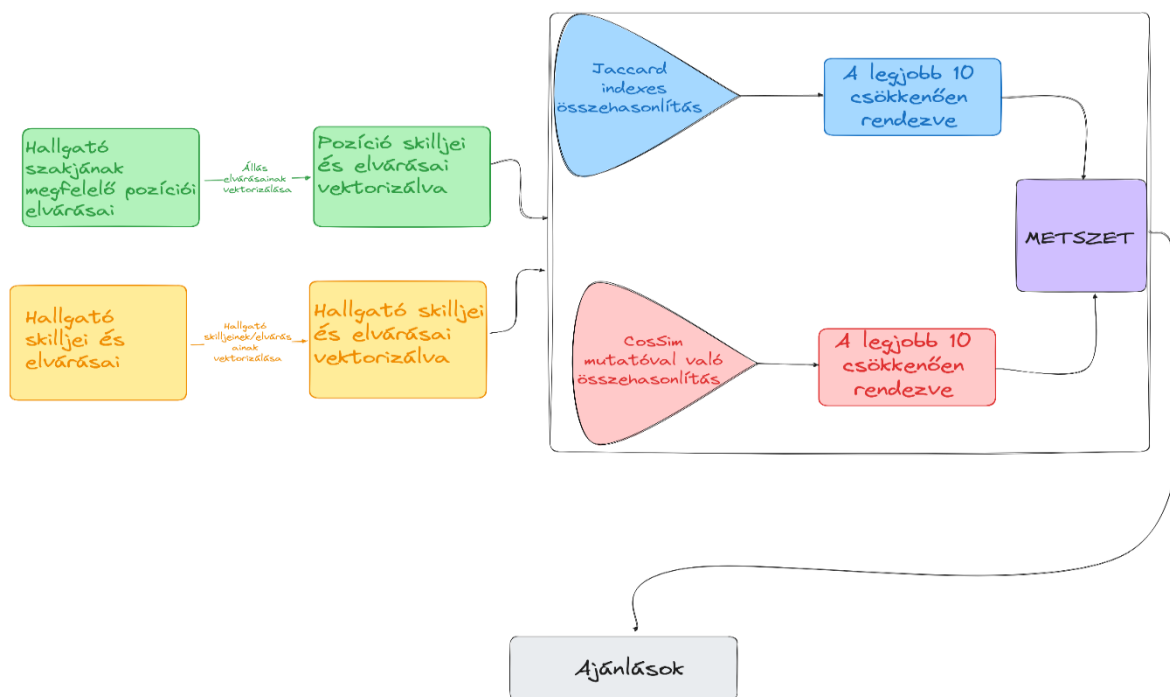
A vektorizálás bemutatása előtt fontos elmondani, hogy a hallgató és a cégek is az elvárásokat és a képességeket priorizálni tudja százalékos arányban. Erre azért van szükség, mert ez alapján az algoritmus figyelembe tudja venni azt, hogy egy hallgató vagy egy pozíció számára mi-milyen arányban fontos vagy nem fontos. Például egy diák képességei között szerepel a Java és a Dart is, azonban a szakmai nyárigyakorlat ideje alatt, inkább a Javát gyakorolná céges környezetben. Ilyenkor megteheti azt, hogy a Javának a prioritását felviszi 75%-ra míg a Dart prioritását 25-re állítja. Ilyenkor a rendszer elsődlegesen azokat a pozíciókat fogja ajánlani, amely esetében az elvárások prioritása hasonló.

A vektorizálás úgy működik, hogy létrehozok egy inicializáló tömböt, az összes lehetséges képesség id-jával, minden id-hoz hozzá mappelve a 0-át, amennyiben a képesség nem is található meg a hallgató (vagy pozíció esetében a pozíció) felsorolt képességei vagy elvárásai között, illetve az adott prioritást, ha a képesség fel van vezetve az adott hallgatóhoz (vagy pozíció esetében a pozícióhoz).

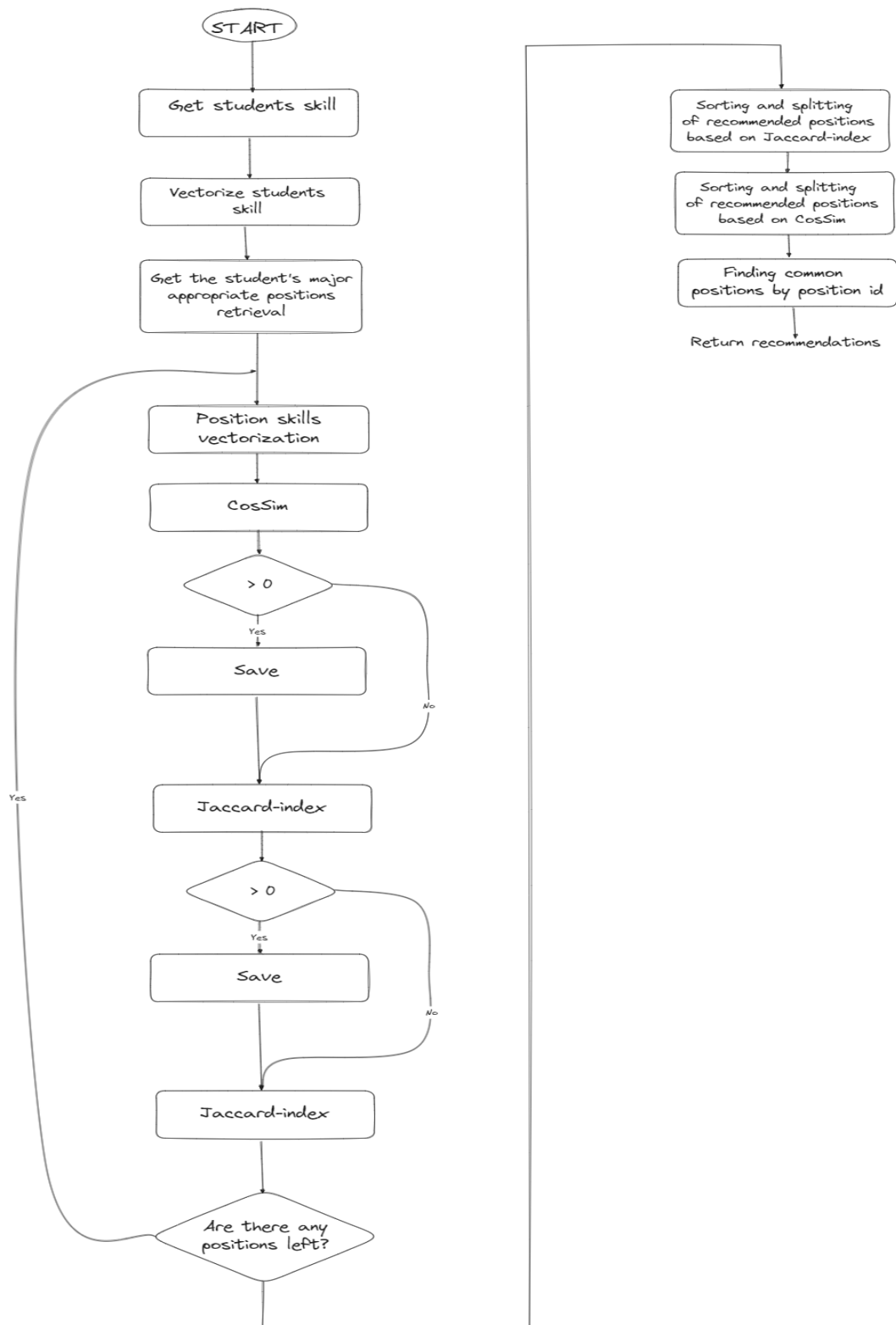
A lekért pozíciókon végig iterálva elkészítem a pozíció inicializáló tömbjét. Ezt az inicializáló tömböt először odaadom a Cosinus hasonlóságot kiszámító algoritmusomnak, majd a Jaccard-indexet kiszámító algoritmusomnak. A két eredményt elmentem egy-egy külön tömbbe a pozícióval együtt, természetesen csak akkor, ha a kapott eredmény nem 0.

Az iteráció lejárta után a rendszer csökkenően rendezi a kapott eredményeket, majd a legjobb 10-10 eredmény közös elemeit visszatéríti. Ez azt jelenti, hogy van két darab maximum 10 elemű listám, amelyek közül az első tartalmazza a Jaccard index alapján ajánlott állások közül a 10 legjobbat, a második pedig a Cosinus hasonlóság alapján ajánlott állások közül a 10 legjobbat, és ebből a két listából visszatérítem azokat az állásokat, amelyek mind a két listában szerepelnek. A közös listában az állások hasonlóság értékéhez a Cosinus hasonlósági százalékot társítom. Ezek tehát már az ajánlott állások

Az ajánló rendszer működési elvét a következő 4.6 ábra mutatja be.



4.6. ábra Ajánló rendszer működési diagrammja



4.7. ábra Az ajánlórendszer folyamatábrája

Kód szinten az ajánlórendszeremet a függelék [1-es pontja](#) mutatja be.



## 4.5. Frontend

### 4.5.1. Konfigurálás

Egy Angular applikáció létrehozása egy egyszerű utasítás lefuttatásából áll, melyet az Angular CLI segítségével kell megtenni. Ez a kód a következő: **ng new name-of-app**. Ezzel létrejön egy alap alkalmazás mely tartalmazza a struktúrát, és a kötelező node moduleokat, melyek ahhoz kellenek, hogy az applikáció elinduljon. Ezután már futtathatjuk is a következő CLI utasítások egyikével: **ng serve**, **npm start**.

Az én applikációm esetében én, egy megvásárolt templateből indultam ki, aminek a kiválasztását sok keresgélés előzte meg. A templatek nagy előnye, hogy sok olyan dologgal nem kell már időt vesztegetni, mint különböző pluginok keresése és verzió összeegyeztetése, ad egy struktúrát a projektnek, aminek köszönhetően egységes felületeket lehet létrehozni, illetve ami szintén fontos, hogy a templateket gyakran frissítik, így, ha kijön egy új Angular verzió, vagy valamelyik plugin elavulttá válik, akkor azt frissítik vagy javítják. Ezáltal sok idő spórolható meg, hogy a funkcionalitásokra koncentrálhassunk. Az én templatem a themeforest.net envatomarket oldaláról volt megvásárolva.[22]

### 4.5.2. Felépítés

Egy Applikáció komponens-alapú architektúrával rendelkezik [7], [8]. Ez azt jelenti, hogy az alkalmazásunkat valamilyen elvek mentén, komponensekre osztjuk. Az elv általában nem más, mint hogy milyen oldalak és nézetek vannak. Ezen felül ez is külön komponensekre bontható, a kód bázis rövidítésének érdekében, megoldható, hogy bizonyos elemeket egy nézetből külön komponensekre bontunk, melyeket paraméterekkel tudunk ellátni, így az újra felhasználhatóvá válik. A komponenseket ezen felül tovább is lehet csoportosítani, úgynevezett modulok segítségével. A modulok szervező szerepet látnak el.

Ezek a modulok mind-mind az **src** mappában található **App** nevű mappában vannak. Az **App** egy module, amely, ha úgy vesszük az alkalmazásunk gyökerét képezi, ebben van beimportálva minden más module.

Az én alkalmazásomban a modulok a szerepek alapján vannak megalkotva, illetve van még három másik modulom, amely nagyjából bármelyik Angular alkalmazásban megvan. Van külön egy modulom a hallgató komponenseinek (**StudentModule**), a cég komponenseinek

(**CompanyModule**) és egy külön az egyetem komponenseinek (**UniversityModule**), amibe beletartoznak a szakkoordinátorok is. Ezeken felül van egy úgynevezett **SharedModule** is, amely az olyan komponenseket tartalmazza, amelyek minden modulban elérhetőek kell legyenek, például a header vagy a footer komponense, illetve az értékelések megjelenítésért felelős csillagos komponensem is. Van továbbá egy külön **WebModuleom**. Ez az a module, amely az alkalmazás kezdőlapját tartalmazza, és létrehozására azért volt szükség, mert sok olyan egyedi, inkább weboldalak esetében hasznos és fontos funkciót kellett implementálni, mint három nyelvésítése vagy egyéb vizuális effektek. Ezen felül a harmadik module, az **AccessModule**, amelye a loginnal és regisztrációval foglalkozó komponenseket gyűjti egybe.

Az **src** mappában helyet kap még a projekt három további fontos mappája, amelyek közül az egyik a **core**. Ez az a mappa, amely tartalmazza a DTO modelleket, a szolgáltatásokat, illetve a csővezeték és egyéb felhasználású segédfüggvényeket, melyek segítségével például az szolgáltatások felügyeletét végzem.

Az **src** mappában a harmadik mappa nem más, mint az **assets**. Ez a mappa úgy tekinthető, mint mondjuk Javában vagy Kotlinban a resources. Minden kép, beépített file vagy egyéb külső erőforrás itt, ebben a mappában van eltárolva.

Az **src** mappában a negyedik mappa az **environments**. Az **environments** úgynevezett környezeti konfigurációs beállítások vannak elmentve. Erre azért van szükség mert egy alkalmazás fejlesztés alatt általában nem ugyanabban a környezetben fut, mint élesben, mondjuk egy szerveren. Itt be tudom állítani azt, hogy az alkalmazás milyen címen éri el a backendet, vagy egyéb olyan címeket melyek az alkalmazás működéséhez szükségesek.

#### 4.5.3. A felhasználói felület

A webalkalmazások fejlődése során egyre nagyobb hangsúlyt kap a felhasználói felület kialakítása és minősége. A felhasználói felület a webalkalmazás arca, amely közvetlenül kapcsolatot teremt a felhasználóval, ezért kritikus fontosságú a sikeres felhasználói élmény biztosítása szempontjából.

A koherens felépítés alapvető eleme a felhasználóbarát webalkalmazásnak. Az egyértelmű navigáció, logikus elrendezés és konzisztens megjelenés segíti a felhasználókat abban, hogy könnyen megtalálják, amit keresnek, és egyszerűen eligazodjanak az alkalmazásban. A logikus menürendszer, a kategóriák jól definiált elhelyezése és az információ könnyen érthető módon történő megjelenítése mind hozzájárulnak a felhasználói felület koherenciájához.

A reszponzivitás a modern webalkalmazások alapvető követelménye. A felhasználók különböző eszközöket használnak a webalkalmazások eléréséhez, mint például asztali számítógépek, laptopok, tabletek és okostelefonok. A felhasználói felületnek alkalmazkodnia kell ezekhez az eszközök különböző képernyőméreteihez és kijelzőfelbontásaihoz. A reszponzív dizájn lehetővé teszi, hogy az alkalmazás átméretezze és átrendezze az elemeket, hogy optimális felhasználói élményt nyújtson bármilyen eszközön.

A felhasználóbarát viselkedés az egyik legfontosabb tényező a felhasználói felület tervezése során. Az alkalmazásnak intuitívnak kell lennie, és a felhasználói műveleteknek természetesnek kell tűnniük. Például a gomboknak és linkeknek megfelelően kell reagálniuk a kattintásra, és az interakcióknak gyorsnak és zökkenőmentesnek kell lenniük. A hibák esetén a felhasználók számára világos és érthető visszajelzést kell adni, hogy segítsék őket a problémák megoldásában.

A vizuális megjelenés a felhasználói felület egy másik alapvető eleme. Az esztétikus és vonzó dizájn vonzza a felhasználókat, és pozitív érzéseket kelt bennük az alkalmazással kapcsolatban. Az egységes színpaletta, az élénk és jól látható elemek, valamint a megfelelő tipográfia mind hozzájárulnak a felhasználói felület vizuális vonzerejéhez. Emellett fontos figyelembe venni a felhasználói felület igényeit és a célcsoport preferenciáit is, hogy a megjelenés összhangban legyen az alkalmazás céljával és a felhasználók elvárásaival.

Ezen szempontok alapján terveztem meg tehát én is a felületemet, inspirálódva nagy és jól működő webalkalmazásoktól mint mondjuk a Prisma Platform vagy a LinkedIn.

Az alkalmazásom felületének arculati elemeit is úgy terveztem meg, hogy egy márkaként is működő képes legyen, mégis a felhasználók számára ne legyenek zavaró a rikító színek, vagy a túldizájnlott logóval.

Az applikáció interakciói egyértelműek, például egy űrlap vagy egy művelet közben is, így jelzi a felhasználónak, ha például valamit nem, vagy helytelenül töltött ki, de arról is kap visszajelzést, ha a műveletet sikeresen elvégezte. Az alkalmazás felülete figyelmeztet, ha a felhasználó olyat próbál végrehajtani, ami visszavonhatatlan (például egy állás törlése).

A felhasználói felületről részletesebb képeket a függelékek [2. pontjában](#) lehet találni.

## 4.6. Verziókövetés és projektmenedzsment és egyéb fejlesztői eszközök

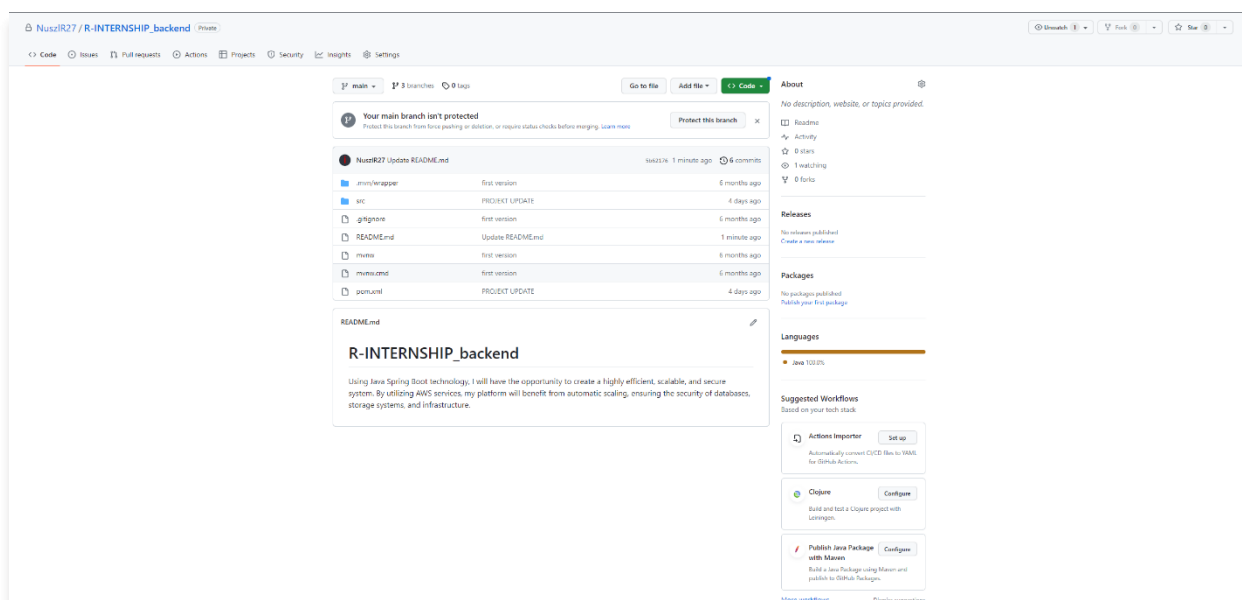
### 4.6.1. A GitHub

A projekt fejlesztése során a verzió követő rendszer, amit használtam az nem volt más, mint a GitHub [22].

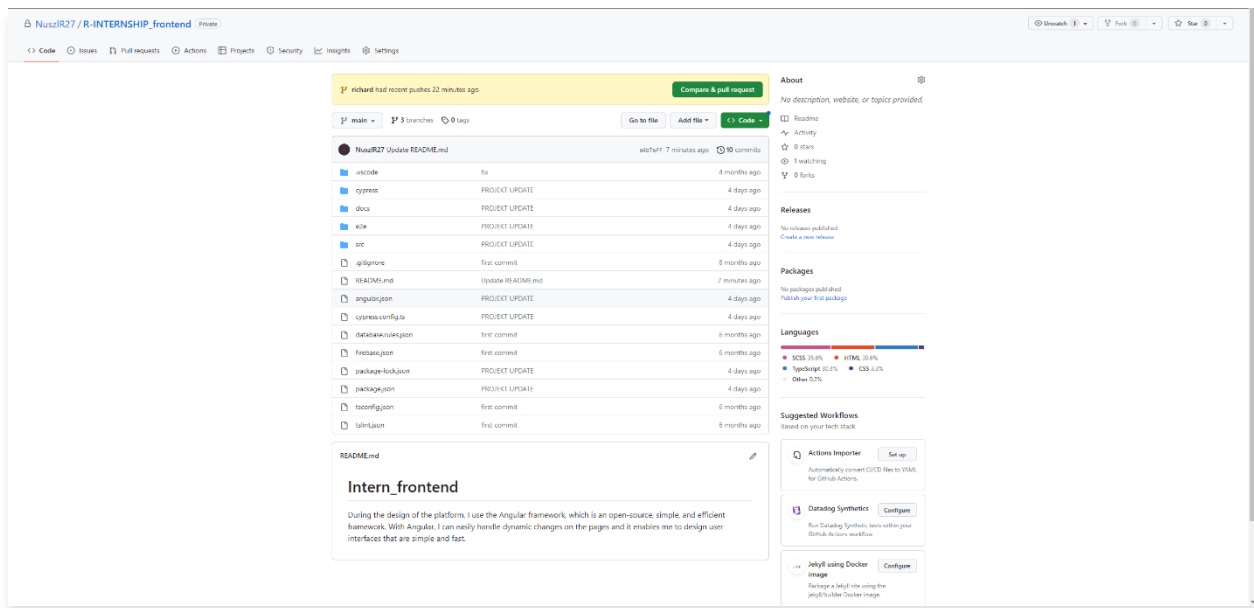
A GitHub egy verziókövetési platform, amely rengeteg előnnyel jár a szoftverfejlesztők számára. Az egyik legfontosabb előnye, hogy egy központi helyet biztosít a projekt dokumentációjának, kódjának és erőforrásainak tárolására.

A GitHub használata során, lehetőségem volt létrehozni 2 branchet, illetve egy **main** branch alpból adva van. A két branch, amit létrehoztam a **richard** és a **dev**. A **richard** branchet használtam arra, hogy az aktuális állapotot, amelyben még lehetnek bugok, de nem szintaktikai hibák elmentsem, majd, ha a funkciót késznek és működőképesnek ítéltém, akkor került be a **dev**-be. A main branch tartalmazza az alkalmazás aktuálisán kész verzióját például amelyik a szerveren van.

Mivel a projektem két nagy részből (frontend és backend) áll, melyeket nem szerettem volna egy repository segítségével tárolni, ezért két repositoryt hoztam létre, egyet a backendnek és egyet a frontendnek.



4.8. ábra A backend repositoryjának dashboardja



4.9. ábra A frontend repositoryjának dashboardja

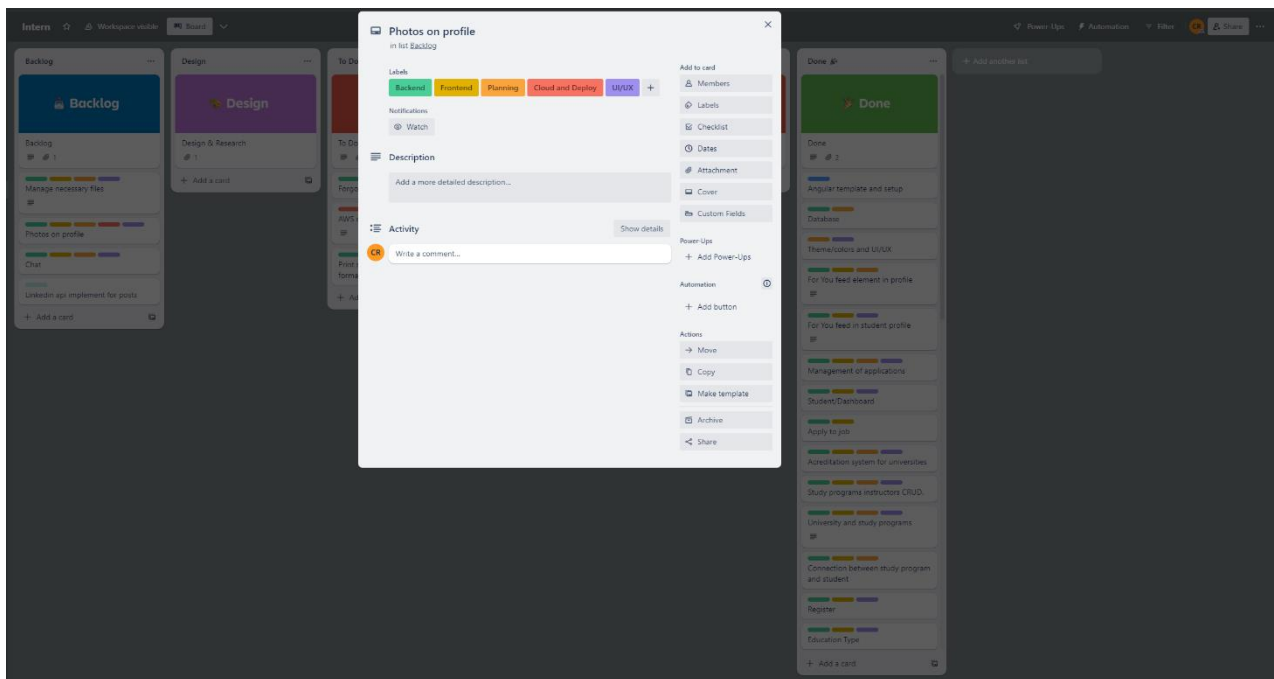
#### 4.6.2. Trello

A Trello [23] egy népszerű projektmenedzsment eszköz, amely segít a csapatoknak szervezettebben és hatékonyabban dolgozni. A Trello alapja a kanban tábla, amelyen a taskokat kártyák formájában lehet nyomon követni. A kártyák könnyedén mozgathatók, így a csapatok átláthatják a folyamatokat és következetesen haladhatnak a projektekkel.

A Trello nagy előnye, hogy a taskokat címkékkel lehet ellátni, amelyek segítségével meg lehet például határozni, hogy az adott task milyen elvárásokat támaszt a fejlesztő felé, de akár ellenőrző listát vagy csatolmányt is lehet a taskhoz kötni, hogy a fejlesztés során már egyértelmű legyen az adott task.

A Trellon belül én 6 nagy listát hoztam létre, amibe az egyes taskok kerülhetnek. Ezek a listák a következők:

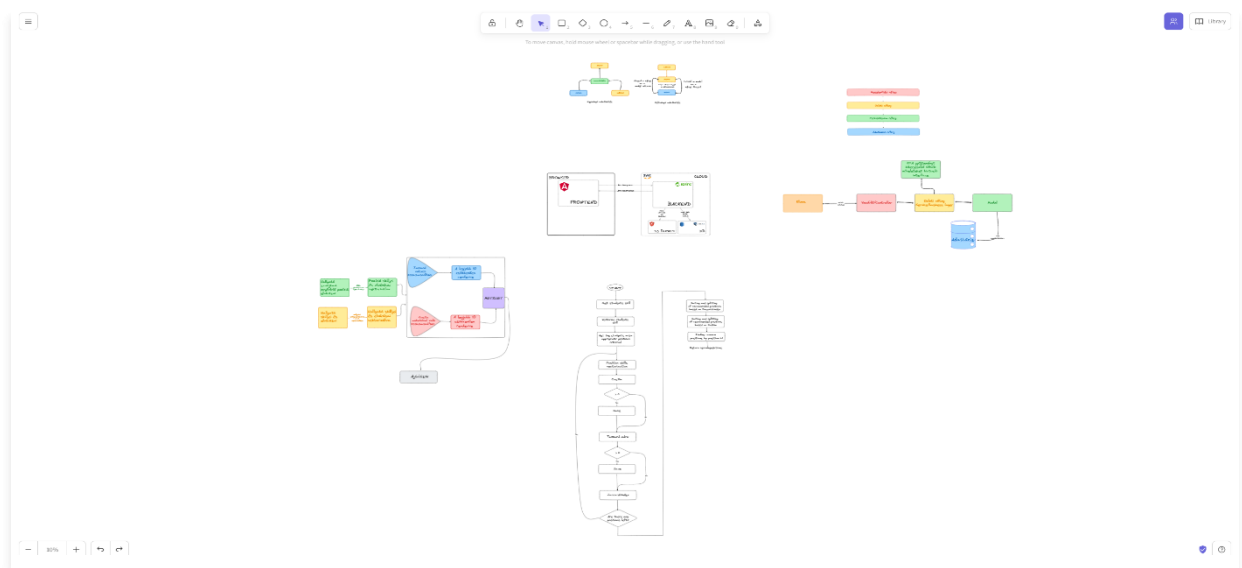
1. Backlog: Ez tartalmazza a projekt során az összes taskot, amely elvégzésre vár.
2. Design: Ez tartalmazza azokat a taskokat, amelyek éppen tervezés alatt állnak
3. ToDo: Ez tartalmazza aktuálisan azokat a taskokat, amelyeket egy adott időszakban el kell végezni
4. Doing: Ez tartalmazza aktuálisan azokat a taskokat, amelyek készülnek
5. Testing: Ez tartalmazza azokat a taskokat, amelyek be lettek fejezve, azonban még tesztelés alatt állnak
6. Done: Ez tartalmazza azokat a taskokat, amelyek be lettek fejezve és le is lettek tesztelve



4.10. ábra Trello board és az általam használt címkézések

### 4.6.3. Excalidraw

Vázlatok és diagrammok elkészítésére egy online elérhető eszközt, az Excalidrawt [24] használtam. Az Excalidraw egy online rajzolóeszköz, amely lehetővé teszi a felhasználók számára egyszerű és intuitív módon készíteni képernyőterveket, vázlatokat és diagramokat. Az Excalidraw interaktív és eszközbarát felülettel rendelkezik, amely lehetővé teszi a felhasználók számára a szabadkezes rajzolást, elemek mozgatását, méretezését és azok közötti kapcsolatok létrehozását.



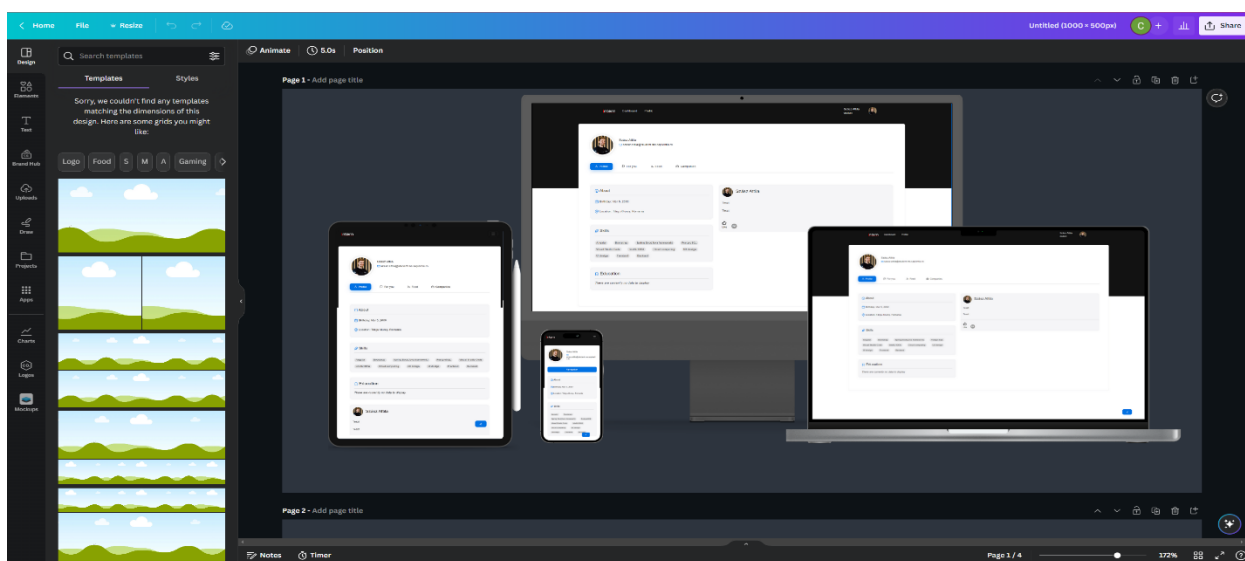
4.11. ábra Excalidraw felület

#### 4.6.4. Canva és Storyset

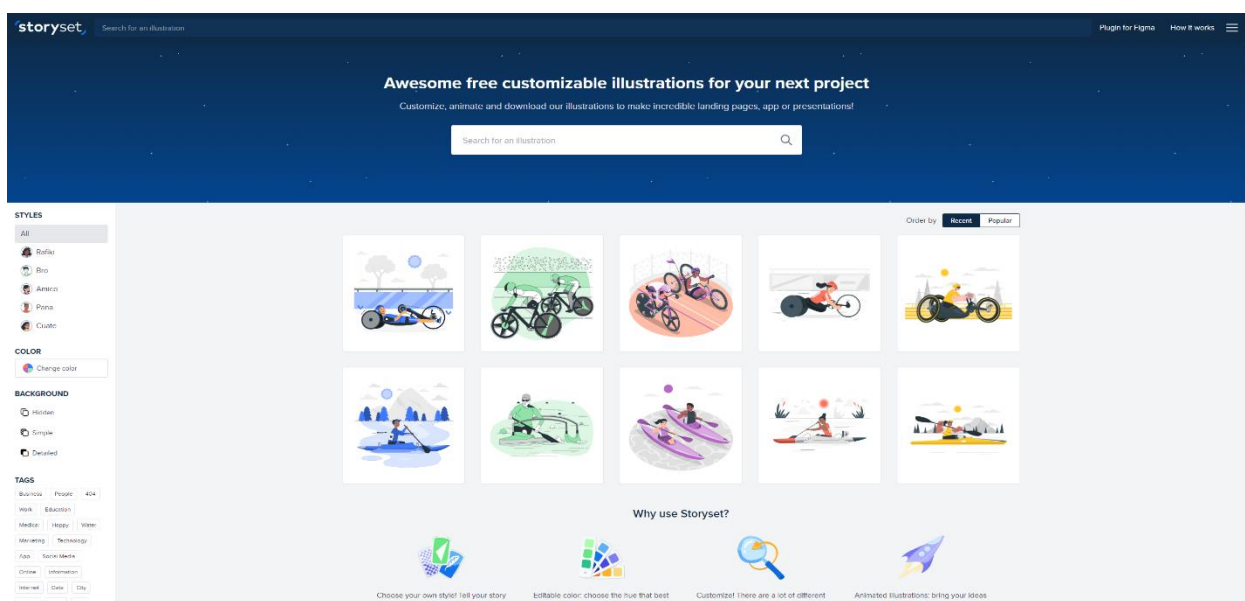
A platform illusztrációinak és képeinek szerkesztéséhez két webalkalmazást használtam. Ez a két alkalmazás a Canva és a Storyset.

A Canva egy remek eszköz akár egy teljes márkaaarculat megtervezésére is, a logótól kezdve a színeken keresztül, egészen a betűtípusig. Remek illusztrációkat és designokat lehet rajta találni, amelyek segítségével látványterveket is össze lehet könnyedén állítani.

A Storyset szintén egy remek webalkalmazás, amely teljesen ingyenes. Minőségi és koherens illusztrációkat lehet a segítségével megtalálni témák szerint csoportosítva, amelyeket testre lehet szabni a megfelelő színekkel, majd a megfelelő formátumba ki lehet exportálni.



4.12. ábra Példa az oldalon szereplő egyik kép elkészítésére Canvaban



4.13. ábra Storyset kereső

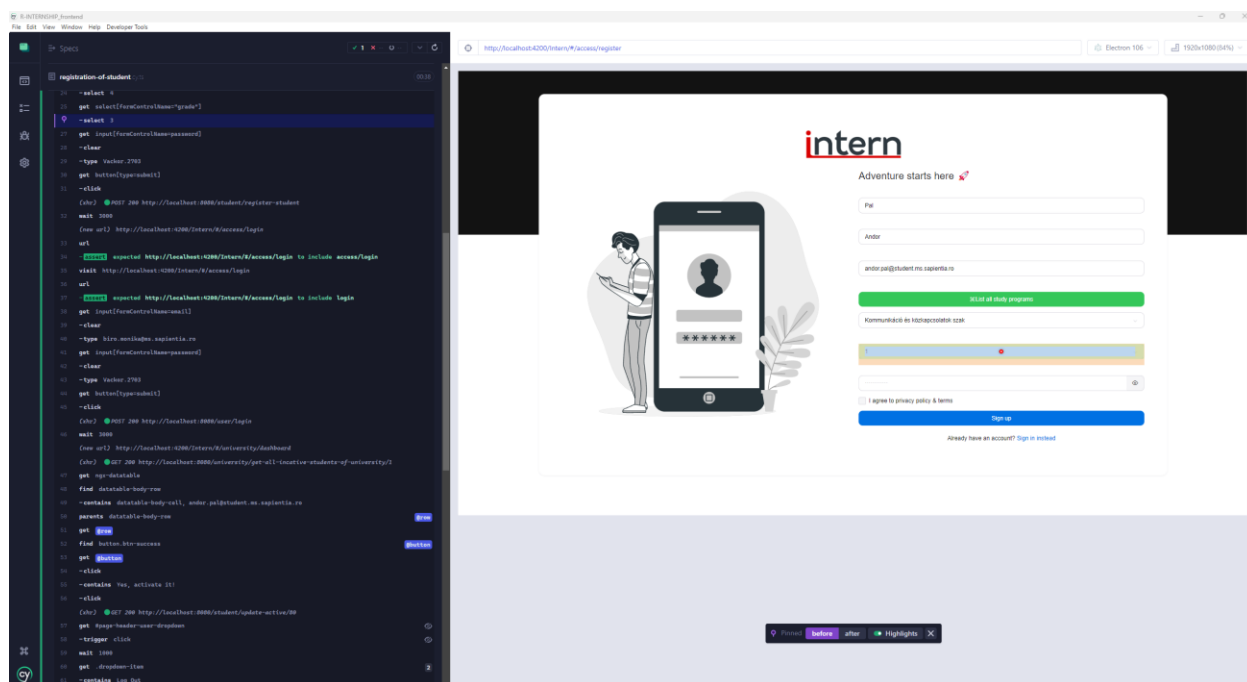
## 4.7. Tesztelés

Az alkalmazásomon belül van néhány olyan folyamat, amely fontos, zökkenőmentesen működjön. Ezeket a folyamatokat úgynevezett szcenárió teszteléssel lehet ellenőrizni, hogy élesben is működik-e.

A szcenárió tesztelés egy olyan tesztelési módszer, amely során a tesztek egy adott üzleti folyamat vagy felhasználói szcenárió alapján kerülnek összeállításra. Ez a tesztelési módszer általában az alkalmazás funkcióinak teljes körű tesztelését jelenti, beleértve a frontend és backend részeket is, azonban a tesztek fókuszja a felhasználói szemszögből történő tesztelésre kerül.

Az én projektem esetében egy diák beregisztrálásának szcenárióját teszteli le. A platformom esetében, egy adott egyetem diákjaként ingyenesen lehet beregisztrálni, az egyetemi email cím segítségével. Erre azért van szükség, hogy a diák általános adatait ne az egyetem megbízottja kelljen bevigye a rendszerbe, ezzel is sok időt és pénzt megtakarítva, hanem minden diáknak lehetősége legyen erre.

Természetesen, az egyetem megbízottja egy jóváhagyást kell adjon a regisztrációra, ezzel kiküszöbölve azt, hogy bárki regisztráljon. Ha a jóváhagyás megtörtént, akkor a diák beléphet a platformra, és máris keresgélhet az állások között.



4.14. ábra A szcenárió tesztelés egyik állapota



## 5. Üzembe helyezés és kísérleti eredmények

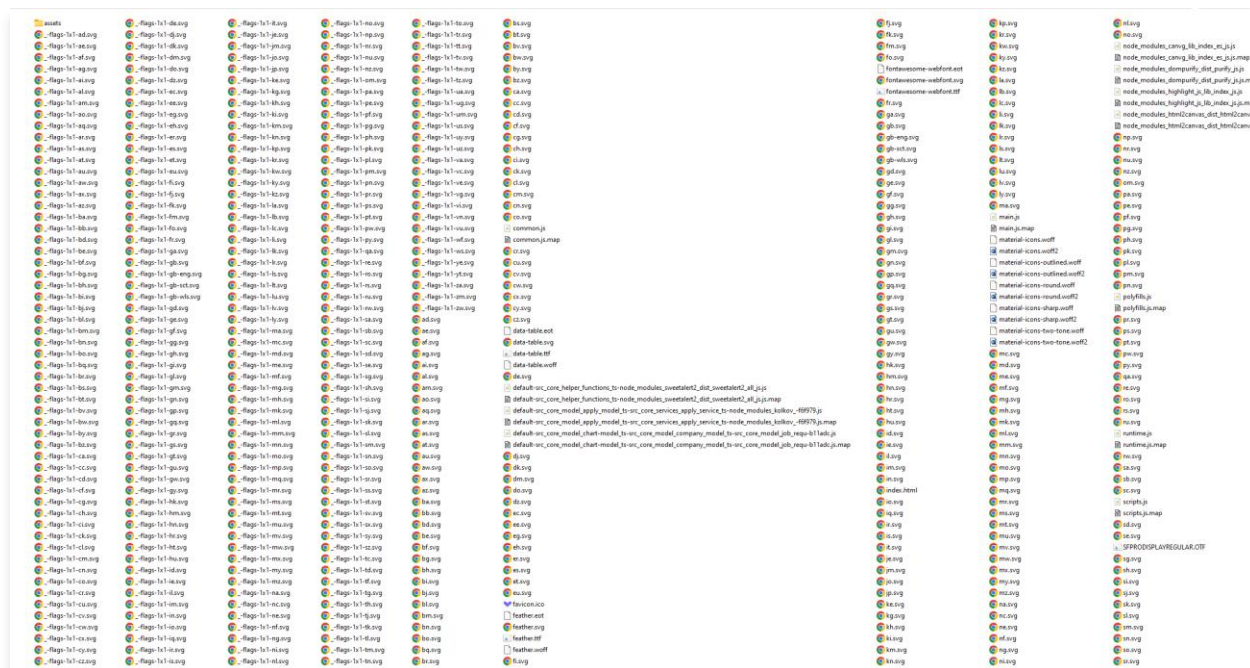
### 5.1. Üzembe helyezési lépések

A software egységei külön folyamatokon kell végig menjenek az üzembe helyezés előtt.

#### 5.1.1. A frontend

A frontend egy Angular alkalmazás, amelyet a szerveren való futtatás érdekében egy buildelési folyamaton kell keresztülvinni. Ehhez a következő pontokat kell követni

- cloneozzuk le a frontend GitHub repositoryját és lépünk át a main branchre
- az **environment.prod.ts** nevű fileban meg kell adni a backend elérhetőségét, illetve egyéb olyan elérési utakat, amelyek ott definiálva vannak
- futtassuk le a következő Angular CLI utasítást: **npm run build**. Ez a parancs buildeli az alkalmazást, azaz JavaScript futtatható állománycsomagot hoz létre a kódbázisunkból, és az éles futásnak megfelelő elérési utakat használja
- nyissuk meg az alkalmazás újonnan létrejött **dist** mappáját és jelöljük ki a tartalmát majd tömörítsük **.zip** formátumba
- nyissuk meg az Angular applikáció futtatására képes szerverünk **cpanel** felületét
- a már konfigurált **domain public\_html** folderébe töltsük fel a tömörített applikációt



5.1. ábra A dist könyvtár a buildelés folyamatát követően

### 5.1.2. Backend

A backend egy Java Spring Boot alkalmazás [25], amit az AWS EC2 szolgáltatásán kell futtatni. Ehhez a következő pontokat kell követni:

- cloneozzuk le a backend GitHub repositoryját és lépünk át a main branchre
- Spring Boot projektjének hozzuk létre a **EC2**-re telepíthető **WAR** fileját
- hozzunk létre egy **EC2** példányt az **AWS** konzolon
- telepítsük a **Java**-t és a **Tomcat** szerverét az EC2 példányra
- adjunk engedélyt a felhasználónak a Tomcatban a "Manage apps" hozzáféréséhez a grafikus felületen
- távolítsuk el az alapértelmezett localhost URL-t
- telepítsük az elkészült **WAR** filet

## 5.2. Felmerült problémák és megoldásaik

Amennyiben a frontendet megpróbáljuk az **npm start** paranccsal ám az nem sikerül, futtassuk le az **npm install** parancsot Angular CLI segítségével. Ez az összes külső függőséget telepíteni fogja, amelyek a **package.json** nevű fileban definiálva van.

Amennyiben a backendet megpróbáljuk lokálisan futtatni ám az nem sikerül akkor az valószínűleg azért van, mert az adatbázisunk lokálisan nem létezik. Ebben az esetben, nyissuk meg a **pgAdmin**t, és hozzunk létre egy új adatbázist, **internship** névvel.

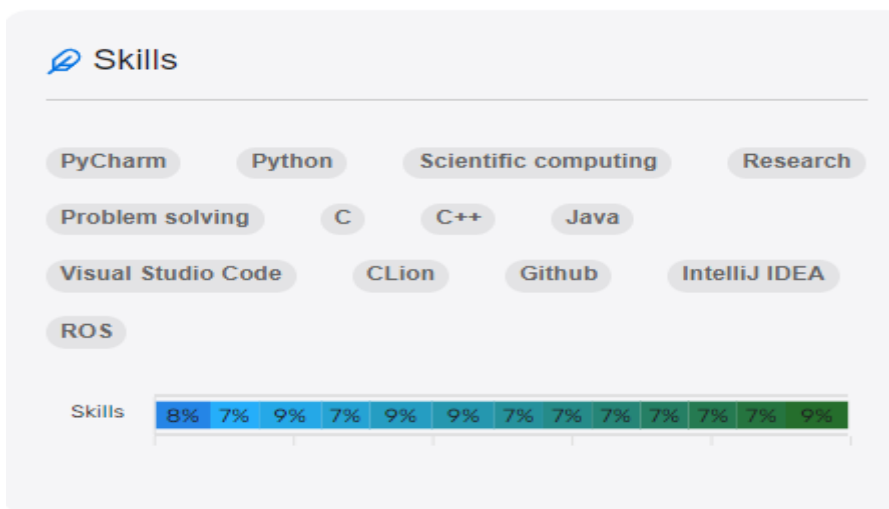
Amennyiben a backend továbbra sem akar elindulni, akkor abban az esetben próbáljuk a **Maven projektet** újra tölteni, majd egy **clean install** elindítása és lejárása után, indítsuk újra az applikációt.

### 5.3. Az ajánlórendszer tesztelési eredményei

- manuális teszteléssel történt az ajánlórendszer hangolása
- feltöltött állások és hallgatók segítségével
- széles választék a hallgatók képességei tekintetében és a kiírt állások között, amely által szélsőséges esetek is tesztelve lettek
- **11** hallgató, **20** meghirdetett állás
- **74** ajánlásból **64** volt helyes és **10** a szak miatt helytelen => ez így **86%** jó ajánlás

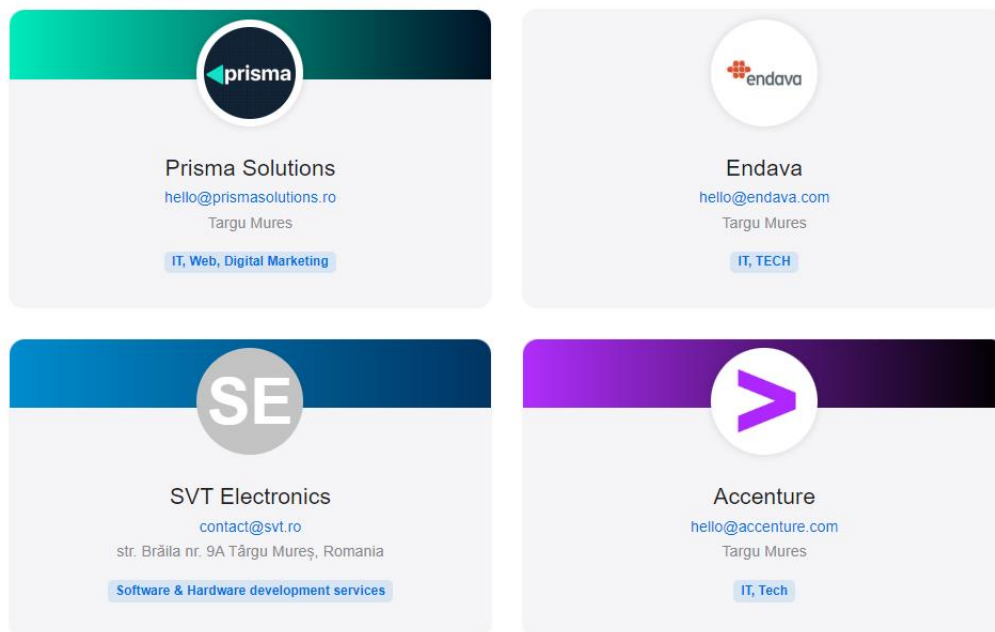
Fontos megemlíteni, hogy a 14%-os hibaarány, abból adódik, hogy létezhetnek olyan cégek is, amelyek több területen is hirdetnek meg állást. A példa esetben az egyik, alapvetően informatika cég, aki főként webalkalmazások és weboldalak fejlesztésével foglalkozik, közzétett SEO szakértő állást is, aminek az elvárásai között olyanok is voltak, amelyekkel adott esetben egy számítástechnika szakos vagy informatika szakos hallgató is rendelkezik. Így rendszer, mivel az ajánlat, a diák képességei szerinti hasonlóság miatt belefért a legjobb 10 ajánlásba, ajánlotta is azt. Ez a helyzet fordítottn is elképzelhető, így tehát világosan látszik, hogy szükséges még a rendszer további fejlesztése, amely a hasonló eseteket is kiszűri.

A tesztelés manuális feltöltéssel kezdődött. Létrehoztam 11 darab felhasználót, amelyeket véletlenszerűen rendeltem hozzá egyetemi szakokhoz (informatika, számítástechnika, kommunikáció és tájépítészet) és évfolyamokhoz. Ezek után, figyelembe véve a már kiválasztott szakokat, feltöltöttem minden hallgató profilját véletlenszerűen képességekkel. Például a [testfirstname1.testlastname1@student.ms.sapientia.ro](mailto:testfirstname1.testlastname1@student.ms.sapientia.ro) email címhez tartozó profil tulajdonosa, aki informatika szakos és a következő képességekkel rendelkezik, 5.2-es ábra



5.2. ábra TestFirstName1 TestLastName1 hallgató képességei

A rendszer ennek megfelelően olyan állásokat kell ajánljon, amelyeket olyan cégek tettek közzé, amelyek akkreditálva vannak a szakkoordinátor által, 5.3-as ábra, és ezen belül is, amelyek a hallgató képességei alapján megfelelnek.



5.3. ábra Informatika szakhoz akkreditált cégek

A várakozásoknak megfelelően a hallgató ([testfirstname1.testlastname1@student.ms.sapientia.ro](mailto:testfirstname1.testlastname1@student.ms.sapientia.ro)), a következő ajánlásokat kapta:

- SVT Electronics: Junior Robotikai Mérnök, 74%-os megegyezés
- SVT Electronics: Junior Szoftverfejlesztő, 55%-os megegyezés
- SVT Electronics: Rendszergazda, 37%-os megegyezés
- Prisma Solutions: Spring Boot developer, 17%-os megegyezés
- Prisma Solutions: SEO szakértő, 13%-os megegyezés
- Prisma Solutions: Digital marketing asszisztens, 12%-os megegyezés
- Prisma Solutions: Szoftvertervező mérnök, 9%-os megegyezés
- Prisma Solutions: Angular developer, 7%-os megegyezés
- Accenture: Tesztmérnök, 5%-os megegyezés

Ebben a példában tehát a Junior Robotikai Mérnök és a Junior Szoftverfejlesztő a legmegfelelőbb ajánlat, azonban például a diák képességei alapján megfelelő lenne számára egy SEO szakértő pozíció is, ami azonban, ha a diák szakját figyeljük már nem megfelelő. Ez amiatt van, hogy például a SEO szakértő pozíciót egy olyan informatikai cég tette közzé, amelyet a szakkoordinátor jóváhagyott, a hallgató pedig rendelkezik néhány olyan képességgel, amely a pozíció betöltéséhez szükséges lehet. Így a rendszer által visszatérített ilyen pozíciókat hibás ajánlásként kezelem.

[illegible]

5.4. ábra Az ajánlómotor eredményességének mérése

## 6. A rendszer felhasználása

A rendszer használata három célcsoportnak nyújthat segítséget.

Az első célcsoport az egyetemi hallgatóság, akik számára fontos a szakmai fejlődés, nem csak elméleti de gyakorlati téren is. A platform segítséget nyújt számukra abban, hogy az elvárásaikhoz és a tudásukhoz kapcsolódó állásajánlatokat találjanak, illetve akár egy első munkahelyhez is jussanak, amely az egyetemi évek alatt és után is nagy előnyt jelent számukra a munkaerő piacon és a személyes fejlődésükben is.

A másik célcsoport a céges világ. Arról ugyan nincsen pontos adat, hogy a Romániában jelenlévő cégek közül átlagosan mennyit költenek tanításra, illetve betanításra, vagy esetleg mennyit költenek szakmai nyárigyakorlatok szervezésére, viszont az köztudott, hogy minden cég, a felvétel utáni egy-másfél hónapot betanításra szánják. Ez az egy-másfél hónap a cégek esetében lecsökkenthető nagyjából két hétre, ha olyan alkalmazottakat vesznek fel, akik az adott cégnél voltak már szakmai nyárigyakorlaton vagy esetleg pont egy szakmai nyárigyakorlat után ajánlottak neki állást. Ha ezt a gondolatmenetet követjük, akkor a cégeknek, már csak az oldalban levő reklám lehetőségek mellett is megéri, ha ezáltal lecsökkenthetik az új alkalmazottak betanítására szánt időt és pénzt.

A harmadik célcsoport az egyetemek és azok adminisztrátorai, akik a leegyszerűsített folyamatoknak hála, könnyedén, időt és ezáltal pénzt spórolva tudják a diákok által elvégzett szakmai nyárigyakorlatot értékelni, és megszerezni, a visszajelzéseket átnézni, ezzel a folyamatot is tudják finomítani és a cégekkel való partneri kapcsolatot is felül tudják vizsgálni.

## **7. Következtetések**

### **7.1. A rendszer állapota**

A rendszer felülete rendkívül könnyen használható, felhasználóbarát és reszponzív. Az intuitív tervezés és az egyszerű navigáció lehetővé teszi a felhasználók számára, hogy könnyedén megtalálják és használják a rendszer funkcióit. A felhasználói élmény kiemelkedő, hiszen a felület reagál az eszköz méretére és képességeire, így a felhasználók bármilyen készüléken, legyen az számítógép, tablet vagy okostelefon, zökkenőmentesen élvezhetik a rendszer előnyeit.

A rendszer kiválóan teljesíti a kiírt funkcionális követelményeket. A tervezés és fejlesztés során minden egyes követelményre figyelmet fordítottam, és az eredmény egy olyan rendszer, amely 100%-ban megfelel ezeknek az elvárásoknak. A felhasználók minden egyes funkciót használhatnak, amelyeket a rendszer előír.

Ezen felül, a rendszerbe sikeresen implementáltam egy jól működő ajánló rendszert, amely tartalom alapú. Ez a funkció lehetővé teszi a felhasználók számára, hogy személyre szabott ajánlásokat kapjanak a rendszer által.

A rendszer jelenleg a szerverre és a felhőbe való feltöltésre vár, ami azt jelenti, hogy a rendszer működőképés állapotban van, de még nem érhető el online vagy távoli eléréssel. Amint ezek a feltöltések megtörténnek, a rendszer bármikor és bárhol hozzáférhető lesz a felhasználók számára.

### **7.2. Hasonló rendszerekkel való összehasonlítás**

A piacon a legnagyobb professzionális orientált közösségi platform egyértelműen a LinkedIn.

A LinkedInhez hasonlóan az én platformomat is közösségi platformnak terveztem meg. Hasonlóan a LinkedInhez, itt is van céges és felhasználói profil, illetve az én alkalmazásom is alkalmaz tartalom alapú ajánló rendszert.

A LinkedIn nagy előnye, hogy az ajánlórendszer több megközelítést alkalmaz, nem csak tartalom alapút, hanem gépi tanulást is, illetve a felhasználói visszajelzéseket is tanulmányozza a minél pontosabb szűrés érdekében.

## **7.3. Továbbfejlesztési lehetőségek**

### **7.3.1. A platform**

Az továbbfejlesztési lehetőségek között számos fontos pont található, amelyek javíthatják az alkalmazás funkcionalitását és felhasználói élményét.

Az egyik lehetőség az egyéb külső rendszerek integrálása, például a Google Drive vagy a Moodle. Ez lehetővé teszi a felhasználók számára, hogy hivatalos papírjaikat rendezett formában tárolják és kezeljék az alkalmazáson belül.

Egy másik lehetőség a továbbfejlesztésre a cv kibővítése, például elkészült projektekkal, amelyeket fel lehetne tölteni a platformra PDF formátumban is.

A LinkedIn integrálása a cégek és hallgatók profiljainak szinkronizálásához szintén nagy előrelépést jelenthet. Ez lehetővé teszi a felhasználók számára, hogy a LinkedIn profiljukat összekapcsolják az alkalmazással, és automatikusan frissítsék az információkat. Ezáltal a felhasználók friss és pontos információkat tartalmazó profilokkal rendelkeznek mind a hallgatói, mind a vállalati oldalon. Ez a funkció nagymértékben segíthet a toborzási folyamatokban és a kapcsolatépítésben.

A chat implementációja egy másik kiemelkedő továbbfejlesztési lehetőség lenne. Ez lehetővé tenné a felhasználóknak, hogy közvetlenül kommunikáljanak egymással az alkalmazáson belül. A chat funkció egyszerű és gyors kommunikációt biztosítana a felhasználók között, legyen szó hallgatók vagy vállalatok közötti interakcióról. Ezáltal gyorsabban és hatékonyabban lehetne kommunikálni és kapcsolatot építeni.

Végül, egy mobilalkalmazás fejlesztése még nagyobb rugalmasságot és hozzáférhetőséget biztosíthatna a felhasználók számára. Ez lehetővé tenné számukra, hogy az alkalmazást bárhol, bármikor használják, ami kényelmesebb felhasználói élményt eredményezne, ráadásul a platformról érkező értesítéseket is azonnal megkapná a felhasználó.

### **7.3.2. Az ajánlórendszer**

Az alkalmazás ajánlórendszere jelenleg csak tartalom alapú ajánlást használ. A továbbfejlesztés során a platformot el lehet vinni abba az irányba, hogy az ajánlórendszer több módszert alkalmazzon, például szövegbányászatot és a közösségi platform nyújtotta előnyöket, valamint gépi tanulást is. Mivel a platformba már integrálva van egy értékelési rendszer, így például a közösségi platform előnyeinek kihasználása viszonylag könnyedén megoldható lenne.



## 8. Irodalomjegyzék

- [1] F. Ricci, *Recommender systems handbook*. New York, NY: Springer Science+Business Media, 2015.
- [2] D. Jannach, Szerk., *Recommender systems: an introduction*. New York: Cambridge University Press, 2011.
- [3] A. A. Amer és L. Nguyen, „Combinations of Jaccard with Numerical Measures for Collaborative Filtering Enhancement: Current Work and Future Proposal”.
- [4] „Building a Job Recommendation Strategy for LinkedIn and XING - Part1”. <https://www.linkedin.com/pulse/building-job-recommendation-strategy-linkedin-xing-part1-uppal> (elérés 2023. június 14.).
- [5] „Strategizing a Recommendation System on a Jobs Platform | Personalising Jobs Feed for LinkedIn - Part 2”. <https://www.linkedin.com/pulse/strategizing-recommendation-system-jobs-platform-feed-shaurya-uppal> (elérés 2023. június 14.).
- [6] S. Bhattacharya és A. Lundia, „MOVIE RECOMMENDATION SYSTEM USING BAG OF WORDS AND SCIKIT-LEARN”, *IJEAST*, köt. 04, sz. 05, o. 526–528, okt. 2019, doi: 10.33564/IJEAST.2019.v04i05.076.
- [7] Dhaduk H., „Angular vs React: Which to Choose for Your Front End in 2023?”, *Simform - Product Engineering Company*, 2023. április 6. <https://www.simform.com/blog/angular-vs-react/> (elérés 2023. június 14.).
- [8] „Angular”. <https://angular.io/> (elérés 2023. június 14.).
- [9] „9. Angular + IONIC”.
- [10] „Prettier · Opinionated Code Formatter”. <https://prettier.io/index.html> (elérés 2023. június 14.).
- [11] „JavaScript Component Testing and E2E Testing Framework | Cypress”. <https://www.cypress.io/> (elérés 2023. június 14.).
- [12] „Mi az a Java Spring Boot?– Bevezetés a Spring Boot-rendszerbe | Microsoft Azure”. <https://azure.microsoft.com/hu-hu/resources/cloud-computing-dictionary/what-is-java-spring-boot/> (elérés 2023. június 14.).
- [13] „Maven – Welcome to Apache Maven”. <https://maven.apache.org/> (elérés 2023. június 14.).
- [14] „Spring Data JPA”, *Spring Data JPA*. <https://spring.io/projects/spring-data-jpa> (elérés 2023. június 14.).
- [15] „JPA Buddy - IntelliJ IDEA plugin supporting Hibernate, EclipseLink, SpringData, Liquibase, Flyway and other JPA related tech”. <https://jpa-buddy.com/> (elérés 2023. június 14.).
- [16] „IntelliJ IDEA – the Leading Java and Kotlin IDE”, *JetBrains*. <https://www.jetbrains.com/idea/promo/> (elérés 2023. június 14.).
- [17] „PostgreSQL: About”. <https://www.postgresql.org/about/> (elérés 2023. június 14.).
- [18] „Liquibase Documentation”. [https://docs.liquibase.com/home.html?\\_ga=2.217581206.1608324242.1686753959-1317798438.1686753959](https://docs.liquibase.com/home.html?_ga=2.217581206.1608324242.1686753959-1317798438.1686753959) (elérés 2023. június 14.).
- [19] „Database Migrations: What are the Types of DB Migrations?”, *Prisma’s Data Guide*. <https://www.prisma.io/dataguide/types/relational/what-are-database-migrations> (elérés 2023. június 14.).
- [20] „SQL Database Design Tool, ER Diagram Online | SqlDBD”, *SqlDBD.com*. <https://sqldb.com/> (elérés 2023. június 17.).
- [21] K. Malvoni és J. Knezovic, „Are Your Passwords Safe: Energy-Efficient Bcrypt Cracking with Low-Cost Parallel Hardware”.

- [22] „Vuexy - Vuejs, React - Next.js, HTML, Laravel & Asp.Net Admin Dashboard Template Preview - ThemeForest”. [https://preview.themeforest.net/item/vuexy-vuejs-html-laravel-admin-dashboard-template/full\\_screen\\_preview/23328599?\\_ga=2.252295436.461391074.1678111924-1018890781.1643616715&\\_gac=1.61432158.1678111924.CjwKCAiAu5agBhBzEiwAdiR5tIgkf35u18hwBhOS\\_iZFj-eWZ3-W9mWrZ67jVWNRv-9qB20blyBRtxoCAdoQAvD\\_BwE](https://preview.themeforest.net/item/vuexy-vuejs-html-laravel-admin-dashboard-template/full_screen_preview/23328599?_ga=2.252295436.461391074.1678111924-1018890781.1643616715&_gac=1.61432158.1678111924.CjwKCAiAu5agBhBzEiwAdiR5tIgkf35u18hwBhOS_iZFj-eWZ3-W9mWrZ67jVWNRv-9qB20blyBRtxoCAdoQAvD_BwE) (elérés 2023. június 16.).

## 9. Függelék

### 9.1. Ajánlórendszer kódja

```
@Override
    public List<RecommendationDto> getAllRecommendedPosition() throws
NoAuthority {
    UserEntity currentUser = null;
    try {
        currentUser = utility.getCurrentUser();
    } catch (Exception e) {
        throw new NoAuthority("User not authorized");
    }

    List<SkillsDto> skills = null;
    try {
        skills = skillsService.getByUser(currentUser.getId());
    } catch (Exception e) {
        throw new IllegalArgumentException("Error retrieving user's
skills");
    }

    List<JobRequirementOptionDto> allJobRequirements = null;
    try {
        allJobRequirements = jobRequirementOptionService.getAll();
    } catch (Exception e) {
        throw new IllegalArgumentException("Error retrieving all job
requirement options");
    }

    List<Long> initialVectorOfStudent =
createSkillVectorOfStudent(skills, allJobRequirements);
    List<PositionDto> positionDtos = null;
    try {
        positionDtos =
positionService.getAllOpenPosition(currentUser.getStudyProgramEntity().getId(
));
    } catch (Exception e) {
        throw new IllegalArgumentException("Error retrieving all open
positions");
    }

    List<RecommendationDto> recommendedByCosinusSimilarity = new
ArrayList<>();
    List<RecommendationDto> recommendedByJaccardIndex = new
ArrayList<>();
    for (PositionDto positionDto : positionDtos) {
        List<Long> initialVectorOfPosition =
createSkillVectorOfPosition(positionDto.getJobSkillsDtos(),
allJobRequirements);
        double cosine = cosineSimilarity(initialVectorOfStudent,
initialVectorOfPosition);
        double jaccardIndex = jaccardIndex(skills,
positionDto.getJobSkillsDtos());
```

```

        if (cosine > 0) {
            RecommendationDto recommendationDto = new
RecommendationDto();
            recommendationDto.setPositionDto(positionDto);
            recommendationDto.setRatio(cosine);
            recommendedByCosinusSimilarity.add(recommendationDto);
        }
        if (jaccarIndex > 0) {
            RecommendationDto recommendationDto = new
RecommendationDto();
            recommendationDto.setPositionDto(positionDto);
            recommendationDto.setRatio(jaccarIndex);
            recommendedByJaccardIndex.add(recommendationDto);
        }
    }
    recommendedByCosinusSimilarity.sort((r1, r2) ->
Double.compare(r2.getRatio(), r1.getRatio()));

    recommendedByJaccardIndex.sort((r1, r2) ->
Double.compare(r2.getRatio(), r1.getRatio()));

    Map<Long, Double> positionRatios = new HashMap<>();

    List<RecommendationDto> combinedRecommendations = new ArrayList<>();
    for (RecommendationDto recommendation : recommendedByJaccardIndex) {
        long positionId = recommendation.getPositionDto().getId();
        positionRatios.put(positionId, recommendation.getRatio());
    }

    for (RecommendationDto recommendation :
recommendedByCosinusSimilarity) {
        long positionId = recommendation.getPositionDto().getId();
        if (positionRatios.containsKey(positionId)) {
            RecommendationDto combinedRecommendation = new
RecommendationDto();

            combinedRecommendation.setPositionDto(recommendation.getPositionDto());
            combinedRecommendation.setRatio(recommendation.getRatio());
            combinedRecommendations.add(combinedRecommendation);
        }
    }

    combinedRecommendations.sort((r1, r2) ->
Double.compare(r2.getRatio(), r1.getRatio()));

    return combinedRecommendations.subList(0, Math.min(10,
combinedRecommendations.size()));
}

```

## 9.2. Felhasználói felület

The screenshot displays a web application interface for a company profile. At the top, a navigation bar includes the 'intern' logo and links for 'Dashboard', 'Company', and 'Applications'. The user 'Levente Kiss' is logged in as 'manager'. The main content area features a company profile for 'Prisma Solutions' with a circular logo and contact email 'hello@primasolutions.ro'. Below the profile, there are sections for 'About us', 'Info', 'Online presence', and 'Rating by students'. The 'Open positions' section lists two roles: 'Spring Boot developer' and 'Angular developer', each with a salary of 1700 RON and a 'Show more' link. The interface is clean and modern, with a light gray background and blue accents.

**intern** Dashboard Company Applications Levente Kiss manager

**prisma**

Prisma Solutions  
hello@primasolutions.ro

**About us** Edit

We provide a wide range of digitalisation services and solutions, including consultation, strategy, and implementation.

**Info**

Date of foundation:

Location: Targu Mures

Industry: IT, Web, Digital Marketing

**Online presence**

Facebook LinkedIn Instagram

**Rating by students**

☆☆☆☆☆

**Open positions** Add

**Spring Boot developer** Delete Edit

Prisma Solutions  
Created Jun 6, 2023

Spring Boot developer

Spring Boot(Java framework) PostgreSQL Amazon AWS Databases Backend Java

PgAdmin Postman IntelliJ IDEA

1700 RON Jun 10, 2023 Jun 21, 2023

Show more

**Angular developer** Delete Edit

Prisma Solutions  
Created Jun 6, 2023

Angular developer

Angular TypeScript Bootstrap Postman Visual Studio Code

1700 RON Jun 9, 2023 Jun 21, 2023

Show more

9.1. ábra Céges profil

**Add Position**

**Title** Setup title and description > **Skills** Add required skills > **General details** Set salary and status

**Skills**

Analytical

Art

Business

Communication

Design

Branding and Identity Database design UI design UX design

Language

**Progress Visualization:**

- Összesen 100
- Angular: 71 / 0
- UI design: 15 / 0
- UX design: 14 / 0

9.3. ábra Új pozíció létrehozása

**intern** Dashboard Profile

Szász Attila student

**General**

Social

Skills

Education

**Profile:**

**Upload** Allowed JPG, GIF or PNG.

**First Name:** Szász **Last Name:** Attila

**Bio:** A highly motivated computer science student seeking internship/job opportunities to apply and enhance my skills in software development and contribute to innovative projects in the technology industry.

**Birthday:** 03/04/2000 **Country:** Romania **City:** Târgu-Mureș

**Save changes**

9.2. ábra Hallgató profiljának beállításai


intern


Dashboard

Profile

Szász Attila

student





Szász Attila

szasz.attila@student.ms.sapientia.ro

Profile

For you

Feed

Companies

Bio

A highly motivated computer science student seeking internship/job opportunities to apply and enhance my skills in software development and contribute to innovative projects in the technology industry.

About

Birthday: Mar 5, 2000

Location: Târgu-Mureș, Romania

f

in

tw

ig

envelope

yt

Skills

CorelDraw

Adobe Photoshop

CSS3

Bootstrap

Visual Studio Code

Skills

20%

20%

20%

20%

20%

Education

Major: Matek-Info


At: Bolyai Liceum

From: Sep 10, 2015 To: Jun 14, 2019

Major: Számítástechnika

At: Sapientia EMTE Marosvásárhely

From: Sep 22, 2019 To: Jul 1, 2023



Szász Attila

Accentures tapasztalat

Sziaztok, barátaim! Szeretném megosztani veletek az izgalmas és tanulságos nyári gyakorlati tapasztalataimat az Accenture-nál, ahol az idei évben töltöttem el a nyári szünetet. Mint harmadéves számítástechnika szakos hallgató, nagyon vártam ezt a lehetőséget, hogy bepillantást nyerjek a valódi szakmai világba, és tapasztalatokat szerezzek a területemen. Az Accenture egy világszínvonalú technológiai tanácsadó cég, és már az első pillanattól kezdve lenyűgözött a körülöttük lévő lendület és innováció. Az első nappal kezdődött a csapatba való beilleszkedés, ahol barátságos és támogató környezetet fogadtam. Azonnal éreztem, hogy a része vagyok egy dinamikus csapatnak, akik mindannyian elképezték a kiválóság iránt. A nyári gyakorlatom során rengeteg izgalmas projekten dolgozhattam, amelyek kihívást jelentettek számomra, de lehetőséget adtak arra, hogy fejlesszem a szakmai és problémamegoldó készségeimet. Az egyik projekt során részt vettem egy nagyvállalati alkalmazás fejlesztésében, ahol kódot írtam és teszteltem. Nagyon izgalmas volt látni, hogyan épül fel egy komplex rendszer, és hogyan működik együtt a fejlesztői csapat, a tervezők és az üzleti elemzők. Az Accenture-nál a mentorom is rendkívül segítőkész volt. Az ő irányítgatása és tapasztalatai sokat segítettek nekem a projektek során. Számos tréninget és workshopot is kaptam, ahol továbbfejleszthettem a technikai ismereteimet és megismerhettem az Accenture által alkalmazott legújabb módszereket és technológiákat. A legjobb része a nyári gyakorlatomnak azonban a csapatban dolgozás és az új emberekkel való kapcsolatteremtés volt. Olyan inspiráló és tehetséges emberekkel találkoztam, akik osztották a szenvedélyemet a technológia és az innováció iránt. Új barátságokra tettem szert, és olyan kapcsolatokat építettem, amelyekre a jövőben is támaszkodhatok. Az Accenture-nál eltöltött nyári gyakorlatom minden várakozásomat felülmúlta. Szakmai fejlődést tapasztaltam

Like

PA

9.4. ábra Hallgató profilja

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA  
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREȘ  
SPECIALIZAREA CALCULATOARE

Vizat decan  
Conf. dr. ing. Domokos József

Vizat director departament  
Ș.l. dr. ing Szabo László Zsolt