Szakdolgozat

Miskolci Egyetem

Szakdolgozat bírálati folyamat és Záróvizsga menedzselő webalkalmazás készítés

Készítette:

Drig Dávid

Programtervező informatikus

Témavezető:

Gégény Dávid

Miskolc, 2024

Miskolci Egyetem
Gépészmérnöki és Informatikai Kar
Alkalmazott Matematikai Intézeti Tanszék **Szám:**

Szakdolgozat Feladat

Drig Dávid (EZ3YRC) programtervező informatikus jelölt részére.

A szakdolgozat tárgyköre: Webalkalmazás fejlesztés

A szakdolgozat címe: Szakdolgozat bírálati folyamat és Záróvizsga menedzselő webalkalmazás készítés

A feladat részletezése:

A záróvizsgák megszervezése és lebonyolítása egy több szereplős és több lépéses folyamat. A dolgozat célja, hogy a szervezéséhez, az adatok kezeléséhez egy online elérhető webalkalmazás formájában segítséget biztosítson. A tervezett rendszer a szakdolgozat benyújtásától kezdve a bírálati folyamaton keresztül, egészen a jegyzőkönyvek kinyomtatásáig eszközt adna mind a hallgatók, témavezetők, bírálók és a bizottság részére egyaránt. A webalkalmazás elsősorban a Miskolci Egyetem Programtervező informatikus BSc szakának az elvárásaihoz igazodik, de szempont, hogy egyszerűen alkalmassá tehető legyen más szakok számára is. A backend alkalmazás Java Spring Boot keretrendszerrel, a frontend pedig Angular keretrendszerrel készül.

Témavezető: Gégény Dávid (egyetemi tanársegéd)

A feladat kiadásának ideje: 2023. szeptember 21.

. . . . . .
szakfelelős

Eredetiségi Nyilatkozat

Alulírott **Drig Dávid**; Neptun-kód: `EZ3YRC` a Miskolci Egyetem Gépészmérnöki és Informatikai Karának végzős Programtervező informatikus szakos hallgatója ezennel büntetőjogi és fegyelmi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy *Szakdolgozat bírálati folyamat és Záróvizsga menedzselő webalkalmazás készítése* című szakdolgozatom saját, önálló munkám; az abban hivatkozott szakirodalom felhasználása a forráskezelés szabályai szerint történt.

Tudomásul veszem, hogy szakdolgozat esetén plágiumnak számít:

- szószerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

Alulírott kijelentem, hogy a plágium fogalmát megismertem, és tudomásul veszem, hogy plágium esetén szakdolgozatom visszautasításra kerül.

Miskolc, ...év ..... .hó ..... .nap

.....

Hallgató

1.

szükséges (módosítás külön lapon)

A szakdolgozat feladat módosítása

nem szükséges

....

dátum témavezető(k)

2. A feladat kidolgozását ellenőriztem:

témavezető (dátum, aláírás): konzulens (dátum, aláírás):

....

....

....

3. A szakdolgozat beadható:

....

dátum témavezető(k)

4. A szakdolgozat ..... szövegoldalt

..... program protokollt (listát, felhasználói leírást)

..... elektronikus adathordozót (részletezve)

.....

..... egyéb mellékletet (részletezve)

.....

tartalmaz.

....

dátum témavezető(k)

5.

bocsátható

A szakdolgozat bírálatra

nem bocsátható

A bíráló neve: .....

....

dátum szakfelelős

6. A szakdolgozat osztályzata

a témavezető javaslata: .....

a bíráló javaslata: .....

a szakdolgozat végleges eredménye: .....

Miskolc, ...

a Záróvizsga Bizottság Elnöke

Tartalomjegyzék

1. fejezet

Bevezetés

A digitális technológiák rohamos fejlődése új lehetőségeket teremtett az oktatás és a tudomány terén is. A felsőoktatási intézményekben egyre jobban elterjednek az informatikai megoldások, amelyek segítségével hatékonyabban tudják kezelni az adminisztratív feladatokat illetve támogatni a hallgatókat és az oktatókat a tanulás folyamatában. Az egyik ilyen kiemelkedő terület a szakdolgozatok és záróvizsgák menedzselése, amelyek jelentős mérföldkövei a hallgatók felsőoktatási pályafutásának.

Az ilyen alkalmazások lehetővé teszik a folyamatok hatékonyabb és átláthatóbb kezelését mind a hallgatók és az oktatók számára. Ennek segítségével könnyebb lehet az adminisztráció, az értékelés, valamint a kommunikáció az érintett felek között. A modern webalkalmazások rugalmas szerkezete lehetővé teszi az automatizált folyamatok bevezetését így segítve a hatékonyabb munkavégzést és az idő megtakarítását. Az ilyen megoldások a digitális oktatásban egyre nélkülözhetetlenebb szerepet töltenek be, hozzájárulva a hallgatók és az oktatók sikeres együttműködéséhez és az eredményes tanulási folyamathoz.

Ebben a szakdolgozatban egy olyan webalkalmazás koncepcióját és megvalósítását szeretném szemléltetni, amelynek célja a szakdolgozatok bírálati folyamatának és záróvizsgák menedzselésének támogatása. A rendszer tervezése és implementációja során figyelembe veszem az oktatók és a hallgatók igényeit, valamint a modern webfejlesztés korszerű trendjeit és technológiáit.

A továbbiakban áttekintem a jelenlegi helyzetet a szakdolgozatok és záróvizsgák adminisztrációjában, elemzek néhány már létező megoldást ezen a területen, majd részletesen bemutatom saját alkalmazásom architektúráját és funkcióit. Végül pedig kiértékelem a fejlesztés során szerzett tapasztalatokat és lehetőségeket a jövőbeni továbbfejlesztésre. A cél az, hogy egy olyan innovatív és hasznos eszközt nyújtsak, amely segíti az Alkalmazott Matematikai Intézeti Tanszéket a hatékonyabb és átláthatóbb adminisztrációban, valamint támogatja a hallgatókat a sikeres szakdolgozatok és záróvizsgák előkészítésében és lebonyolításában.

2. fejezet

Egyéb alkalmazások és technológiai összehasonlítás

Ebben a fejezetben bemutatom a bírálati folyamat és a záróvizsga menedzselését, néhány kurzus kezelő szoftvert és a felhasznált technológiákat.

2.1. A bírálati folyamat és a záróvizsga menedzselése

A bírálati folyamat és a záróvizsga menedzselése kiemelkedő fontosságú a tanulmányok vagy képzések során, mivel ezek az események segítenek az értékelésben és az eredmények összegzésében.

Általánosságban a folyamat a következő lépéseket foglalja magában:

- Előkészületek: A hallgatók és tanárok felkészülnek az értékelésre vagy záróvizsgára. Ez magában foglalja a tananyagok áttekintését, dokumentumok előkészítését.
- Kritériumok meghatározása: Fontos az értékelési kritériumok vagy a záróvizsga szempontjainak meghatározása, amelyek alapján a teljesítményt értékelik. Ezek általában a tudás, készségek, alkalmazott módszerek és prezentációs készségek lehetnek.
- Vizsgák vagy prezentációk megszervezése: Fontos megszervezni a vizsgákat vagy prezentációkat, ideértve az időpontok kijelölését, helyszínek meghatározását és technikai támogatások biztosítását.
- Értékelés: A tanárok vagy vizsgáztatók értékelik a hallgatók teljesítményét a meghatározott kritériumok alapján, legyen az írásbeli teszt, gyakorlati feladatok vagy szóbeli prezentáció.

2.2. Kurzus kezelő szoftverek

A *kurzuskezelő szoftverek* [1] olyan platformok, amelyeket általában oktatási intézmények, tréningközpontok vagy vállalatok használnak a kurzusok, tanfolyamok vagy képzések hatékony kezelésére és adminisztrációjára. Ezek a szoftverek általában számos funkciót kínálnak, például regisztráció és jelentkezés kezelése, kurzusanyagok és feladatok feltöltése, tanulói teljesítmény nyomon követése, értékelési eszközök, kommunikációs eszközök és egyéb adminisztratív funkciók.

A legtöbb *kurzuskezelő szoftver* online alapú, ami lehetővé teszi a hozzáférést és a kezelést bármely internetkapcsolattal rendelkező eszközről. Ezek a platformok gyakran könnyen testreszabhatóak és skálázhatóak az intézmény vagy vállalat egyedi igényei szerint. Emellett sok kurzuskezelő szoftver támogatja a tanulói interakciókat és együttműködést, például csoportos munka lehetőségét vagy fórumokat biztosítva.

Ezek a szoftverek hatékony eszköztárat kínálnak az oktatási és képzési folyamatok digitalizálására és automatizálására, ami segít az adminisztratív terhek csökkentésében és a tanulási élmények javításában.

2.2.1. ETR (Egységes Tanulmányi Rendszer)

A Magyarországon elterjedt felsőoktatási szoftverek egyike, az *Egységes Tanulmányi Rendszer (ETR)* [2], lehetővé tette a diákok számára, hogy online platformon keresztül intézzék az oktatási intézménnyel kapcsolatos ügyeiket. Az *ETR* keretében minden regisztrált hallgatónak lehetősége nyílt személyes és tanulmányi adatainak megtekintésére, pénzügyeik nyomon követésére, valamint vizsgákra jelentkezésre és kurzusok felvételére. Az *ETR* technológiai alapjai az adott fejlesztőcsapat vagy az egyetem preferenciáitól függnek, így pontos fejlesztési technológiát nem lehet egyértelműen megállapítani.

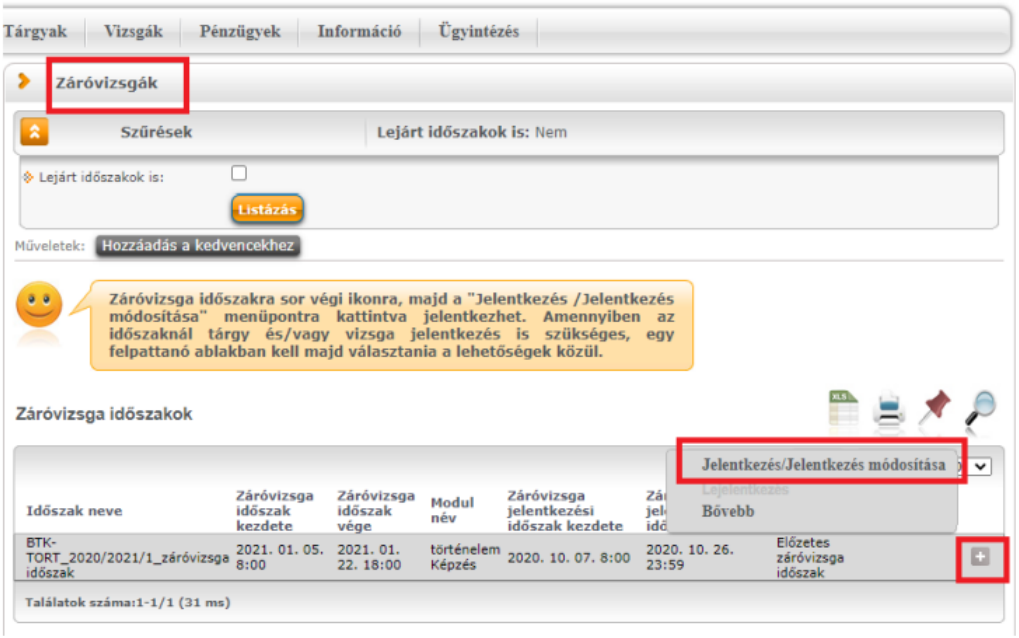
2015-től kezdve az *ETR* funkcionalitása fokozatosan beépült a *Neptun.NET* [3] tanulmányi rendszerbe, és az *ETR*-t használó felsőoktatási intézmények 2018-ig áttértek a *Neptun* rendszerére.

2.2.2. Neptun

A *Neptun* [3] egy egyetemi tanulmányi adminisztrációs alkalmazás, amely *.NET* [4] technológiával készült. A rendszer támogatja a bírálati folyamatot és a záróvizsga menedzselését is a felsőoktatási intézményekben. A bírálati folyamat során az oktatók számos feladatot végeznek, például tanulmányi eredmények értékelése, visszajelzések megadása, valamint a hallgatók előrehaladásának nyomon követése. A rendszer lehetőséget biztosít az oktatóknak arra, hogy ezeket a feladatokat hatékonyan elvégezzék, beleértve az eredmények rögzítését és a visszajelzések megadását is.

A záróvizsga menedzselése is fontos része a *Neptun* platformnak. Ennek során az intézményeknek lehetőségük van előre meghatározott időpontokban és helyszíneken szervezni a záróvizsgákat, valamint regisztrálni a hallgatókat ezekre a vizsgákra.

A *Neptun* szolgáltatás általában rugalmas konfigurációs lehetőségeket kínál az intézmények számára, hogy testreszabhassák a bírálati folyamatot és a záróvizsga menedzselését az adott intézet saját igényeihez és eljárásaihoz. Ezáltal az intézmények hatékonyan tudják kezelni ezeket a fontos tanulmányi folyamatokat, és biztosítani tudják a hallgatók és oktatók számára az átlátható kommunikációt és adminisztrációt.



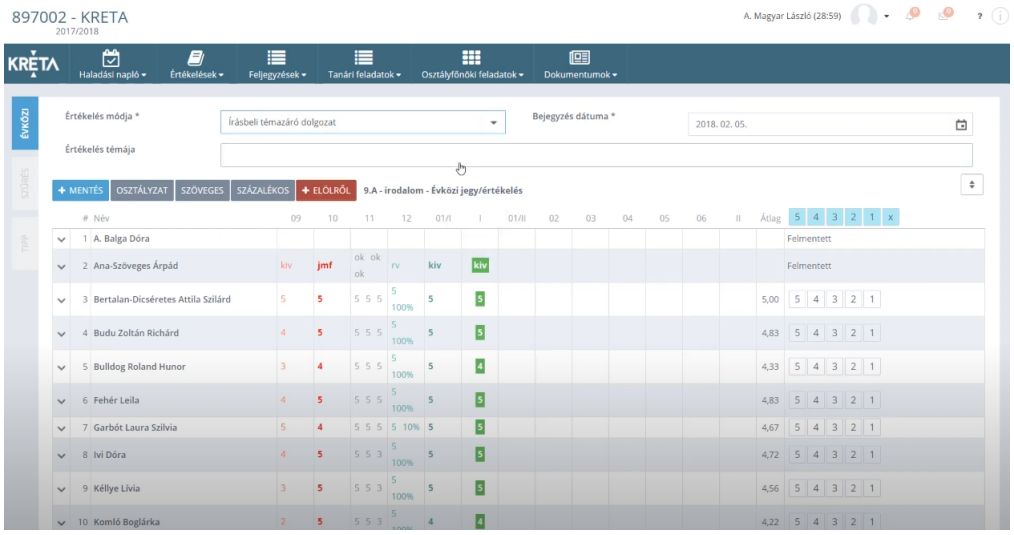
2.1. ábra. Záróvizsga jelentkezés a *Neptun* rendszerben[6]

2.2.3. KRÉTA

A *KRÉTA (Köznevelési Regisztrációs és Tanulmányi Alaprendszer)* [7] kurzuskezelő szoftver számos funkciót kínál a tanulmányi folyamatok hatékony kezelésére és adminisztrációjára, ami PHP [5] technológiával készült. Az oktatók számára lehetőség van a tanulmányi eredmények értékelésére, visszajelzések megadására, valamint a diákok teljesítményének nyomon követésére. Ezáltal az oktatók hatékonyan tudják értékelni a diákok munkáját és fejlődését, és szükség esetén visszajelzést adni nekik a további fejlődés érdekében.

A bírálati folyamatok integrációja a *KRÉTA* kurzuskezelő rendszerébe lehetővé teszi az intézmények számára, hogy átláthatóan és hatékonyan kezeljék az értékelési folyamatokat. Emellett segíti az oktatókat abban is, hogy időben és pontosan értékeljék a diákok munkáját, és visszajelzést adhassanak a tanulmányi előrehaladásukról.

Így a *KRÉTA* nemcsak a tanulmányi folyamatok kezelését segíti, hanem a bírálati folyamatokat is integrálja, hogy teljes körű támogatást nyújtson az oktatási intézmények számára.

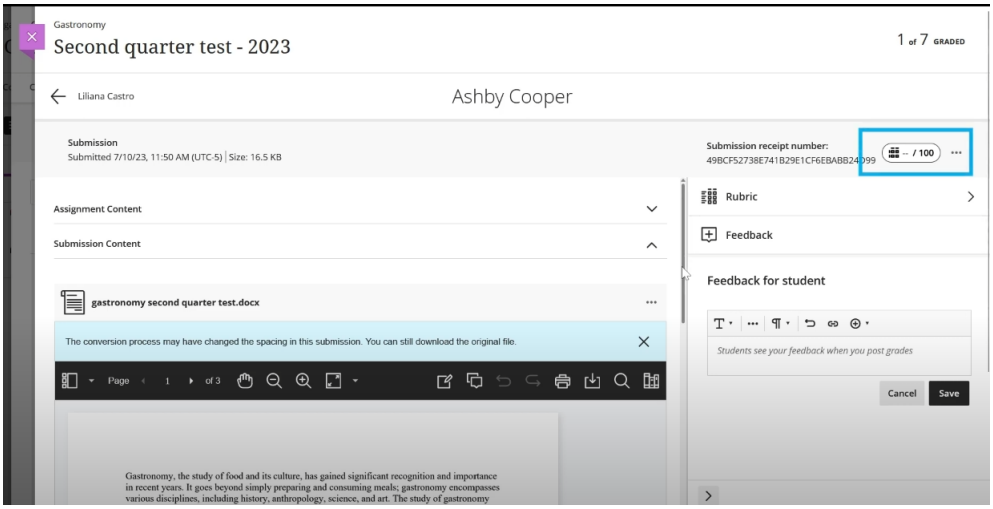


2.2. ábra. Értékelés a *KRÉTA* rendszerben[8]

2.2.4. Blackboard Learn

A *Blackboard Learn* [9] egy kifinomult e-learning platform, melyet széles körben használnak oktatási intézmények világszerte. Ez az alkalmazás lehetővé teszi az oktatók számára, hogy létrehozzanak és kezeljenek online tananyagokat, interaktív kurzusokat, valamint feladatokat és teszteket. A hallgatók számára a *Blackboard Learn* egy hozzáférhető, felhasználóbarát felületet biztosít, ahol könnyen hozzáférhetnek a tananyagokhoz, kommunikálhatnak az oktatókkal és társaikkal, valamint beadhatják feladataikat és vizsgáikat.

Az oktatók az alkalmazáson keresztül képesek értékelni a hallgatók munkáját, visszajelzéseket adni nekik, valamint nyomon követni az előrehaladásukat és teljesítményüket. Emellett a *Blackboard Learn* lehetőséget biztosít az intézményeknek a vizsgák szervezésére, menedzselésére és értékelésére. Ez az alkalmazás segít az oktatási folyamatok digitalizálásában és hatékonyabbá tételében, elősegítve a hallgatók és oktatók közötti interakciót és együttműködést.



2.3. ábra. Értékelés és bírálat készítés a *Blackboard Learn* rendszerben [10]

2.3. Felhasznált technológiák

A felhasznált technológiák megválasztásakor elterjedt és stabil eszközök alkalmazása volt a cél, mivel ezek a megbízható platformok lehetővé teszik a hatékony munkavégzést és minimalizálják az esetleges technikai problémák kockázatát.

2.3.1. Adatbázis

Ahhoz, hogy hatékonyan kezeljem az adatokat és az eredményeket, a *MySQL* [11] relációs adatbázis-kezelő rendszert választottam. A *MySQL* rendszer biztosítja a szükséges funkcionalitást az adatbázis létrehozásához és kezeléséhez, valamint lehetővé teszi az adatok gyors és hatékony lekérdezését.

Az adatbázis-kezeléshez a MySQL Community Edition csomag részeként könnyen elérhető *MySQL Workbench* [12] alkalmazást választottam. Ennek segítségével könnyedén létrehozhatom az adatbázis sémáját, végrehajthatom az SQL utasításokat és optimalizálhatom azokat. A *MySQL Workbench* számos hasznos eszközt kínál, mint például a szintaxiskiemelést és az automatikus kiegészítést, ami jelentősen megkönnyíti a fejlesztési folyamatot.

A *MySQL* helyett választható lett volna például az *SQLite*, *MariaDB*. Emellett az adatok nyilvántartására maximálisan alkalmas a *NoSQL MongoDB* is, mivel a kezelendő információk nem teszik elengedhetetlenné relációs adatbázis választását.

2.3.2. Java Spring Boot

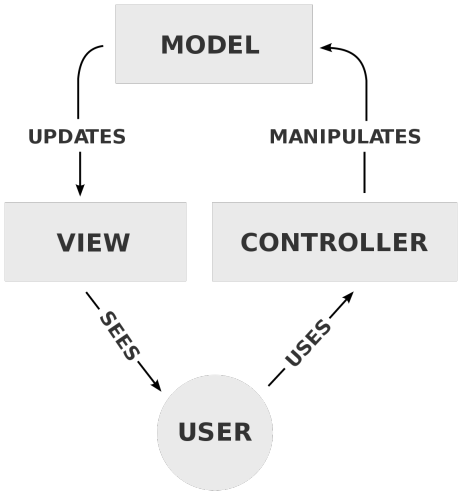
A webalkalmazásom backend részéhez a *Java* [13] programozási nyelvet választottam. A *Java* egy kiválóan használható, objektumorientált nyelv, amely széles körben elterjedt a fejlesztői közösségben.

A backend implementációjához a *Spring Boot* [14] keretrendszert választottam, amely segíti a gyors és egyszerű webalkalmazások fejlesztését. A *Spring Boot* automatikus konfigurációval rendelkezik, ami megkönnyíti a fejlesztési folyamatot. Támogatja az *MVC (Model-View-Controller)* [15] programtervezési mintát, és beágyazott Tomcat szerveren futtatja az alkalmazást.

Az *MVC* modell három fő logikai egységből áll melyek a következők:

- **Controller:** Az *MVC (Model-View-Controller)* modellben a vezérlő feladata a felhasználói interakciók kezelése és az üzleti logika irányítása. A vezérlő fogadja a kéréseket a felhasználótól, feldolgozza azokat, majd frissíti a modellt vagy változtatást eszközöl a nézeten.
- **View:** A nézet az adatok megjelenítéséért felelős rész. A modellből származó adatokat a felhasználó számára értelmezhető formában jeleníti meg. A nézetek lehetnek grafikus felületek, weboldalak, vagy akár más típusú kimenetek is.
- **Model:** A modell reprezentálja az alkalmazás adatait és üzleti logikáját. Az adatokat tárolja, feldolgozza és manipulálja a modell, valamint a szükséges művele-

teket elvégzi az adatok tartós tárolásához vagy az alkalmazás állapotának fenntartásához. A modell nem kommunikál közvetlenül a felhasználóval, helyette a vezérlő által történik az adatok elérése és kezelése.



2.4. ábra. *MVC* modell [15]

Bár más technológiák is elérhetőek lettek volna, mint például a *Node.js*, a *Python Django* vagy a *C# ASP.NET Core*, azonban tapasztalataim alapján a *Java* [13] és kifejezetten a *Spring Boot* [14] a legmegfelelőbb választásnak tűnt a projekt szempontjából.

2.3.3. Angular

A fejlesztett webalkalmazás vékony kliens alapú, ami azt jelenti, hogy a szükséges funkciókat és üzleti logikát a távoli szerveroldalon, vagyis a backenden kerül implementálásra. A kommunikáció a kliens és a szerver között *HTTP* [16] protokollon keresztül zajlik, ami egy alkalmazás szintű, állapotmentes kérés-válasz alapú protokoll elosztott rendszerek számára. Emellett a kliensoldalon a GET, POST, PUT és DELETE *HTTP* metódusokat használom az adatok kezelésére.

Az alkalmazás frontend részét az *Angular* [17] keretrendszer segítségével valósítottam meg. Az *Angular* egy nagy teljesítményű, bővíthető és funkciókban gazdag frontend eszközkészlet, amely lehetővé teszi összetettebb webalkalmazások fejlesztését különböző komponensekből, amelyek *HTML* [18], *TypeScript* [19] és *CSS* [20] fájlokra épülnek. A felhasználói felület megjelenésének stílusához *Bootstrap* [21] eszközkészletet is alkalmaztam, ami rengeteg *HTML* és *CSS* alapú tervezősablont és mintát tartalmaz számos komponenshez.

Lehetőség lett volna a *Vue* [22] vagy a *React* [23] használatára is a webalkalmazás frontendjének megvalósításához, az *Angular* [17] mellett döntöttem. Az *Angular* széles körű támogatottsága és népszerűsége miatt könnyebb volt találni hozzá dokumentációt, oktatóanyagokat és közösségi támogatást, ami gyorsabb fejlesztést eredményezett.

3. fejezet

Tervezés

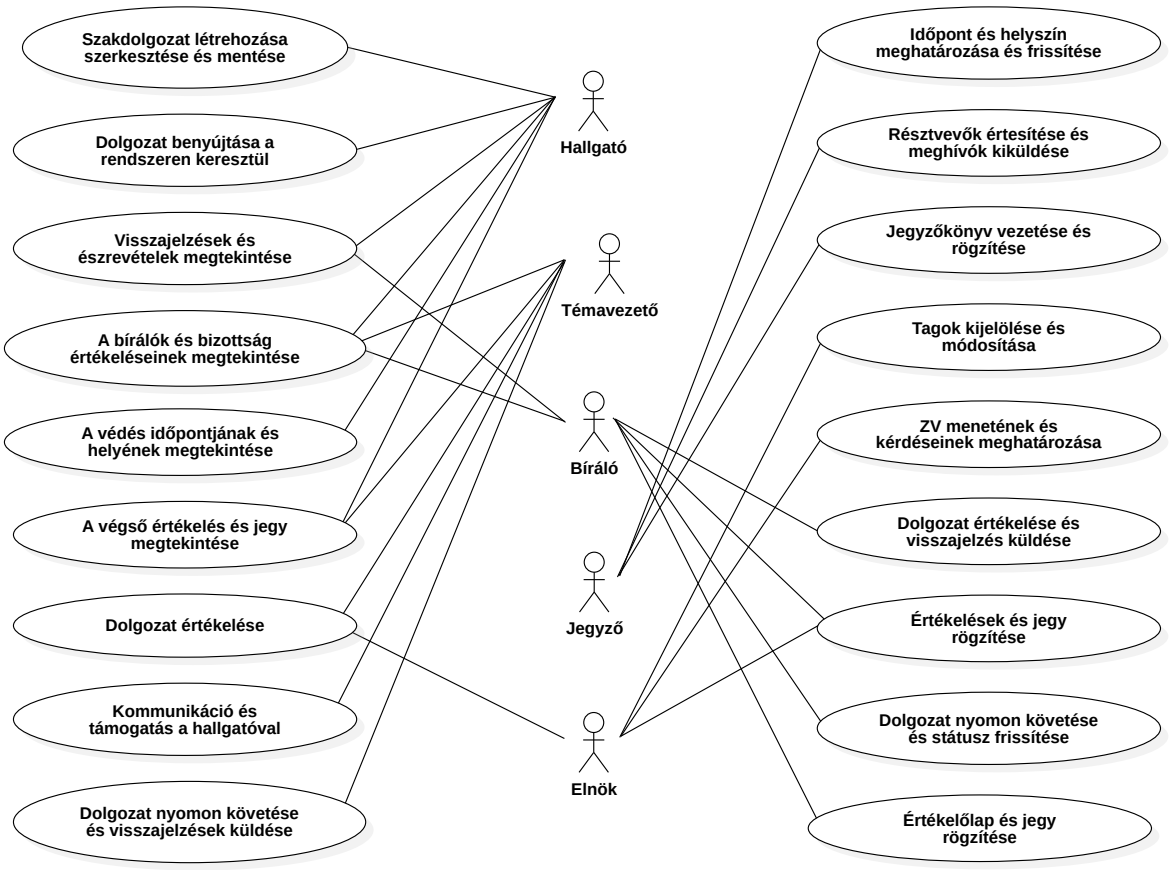
A következőkben bemutatom a különböző szerepköröket és folyamatokat a szekvenciadiagramok és folyamatábrák segítségével. Részletes elemzésre kerülnek a lapok közötti átmenetek és az adatmodell is. Ezek a lépések nélkülözhetetlenek a hatékony és átlátható tervezéshez, hogy sikeresen elkészülhessen a projekt.

3.1. Szerepkörök leírása

A webalkalmazás kialakítása során hat különböző szerepkört határoztam meg, amelyek mindegyike különös felelősséggel és jogosultságokkal rendelkezik a platformon. Ezek a szerepkörök a következők:

1. Hallgató
    - Szakdolgozat létrehozása, szerkesztése és mentése a rendszerben.
    - Az elkészült dolgozat benyújtása a rendszeren keresztül.
    - Visszajelzések és észrevételek megtekintése a dolgozatról a bírálók, témavezető vagy a bizottság részéről.
    - A bírálók és a bizottság értékeléseinek megtekintése.
    - A védés időpontjának és helyszínének megtekintése és kezelése.
    - A végső értékelés és jegy megtekintése.
2. Bíráló
    - A dolgozat értékelése és visszajelzés küldése a hallgatónak.
    - A dolgozat nyomon követése és annak státuszának frissítése a rendszerben.
    - Az értékelőlap és a végső jegy rögzítése a rendszerben.
3. Záróvizsga jegyzője
    - A záróvizsga időpontjának és helyszínének meghatározása és frissítése.
    - A záróvizsga résztvevőinek (hallgató, bírálók, bizottság) értesítése és meghívók kiküldése.
    - A záróvizsga jegyzőkönyvének vezetése és rögzítése a rendszerben.

4. Záróvizsga bizottság elnöke
    - A bizottság tagjainak kijelölése és módosítása.
    - A záróvizsga menetének és a kérdéseknek meghatározása.
    - A hallgató szakdolgozatának értékelése a bizottság véleménye alapján.
    - A végső értékelés rögzítése a rendszerben.
5. Témavezető
    - A hallgatóval való kommunikáció és támogatás a szakdolgozat elkészítése során.
    - A hallgató dolgozatának nyomon követése és visszajelzések küldése.
    - A bírálók és a bizottság értékeléseinek megtekintése.
    - A végső értékelés és jegy rögzítése a rendszerben.
6. Adminisztrátor
    - Az adminisztrátor a fentiek mindegyikével rendelkezik.



3.1. ábra. Az alkalmazás use case diagramja

3.2. Folyamatok ábrázolása

Az alkalmazás tervezésekor több folyamatot is figyelembe kellett vennem, mint például a bírálati folyamatot és a bírálat feltöltésének lépéseit. Ezeket a folyamatokat szekvenciadiagramon és folyamatábrán ábrázoltam.

3.2.1. Bírálati folyamat

1. **Adminisztrátor regisztrálja a hallgatót:**

   Az adminisztrátor bejelentkezik az alkalmazásba és regisztrálja a hallgatót a rendszerbe. A regisztráció során az adminisztrátor rögzíti a hallgató nevét, e-mail címét, jelszavát és egyéb szükséges információkat.

2. **Témavezető értesíti az adminisztrátort a bíráló elérhetőségéről:**

   A témavezető értesíti az adminisztrátort arról, hogy mely bírálók érhetők el a szakdolgozat bírálásához.

3. **Adminisztrátor regisztrálja a bírálót:**

   Az adminisztrátor rögzíti a bíráló adatait az alkalmazásban, beleértve a nevet, e-mail címet, jelszót és egyéb szükséges információkat. A regisztráció után a bíráló jogosult lesz részt venni a bírálati folyamatban.

4. **Elnök felkéri a témavezetőt és a bírálót a bírálatra:**

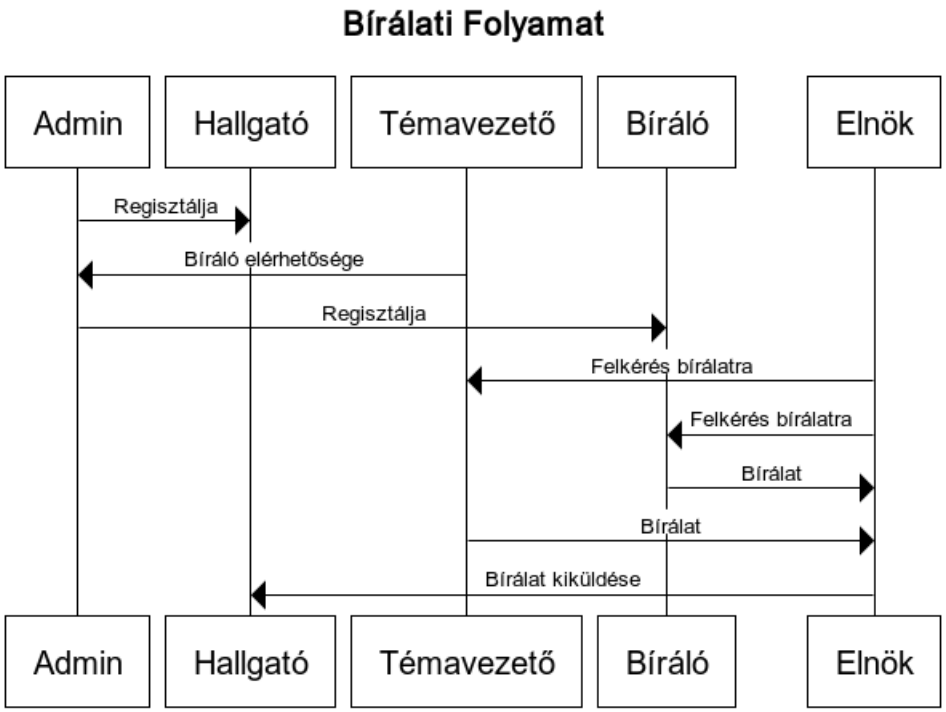
   Az elnök bejelentkezik az alkalmazásba és kiválasztja a szakdolgozatot, amelyre bírálatot kíván kérni. Az elnök elküldi a felkérést a bírálat elvégzésére a témavezetőnek, és a kiválasztott bírálónak.

5. **Bíráló és a témavezető elkészíti a bírálatot:**

   A bíráló és a témavezető bejelentkezik az alkalmazásba, ahol elérhetik a kijelölt szakdolgozatot. A bíráló és a témavezető részletesen értékeli a dolgozatot, és dokumentálja az észrevételeit.

6. **Elnök kiküldi a bírálatot a hallgatónak:**

   Az elnök bejelentkezik a platformra, ahol hozzáfér a bírálati eredményekhez. Az elnök elküldi a bírálati eredményeket a hallgatónak, lehetővé téve számára azok megtekintését.



3.2. ábra. Bírálati folyamat szekvenciadiagramja

3.2.2. Bírálat feltöltésének lépései

1. **Bíráló belép a rendszerbe és listázza a felkéréseket:**

   A bíráló bejelentkezik az alkalmazásba. Miután sikeresen bejelentkezett, a rendszer megjeleníti a bíráló számára elérhető felkéréseket. A felkérések listája tartalmazza az összes olyan szakdolgozatot, amelyekre a bírálót felkérték a bírálat készítésére.

2. **Bíráló kiválaszt egy felkérést a listából:**

   A bíráló kiválasztja a listából azt a szakdolgozatot, amelyre bírálatot szeretne készíteni.

3. **Bíráló elkezdi szerkeszteni a bírálatot:**

   Elkezdi szerkeszteni a bírálatot. Részletesen értékeli a dolgozatot és dokumentálja az észrevételeit, pontozását vagy egyéb megjegyzéseit a bírálatban.

4. **Bíráló beküldi a szerkesztett bírálatot:**

   Miután a bíráló elkészítette a bírálatot, beküldi azt a rendszerbe.

5. **Ellenőrzés:**

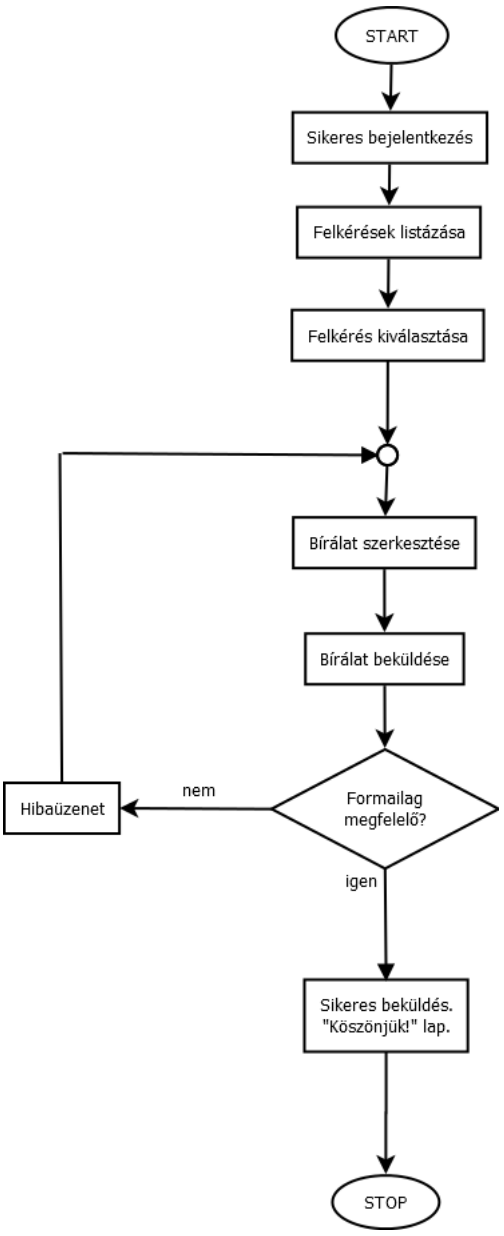
   A rendszer ellenőrzi a beküldött bírálatot, hogy megfelel-e a formai követelményeknek.

   - Ha a bírálat formailag nem megfelelő: A rendszer hibaüzenetet jelenít meg, és figyelmezteti a bírálót a hibákra. A bíráló visszairányításra kerül a bírálat szerkesztéséhez, hogy kijavítsa a hibákat.

- Ha a bírálat megfelelő formában van beküldve: A rendszer értesítést küld a bírálónak a sikeres beküldésről, illetve a dokumentum azonnal letöltődik, amit meg tud tekinteni.

6. **Sikeres beküldés:**

   Miután a bíráló sikeresen ellenőrizte és beküldte a bírálatot, a folyamat befejeződik, és a bíráló visszatér az alkalmazás főoldalára vagy más tevékenységekhez.

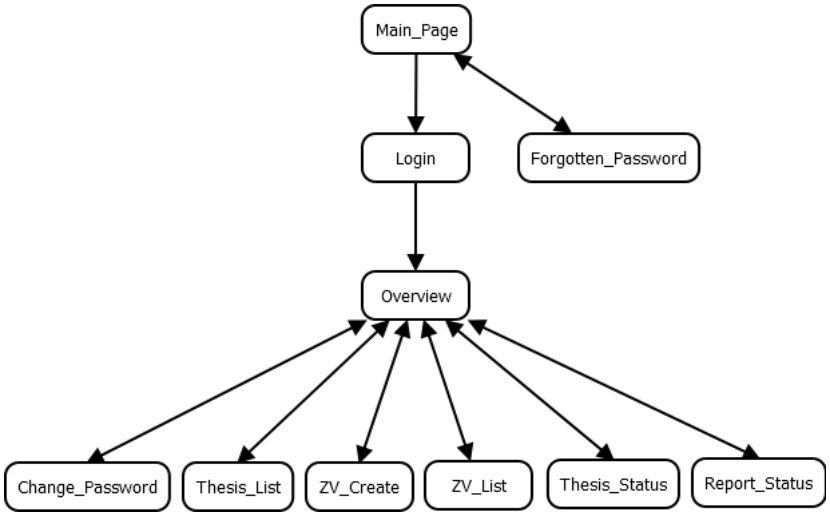


3.3. ábra. Bírálati feltöltéseinek lépései

3.3. Lapok közötti átmenetek

Az alábbi ábrák és a leírások különböző szerepkörök nézőpontjából ábrázolják, hogy melyik oldalhoz férhetnek hozzá.

3.3.1. Elnök és Jegyző

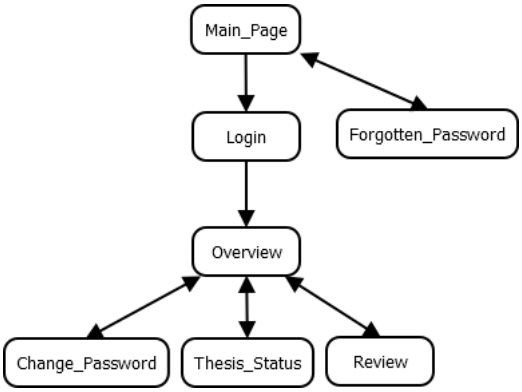


3.4. ábra. A webalkalmazás az elnök szemszögéből

- **Bejelentkezés:** Minden felhasználó lehetőséget kap a belépésre az alkalmazásba, ahhoz hogy azonosíthassák magukat és hozzáférjenek a rendszerhez. Ehhez felhasználónevet és jelszót kell megadniuk.
- **Elfelejtett jelszó:** Ha a felhasználók elfelejtik a jelszót, lehetőségük van új igénylésére. Erre az Elfelejtett jelszó menüpont szolgál, amely segítségével új jelszó kérése vagy generálása lehetséges.
- **Jelszó változtatás:**Az alkalmazás felhasználói képesek megváltoztatni jelszavukat, ha szükségük van rá. Erre a Jelszó változtatás lap szolgál, ahol lehetőség van a jelszó módosítására.
- **Szakdolgozatok listázása:** Az elnök és a jegyző is képes kilistázni a rendszerben található összes szakdolgozatot. Ez lehetőséget ad arra, hogy áttekintsék a szakdolgozatok részleteit.
- **Záróvizsgák létrehozása és listázása:** Mindkét szerepkörnek lehetősége van létrehozni és kilistáztatni záróvizsgákat a platformon.
- **Szakdolgozatok státusza:** Az elnök és a jegyző képes megtekinteni a szakdolgozatok aktuális státuszát az alkalmazásban. Ez lehetőséget ad számukra, hogy nyomon kövessék a dolgozatok előrehaladást.
- **Bírálat státusza:** Az elnök és a jegyző bírálókat tud felkérni a szakdolgozatok értékelésére, valamint nyomon tudják követni a bírálati folyamatot. Ezenkívül lehetőségük van letölteni a bírálatokat és e-mailben elküldeni a hallgatóknak.

- **Záróvizsga jegyzőkönyv generálása** (jegyző számára): A jegyző külön lehetőséget kap arra, hogy generáljon záróvizsga jegyzőkönyveket az alkalmazásban, ami segíti az adminisztrációt és a dokumentációt a záróvizsgákhoz kapcsolódóan.

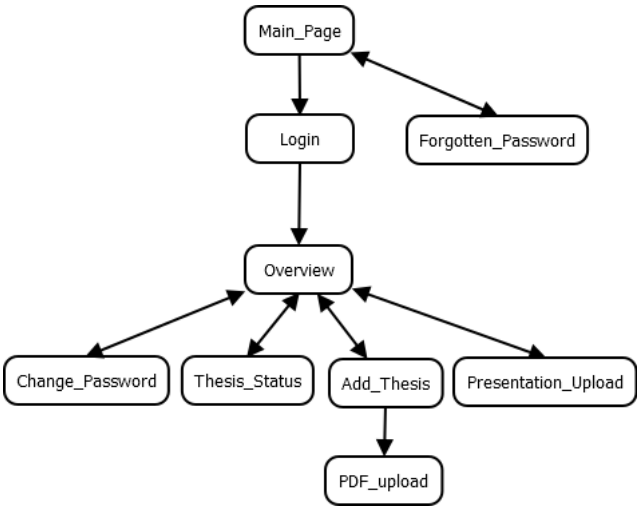
3.3.2. Bíráló és Témavezető



3.5. ábra. A webalkalmazás a bíráló és témavezető szemszögéből

- **Bírálat írása:** A témavezető és a bíráló bírálatokat tud írni a szakdolgozatokról. Fontos megjegyezni, hogy csak azokra a dolgozatokra tudnak bírálatot írni, amelyekre fel lettek kérve. A bírálatok általában értékelést és észrevételeket tartalmaznak a dolgozat tartalmával kapcsolatban.
- **Szakdolgozatok státusza:** Mindkét felhasználó képes megtekinteni azokat a szakdolgozatokat, amelyekre fel lettek felkérve. Ez lehetőséget ad számukra, hogy letölthessék és megtekintsék a dolgozatokat.

3.3.3. Hallgató



3.6. ábra. A webalkalmazás a hallgató szemszögéből

- **Szakdolgozat adatainak kitöltése:** A hallgatónak a feltöltött szakdolgozathoz kapcsolódó adatokat kell megadnia, például a dolgozat címét, szerzőjét, konzulensét stb. Ez lehetőséget ad az adminisztrátornak és a többi felhasználónak a dolgozat könnyebb azonosítására és kezelésére.
- **Szakdolgozat feltöltése:** A hallgatónak lehetősége van a szakdolgozatát feltölteni az alkalmazásba. A szakdolgozat adatainak kitöltése után a rendszer átnavigálja a hallgatót erre a felületre, ahol feltöltheti a szakdolgozat dokumentumát és a csatolmányt.
- **Szakdolgozat státusza:** A hallgatónak lehetősége van megtekinteni a feltöltött szakdolgozatának aktuális státuszát az alkalmazásban.
- **Prezentáció feltöltése:** Ez egy olyan funkció, ami lehetőséget biztosít a hallgatónak arra, hogy a szakdolgozatának megvédéséhez szükséges prezentációt is feltölthesse az alkalmazásba.

3.3.4. Adminisztrátor

Az adminisztrátor szerepe az alkalmazásban a legteljesebb hozzáférést biztosítja az összes funkcionalitáshoz, valamint az alábbi plusz feladatokat látja el:

- **Felhasználók felvitele:** Az adminisztrátornak lehetősége nyílik új felhasználókat regisztrációjára.
- **Felhasználók listázása és kezelése:** Az admin képes kilistázni az összes felhasználót az alkalmazásban. Lehetősége van az adatokat megtekinteni, módosítani és törölni.

4. fejezet

Megvalósítás

4.1. A rendszer felépítése

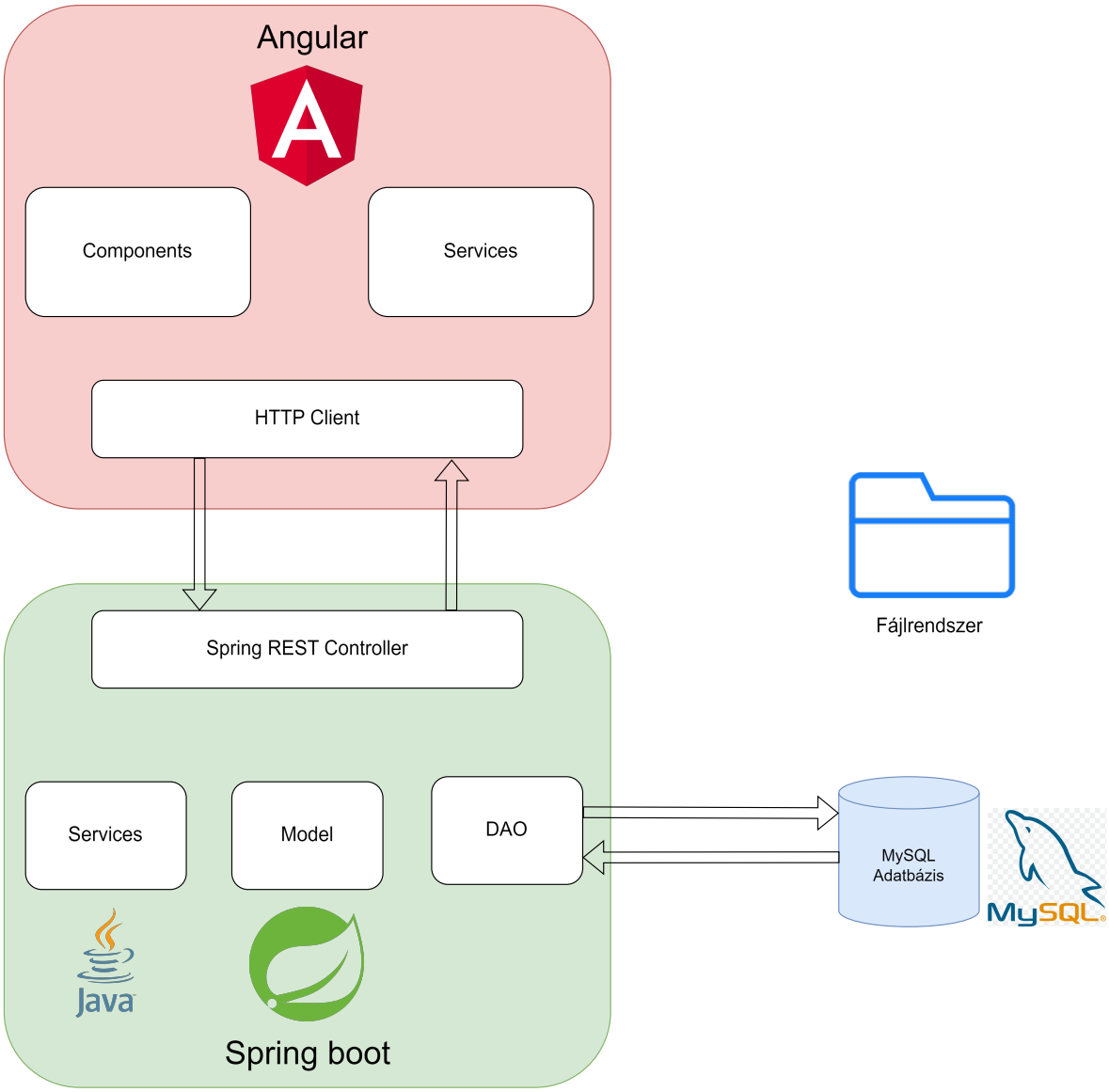
A webalkalmazás fejlesztése során fontos szempont volt az alapvető technológiai döntések meghozatala. Az alkalmazásom három fő komponensre épül: adatbázis, backend és frontend. Az adatbázis rétegként MySQL-t használok, mely hatékonyan tárolja és kezeli az adatokat. A backendet Java Spring Boot keretrendszer segítségével építem fel, ami biztosítja az üzleti logika végrehajtását és az adatok kezelését. A frontend felelős a felhasználói felület megjelenítéséért és kezeléséért. Ehhez az Angular keretrendszert alkalmazom, ami modern és komponensalapú megközelítést biztosít.

Ezek a technológiák összehangoltan működnek, hogy egy erőteljes, hatékony és felhasználóbarát webalkalmazást hozzanak létre. A MySQL adatbázis, a Java Spring Boot backend és az Angular frontend kombinációja lehetővé teszi számomra, hogy magas színvonalú alkalmazást fejlesszek ki. Ez megfelel a felhasználók elvárásainak és az iparági követelményeknek.

4.2. Perzisztens tárolás

A webalkalmazás fájlrendszere a dolgozatok, prezentációk, bírálatok és záróvizsga jegyzőkönyvek tárolását szolgálja. Az alkalmazásban a fájlok tárolásához különböző jegyzékstruktúrákat használok fájltípusok szerint. A fájlok elérésének szempontjából az alkalmazás egyedi azonosítót (továbbiakban: 'id') és Univerzálisan Egyedi Azonosítót (továbbiakban: 'uuid') használ. Az id az adatbázisban tárolt entitások egyedi azonosítója, míg az uuid egy véletlenszerűen generált, egyedi azonosító, amelyet a fájlokhoz rendelek. Használhattam volna csak az id-t is viszont biztonságtechnikai szempontból a uuid mellett döntöttem. Ezáltal a uuid megnehezíti az illetéktelen hozzáférést a fájlokhoz, és növeli az adatbiztonságot azáltal, hogy komplikáltabbá teszi a manuális fájl elérését. A uuid használata más szempontból is előnyösnek bizonyult: Amikor a hallgató feltölti a szakdolgozatot, az alkalmazás egy uuid-t rendel hozzá, amely garantálja, hogy a fájl neve egyedülálló legyen. Ez különösen hasznos ebben az esetben, ha az összes szakdolgozat egy közös jegyzékben található, így elkerülhető a névütközés.

Az alábbiakban felsorolom a jegyzékstruktúrákat és az egyes fájltípusok tárolásának módját:



4.1. ábra. A webalkalmazás felépítése

A dolgozatok hordozható dokumentum formátum (továbbiakban: 'pdf') és tömörített fájlformátum (továbbiakban: 'zip') kerülnek tárolásra.

- Dolgozatok dokumentációjai (pdfs):

  A dolgozatok dokumentációját a "pdfs" nevű jegyzékben tárolom. A fájl feltöltésekor generálódik egy UUID, mely az első részét képezi a fájl nevének, majd ezt követi a hallgató által megadott elnevezés része.

- Dolgozatok mellékletei (zip):

  A szakdolgozatok mellékleteit az "attachment" jegyzékben raktározom. A fájl elnevezése ugyanaz mint a dokumentációnál.

- Prezentáció (pptx):

  A prezentációk eltárolása érdekében is egy külön jegyzéket "ppts" hoztam létre, amelyben a fájlok elnevezése szintén megegyezik a fentebb említett két verzióval.

- Bírálatok (reviews):

  A bírálatokat word formátumban tárolom le egy külön "reviews" jegyzékben. A fájlnév szerkezete a következő: először az "Review" szó áll, majd a feltöltés dátuma, a hallgató neve, és végül egy uuid következik.

- Záróvizsga Jegyzőkönyvek (zv):

  A záróvizsga jegyzőkönyveket is word formátumban tárolom a "zv" jegyzékben. Az elnevezési konvenció hasonló az bírálathoz viszont a feltöltés dátuma nem jelenik meg benne.

A perzisztens tárolás során az alkalmazás adatmodelljét relációs séma szerint terveztem meg. Ennek értelmében az adatok strukturált formában, relációs adatbázisban lesznek tárolva a későbbi hatékony lekérdezés és adatmanipuláció érdekében. A relációs adatmodell lehetővé teszi az adatok logikai kapcsolatainak és struktúrájának pontos meghatározását, amely segíti az adatok egyszerű és hatékony kezelését, valamint az adatintegritás megőrzését. Az adatmodell megtervezése során figyelembe vettem az alkalmazás üzleti logikáját és funkcióit, hogy az adatbázis struktúrája és relációi pontosan tükrözzék az alkalmazás igényeit és funkcionalitását.

4.3. Backend

A backend rész az alkalmazás egyik kulcsfontosságú része, amely az adatok kezelését, üzleti logikájának végrehajtását és az ügyfélkérések feldolgozását végzi. Ebben a részben bemutatom az alkalmazás Data Access Object (továbbiakban: 'DAO'), Entitás , Service és Vezérlő rétegeit.

4.3.1. DAO

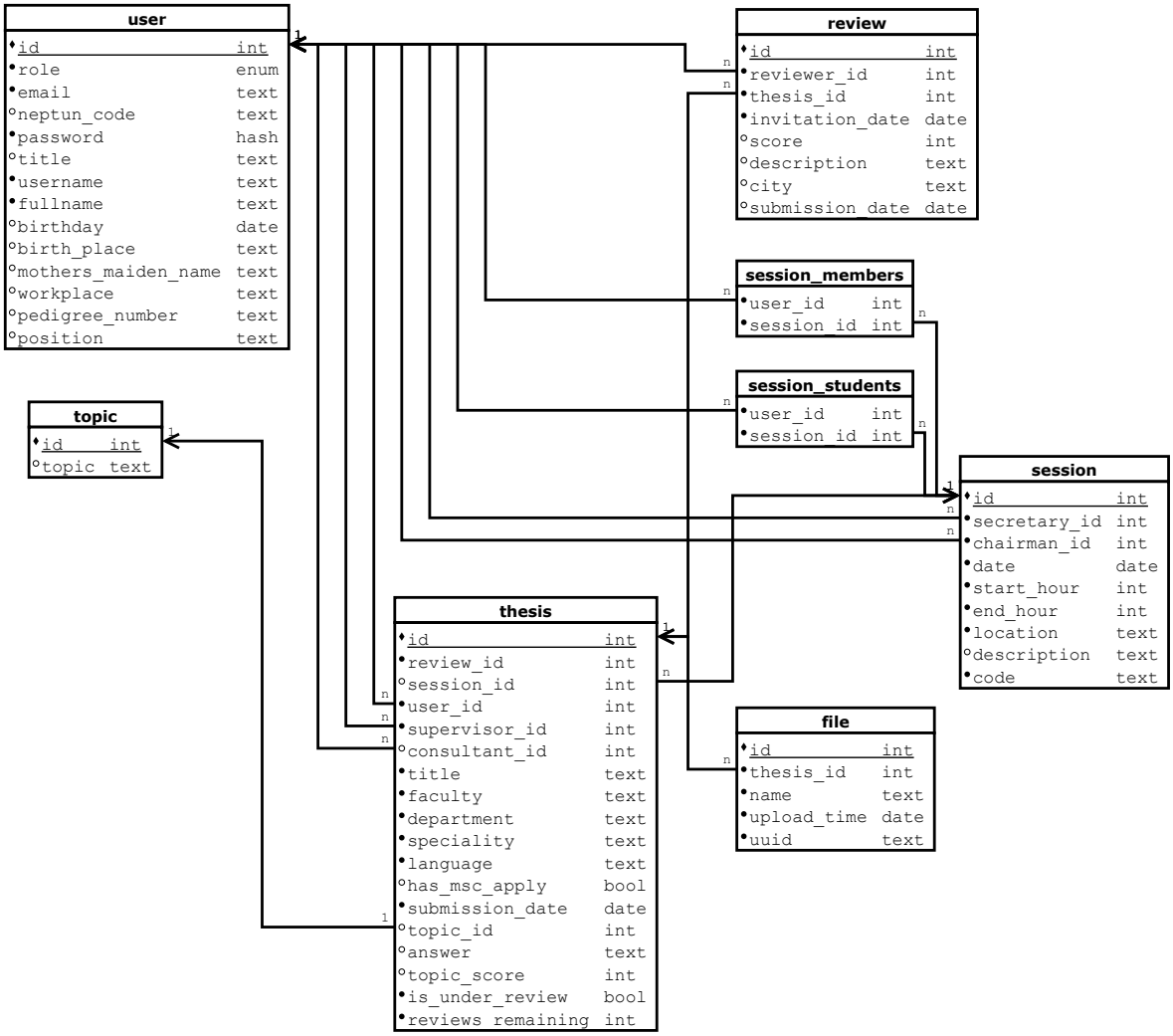
A DAO réteg egy tervezési minta, amelyet szoftverfejlesztés során alkalmaznak, különösen az adatok perzisztens tárolásának és hozzáférésének kezelésére. A Spring Boot alkalmazásokban is gyakran használják a DAO réteget az adatbázis műveletek kiszolgálására. Néhány fő ok, hogy miért szükséges:

- A DAO réteg segít elválasztani az üzleti logikát az adathoz való hozzáférés rétegétől. Ezáltal könnyebbé válik a kód karbantarthatósága és kiterjeszthetősége.
- Az üzleti logika nem közvetlenül függ az adatbázis technológiájától vagy a konkrét adatmodelltől, ami könnyebb tesztelést és kód újrafelhasználást tesz lehetővé.
- A DAO réteg lehetővé teszi az adatbázis műveletek optimalizálását.
- Ha az adatbázis vagy az adatmodell változik, a DAO réteg módosítása ezt a változást kezelheti, anélkül hogy az üzleti logikát érintené.
- Ha az alkalmazásnak későbbiekben változnia kell az adatbázis technológiájában (pl. MySQL-ről PostgreSQL-re), a DAO réteg könnyen cserélhető anélkül, hogy az alkalmazás más részeiben jelentős módosításokat kellene végrehajtani.

A webalkalmazásban található SQL query-k olyan mezőket kérnek le egyes táblákból, amelyek az adott lekérdezéshez szükségesek lehetnek. Azokra az adatokra amelyekre nincs szükség egy adott lekérdezés során a 'null as' kulcsszóval látom el.

A Spring JDBC Template-t is használhatom az SQL lekérdezések végrehajtására, és ennek segítségével is könnyedén érhetem el az adatbázist. A JDBC templatet viszont a service osztályban valósítom meg.

4.3.2. Entity



4.2. ábra. Az alkalmazás adatmodelljének sémája

Először létrehoztam a Java osztályokat az adatbázistáblákhoz, amelyeket JPA entitásokként definiáltam. Ezek az osztályok reprezentálják az adatbázistáblákat és tartalmazzák a tábla attribútumait. Konfiguráltam a Spring Boot alkalmazást, hogy csatlakozzon az adatbázishoz és használja a JPA-t az adatok kezelésére. Ehhez a application.properties fájlban megadtam az adatbázis kapcsolódási információit és konfigurációit. A Spring Data JPA segítségével Repository interfészeket hoztam létre, amelyek a JPA entitásokhoz kapcsolódnak. Ezek a interfészek kiterjesztik a Spring Data CrudRepository vagy JpaRepository interfészeket, amelyek előre definiált adatműve-

leteket nyújtanak az entitásokhoz. Minden entitás osztályra alkalmaztam az @Entity annotációt, hogy jelezzem, hogy az adott osztály egy JPA entitás, és kapcsolódik egy adatbázistáblához.

Minden entitáshoz létrehoztam getter-setter metódusokat az attribútumok kezelésére és beállítására, valamint egy toString() metódust az objektumok olvasható formában történő megjelenítésére.

User

A User entitásban vagyis a User.java fájlban található a felhasználó osztály. Ebben az osztályban határozom meg, hogyan kerülnek az adatok a User táblába az adatbázisban. Az id mező az adatbázisban az egyedi azonosítókat tárolja, amelyek segítségével azonosítani lehet egy adott felhasználót a rendszerben. Ezt a mezőt az adatbázis automatikusan generálja a "GenerationType.AUTO" stratégiát alkalmazva. A többi entitás azonosító mezőjénél is a "GenerationType.AUTO" stratégiát használom az elsődleges kulcs generálásához. A User entitás kapcsolatban áll a Thesis, a Review és a Session táblákkal. A role mező egy enum attribútum, amely lehetővé teszi a különböző felhasználói szerepkörök (például adminisztrátor, hallgató stb.) könnyebb azonosítását és kezelését az alkalmazásban. A username, a fullname és az email segíti a felhasználók azonosítását és kommunikációját az alkalmazásban. A neptun kód, születési dátum, születési hely, anyja leánykori neve és a törzskönyvi szám mezők hozzájárulnak a hallgató könnyebb beazonosításához, valamint elengedhetetlen adatok lesznek később a záróvizsga jegyzőkönyv generálásához. A munkahely, a név előtag és a beosztás a többi szerepkör meghatározásához járulnak hozzá. A jelszó mező tárolja a felhasználó jelszavát, amelyet az adatbázisban hash-elt formában tárolok, így a tényleges jelszavak nem láthatóak a rendszerben.

Thesis

A Thesis entitás az alkalmazásban olyan információkat tárol, amelyek az egyetemi hallgatók által írt szakdolgozatokra vonatkoznak. Az id mező a Thesis táblában egyedi azonosítót jelöl, amely segíti az adatbázisban tárolt szakdolgozatok egyértelmű azonosítását és kezelését. A Thesis tábla összeköttetésben áll az adatbázisban szereplő összes entitással. A session_id mező egy idegen kulcs amely a session tábla id mezőjére mutat. A két tábla között egy-több kapcsolat áll fenn, mivel egy záróvizsgához több szakdolgozat is tartozhat. A kapcsolatot Java-ban a @ManyToOne és @OneToMany annotációval valósítottam meg. A kapcsolatot a @JoinColum annotációval egy mezőre képeztem le ez esetben ez a sessionId mező lenne. Az insertable = false és updatable = false attribútumok azt jelzik, hogy az adatbázisba nem kell beilleszteni vagy frissíteni a thesis entitásban lévő sessionId mezőt, mert a kapcsolatot a session entitásban kell kezelni. Az idegenkulcsok kezelése hasonló a fent bemutatott példához más entitásoknál is. A userId, supervisorId és a consultantId mezők is idegenkulcsok melyek a user tábla userId mezőjére mutatnak. Itt is egy-több kapcsolatot valósítottam meg, mivel egy hallgatónak, témavezetőnek és konzulensnek lehet több szakdolgozata. A topicsId adattag a topic tábla id mezőjére mutat és egy-több kapcsolatot valósítottam meg itt is. A title, faculty, department, speciality, language, hasMscApply, submissionDate, answer mezők a szakdolgozatok könnyebb beazonosítására szolgálnak. Az isUnderReview a folyamatban lévő szakdolgozatra történő bírálatot szeretné reprezentálni, míg

a reviewsRemaining mező a hátralévő bírálatokat tartja számon, amely alapesetben beállításra kerül 2-es értékkel.

Topic

A topic entitás a záróvizsgán húzott tételeket tartalmazza. Az id mező a topic táblában egyedi azonosítót jelöl, amely a tétel sorszámát tárolja. A topic mezőben pedig a záróvizsga tételek szöveges formátumban találhatóak meg. A topic mezőnél szükség volt a String típusnál nagyobb kapacitásra, mivel a záróvizsga témák leírása ennél hosszabb. Ennek megfelelően a topic attribútum típusát VARCHAR(7500) karakterre növeltem, hogy elegendő helyet biztosítsak a témák részletes leírásához.

Review

Az review entitás az alkalmazásban az egyetemi szakdolgozatok értékeléseit tárolja. A szakdolgozat értékeléseket különböző szempontok alapján végezzük, és ezeket az adatokat rögzítjük az entitásban. A táblában az reviewId egyedi azonosítóként szerepel. A reviewerId egy idegenkulcs, amely a user tábla userId mezőjére mutat és egy-több kapcsolatot hoztam ehhez létre. A thesisId mező szintén egy idegenkulcs amely a thesis tábla id mezőjére mutat. Emellett az invitationDate, score, description, city és submissionDate mezők szolgálnak az értékelések egyértelmű azonosítására és részletezésére. A description mező esetében felmerült az adatméret korlátozottsága, ezért a @Column-Definition annotáció segítségével kiküszöböltem a problémát, beállítva a mező típusát LONGTEXT-re, ezáltal lehetővé téve a bírálatok részletes leírásának tárolását.

File

A file entitás az alkalmazásban a fájlok tárolását szolgálja a szakdolgozatokhoz és egyéb kapcsolódó dokumentumokhoz. Az id mező az file entitásban egyedi azonosítóként szolgál, amely segíti az adott fájl egyértelmű azonosítását az adatbázisban. A thesisId idegenkulcsként hoztam létre amely a thesis tábla id mezőjével áll kapcsolatban. Egy-több kapcsolatot valósítottam meg mivel egy szakdolgozathoz több fájl is rendelhető. A name, uploadTime és uuid mezők a fájlok egyértelműbb beazonosítására szolgálnak. Az UUID (Universally Unique Identifier) egy olyan azonosító, amelyet egyedülállóan generálódik minden egyes fájl esetén. Ez segít elkerülni az azonos nevű vagy azonos tartalmú fájlok ütközését, valamint könnyen lehetőséget nyújt az adatok azonosítására és kezelésére.

Session

A Session entitás az alkalmazásban a záróvizsgák adatait tárolja. Az id mező az Session entitásban egyedi azonosítóként szolgál, amely lehetővé teszi az adott záróvizsga egyértelmű azonosítását az adatbázisban. A notary és a president idegenkulcsok, amelyek segítségével könnyen beazonosítható a jegyző és az elnök személye. A members és students mezők a záróvizsga tagokat és hallgatókat reprezentálja ezért egy listába tárolom mindkét adattagot. Itt egy több-több kapcsolatot hoztam létre mivel több tag és több hallgató is részt vesz egy záróvizsgán. A date, startHour, endHour, location, description és a code adattagok a záróvizsgák egyértelműbb beazonosítására szolgálnak.

4.3.3. Service

A Spring Boot alkalmazásban a service osztályok általában az üzleti logika rétegét képviselik. Ezek az osztályok felelősek az üzleti folyamatok végrehajtásáért, adatfeldolgozásért és egyéb műveletekért, amelyeket az alkalmazás végrehajt.

UserService

A UserService osztály a felhasználók kezeléséért és azokhoz kapcsolódó műveletek végrehajtásáért felelős. Az addUser metódus lehetővé teszi, hogy új felhasználót adjunk hozzá az adatbázishoz, míg az UpdateUsers metódussal felülírhatjuk a felhasználók egyes adatait illetve a deleteUserbyId metódussal törölhetünk is akár felhasználót az egyedi azonosítója alapján az adatbázisból. Ezenkívül létrehoztam különböző metódusokat amelyek az adott felhasználó beazonosítására szolgálnak amellyel meg tudjuk őket jeleníteni őket egy táblázatban illetve a drop-down listában tudjuk megjeleníteni őket. A felhasználók beazonosíthatóak név, egyedi azonosító és szerepkör alapján. Ezekhez a keresésekhez alkalmaztam a dao rétegben létrehozott sql querry-ket annak érdekében hogy minden felhasználóról csak bizonyos adatok jöjjenek létre. A findStudentsByLoggedInReviewer függvényre a thesisstatus lap miatt van szükség, mivel szerettem volna úgy megcsinálni az oldalt, hogy a bíráló és a témavezető is csak azt a hallgatót lássa amire fel lettek kérve bírálatra. Az init metódusokkal igazából inicializálok különböző szerepkörökből felhasználókat azért, hogy amikor elindul az alkalmazás ezek már létrejönnek azonnal. Ez sokban segítette néhány funkció tesztelését, hogy nem kellett mindig létrehozni felhasználót, szakdolgozatot stb.-t hogy egyből ki tudjak próbálni egy új funkciót. A jelszóváltoztatást segítő metódusokat is itt készítettem el. Készítettem először is egy getEncodedPassword metódust amelyben használom a PasswordEncoder osztály passwordEncoder metódusát annak érdekében hogy a jelszó titkosítva legyen hash formátumban az adatbázisban.

ThesesService

A ThesesService osztály felelős a szakdolgozatokkal kapcsolatos műveletek végrehajtásáért és az azokhoz kapcsolódó adatbázis-műveletek kezeléséért. Az addTheses updateTheses és a deleteThesesbyId metódusok funkciója nem tér el a userService-ben látottakhoz képest annyi hogy most szakdolgozatokkal végezhető el ez a művelet. Itt is megtalálhatóak különböző metódusok, amelyek a szakdolgozat pontos beazonosítására szolgálnak akár egyedi azonosító alapján és a felhasználó egyedi azonosítója alapján is.

TopicService

A TopicService osztály felelős a záróvizsga tételekkel kapcsolatos műveletek végrehajtásáért és az azokhoz kapcsolódó adatbázis-műveletek kezeléséért. Az addTopics metódus segítségével új záróvizsga tételt lehet hozzáadni egy szakdolgozathoz. A chooseTopic eljárásban valósítottam meg egy jdbcTemplattel azt hogy amikor kiválasztásra kerül a topic akkor az a theses tábla topicId mezőjében is megjelenjen. A getTopic metódussal visszatérítem a tételt és a tételhez tartozó egyedi azonosítót. Az initTopics metódussal inicializálom a tételeket. A tételeket egy tömbben tárolom el. Mivel amikor újból indítom az alkalmazást akkor mindig legenerálódik ismételten a 27 db tétel ezért írtam rá egy sql querry-t amely megszámolja hogy mennyi szakdolgozat van és ha még nincs szakdolgozat akkor inicializálja a tételeket.

FileService

A FileService felelős a fájlok kezeléséért, beleértve a feltöltést, letöltést, fájlnevek lekérdezését és törlését. Az assembleFileWithStudent metódus összerendezi a fájlt a megadott szakdolgozattal. Ellenőrzi, hogy a fájl és a szakdolgozat megtalálható-e az adatbázisban. Ha megtalálhatóak, akkor hozzárendeli a fájlt a szakdolgozathoz. A getFilenameByUUID metódus az adatbázisból keresi és visszaadja a fájl nevét a UUID alapján. Azért van erre szükség mert a fájl nevébe belegenerálódik a uuid ami biztonsági szempontból szerintem előnyösebb mint az egyedi azonosító mivel azt könnyebb lenne kitalálni és ezáltal könnyebb lenne hozzáférni a fájlhoz. A getFile metódus visszaadja a fájlt mint FileSystemResource objektumot a megadott fájlnév alapján. Ellenőrzi, hogy a fájl létezik-e, és ha igen, akkor visszaadja azt. A deleteFileById metódus uuid alapján töröl az adatbázisból fájlokat. Az uploadFile metódussal feltölti a felhasználó a megadott fájlt. Először létrehoz egy új File objektumot a feltöltéshez. A fájl típusától függően megfelelő mappákba menti a fájlt, és beállítja a fájl nevét és egyéb adatait az adatbázisban.

EmailSenderService

Ez az osztály felelős az e-mail küldés funkcióért. A Spring JavaMailSender interfészét használja a levelek küldésére.

A sendEmailWithAttachment metódus egy olyan e-mailt küld, amelynek van csatolmánya. A paraméterei a címzett e-mail címe (toEmail), a levél tárgya (subject), a levél szövege (body) és a csatolt fájl elérési útvonala (pathToAttachment). Ez a metódus létrehoz egy MimeMessage objektumot, amelyet a JavaMailSender segítségével állít be. A csatolt fájlt a FileSystemResource segítségével adja hozzá a levélhez. Ha a csatolt fájl nem található, akkor kivételt dob. Végül a mailSender.send(message) hívással elküldi az e-mailt, majd a konzolra kiírja, hogy sikeresen kiküldésre került az e-mail.

A sendEmail metódus egy olyan e-mailt küld, amelynek nincs csatolmánya. Paraméterei ugyanazok, mint a sendEmailWithAttachment metódusnak, csak itt nincs pathToAttachment paraméter. A funkciója ugyanaz, csak csatolmány nélkül küldi el az e-mailt.

ForgottenPasswordService

Ez a szolgáltatás osztály kezeli a jelszófelejtéssel kapcsolatos folyamatokat, például az új jelszó kérést és az e-mailen keresztüli jelszóvisszaállító token küldését.

sendForgottenPasswordTokenByEmail(String username): Ez a metódus küld egy e-mailt a felhasználónak, amely tartalmazza a jelszóvisszaállító token-t. Az e-mail tartalmazza a token-t és egy linket a jelszó visszaállításához. Ha a felhasználó nem található meg a megadott felhasználónév alapján, akkor egy UsernameNotFoundException kivételt dob.

setNewPassword(String username, String token, String newPassword): Ez a metódus lehetővé teszi a felhasználó számára, hogy új jelszót állítson be a kapott token segítségével. Ellenőrzi, hogy a token érvényes-e, majd megváltoztatja a felhasználó jelszavát a megadott új jelszóra.

setNewPasswordtwo(Integer userId, String oldPassword, String newPassword): Ez a metódus lehetővé teszi a felhasználó számára, hogy új jelszót állítson be a régi jelszó-

val. Ellenőrzi, hogy a megadott régi jelszó helyes-e, majd megváltoztatja a felhasználó jelszavát a megadott új jelszóra.

isValidPassword(Integer userId, String password): Ez a metódus ellenőrzi, hogy a megadott jelszó helyes-e a felhasználóhoz tartozó jelszóval.

isValidToken(String username, String token): Ez a metódus ellenőrzi, hogy a megadott token érvényes-e a felhasználóhoz.

JWT alapú autentikáció és azonosítás kezelése

Az általad létrehozott JWT alapú autentikációs és azonosítási rendszer kifejezetten fontos szerepet tölt be az alkalmazásban, biztosítva a felhasználók biztonságos hozzáférését és azonosítását. Az alábbiakban részletesen bemutatom az általad létrehozott osztályokat és szolgáltatásokat:

JwtRequest és JwtResponse entitások: JwtRequest: Ez az osztály a felhasználónév és a jelszó mezőket tartalmazza. Az alkalmazás ezen osztály segítségével fogadja a felhasználói bejelentkezési adatokat. JwtResponse: A sikeres autentikáció után ez az osztály tartalmazza a felhasználót és a hozzá tartozó JWT tokent. Ezt a tokent kliensoldalon lehet tárolni és későbbi hitelesítéshez használni. JwtRequestFilter és JwtAuthenticationEntryPoint: JwtRequestFilter: Ez a szűrő felelős a bejövő HTTP kérések figyeléséért. Amikor egy kérés érkezik, kinyeri belőle a JWT tokent a fejlécből, majd ellenőrzi annak érvényességét. Ha a token érvényes, akkor beállítja a felhasználói azonosítást a Spring Security SecurityContextHolder objektumban. JwtAuthenticationEntryPoint: Ez az osztály kezeli az autentikációs hibákat. Amennyiben a kliens nem rendelkezik megfelelő hitelesítő adatokkal vagy érvénytelen tokent küld, a rendszer HTTP 401 Unauthorized választ küld vissza. JwtService: JwtService: Ez a szolgáltatásosztály felelős az autentikációs folyamat teljes körű kezeléséért. Itt valósul meg a felhasználók azonosítása, az autentikáció során a JWT tokenek generálása és validálása. Emellett az osztály implementálja a Spring Security UserDetailsService interfészt, így lehetővé teszi a felhasználói adatok betöltését az alkalmazás adatbázisából. Ezen osztályok és szolgáltatások kombinációja biztosítja a teljes körű JWT alapú autentikációs rendszer működését az alkalmazásban. A JwtRequestFilter figyeli és kezeli az érkező kéréseket, míg a JwtService végzi az autentikációs logika megvalósítását és a JWT tokengenerálást. Ezáltal garantálva van a felhasználók biztonságos hozzáférése és azonosítása az alkalmazásban.

ReviewService

A ReviewService a bíráló és a témavezető által készített bírálatok kezeléséért felelős a rendszerben. Az addReview metódus felel a bírálatok adatbázisba mentéséért. Itt külön beállításra kerülnek a remainingReview és az underReview mezők amelyek a hátra lévő és folyamatban lévő bírálatokat kezelik. A getReviewData metódus a bírálatokat kigyűjtésére szolgál. A findThesesByUserIdAndReviewerId metódus a szakdolgozatokat keresi meg témavezető és bírálói azonosító alapján. A findReviewByThesisId metódus pedig szakdolgozat egyedi azonosítója alapján keresi meg a bírálatot. A Proba metódusban történik a bírálat word dokumentummá generálása. A felhasználó amint megadja az adatokat kliensoldalon az elmegy a szerver felé aki feldolgozza ezeket az adatokat és ezek alapján az információk alapján egy word dokumentumot készít. A metódus paraméterként egy objektumot kap, amely tartalmazza a bírálat részleteit.

Az objektumot String formátumra alakítja, majd feldolgozza és kinyeri belőle a szükséges információkat, mint például a hallgató nevét, a dolgozat címét, a bíráló nevét, a bírálat részletes leírását stb. Az Apache POI segítségével létrehoz egy üres Word dokumentumot. Beállítja a dokumentum fejlécét, amely tartalmazza az egyetem nevét és logóját. Beilleszti a dokumentumba a felhasználók által megadott információkat, mint például a hallgató nevét, a dolgozat címét, a bíráló nevét, a bírálatot stb. Ezeket a részleteket formázott szövegként adja hozzá a dokumentumhoz. A generált Word dokumentumot elmenti egy kijelölt könyvtárba. A fájlnevet a dokumentumhoz kapcsolódó adatokból generálja, például a hallgató nevét és egy uuid-t is tartalmazza. A metódus elmenti a generált Word dokumentum adatait az adatbázisba. Az adatbázisban tárolja a fájl nevét, feltöltési idejét és egyedi azonosítóját, hogy később könnyen visszakereshető legyen. A fájl mentése után A Word dokumentum generálása során számos formázási lehetőséget alkalmazok a szöveg és egyéb elemek megjelenítéséhez. Ezek közé tartozik a setText, setFontSize(12), setFontFamily metódusok használata, amelyekkel beállíthatom a szöveg stílusát, betűméretét és betűtípusát. Emellett lehetőségem van új bekezdéseket létrehozni a szöveg tagolásához, valamint képeket is beilleszthetek a dokumentumba.A táblázatokat is könnyedén formázhatom, például a cellák méretét vagy szegélyeit állíthatom be. Emellett a táblázatok celláinak láthatóságát is szabályozhatom, hogy megfeleljenek az adott formai követelményeknek. Ezen funkciók segítségével teljes körűen testreszabhatom és formázhatom a generált Word dokumentumokat, hogy azok megfeleljenek az elvárásoknak és esztétikusak legyenek. A táblázatok esetében létrehoztam egy külön függvényt a méretük beállítására, ami növeli a kód olvashatóságát és karbantarthatóságát. Ennek köszönhetően könnyen változtathatom a táblázatok méretét az alkalmazás különböző részein, ami javítja a fejlesztés hatékonyságát és karbantarthatóságát. A külön függvény segítségével elkerülhető az ismétlődő kód, így javítva a programstruktúrát.

GenerateDocxService

A generateDocxService osztály felelős a záróvizsga jegyzőkönyv generálásáért. A függvény egy Object típusú paramétert kap, majd ebből a paraméterből kinyeri a szükséges adatokat, mint például a hallgató nevét, a szakdolgozat címét, a bizottság tagjainak adatait stb. Ezeket az adatokat felhasználva létrehoz egy Word dokumentumot, amelyben formázott módon jeleníti meg a jegyzőkönyvet. A dokumentumban címeket, bekezdéseket, táblázatokat és egyéb formázott szövegeket helyez el a megfelelő helyeken a jegyzőkönyv struktúrájának megfelelően. A végeredmény egy Word (.docx) fájl lesz, amely tartalmazza a generált záróvizsga jegyzőkönyvet.

SessionService

A SessionService egy szolgáltatásréteg az alkalmazásban, amely a záróvizsga (session) entitással kapcsolatos üzleti logikát valósítja meg. A findSessionById metódus lehetővé teszi egy záróvizsga entitás keresését az azonosító alapján a SessionDao rétegen keresztül. Az addSessions metódus a záróvizsga felvitelét végzi el. Először lementi a záróvizsgát az adatbázisba, ezután emaileket küld az összes érintett félnek (tagok, hallgatók, elnök, jegyző, stb.) az emailSenderService osztály sendEmail metódus segítségével. A további metódusok arra szolgálnak, hogy az eddigi záróvizsgákat lekérjük a rendszerből.

StatusService

A StatusService a szakdolgozatok állapotának kezelésével és azokhoz kapcsolódó folyamatok végrehajtásával foglalkozik. A getNeccessaryData metódus lekéri azokat az adatokat, amelyek szükségesek a szakdolgozatok állapotának és azokhoz kapcsolódó folyamatoknak a kezeléséhez. A metódus lefuttatja a megfelelő adatbázis-lekérdezést a FileDao segítségével, majd feldolgozza az eredményeket és visszaadja egy listában. A findFilesByThesesId metódus megkeresi és feldolgozza az adott thesishez tartozó fájlokat. Először lekéri a megfelelő szakdolgozatot az azonosítója alapján. Ezután lekéri a szakdolgozathoz tartozó hallgató adatait és a kinyert email címre elküldi a szakdolgozathoz tartozó bírálatokat egy email-ben.

ReportstatusService

A ReportStatusService a szakdolgozatok állapotának jelentésével kapcsolatos üzleti logikát valósítja meg. A requestForReview metódus a bírálónak küld felkérést egy szakdolgozat bírálatra. A bíráló felkérése után rögtön email megy ki a bírálónak az email címére hogy felkérés érkezett benne a szakdolgozat címével. Ezután a szakdolgozat állapota az isunderReview mezőben megváltozik. A findThesesUnderReview metódus visszaadja az összes olyan szakdolgozatot, amely jelenleg bírálat alatt áll, azaz az "isUnderReview" mező értéke igaz. Ez lehetővé teszi a rendszer számára, hogy kezelje és megjelenítse ezeket a szakdolgozatokat a jegyző és az elnök számára. A findReviewedTheses metódus visszaadja az összes olyan szakdolgozatot, amely már átesett a bíráláton. Ezeket a szakdolgozatokat általában megfeleltnek vagy nem megfeleltnek minősítették, és az értékelésük befejeződött.

4.3.4. Vezérlők

4.4. A REST API áttekintése

4.5. Frontend

5. fejezet

Tesztelés

A fejezetben be kell mutatni, hogy az elkészült alkalmazás hogyan használható. (Az, hogy hogyan kell, hogy működjön, és hogy hogy lett elkészítve, az előző fejezetekben már megtörtént.)

Jellemzően az alábbi dolgok kerülhetnek ide.

- Tesztfuttatások. Le lehet írni a futási időket, memória és tárigényt.
- Felhasználói kézikönyv jellegű leírás. Kifejezetten a végfelhasználó szempontjából lehet azt bemutatni, hogy mit hogy lehet majd használni.
- Kutatás kapcsán ide főként táblázatok, görbék és egyéb részletes összesítések kerülhetnek.

6. fejezet

Összefoglalás

Hasonló szerepe van, mint a bevezetésnek. Itt már múltidőben lehet beszélni. A szerző saját meglátása szerint kell összegezni és értékelni a dolgozat fontosabb eredményeit. Meg lehet benne említeni, hogy mi az ami jobban, mi az ami kevésbé jobban sikerült a tervezettnél. El lehet benne mondani, hogy milyen további tervek, fejlesztési lehetőségek vannak még a témával kapcsolatban.

Irodalomjegyzék

[1] Kurzus kezelő szoftverek
Course Management Software explained
https://www.arlo.co/features/short-course-management-software

[2] Egységes Tanulmányi Rendszer
Egységes Tanulmányi Rendszer
https://hu.wikipedia.org/wiki/Egys%C3%A9ges_Tanulm%C3%A1nyi_Rendszer

[3] Neptun
HALLGATÓI FELHASZNÁLÓI SEGÉDLET
https://neptun.unideb.hu/download/HWEB_6_3.pdf

[4] .NET
.NET Build. Test. Deploy.
https://dotnet.microsoft.com/en-us/

[5] PHP
PHP Official Website
https://www.php.net/

[6] Neptun Záróvizsga jelentkezés
Záróvizsga neptunos jelentkezés
https://btk.pte.hu/sites/btk.pte.hu/files/2020-10/zv%20neptunos%20jelentkez%C3%A9s2020okt%C3%B3ber.pdf

[7] KRÉTA
A KRÉTA Elektronikus napló
https://tudasbazis.ekreta.hu/

[8] KRÉTA értékelés
KRÉTA e-Napló modul Évközi értékelések rögzítése
https://youtu.be/xwqjan45l18?si=ocG4mcJBfzX4N-_C&t=46

[9] Blackboard Learn
Blackboard Learn Homepage
https://help.blackboard.com/Learn/Student/Ultra/Getting_Started/Log_in_to_Learn

[10] Blackboard Learn Review
Grading Assignments in Blackboard Learn
https://youtu.be/dMw0yvKL4Ys?si=0RIplrefVgypOpSB&t=44

[11] MySQL homepage
About MySQL
https://www.mysql.com/about/

[12] MySQL Workbench
About MySQL Workbench
https://www.mysql.com/products/workbench/

[13] Java
Java programozási nyelv
https://hu.wikipedia.org/wiki/Java_(programoz%C3%A1si_nyelv)

[14] Spring Boot
Java Spring Boot
https://www.ibm.com/topics/java-spring-boot

[15] Model-View-Controller
Model-View-Controller
https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

[16] HTTP
HTTP dokumentáció
https://datatracker.ietf.org/doc/html/rfc2616

[17] Angular
Angular dokumentáció
https://angular.io/guide/component-overview

[18] HTML
HTML dokumentáció
https://www.w3schools.com/html/

[19] Typescript
Typescript dokumentáció
https://www.typescriptlang.org/

[20] CSS
CSS dokumentáció
https://www.w3schools.com/css/

[21] Bootstrap
Bootstrap dokumentáció
https://getbootstrap.com/

[22] Vue
Vue dokumentáció
https://vuejs.org/

[23] React
React dokumentáció
https://react.dev/

CD Használati útmutató

Ennek a címe lehet például *A mellékelt CD tartalma* vagy *Adathordozó használati útmutató* is.

Ez jellemzően csak egy fél-egy oldalas leírás. Arra szolgál, hogy ha valaki kézhez kapja a szakdolgozathoz tartozó CD-t, akkor tudja, hogy mi hol van rajta. Jellemzően elég csak felsorolni, hogy milyen jegyzékek vannak, és azokban mi található. Az elkészített programok telepítéséhez, futtatásához tartozó instrukciók kerülhetnek ide.

A CD lemezre mindenképpen rá kell tenni

- a dolgozatot egy `dolgozat.pdf` fájl formájában,
- a LaTeX forráskódját a dolgozatnak,
- az elkészített programot, fontosabb futási eredményeket (például ha kép a kimenet),
- egy útmutatót a CD használatához (ami lehet ez a fejezet külön PDF-be vagy MarkDown fájlként kimentve).