

**SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR,
INFORMATIKA SZAK**



SAPIENTIA
ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM

Programozói versenyfeladatok elemzését és értékelését segítő
webalkalmazás

DIPLOMADOLGOZAT

Témavezető:
Dr. Osztián Erika,
Egyetemi tanár

Végzős hallgató:
Molnár Balázs

2021

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,
SPECIALIZAREA INFORMATICĂ



UNIVERSITATEA
SAPIENTIA

Aplicație web pentru analiza și evaluarea problemelor de concurs
de programare

LUCRARE DE DIPLOMĂ

Coordonator științific:
Dr. Osztián Erika,
Profesor universitar

Absolvent:
Molnár Balázs

2021

**SAPIENTIA HUNGARIAN UNIVERSITY OF
TRANSYLVANIA
FACULTY OF TECHNICAL AND HUMAN SCIENCES
COMPUTER SCIENCE SPECIALIZATION**



SAPIENTIA
HUNGARIAN UNIVERSITY
OF TRANSYLVANIA

Web application for analyzing and evaluating programming
competition tasks

BACHELOR THESIS

Scientific advisor:
Dr. Osztián Erika,
Full Professor

Student:
Molnár Balázs

2021

Declarație

Subsemnatul/a MOLNÁR BALÁZS, absolvent(ă) al/a specializării INFORMATICĂ, promoția 2018-2021 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, Târgu Mureș

Data: 06.07.2021

Absolvent

Semnătura.....

Kivonat

Dolgozatom témája: programozói versenyfeladatok elemzését és értékelését segítő webalkalmazás. A COVID-19-es világjárvány során számos megoldásra váró problémával szembesültünk az élet szinte minden területén. Az oktatás és tanügyi rendszernek hatalmas változásokon kellett keresztül mennie, ahhoz hogy megfelelő szintű és minőségű tudást tudjanak átadni a jövő generációi számára.

Dolgozatomban egy ilyen problémára próbáltam megoldást találni. A probléma forrása nem újkeletű, eddig is érzékelhető volt a magyar nyelvű programozás oktatása során. A diákok általában nem kapnak megfelelő visszajelzést az általuk elkészített és beküldött feladatokról, csupán egy bizonyos pontszámot, a megoldott feladatok helyességének függvényében. Továbbá eddig nem volt egy olyan magyar nyelvű, internetes feladatgyűjtemény, amely azok számára nyújt segítséget, akik nem megfelelő idegennyelvű ismerettel rendelkeznek.

Így jött az ötlet, hogy egy olyan magyar felületű webalkalmazást készítsek, amely könnyen elérhető és használható mindenki számára, aki rendelkezik internet hozzáféréssel.

A rendszer egyszerű mégis hatékony, számos bővítési lehetőséggel, a tanárok és diákok igényei szerint könnyen alakítható. Az alkalmazásra regisztrálhatnak úgy diákok, mint tanárok. Minden diák egy, a tanára által létrehozott osztályhoz csatlakozhat, majd feladatok tucatjait oldhatja meg, amelyeket akár a saját tanáruk, akár más tanárok készítettek vagy töltöttek fel. A tanuló által kidolgozott és feltöltött megoldásokat pedig a vezető tanár könnyedén értékelheti, és elősegítheti a hatékony fejlődését javaslataival.

Bár nagyon nehéz volt a kezdés, örömmel tölt el az a tudat, hogy dolgozatomban végül sikerült elérnem az elsődleges céljaimat, és képes voltam egy mindenki számára könnyen elérhető magyar nyelvű, versenyszintű feladatok megoldására és kiértékelésére alkalmas webalkalmazást létrehozni. Ennek ellenére be kellett látnom, hogy ez egy hatalmas projekt, számos továbbfejlesztési lehetőséggel, melyre akár éveken keresztül is lehet dolgozni. Ez akkor is bebizonyosodott, amikor a már létező webalkalmazások felkutatása során számos olyan alkalmazást találtam, amely céljában hasonló az enyémhez, de több év nehéz munkájának az eredménye.

Kulcsszavak: webalkalmazás, oktatás, kommunikáció, tanári jelenlét, hatékonyság.

Rezumat

Subiectul disertației mele este o aplicație web, care ajută la analiza și evaluarea sarcinilor de concurență ale programatorilor. În timpul pandemiei COVID-19, ne-am confruntat cu o serie de probleme de rezolvat în aproape toate domeniile vieții.

Sistemul de educație și formare a trebuit să treacă prin schimbări enorme pentru a transmite nivelul și calitatea corectă a cunoștințelor generațiilor viitoare. În disertație am încercat să găsec o soluție la o astfel de problemă. Sursa problemei nu este nouă, s-a simțit până acum în predarea programării în limba maghiară. De obicei, elevii nu primesc feedback adecvat cu privire la sarcinile pe care le-au pregătit și le-au prezentat, ci doar un anumit scor, în funcție de corectitudinea sarcinilor rezolvate. Mai mult, până acum nu a existat nici o colectare de sarcini pe Internet în limba maghiară care să îi ajute pe cei care nu au o bună cunoaștere a unei limbi străine.

Atât studenții, cât și profesorii se pot înregistra pentru aplicație. Fiecare elev se poate alătura unei clase create de profesorul său și apoi poate completa zeci de sarcini pregătite sau completate fie de propriul profesor, fie de alți profesori.

Soluțiile dezvoltate și încărcate de elev pot fi ușor evaluate de către profesorul lui și pot ajuta la dezvoltarea eficientă a acestuia cu sugestiile sale. Cu toate acestea, am realizat că proiectul este o muncă uriașă, cu multe oportunități de dezvoltare ulterioare la care s-ar putea lucra ani de zile. Acest lucru dovedesc aplicațiile găsite, care au avut același scop: a contribui la eficiența muncii atât ai elevilor cât și ai profesorilor.

Cuvinte de cheie: aplicație web, educație, comunicare, prezența profesorului, eficiență.

Abstract

The topic of my dissertation is a web application that helps to analyze and evaluate programmer competition tasks. During the COVID-19 pandemic, we faced a number of problems to be solved in almost every area of life. The education system had to go through enormous changes in order to be able to pass on the right level and quality of knowledge to future generations.

In my dissertation I tried to find a solution to such a problem. The source of the problem is not new, it has been felt so far in the teaching of Hungarian language programming. Students usually do not receive adequate feedback on the tasks they have prepared and submitted, only a certain score, depending on the correctness of the solved tasks. Furthermore, so far there has been no online collection of tasks in Hungarian that provides help to those who do not have adequate foreign language skills.

This is how I came up with the idea to create a web-based application that is easy to access and use for anyone with internet access. The system is simple yet efficient, with many expansion options, and can be easily adapted to the needs of teachers and students. Both students and teachers can register for the app. Each student can join a class created by their teacher and then complete dozens of assignments created or uploaded by either their own teacher or other teachers. The solutions developed and uploaded by the student can be easily evaluated by the lead teacher and their effective development can be promoted with his suggestions.

Although it was very difficult to start, I am pleased to know that I finally achieved my primary goals to create a web application suitable for solving and evaluating competition-level tasks. Nevertheless, I had to realize that this was a huge project, with many opportunities for improvement that could be worked on for years. This was also proven when I found a number of applications that were similar in purpose to mine in my search for preexisting web applications.

Keywords: web application, education, communication, teacher presence, efficiency.

Tartalomjegyzék

1. Bevezető	11
1.1. Célkitűzések	12
1.2. Indoklás	12
2. Előzetes felmérés	13
3. Létező webalkalmazások felkutatása	15
3.1. LeetCode vs. Kódolj!	15
3.2. HackerRank vs. Kódolj!	16
3.3. Pbinfo vs. Kódolj!	17
3.4. Eötvös Loránd Tudományegyetem Informatikai Kar MESTER - online programozási feladatbank	17
4. Technológiai áttekintés	19
4.1. Adatbázis-kezelő rendszerek	19
4.1.1. SQLite	19
4.1.2. PostgreSQL	20
4.1.3. SQLite vs PostgreSQL	20
4.2. Webes keretrendszerek	22
4.2.1. Django	23
4.3. Amazon Web Services (AWS) - Cloud Computing Services	24
4.3.1. Amazon Relational Database Service (Amazon RDS)	25
4.3.2. Amazon Simple Storage Service (Amazon S3)	25
4.4. Apache HTTP Server vs Heroku	25
5. A rendszer specifikációja	27
5.1. Felhasználói követelmények	27
5.2. Rendszerkövetelmények	30
5.2.1. Funkcionális követelmények	30
5.2.2. Nem funkcionális követelmények	32
6. Tervezés	33
6.1. A fejlesztéshez használt technológiák	33
6.2. A rendszer architektúra	33
6.2.1. Front-end	33
6.2.2. Back-end	36
6.2.3. Adatbázis	39

6.2.4. Felhasznált csomagok és könyvtárak	45
6.2.5. Modulok bemutatása	47
7. A felhasználói felület bemutatása	51
7.1. Regisztráció	51
7.2. Bejelentkezés	52
7.3. Kezdőlap	53
7.4. Feladatok	55
7.5. Profil	56
7.6. Osztályok megtekintése	57
Összefoglaló	59
7.7. Továbbfejlesztési lehetőségek	59
Ábrák jegyzéke	60
Irodalomjegyzék	63

1. fejezet

Bevezető

Világunk folyamatos változásokon megy keresztül szinte az élet minden területén. Nincs ez másképp az informatika területén sem. Sőt, talán ez a tudományágazat mutatja a legdinamikusabb, legsokoldalúbb fejlődést. Naponta jelennek meg az okosabbnál okosabb eszközök, újfajta technológiák és programozási nyelvek. Ez egy csodálatos világ, számos lenyűgöző lehetőséggel, csak nehéz lépést tartani különösen az oktatóknak és a diákoknak. Aki ezzel a tudománnyal foglalkozik, szinte biztosan tapasztalta már azt az örömet, amelyet egy komoly probléma sikeres megoldása során érez az ember. Talán ez az öröm az egyik hajtóereje ennek a hihetetlenül gyors fejlődésnek, persze sok más tényező mellett. A folyamatos küzdelem, próbálkozás, kudarok után elért sikerélmény, újabb és újabb lendülettel vezérli a 21. századi diákot.

Vannak könnyebb és nehezebb, időigényes és kevésbé időigényes, sok tapasztalatot vagy akár kevés tapasztalatot igénylő feladatok. Az emberek sokszínűek, így mindenki számára minden feladat más-más kihívást jelent. Egy valami viszont mindenki számára kulcsfontosságú: a munkájukra adott visszajelzések. Hogyan lehetett volna jobb megoldást találni? Hol hibáztam el az algoritmust? Miért nem működött az általam megfogalmazott módszer? Vagy éppen miért működött? Ezekre a kérdésekre sokan keresik a választ, akár egy egyszerűbb, akár egy nehezebb feladat megoldása során. Sokat segíthet egy hiba pontos magyarázata. Ez megakadályozhatja, hogy a jövőben ne kövessünk el hasonló hibákat, ugyanakkor elősegítheti a fejlődésünket.

Egy diák számára elengedhetetlen a munkájának a pontszám alapján történő kiértékelése, de ugyanakkor ez elveszíti a jelentőségét, ha nem kap magyarázatot a hibáira, illetve a helyes válaszaira.

1.1. Célkitűzések

A projekt célja, hogy egy olyan felhasználóbarát környezet létrehozása, mind a diákok, mind a tanárok részére, amelyen keresztül könnyedén és hatékonyan tudnak egymással kommunikálni, annak érdekében, hogy a diákok minél inkább megtanulják és elsajátítsák a magasabb szintű programozói feladatok megoldásának módszereit. A célkorosztály a 9-12.-es diákok. Úgy vélem, hogy ez a korosztály az, amely a leginkább érdekelt lenne egy ilyen platform használatára.

A dolgozat elkészítése során fontos szempont volt, hogy a diákok megfelelő visszajelzést kapjanak, arról hogy az általuk elkészített feladat megoldása mennyire jó. Továbbá nem csak a diákok, hanem a tanárok munkáját is próbáltam elősegíteni azzal, hogy egy olyan webalkalmazást hozok létre, amely már számos versenyszintű feladat tárháza. Így jelentősen le tudom csökkenteni az azzal eltöltött időt, melyet a tanárok arra kell fordítsanak, hogy folyamatosan feladatokat keressenek a versenyre készülő diákjaik számára, továbbá lehetőséget adtam a tanárok számára, hogy esetleg ők maguk tudjanak hozzáadni feladatokat a webalkalmazáshoz.

1.2. Indoklás

Főként azért találom hasznosnak ezt a webalkalmazást, mert egy teljesen újfajta megközelítése a programozás oktatásának. Számos olyan oldallal találkozhat a diák, ahol miután megoldotta és feltöltötte a kitűzött feladatot, egy automatikus visszajelzést kap, melynek háttérben bizonyos tesztfolyamatok vannak. Ennek az oldalnak a háttérben, ott van a tanár, akinek igencsak fontos szerepe van. Az általa adott visszajelzés irányt ad a diáknak, hogy a jövőben milyen hibákat ne kövessen el és segítséget is nyújthat, amikor problémába ütközik.

Az oldal erőssége abban is rejlik, hogy az oktatónak lehetősége van megfigyelni a diákjai haladását és motiválni is tudja őket a rendszer által felállított rangsor segítségével.

2. fejezet

Előzetes felmérés

A projekt megvalósítása előtt előzetes felmérést végeztem olyan személyek körében, akik a 9-12. osztályban programozói versenyeken vettek részt. A felmérést [Google Űrlapok](#) segítségével valósítottam meg.

Az általam készített és közzétett űrlapot 43 felhasználó töltötte ki, amelyek 67,4% férfi és 32,6% nő volt. Továbbá a felhasználók 79.1% 18-24 év közötti, és 16.3% 18 év alatti (lásd [2.1](#) ábra).



2.1. ábra. Nem és életkor

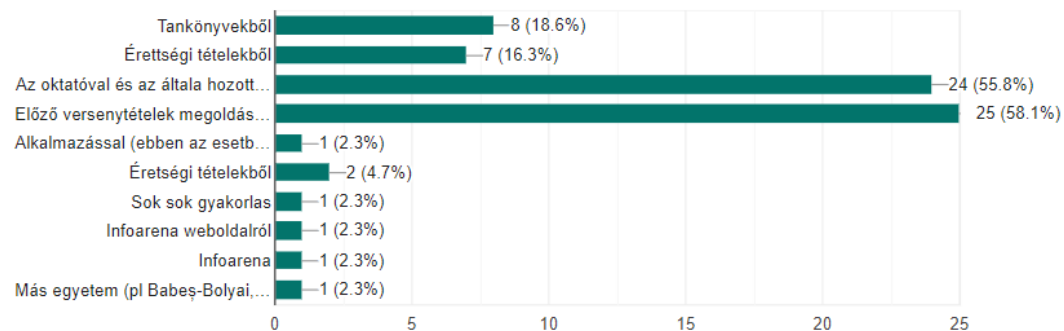
Az űrlapban a kérdések főként arra vonatkoztak, hogy milyen tapasztalataik vannak a felhasználóknak a programozói versenyekre való felkészülés során. Az első igen fontos dolog, amire rákérdeztem, az a versenyekre való felkészülés volt lásd a [2.2](#) ábrán. Az ábrából jól leolvasható, hogy a diákok felkészülése főként az oktató által hozott anyagokból, az előző években kiadott versenytételekből, illetve tankönyvekből és érettségi tételekből áll.

Továbbá az is kiderült, hogy a diákok amennyiben idegennyelvről, például román és angolról, próbálják lefordítani a feladatokat sokat hibáznak. Ez magával hozza a hibás megoldásokat is, nem is beszélve arról, hogy mennyire időigényes ez a folyamat némely esetben (lásd [2.3](#) ábra).

A kutatás kiértékelése során jól láthatóvá vált számomra, hogy nagy igény van egy olyan, könnyen hozzáférhető alkalmazásra, amelyen keresztül a diákok könnyedén, magyar nyelven, programozói verseny szintű feladatokat oldhatnak. Ezt mutatja számunkra a [2.4](#) ábra.

Hogyan/Miből készült fel a versenyre?

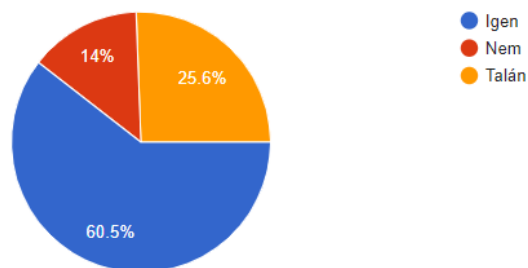
43 responses



2.2. ábra. Diákok által felhasznált eszközök versenyre való készülés céljából

Fordult elő már önnel, hogy csak azért tévedett, mert rossz volt vagy nem volt egyértelmű a fordítás?

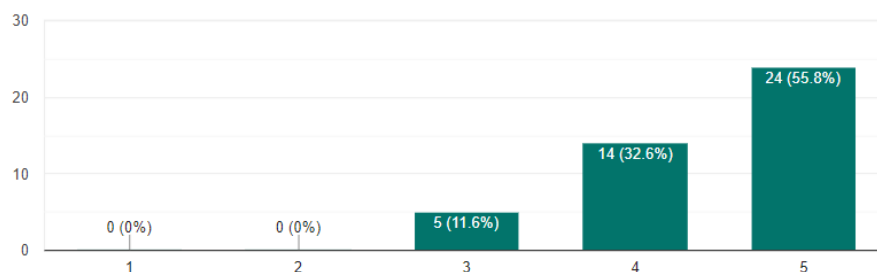
43 responses



2.3. ábra. Nem megfelelő fordítás

Használna egy olyan alkalmazást, amellyel anyanyelvén oldhatna meg versenyszintű feladatokat?

43 responses



2.4. ábra. Igényfelmérés

3. fejezet

Létező webalkalmazások felkutatása

A létező szoftverek, webalkalmazások felkutatása során számos tényezőt figyelembe kellett vennem, ilyenek például a nyelv, hozzáférhetőség, könnyen kezelhetőség. Viszonyítás alapjául hosszasan tanulmányoztam négy, hasonló célt szolgáló webalkalmazást:

- LeetCode
- HackerRank
- Pbinfo
- Eötvös Loránd Tudományegyetem Informatiai Kar MESTER - online programozási feladatbank

A tanulmányozás eredményeként a következő megfigyeléseket szeretném ismertetni, összehasonlítva az általam készített webalkalmazással, ennek előnyeivel és hátrányaival.

3.1. LeetCode vs. Kódolj!

A [LeetCode](#) egy népszerű Coding Challenge oldal, amely több száz különböző tematikájú és nehézségű feladatot tartalmaz. Főként olyan személyek számára ajánlatos az oldal, akik már rendelkeznek némi programozási háttérrel. Számos állásinterjú kérdés, feladat és cikk található benne, amelyek segítenek megérteni bizonyos programozási problémák megoldási sémáit. Mindazonáltal sikeres programozók osztják meg a felhasználókkal a tapasztalataikat bizonyos bejegyzések alatt. Felhasználói felülete könnyen áttekinthető, de mégis látványos.

Előnyei az én webalkalmazásommal szemben:

- A felhasználó ki tudja választani több mint 10 programozási nyelv közül, hogy mely nyelven szeretné megoldani az adott feladatot.
- A feladat leadása után visszajelzi, hogy hány teszt esetre futott le helyesen a program.
- Képes megmutatni, hogy mennyire gyors a kódunk a többi felhasználó kódjához viszonyítva. Például: A kódunk gyorsabb, a felhasználók 75%-ánál.

- Hatalmas feladat gyűjteménnyel rendelkezik.

Hátrányai az én webalkalmazásommal szemben:

- A kiértékelés automatikus, nincs egy tanár vagy bármilyen más felhasználó, aki értékelhetné a kódjukat, esetleg elakadás során segíthetne nekik.
- Nincsenek jól elkülönítve a feladatok szintjei.
- Rengeteg szolgáltatás prémium hozzáférést igényel, ilyen például a feladatok megoldásai vagy ötletek a megoldáshoz.
- Csupán angol nyelven elérhető.

3.2. HackerRank vs. Kódoj!

Ez is egy igen nagy népszerűségnek örvendő Coding Challenge felület. Itt a felhasználó számos lehetőség közül választhat annak függvényében, hogy mivel szeretné próbára tenni magát. Széleskörű választékot biztosít algoritmusok, SQL, AI, funkcionális programozásból egyaránt. Jól látható ezeknek a típusú oldalaknak a jelentősége és fontossága, már csak abból is, hogy milyen hatalmas összegeket fektetnek be az ilyen típusú oldalakra megvalósítására. A [crounchbase](#) adatai szerint a [HackerRank](#) alapító összege 42.9 millió dollár és több száz alkalmazott munkájának köszönhető.

Előnyei az én webalkalmazásommal szemben:

- A felhasználó több mint 30 programozási nyelv közül választhat.
- Elméleti tesztek alapján fel tudja mérni egy adott programozási nyelv ismeretét.
- A felhasználók versenyezni tudnak egymással bizonyos időközönként megrendezett versenyeken.
- Több száz feladat közül válogathatnak a felhasználók.

Hátrányai az én webalkalmazásommal szemben:

- A felhasználók nem kapnak megfelelő visszajelzést a kódjuk minőségéről, helyességéről.
- Túl sok beépített funkciója van az alkalmazásnak, így a felhasználó sokszor percekig kell keresse, hogy hol tudja a számára kedvező feladatokat megoldani.
- Számos prémium fiókot igénylő szolgáltatás van, amelyek nagyon lekorlátozzák a felhasználó mozgásterét.
- Csupán angol nyelven érhető el a felület.

3.3. Pbinfo vs. Kódolj!

A [Pbinfo](#) romániai programozás oktatásában igen gyakran használt webalkalmazás a 9-12. osztályok körében. Ez a weboldal hasonlít leginkább az általam készített webalkalmazáshoz célját tekintve. Az oldal összességében jól használható. Könnyen lehet felhasználói fiókot létrehozni és feladatokat oldani. Kicsit zavaró, hogy néhány nézet érezhetően túlzsúfolt. Az oldal teljes egészében román nyelvű, kezdve a regisztrációnál, az oldalon található fülekig, illetve a feladatok is román nyelven érhetőek el.

Előnyei az én webalkalmazásommal szemben:

- Jól el vannak különítve a 9., 10., 11. és 12. osztályosoknak való feladatok.
- A feladatokat C és C++ nyelven is meg lehet oldani és fel lehet tölteni.

Hátrányai az én webalkalmazásommal szemben:

- A felhasználók nem kapnak megfelelő visszajelzést a kódjuk hatékonyságáról.
- Túl zsúfolt néhány felület.
- Nem rendelkezik egy szemnek tetsző ügyféloldallal.
- Csak román nyelven érhető el a felület.
- A feladatokra automatikus visszajelzéseket kap, amelyek tesztesetek kiértékeléséből születnek.
- A rendszer különös hangsúlyt fektet a feladatok hatékony megoldására. Nem hatékony megoldás esetén a diák 0 pontot is kaphat, amely miatt elveszítheti a motivációját.

3.4. Eötvös Loránd Tudományegyetem Informatikai Kar MESTER - online programozási feladatbank

Kutatásom során csupán egy magyar nyelvű webalkalmazást találtam az enyémhez hasonló céllal, ez pedig az [Eötvös Loránd Tudományegyetem Informatikai Kar MESTER - online programozási feladatbankja](#). Az oldalt alaposan átvizsgálva számos alapvető hibát találtam. Első dolog amit észrevettem, hogy a böngésző méretét változtatva dinamikusán nem változik az oldal külalakja, ami napjainkban már alapvető funkciója kell legyen minden webalkalmazás számára. Az oldalra való regisztráció nincs megfelelően levédve, továbbá a hibaüzenetek nincsenek megfelelő módon kezelve és közölve a látogatóval. Az oldal válaszüzeje nagyon lassú, különösebb mérés nélkül is észrevehető, hogy egy-egy elemre kattintva több másodpercet kell várni, míg bármi történik. A belépéshez Google azonosításra van szükség mégis, annak ellenére, hogy megkapjuk a sikertelen bejelentkezésről szóló üzenetet tovább tudunk haladni az oldalon, és látni tudjuk, amit egy sikeresen bejelentkezett felhasználó lát. Továbbá, alapvető programozói ismerettel könnyen ki lehet

kerülni a bejelentkezést és regisztrációt. Összegzésképp elmondható, hogy a webalkalmazás nem áll készen egyszerre több felhasználó megfelelő minőségű kiszolgálására, és számos esztétikai és biztonsági probléma észlelhető.

4. fejezet

Technológiai áttekintés

Ebben a fejezetben bemutatásra kerül néhány technológia, melyeket megvizsgáltam, hogy a projektet megfelelő technológiák segítségével tudjam létrehozni. A technológiák kiválasztása során fontosnak tartottam, hogy olyanokat használjak, amelyek a mai 21. századi webfejlesztés során javasoltak. Ez a fejezet arra szólal, hogy egy általános képet kapjon az olvasó, az általam felhasznált technológiákról.

4.1. Adatbázis-kezelő rendszerek

Az adatbázis-kezelő rendszerek (*Database Management System, DBMS*) feladata az adatok kezelése, az adatbázis-motor lehetővé teszi az adatokhoz való hozzáférést, zárolást és módosítást. Az adatbázis séma meghatározza az adatbázis logikai felépítését. Ez a három alapelem elősegíti a párhuzamosságot, a biztonságot, az adatok integritását és az egységes adatkezelési eljárásokat. A DBMS által támogatott tipikus adatbázis-adminisztrációs feladatok közé tartozik a változáskezelés, a teljesítmény figyelése és hangolása, a biztonság, valamint a biztonsági mentés és a helyreállítás. A legtöbb adatbázis-kezelő rendszer felelős az automatizált visszaállításokért és újraindításokért, valamint az adatbázisokban végzett tevékenységek naplózásáért és az ezekhez hozzáférő alkalmazásokért is.

A adatbázis-kezelő rendszer központosított képet nyújt azokról az adatokról, amelyekhez több felhasználó, több helyről, ellenőrzött módon férhet hozzá. A DBMS korlátozhatja a végfelhasználó által látott adatokat, valamint azt, hogy a végfelhasználó miként tekintheti meg az adatokat, egyetlen adatbázis-séma sok nézetét biztosítva. A végfelhasználóknak és a szoftver programoknak nincs szükségük annak megértésére, hogy az adatok hol helyezkednek el fizikailag, vagy milyen típusú adathordozókon találhatók, mert a adatbázis-kezelő rendszer kezeli az összes kérést. [1]

4.1.1. SQLite

Az SQLite egy kisméretű, önálló, C forrású program könyvtárként (library) megvalósított relációs adatbázis-kezelő rendszer, illetve adatbázismotor. Az SQLite jelenleg a világ leghasználatosabb adatbázis kezelő rendszere. Be van építve a világ összes mobiltelefonjába és a legtöbb számítógépbe és számos más alkalmazásba is, amelyeket az

emberek minden nap használnak. Az SQLite fájlformátum stabil, platformokon átívelő és visszafelé kompatibilis [2].

Az SQLite adatbázis fájlokat általában tárolóként használják nagy tartalom továbbítására a rendszerek között, továbbá az adatok hosszú távú archiválására. Aktív használatban több mint 1 billió SQLite adatbázis van. Hatalmas előnye, hogy forráskódja nyilvános, és mindenki számára szabadon elérhető. Jelenlegi legfrissebb stabil kiadása a v.3.35.5 [3].

4.1.2. PostgreSQL

A PostgreSQL egy erőteljes, nyílt forráskódú objektum-relációs adatbázis-kezelő rendszer, amely az SQL nyelvet használja és kiterjeszti számos olyan funkcióval kombinálva, amelyek biztonságosan tárolják és méretezik a legbonyolultabb adat terheléseket. A PostgreSQL komoly hírnevet szerzett bizonyítottan jó architektúrájával, megbízhatóságával, adatintegritásával, robusztus szolgáltatáskészletével és bővíthetőségével. A PostgreSQL az összes főbb operációs rendszeren fut (Linux, macOS, Windows, BSD, Solaris). Továbbá számos olyan funkcióval rendelkezik, amelyek célja a fejlesztőknek az alkalmazások, a rendszergazdáknak az adatok integritásának védelme és a hibatűrő környezetek kialakításának elősegítése, valamint az adatok kezelése, függetlenül attól, hogy mekkora az adat mennyiség. Forráskódja nyilvános, és mindenki számára szabadon elérhető. [4] Jelenlegi legfrissebb stabil kiadása a v.13.3.

4.1.3. SQLite vs PostgreSQL

Az SQLite és a PostgreSQL a legelterjedtebb relációs adatbázis-kezelő rendszerek (*Relational Database Management Systems*) közé tartoznak. Mindkettő nyílt forráskódú és ingyenes, de vannak néhány jelentős különbségük, amelyeket figyelembe kell venni a rendszer kiválasztásakor. [5]

1.) Működése

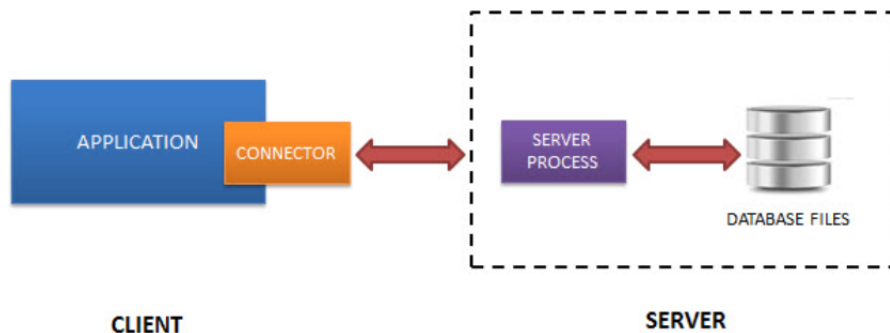
- Az SQLite egy „beágyazott” adatbázis, ami azt jelenti, hogy kiszolgáló nélküli(server-less) és az alkalmazáson belül futtatható (lásd 4.1 ábra¹).



4.1. ábra. SQLite

¹forrás: <https://tableplus.com/blog/2018/08/sqlite-vs-postgresql-which-database-to-use-and-why.html>

- A PostgreSQL viszont kliens-szerver modell alapján működik, amelyhez DB szerver szükséges a hálózat felépítéséhez és futtatásához (lásd 4.2 ábra²).



4.2. ábra. PostgreSQL

2.) Támogatott adattípusok

- Az SQLite csak öt típust támogat: BLOB, NULL, INTEGER, TEXT, REAL.
- A PostgreSQL viszont szinte a világ összes típusát támogatja

3.) Tárolás

- Az SQLite könyvtár kevesebb, mint 500 KB, míg a PostgreSQL sokkal nagyobb méretű.

4.) Hordozhatóság

- Az SQLite egyetlen normál lemezfájlban tárolja az adatbázist, amely bárhol elhelyezhető a könyvtárban. A használt fájlformátum szintén platformokon átívelő, így könnyen másolható és áthelyezhető.
- A PostgreSQL csak akkor hordozható, ha fájlba exportálja és feltölti egy másik szerverre. Ez néha gondot okozhat.

5.) Többszörös hozzáférés

- Az SQLite nem rendelkezik felhasználói felügyelettel, valamint több egyidejű hozzáférést kezel.
- A PostgreSQL nagyon jól kezeli a több felhasználót és jól elkülöníti a hozzáférési szinteket.

6.) Sebesség

- Az SQLite nagyon gyors, a minimális tervezésnek és az egyszerű műveleteknek köszönhetően.

²forrás: <https://tableplus.com/blog/2018/08/sqlite-vs-postgresql-which-database-to-use-and-why.html>

- Ha csak gyors olvasási műveletekre van szüksége, akkor a PostgreSQL túlterhelő lehet, és kevésbé teljesítőnek tűnhet. Ha összetett műveletekről van szó, a PostgreSQL nagyon jól teljesít.

7.) Biztonság és hitelesítés

- Az SQLite nem biztosít hitelesítési rendszert. Maga az adatbázisfájl bárki által frissíthető és olvasható.
- Számos biztonsági funkció, valamint bonyolult konfiguráció van beépítve a PostgreSQL-be az adatbázis védelme érdekében.

8.) Konfigurálás

- Az SQLite DB telepítése és futtatása a legtöbb kezdő felhasználó számára is nagyon egyszerű.
- Fejlett tulajdonságai miatt a PostgreSQL telepítése és konfigurálása sokkal összetettebb.

9.) Megbízhatóság

- Az SQLite nem rendelkezik felhasználói felügyelettel, valamint több egyidejű hozzáférést kezel.
- A PostgreSQL nagyon jól kezeli a több felhasználót és jól elkülöníti a hozzáférési szinteket.

4.2. Webes keretrendszerek

A webes keretrendszer olyan szoftvereszköz, amely lehetőséget nyújt webalkalmazások létrehozására és futtatására. Ennek eredményeként nem kell önállóan kódot írnia, és nem kell időt pazarolnia az esetleges téves számítások és hibák keresésére.

A keretrendszereknek két fő funkciója van: a kiszolgálói oldalon (backend) vagy a kliens oldalon (frontend) dolgozni, a típusoknak megfelelően.

Szerveroldali keretrendszerek:

Ezeknek a keretrendszereknek a szabályai és architektúrája lehetővé teszi egyszerű, különböző típusú oldalak, lekérdezések és űrlapok létrehozását. Ahhoz azonban, hogy jól fejlett felületű webalkalmazást építsen, szélesebb funkcionálitással kell rendelkeznie. Ezek a keretrendszerek a kimeneti adatokat is kialakíthatják, és webes támadások esetén javíthatják a biztonságot. Mindezek egyértelműen leegyszerűsíthetik a fejlesztési folyamatot. A kiszolgálóoldali keretek többnyire olyan konkrét, de fontos részleteken működnek, amelyek nélkül egy alkalmazás nem tud megfelelően működni. Itt van néhány back-end keretrendszer és azok a nyelvek, amelyeken dolgoznak:

- Django - Python
- Zend - PHP

- Express.js - Javascript
- Ruby on Rails - Rubin

Ügyféloldali keretrendszerek:

A kiszolgáló oldaltól eltérően az ügyféloldali keretrendszereknek semmi közük az üzleti logikához. Munkájuk a böngészőben zajlik. Így lehet fejleszteni és bevezetni az új felhasználói felületeket. Számos animációs funkció készíthető a frontend keretrendszerével. Az ügyféloldali keretrendszerek mindegyike különbözik funkciója és felhasználása szempontjából. [6] Összehasonlítás céljából itt vannak:

- Angular - Javascript
- React - Javascript
- Vue.js - Javascript

4.2.1. Django

A Django egy magas szintű, webes keretrendszer, amelyet Python írt a gyors fejlesztés és a pragmatikus tervezés érdekében. A keretrendszer egyszerűen modulok gyűjteménye, amelyek csoportosítva létrehozzák a webalkalmazásokat egy megbízható, már létező forrásból.

A Django-t 2003-ban hozták létre, amikor a Kansas állambeli Lawrence-ben egy újság cégnél dolgozó webfejlesztőknek jobb módszerre volt szükségük kódjuk rendezésére. Mivel a fejlesztőket sok újságíró vette körül, az egyértelmű dokumentáció a „Django” projekt néven ismertté vált projekt szerves részévé vált.

Django azóta hatalmas online nyílt forráskódú közösséggé nőtte ki magát, amelynek szinte mindenre megoldása van, a hitelesítéstől a tartalomkezelő rendszerekig.

A Django előnyei

- *Skálázható:* A Django képes teljesíteni egy nagy projekt forgalmi igényeit.
- *Csomagok:* Számos csomaggal rendelkezik, amelyekkel szabványos webalkalmazás-feladatokat hajthat végre, például hitelesítést, tartalomkezelést vagy lekérdezést.
- *Biztonságos:* a Django beépített biztonsági protokolljaival elkerülhetők a gyakori biztonsági kockázatok a webhelyek közötti hamisítások, a helyek közötti parancsfájlok és az SQL injekciók számára.
- *Dokumentáció:* A Django-nak rendkívül jó dokumentációja van. Könnyen olvasható, még a technikai háttérrel nem rendelkező emberek számára is.

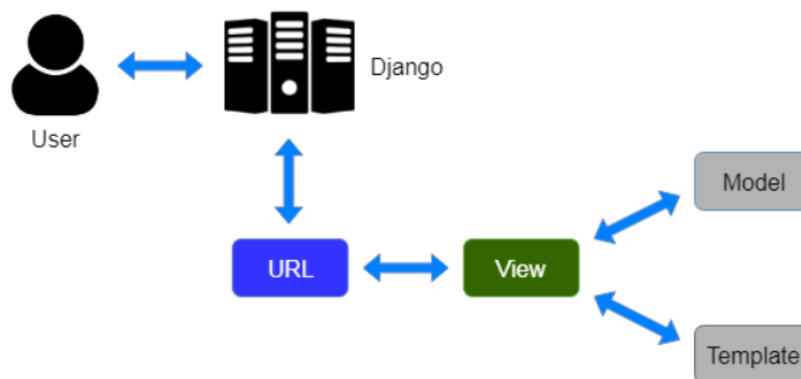
Django sokoldalúságával minden bizonnyal képes megfelelni szinte bármely projekt igényeinek. Az olyan vállalatok, mint a Spotify, a Pinterest, a National Geographic és a Dropbox a Django-t használják üzleti modelljükhöz. [7]

Django tervezési minta(Django Design Pattern)

A Django-nak egy egyedi szoftver tervezési mintája van, amely az *MVT* (Model View Template). Három fontos összetevőből áll: Model, View és Template.

- A *Model* segít az adatbázis kezelésében. Ez egy adatelérési réteg, amely az adatokat kezeli.
- A *Template* egy bemutató réteg, amely teljesen kezeli a felhasználói felület részét.
- A *View* az üzleti logika végrehajtására és egy modellel való interakcióra szolgál adatok hordozására és sablon megjelenítésére.

A Django követi az MVC mintáját, de fenntartja saját konvencióit. Tehát az irányítást maga a keret kezeli. Nincs külön vezérlő, és a teljes alkalmazás a Model, View és a Template alapján történik. Ezért hívják MVT alkalmazásnak. Lásd a következő grafikont, amely az MVT alapú vezérlési folyamatot mutatja 4.3 ábrán³. [8]



4.3. ábra. Django MVT

4.3. Amazon Web Services (AWS) - Cloud Computing Services

Az Amazon Web Services (AWS) egy biztonságos felhőszolgáltatási platform, amely számítási teljesítményt, adatbázis-tárolást, tartalomszolgáltatást és egyéb funkciókat kínál a vállalkozások méretarányos növekedéséhez [9].

Egyszerű szavakkal az AWS lehetővé teszi a következő dolgok elvégzését:

- Web- és alkalmazáskiszolgálók futtatása a felhőben dinamikus webhelyek fogadásához.

³forrás: <https://www.javatpoint.com/django-mvt>

- Az összes fájlját biztonságosan tárolja a felhőben, hogy bárhonnán hozzáférhessen hozzájuk.
- Olyan adatbázisok használata, mint a MySQL, a PostgreSQL, az Oracle vagy az SQL Server az információk tárolásához.
- Gyorsan szállítson statikus és dinamikus fájlokat szerte a világon a Content Delivery Network (CDN) segítségével.
- Tömeges e-mailt küldhet ügyfeleinek.

4.3.1. Amazon Relational Database Service (Amazon RDS)

Az Amazon Relational Database Service (Amazon RDS) megkönnyíti a relációs adatbázis felállítását, üzemeltetését és méretezését a felhőben. Költséghatékony és átméretezhető kapacitást biztosít, miközben automatizálja az időigényes adminisztrációs feladatokat, mint például a hardver kiépítése, az adatbázis beállítása, a javítás és a biztonsági mentések. Felszabadítja a felhasználót, hogy az alkalmazásokra összpontosítson, így gyors teljesítményt, magas rendelkezésre állást, biztonságot és kompatibilitást biztosíthat nekik.

Az Amazon RDS több adatbázis-példány típusokban érhető el - a memória, a teljesítmény vagy az I/O-ra optimalizálva, és hat megszokott adatbázis-motort kínál, amelyek közül választhat, köztük az Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database és SQL Server. Az AWS Database Migration Service segítségével könnyedén migrálhatja a meglévő adatbázisokat az Amazon RDS-be [10].

4.3.2. Amazon Simple Storage Service (Amazon S3)

Az Amazon Simple Storage Service (Amazon S3) egy objektum tároló szolgáltatás, amely az iparág vezető skálázhatóságát, az adatok elérhetőségét, biztonságát és teljesítményét kínálja. Ez azt jelenti, hogy minden méretben és iparágban az ügyfelek bármilyen mennyiségű adatot tárolhatnak és védhetnek különböző felhasználási esetekben, például adatlapok, webhelyek, mobil alkalmazások, biztonsági mentés és visszaállítás, archiválás, vállalati alkalmazások és nagy adatelemzés. Az Amazon S3 könnyen kezelhető kezelési funkciókat kínál, így rendszerezheti adatait és konfigurálhatja a finomhangolású hozzáférés-vezérlőket, hogy megfeleljenek a felhasználó üzleti, szervezeti és megfelelőségi követelményeinek. Az Amazon S3-at alkalmazások millióinak adatait tárolja a vállalatok számára a világ minden tájáról. [11]

A projektben ezt arra használom, hogy a felhasználók fényképeit tároljam, illetve jövőbeli fejlesztésekhez is elengedhetetlenül fontos, amikor a feladatok teszteseinek bemeneteit és kimeneteit vizsgáljuk.

4.4. Apache HTTP Server vs Heroku

Apache HTTP Server: Az Internet legnépszerűbb webservere 1996 óta. Továbbá egy erőteljes és rugalmas HTTP / 1.1-kompatibilis webserver. Eredetileg az NCSA HTTP

Server helyettesítésére tervezték, és az Internet legnépszerűbb webserverevé nőtte ki magát. [12]

Heroku: Egy felhőalkalmazás-platform - a webalkalmazások építésének és telepítésének új módja. A Heroku lehetővé teszi az alkalmazásfejlesztők számára, hogy idejük 100% -át az alkalmazáskódjukra fordítsák, nem a szerverek, a telepítés, a folyamatban lévő műveletek vagy a méretezés kezelésére.

Az Apache HTTP Server eszközként besorolható a *Web szerverek* kategóriába, míg Heroku a Platform mint szolgáltatás (*Platform as a Service*) csoportba tartozik.

Az oka, amiért a Herokut részesítettem előnybe az volt, hogy egyszerű a telepítése, ingyenes kisebb a projektek számára és rengeteg időt meg lehet vele spórolni.

5. fejezet

A rendszer specifikációja

5.1. Felhasználói követelmények

A felhasználók csupán egyetlen feltételnek kell hogy eleget tegyenek, ez pedig az, hogy rendelkezzenek egy olyan eszközzel, amellyel internet kapcsolatot tudnak létrehozni. A rendszert igénybe vevő felhasználói szerepköröket és az ezek számára elérhető funkcionálisokat az 5.1 ábra szemlélteti. Amint az ábrán is látható az alkalmazást négyféle felhasználó veheti igénybe: Látogató, Diák, Tanár és az Admin.

Látogató

A Látogató az a felhasználó, aki számára nem szükséges regisztráció és bejelentkezés az oldal bizonyos tartalmának megtekintéséhez. Ebből kifolyólag sokkal kevesebb funkcionális érhető el számára, összehasonlítva egy Diák vagy Tanár felhasználóval. Az oldalon folytatott tevékenységét nem menti el a rendszer semmilyen formában. Minden látogatása úgy van kezelve, mintha első látogatása lenne a felhasználónak az oldalon. A továbbiakban a Látogató számára elérhető funkcionálisokra térek ki. A könnyen követhetőség érdekében az 5.1 ábrán fentről lefelé haladva vesszük sorjában az elemeket.

Egy Látogató két különböző típusú felhasználói fiókot tud létrehozni a regisztráció során. Az egyik ilyen típusú felhasználó a Diák a másik pedig a Tanár. Értelemszerűen a felhasználó azt a típust választja a kettő közül, amelyiket létre szeretné hozni. A regisztráció könnyű és átlátható, négy kötelezően kitöltésre váró mezővel. A könnyen átláthatóság érdekében, számos, az adatok kitöltésére szolgáló tanács és követelmény van feltüntetve a kitöltéshez. Valamilyen mező nem megfelelő kitöltése esetén jól látható, sokatmondó hiba üzenetek olvashatóak a felhasználó számára.

A regisztráció után lehetősége van az eddig Vendégként kezelt felhasználónak a bejelentkezésre, ami által tovább léphet *Diák* vagy *Tanár* felhasználói státuszba. A bejelentkezéshez csupán a regisztráció során megadott adatokból a felhasználónevet és a jelszót kell megadnia.

Az alkalmazás főoldalán a felhasználó az Admin által megírt bejegyzéseket olvashatja. Itt a szerző nevét kiválasztva megtekintheti a által írt összes bejegyzést, időrendi sorrendben. Továbbá a bejegyzés címét kiválasztva, megtekintheti az adott bejegyzést és annak teljes tartalmát.

Ezek mellet még lehetősége van a Látogatónak átnézni a *Tanár* és *Admin* felhasználó által létrehozott feladatokat, és ezek között tud keresni több kritérium szerint is.

A feladatok nehézségi szint szerint három csoportra vannak osztva: *Könnyű*, *Közepes* és *Nehéz*. A feladatok közül ki lehet választani egyet, amelynek meg lehet tekinteni a teljes szövegét.

Diák

Amennyiben regisztrációnál a *Vendég* azt választotta, hogy *Diák* típusú felhasználót hoz létre, akkor a bejelentkezés után láthatóvá válik számára minden olyan funkció, amely ehhez a felhasználóhoz hozzá van rendelve. Itt megtekinthetők az általa megoldott feladatok, illetve ezeknek a *Tanár* által elkészített kiértékelője. A Diákok számára lehetőség nyílik osztálytermekhez csatlakozni, de csak abban az esetben, ha rendelkeznek egy, a Tanártól kapott kóddal. A Diáknak lehetősége van megtekinteni osztálytermeit, és ezekben a termekben levő társaik neveit, profilképeit és az általuk elért pontszámokat.

A feladatokkal ugyanazokat a dolgokat megteheti egy Diák, amelyre egy Látogató is képes, de plusz funkciókkal is rendelkezik. Képes a feladatok megoldásának megtekintésére, amelyet egy Tanár adhat meg. Továbbá képes a feladatok kidolgozására, és ezek osztályozásra való leadására. Egy feladatot a Diák akár többször is megoldhat és feltölthet. A Diák képes megtekinteni egy feladatra adott válaszainak a kiértékelését.

A Diákok számára elérhető egy eredmény tábla is, melyen nyomon tudják követni, hogy az összes felhasználó közül ki rendelkezik a legtöbb pontszámmal. Hogy minden felhasználó személyazonossága rejtve maradjon, a Diákok azonosítója jelenik meg a felhasználó nevük helyett. Ez motiválja a Diákokat, hogy minél jobban dolgozzanak, ezáltal egy versenyszerűbb környezetet teremtve egymás számára.

A Diák megtudja tekinteni saját profilját, ahol akár meg is változtathatja az adatait.

Amennyiben a felhasználó nem szeretne a továbbiakban bejelentkezve maradni, a kijelentkezés gombra kattintva, kiléphet a fiókjából, és visszatérhet az oldalra, mint Látogató.

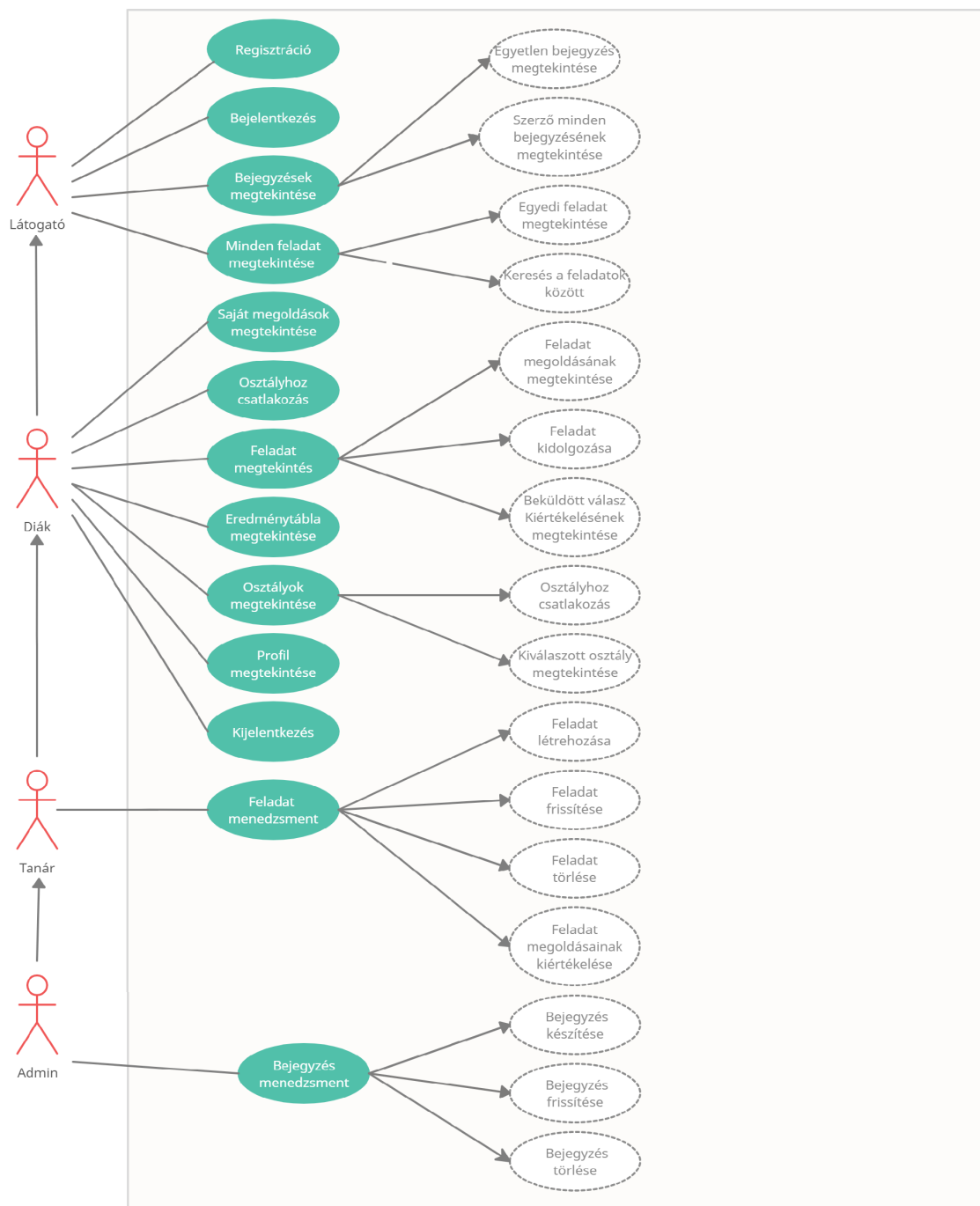
Tanár

A Tanár egyik fő feladata a feladatok menedzselése. A Tanár képes feladatokat létrehozni, szerkeszteni, vagy törölni. Mint említettem, a webalkalmazás egyik fő célja, hogy a Tanárok visszajelzései alapján a Diákok versenyszintű feladatokat oldhassanak meg. Ennek elengedhetetlen része, hogy a Tanár le tudja osztályozni a Diákok által megoldott és beküldött feladatokat, nem csak elért pontszám megadásával, hanem személyes üzenettel, javaslatokkal a Diák részére.

A Tanárnak módjában áll osztálytermeket létrehozni, amelyhez a Diákok egy kód segítségével csatlakozhatnak. A Tanár csak azoknak a Diákoknak látja a leadott megoldásait, s csak azokat értékelheti ki, amelyek vele egy közös osztályban vannak. Így fontos, hogy a Tanár egy osztályt létrehozzon, megossza a Diákokkal a kigenerált osztály belépőkódját, hogy a Diákok csatlakozzanak ide.

Admin

Az Admin rendelkezik minden olyan funkcióval, amely az összes többi típusú felhasználóhoz van rendelve. Továbbá ő felel a bejegyzések menedzseléséért. Létrehozhat új bejegyzéseket, a régieket frissítheti vagy akár ki is törölheti. Továbbá jogában áll a felhasználók, osztályok, feladatok szerkesztésére és törlésére.



5.1. ábra. A rendszer használati eset diagramja

5.2. Rendszerkövetelmények

5.2.1. Funkcionális követelmények

- Bizonyos funkcionalitások csupán bejelentkezést követően legyenek elérhetőek.
- Minden felhasználói típus csak a saját csoportja számára hozzáférhető dolgokat láthasson.
- Jól látható, egyértelmű információk és hiba üzenetek megjelenítése.

Látogató

- Egy bejegyzés írójának nevére kattintva, annak minden bejegyzésének megjelenítése.
- Bejegyzésre kattintva csupán egy bejegyzés megtekintése.
- Az összes elérhető feladat megtekintése táblázat formátumban. Az elemek közti keresés. Keresés esetén figyelembe veszi a feladatok címeit, szintjét, típusát, sorszámát. A táblázatban lévő elemek számának oldalankénti megjelenésének megváltoztatása: 10, 25, 50 és 100 elemre. A feladatok közötti lapozás.
- A feladatok táblázatában a "Megtekintések" gombra kattintva meg lehessen tekinteni a feladat pontos információit: feladat címe, szintje, típusa, maximálisan elérhető pontszáma, illetve a feladat teljes leírása.
- Regisztráció során két típus: Diák és Tanár között lehessen választani.
- Regisztráció során meg kell adni: Felhasználónév, E-mail cím, Jelszó, Jelszó hitelesítés. A "Regisztráció" gombra kattintva, ha a mezők megfelelő módon voltak kitöltve, akkor sikeres a regisztrálás, ellenkező esetben hiba kiírás jelenik meg.
- Az "Elfelejtette jelszavát?" gombra kattintva tudjuk megadni E-mail címünket, melyre egy jelszó visszaállítására vonatkozó üzenetet kapunk a rendszertől, amely tartalmaz egy linket, amelyre rákattintva lehetőségünk van új jelszó beállítására.

Diák

- Minden funkcionalitás, amely a Vendég típusú felhasználó számára elérhető, legyen a számára is használható.
- A feladatok megtekintése esetén lehetősége legyen a feladatok kidolgozására, illetve a feladat megoldásának megtekintésére.
- A feladatok kidolgozása során jelenjen meg a feladat minden információja és lehessen ezek alatt a feladatot kidolgozni és leadni.
- Egy feladatot többször is lehessen megoldani.

- A "Megoldások" menüpontot kiválasztva lehessen megtekinteni az eddig leadott munkánkat egy bizonyos feladatra, illetve a Tanár által feltöltött megoldást, amennyiben van ilyen.
- A felhasználónak lehetősége legyen megtekinteni a saját megoldásait a "Megoldásaim" menüpont alatt, ahol a Diák neve, összpontszáma és az összes eddig elkészített feladata látható. Minden "Kidolgozás"-nál lehetősége legyen a Diáknak újra megoldani a feladatot, és megtekinteni a tanár általi értékeléseket.
- Az "Osztályhoz csatlakozás" menüpontot kiválasztva lehetőség legyen a Diákoknak egy osztályhoz csatlakozni, amennyiben rendelkeznek az osztály kódjával.
- Lehetőségük legyen az osztályok megjelenítésére, illetve az osztályban lévő Diákok neveinek, profil képeinek, illetve eddig elért pontszámainak megtekintésére.
- Az "Eredménytábla" fület kiválasztva, lehetőség legyen a Diákok számára megtekinteni az összes regisztrált felhasználó pontszámát. A felhasználók nevei helyett az azonosítójuk jelenjen meg, biztonsági okok miatt. Ugyan úgy, mint a feladatok esetében lehessen keresni az azonosítók között.
- A felhasználók képesek legyenek megtekinteni saját profiljukat, illetve lehetőségük nyíljon változtatni az adataikon.
- A "Kijelentkezés"-re kattintva a felhasználó kiléphet a jelenlegi munkamenetből, s mint Vendég felhasználóként térjen vissza az oldalra.

Tanár

- Minden funkcionalitás, amely a Diák típusú felhasználó számára elérhető számára is legyen elérhető.
- A Tanár képes legyen új feladatokat létrehozni, a meglévő feladatokat frissíteni, továbbá lehetőségében álljon a feladatok törlésére is. A "Feladatok" menüpont alatt, akár csak a Diák felhasználó számára egy táblázat formájában jelenjenek meg a feladatok. Továbbá egy tanárnak legyen lehetősége a feladatok megoldásainak megadására. A "Válaszok" gombra kattintva a Tanár tudja megtekinteni egy feladatra beküldött minden olyan Diák választát, amely vele egy osztályban van.
- A Diák nevére kattintva a Tanár nézhessen meg a Diák által beküldött összes megoldást.
- A Tanárnak legyen lehetősége a feladatok kiértékelésére, amennyiben egy Diák választának megtekintése során rákattint a "Kiértékelés" gombra.
- A Tanár legyen képes Osztályokat létre hozni, amelyekbe később a Diákok csatlakozhatnak.
- Egy osztály megtekintése során a Tanárnak legyen lehetősége kiválasztani a Diákok profiljait és megtekinteni azokat.

Admin

- Minden funkcionalitás, amely a Tanár típusú felhasználó számára elérhető legyen számára is elérhető.
- Hozzáférése legyen az adatbázisban tárolt adatokhoz.
- Tudjon felhasználói csoportokat létrehozni és ezekhez bizonyos jogosultságokat hozzá csatolni. Ilyen felhasználói csoport például a Diák és a Tanár is.
- Az Admin oldalon lehetősége legyen: felhasználók létrehozására, meglévő felhasználók adatainak megváltoztatására, extra felhasználói jogosultságok megadására, illetve hozzárendelni más felhasználói csoportokat. Adatbiztonsági szempontok miatt a felhasználók jelszavai rejtve maradnak az Admin előtt is. Bejegyzések szerkesztésére is legyen képes. Az Admin oldalon hozzáférése legyen az összes bejegyzéshez és ezek megváltoztatásához, törléséhez, de akár itt is készíthet bejegyzéseket, akár csak a felhasználói felületen. Továbbá jogában legyen létrehozni, frissíteni és törölni bármely osztályt, feladatot, megoldást vagy akár felhasználót is.

5.2.2. Nem funkcionális követelmények

- A webalkalmazás használatához a felhasználónak szüksége van internet kapcsolatra és olyan eszközre, amely lehetővé teszi számára a biztonságos hozzáférhetőséget.
- A rendszer egyszerre maximum 1000 felhasználót tud kiszolgálni, mivel ez a felső határa ez ingyenes Heroku által futatott webalkalmazásnak. Erre a határokra vonatkozó információkról [itt](#) lehet olvani.
- A használt operációs rendszer lehet Linux, macOS, Windows, BSD, Solaris, Andorid és iOS is. Külön telefonos alkalmazás nincs, azonban bármely telefonos böngészőből megnyitható a webalkalmazás.
- A felhasználók adatait egy PostgreSQL (lásd [4.1.2](#)) adatbázis tárolja az Amazon RDS-en (lásd [4.3.1](#) fejezetben).
- A felhasználók által feltöltött profilképeket és minden más média elemet az Amazon által biztosított [4.3.2](#) fejezetben tárgyalt S3 adattároló tartalmazza. Mindez a jövőbeli fejlesztésekhez is fontos szerepet játszik, amennyiben automatikus tesztelést hajtánánk végrehajtani a Diák megoldásain. Ebben az esetben fontos, hogy egy biztonságos és könnyen elérhető helyen legyenek tárolva a tesztesetek.
- A felhasználók azonosítása a Django egy beépített [hitelesítés ellenőrző](#) rendszerével történik meg.

6. fejezet

Tervezés

6.1. A fejlesztéshez használt technológiák

A felhasznált technológiák kiválasztásának fontosabb szempontjait bemutattam a technológiai áttekintés során a 4. fejezetben. Összegezve az ott leírtakat arra a következtetésre jutottam, hogy a számomra leginkább megfelelő adatbázis a PostgreSQL. Fontos tényező volt az adatbázis kiválasztásánál annak figyelembe vétele, hogy ez az adatbázis volt Amazon RDS által leginkább támogatott rendszer, melynek használatát fontosnak éreztem, ahhoz, hogy egy ténylegesen használható webalkalmazást hozzak létre. A front-end fejlesztéséhez nem használtam semmilyen összetettebb keretrendszert, egyedül a Bootstrap CSS Frameworkot. A Back-end fejlesztés során választásom a Django-ra esett, mivel számos kimagasló tulajdonsággal rendelkezik. Hogy a webalkalmazás mindenki számára könnyen hozzáférhető legyen, és könnyen kezelhető Herokut egy felhő alkalmazásplatformot használtam (lásd 4.4 fejezet).

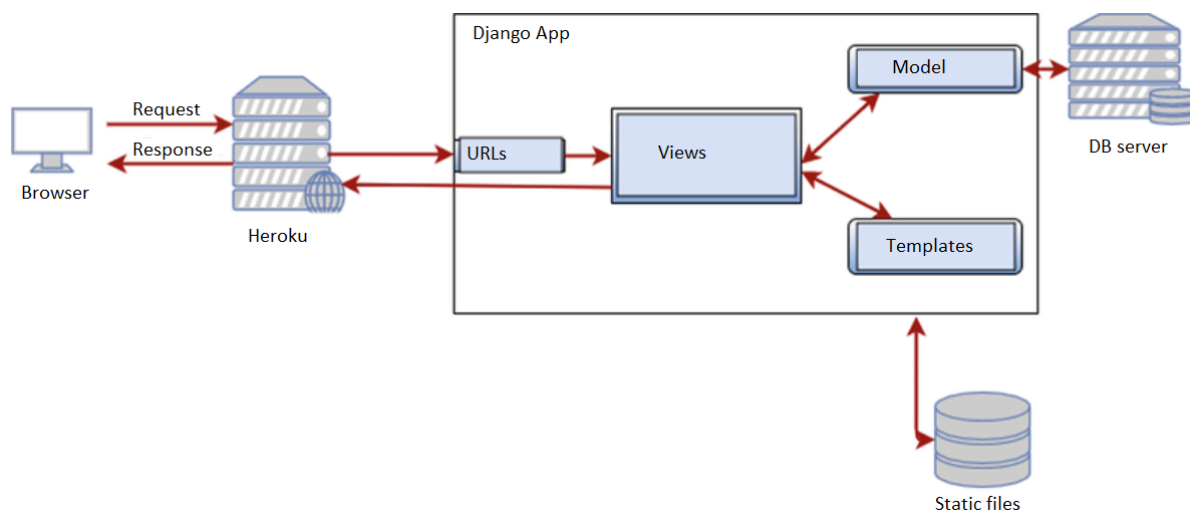
6.2. A rendszer architektúra

A 6.1 ábrán látható a rendszer architektúrája. Először a felhasználó valamilyen internetes kereső segítségével kéréseket intéz a szerver számára. Ezeket a kéréseket először a Heroku fogadja, amely a hozzáférést kezeli a Django projekthez. Az internet minden oldalának meg kell adnia a saját URL-jét. Így az alkalmazás tudja, mit kell megmutatnia egy felhasználó számára, aki megnyitja az URL-t. A Django-ban az URLconf (URL konfiguráció) nevet használjuk. Az URLconf olyan minták összessége, amelyekkel Django megpróbálja egyeztetni a kért URL-t, hogy megtalálja a helyes nézetet (view) [13]. Majd a következő lépésben a Django MVT (lásd 4.2.1 fejezet) tervezési minta segítségével feldolgozásra és kiszolgálásra kerülnek a beérkező kérések.

6.2.1. Front-end

A front-end web fejlesztés, más néven kliensoldali web fejlesztés az a gyakorlat, hogy HTML, CSS és JavaScript-et állítanak elő egy webhelyhez vagy webalkalmazáshoz, hogy a felhasználó közvetlenül lássa és kölcsönhatásba léphessen velük.

A jó Front-end célja annak biztosítása, hogy amikor a felhasználók megnyitják az oldalt, könnyen olvasható és releváns formátumban látják az információkat. Ezt tovább



6.1. ábra. A rendszer architektúrája

bonyolítja az a tény, hogy a felhasználók most már sokféle eszközt használnak változó képernyő mérettel és felbontással, így arra kényszerítve a tervezőt, hogy ezeket a szempontokat vegye figyelembe a webhely megtervezésekor. Biztosítaniuk kell, hogy webhelyük megfelelő böngészőben (cross-browser), különböző operációs rendszerekben (cross-platform) és különböző eszközökben (cross-device) megfelelően jelenjen meg, ami pontos munkát igényel a fejlesztő oldalán [14].

A front-end elkészítése során nagy segítségemre volt a Bootstrap. A Bootstrap az egyik legnépszerűbb HTML, CSS és JavaScript keretrendszer egy érzékeny és mobilbarát weboldal fejlesztéséhez. Teljesen ingyenesen letölthető és használható. Ez egy olyan front-end keretrendszer, amelyet könnyebb és gyorsabb webfejlesztéshez használnak. HTML és CSS alapú tervezősablonokat tartalmaz tipográfiához, űrlapokhoz, gombokhoz, táblázatokhoz, navigációhoz, kép megjelenítéséhez és még másokhoz [15]. A Bootstrap használatához a `base.html`-ben, ami az összes template alapjául szolgál. Be kellett helyezni a Bootstrap által kért linkeket, a használatához. Részletes információk a telepítésről és a projektekhez való kapcsolódásról [itt](#) érhetőek el.

Továbbá nagy segítségemre volt a Django sablonrendszer, amely olyan címkéket biztosít, amelyek hasonlóan működnek, mint egyes programozási szerkezetek, if címke logikai tesztekhez, stb. Alapértelmezés szerint csak néhány címkét, szűrőt és szintaxist támogat, de ezekhez hozzá adhatjuk saját kiterjesztéseinket a sablonjainkat is [16].

Base.html

Fontosnak tartottam kiemelni a `base.html` templatet, amely a projekt fő sablonja. Erre épül az összes többi sablon, ami a Django sablon rendszernek köszönhető. Használata nagyon egyszerű, gyors és könnyen kezelhető. Ide importáljuk be az összes elemet, amit meg szeretnénk hívni a projekt során, itt adtam meg a weboldal általános kinézetét, a navigációs mezőben megjelenő elemeket, a megköötéseket, hogy mely típusú felhasználó számára mi látható. Az alábbi kódrészletet a HTML-be a következő képpen írtam be. A 6.1 képen látható kódrészlet megvizsgálja, hogy a felhasználónk be van-e jelentkezve, amennyiben be van jelentkezve a Tanár felhasználói csoport tagja-e, s amennyiben igen

az utána megadott elemeket láthatja. A `{% url 'practices' %}` kódrészlet az URL címre való csatlakozást valósítja meg.

```
{% if user.is_authenticated %}
  {% if request.user|has_group:"teacher" %}
    <li class="nav-item dropdown">
      <a class="nav-link dropdown-toggle" href="#"
        id="navbarDropdownMenuLink" data-toggle="dropdown"
        aria-haspopup="true" aria-expanded="false">Feladatok</a>

      <div class="dropdown-menu"
        aria-labelledby="navbarDropdownMenuLink">
        <a class="dropdown-item" href="{% url 'practices'
          %}">Feladatok</a>
        <a class="dropdown-item" href="{% url 'create_exercise' %}">Új
          feladat</a>
      </div>
    </li>
  {% endif %}
{% endif %}
```

6.1. kódrészlet. base.html

Egy sablonban könnyen meg tudjuk hívni a base.html-t . Annyi a dolgunk, hogy az új HTML fájlba kiterjesztjük (lásd 6.2 kódrészlet). Viszont mielőtt ezt megtennénk be kell írunk a fő sablonba a `{% block content %}{% endblock %}` parancsot oda, ahova meg szeretnénk jeleníteni az új tartalmat.

```
{% extends "blog/base.html" %}
{% block content %}
...
{% endblock content %}
```

6.2. kódrészlet. base.html új sablonban való kiterjesztése

Továbbá a sablonokon lévő táblázatok megjelenítéséhez, szűréséhez és lapozásához Javascriptet is használnom kellett, hogy az oldalt dinamikussá alakítsam (lásd 6.3. kódrészlet).

```
$(document).ready( function () {
  $('#myTable').DataTable( {
    paging: true,
    pagingType: "full_numbers",
    searching: true,
    autoWidth: true,
    oLanguage: {
      oPaginate: {
        sNext: 'Következő',
        sPrevious: 'Előző',
        sLast: 'Utolsó',
        sFirst: 'Első',
```

```

    },
    sZeroRecords: "Nem található ilyen feladat",
    sLengthMenu: "Elemek száma oldalanként: _MENU_ ",
    sInfo: "Oldal: _PAGE_/_PAGES_",
    sInfoEmpty: "Nem található ilyen feladat a mezok között! ",
    sInfoFiltered: "(Keresés mind a _MAX_ bejegyzés között)",
    sSearch: 'Keresés',
  },
  } );
} );

```

6.3. kódrészlet. Táblázatok megjelenítése

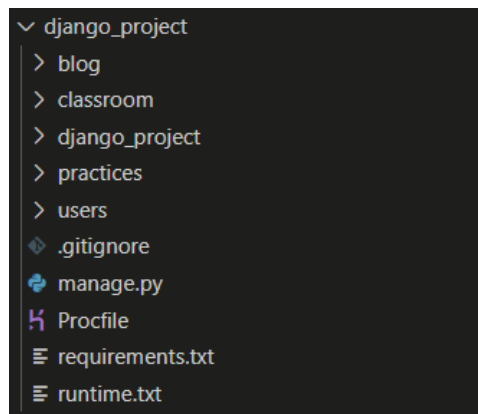
6.2.2. Back-end

A projektem back-endje Pythonban lett implementálva, keretrendszernek pedig a Django-t használtam, mivel számos olyan könyvtárral rendelkezik, amely sokat segíthet a projektem megvalósítása során, további részleteket a Django-ról a 4.2.1 fejezetben olvashatnak. A Django projektem a 3.0 verziószámú. A Django projektet a következő képpen hoztam létre, a megadott parancsokat a parancssorba kell írni:

- Első lépésként az a dolgunk, hogy telepítsük a Python-t. Én a Python 3.7.4 verziószámút telepítettem és használtam a projekt során.
- A következő lépésben létre kell hoznunk egy virtuális környezetet(virtualenv) egy általunk kiválasztott könyvtárban. A *extvirtualenv* egy olyan eszköz, amely olyan Python környezetek létrehozására szolgál, amelyek tartalmazzák a saját Python másolatot, a pip-ot és a saját helyüket a PyPI-ből telepített könyvtárak megőrzéséhez. Úgy tervezték, hogy lehetővé tegye, hogy ugyanazon a gépen egyszerre több különböző függőségű projekten dolgozzon [17].
- A következő lépésként telepíteni kell a Django-t, amelyet megtehetünk a *pip install Django==3.0.0* paranccsal.
- A következőkben létre kell hoznunk a projektünket, amelyet megtehetünk a következő paranccsal: *django-admin startproject django_project*. Amennyiben eddig mindent jól csináltunk, létre kellett jöjjön szerver, ami futtatható a *python manage.py runserver* paranccsal.
- Ezek után, amennyiben dolgozni szeretnénk a projekttel, szükséges különböző applikációkat létrehozni és csatlakoztatni a projekthez, amit majd a szerver futtat és megjelenít számunkra. Ezt a *python manage.py startapp app_name* paranccsal tehetjük meg, majd a *settings.py*-ban regisztrálnunk kell a használt alkalmazásokat. Minden alkalmazás, amelyet Django-ban ír, egy Python csomagból áll, amely egy bizonyos konvenciót követ. A Django olyan segédprogrammal érkezik, amely automatikusan létrehozza az alkalmazás alapkönyvtárának felépítését, így könyvtárak létrehozása helyett a kódolásra koncentrálhat [18] .

1.) django_project gyökérkönyvtár (lásd 6.2 ábra)

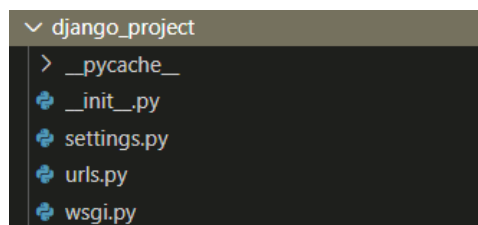
- A külső *django_project* a gyökérkönyvtár, amely a projekt tárolója. Fontos, hogy ezen a könyvtáron kívül legyen az általunk létrehozott virtuális környezet.
- *blog*, *classroom*, *practices*, *users*: Ezek az általam írt és használt alkalmazások.
- *manage.py*: Parancssori segédprogram, amely lehetővé teszi a Django projekttel való különféle kapcsolattartást.
- A belső *django_project* a projekt tényleges csomagja. A neve a Python csomag neve, amelyet a benne lévő elemek importálásához kell használni (pl. *django_project.urls*).
- *runtime.txt*, *requirements.txt*, *Procfile*, *.gitignore* ezek a fájlok szükségesek a Heroku számára, hogy futtathatja a szerverünket.



6.2. ábra. A projekt gyökérkönyvtárának tartalma

2.) *django_project* belső könyvtár (lásd 6.3 ábra)

- *__init__.py*: Egy üres fájl, amely megmondja a Pythonnak, hogy ezt a könyvtárat Python csomagnak kell tekinteni.
- *settings.py*: A Django projekt beállításai, konfigurációja.
- *urls.py*: A Django projekt URL-deklarációi; a Django által üzemeltetett webhely „tartalomjegyzéke”.
- *wsgi.py*: WSGI-kompatibilis webserverek belépési pontja a projekt kiszolgálására.

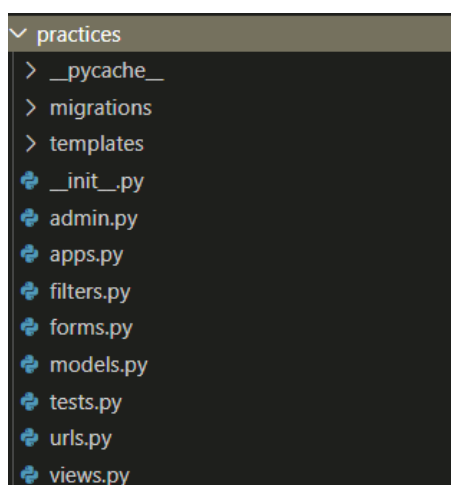


6.3. ábra. A projekt belső könyvtárának tartalma

3.) Practices alkalmazás könyvtára (lásd 6.4 ábra)

Mivel a *blog*, *classroom*, *practices*, *users* alkalmazásoknak hasonlóak itt csak a *Practices* appon belül magyarázom a fájlrendszert.

- *migrations*: A modelleken végrehajtott változtatásokat tartalmazza, hogy vissza követhetők legyenek az elvégzett műveletek.
- *templates*: Ez tartalmazza az általam létrehozott sablonokat, amelyeket meghívhat a *views.py*.
- *admin.py*: Ide kell regisztrálnunk a modelljeinket, ahhoz hogy elérhetőek legyenek számunkra az Admin oldalon.
- *apps.py*: Ez tartalmazza a projekt konfigurációs nevét, amelyet meg kell adnunk a *settings.py*-ban, hogy a szerver számára is elérhető legyen az alkalmazás.
- *filters.py*: Amennyiben vannak olyan adataink, amelyeken szűréseket szeretnénk végezni megjelenítés előtt vagy akár megjelenítés alatt itt tudunk szűrőket beállítani. Ez nem alapértelmezetten jön létre az alkalmazás létrehozatalánál, hanem nekünk kell létrehoznunk.
- *forms.py*: Tartalmazza az űrlapokat, azoknak megjelenítését, illetve itt tudjuk őket személyre szabni, specifikusabbá alakítani.
- *models.py*: Ez tartalmazza az adatbázis létrehozásához szükséges modelleket, kapcsolatok beállításokat.
- *tests.py*: Ide tudunk teszteket írni az alkalmazás számára.
- *urls.py*: A jobb átláthatóság érdekében meg tudjuk azt csinálni, hogy helyileg minden alkalmazásnak megadjuk a hozzá tartozó URL címét.
- *views.py*: Szerepe a kérések felvétele és ezekre való válasz adása. Tartalma lehet egy HTML fájl, egy valahova való átirányítás, egy rendszerüzenet, akár egy JSON vagy szinte bármi.



6.4. ábra. Practices alkalmazás könyvtára

6.2.3. Adatbázis

A 4. fejezetben vett összehasonlítás illetve a Heroku és AWS-el való összeférhetőség tekintetében arra a következtetésre jutottam, hogy számomra a legmegfelelőbb a PostgreSQL adatbázis kezelő rendszer használata lesz. Ennek használatához sorban a következő lépéseket követtem:

- Először létre kell hoznunk egy Amazon AWS felhasználói fiókot. Itt létre kell hoznunk egy RDS: PostgreSQL típusú adatbázist. Ezek után számos dolgot kell beállítani. Ezek közül a legfontosabbak az adatbázis hozzáférhetőség nyilvánosra állítása, a Port megadása, és megadjunk egy úgynevezett mester felhasználónevet és jelszavat, amelyek segítségével össze tudjuk kapcsolni az adatbázist a projektünkkel. Továbbá számos más lehetőség áll rendelkezésünkre, amiről [itt](#) olvashatunk bővebben.
- A beállítások mentése után az AWS létrehozza az immár használható adatbázist, amihez annyi a dolgunk, hogy hozzá csatlakozzunk.
- A következő lépésben [PostgreSQL](#) hivatalos oldaláról le kell tölteni és telepíteni az adatbázis elérhető legfrissebb változatát. Én a 13-as verzióval dolgoztam.
- A továbbiakban szükséges letölteni a [pgAdmin-4](#)-et, ami lehetővé teszi számunkra az adatbázissal való munkát. Ebből is a lehető legfrissebb elérhető rendszert választottam ami a v.5.3. A letöltés és telepítés után szükséges egy felhasználói fiókot létrehozni a pgAdmin számára.
- A következő lépésben egy szerveret kell létrehozni a pgAdmin-4 számára és meg kell adnunk pár alap beállítást: szerver nevét, a szerver futtató host-ot, ami a mi esetünkben: `database-1.c*****x.us-east-2.rds.amazonaws.com`. Ezt a címet az adatbázis létrehozásakor generálja nekünk az RDS. Továbbá meg kell adnunk ugyanazt a Portot, amelyet megadtunk az RDS számára. Meg kell adnunk az AWS-nél megadott felhasználó nevünket és jelszavunkat. Amennyiben itt minden adat rendben van hozzá fogunk kapcsolódni a pgAdmin-hoz.
- A következő lépésben a Django projektünkkel kell hozzá csatlakozni az adatbázishoz, és készen is vagyunk. Előtte viszont le kell telepíteni a Psycopg2 könyvtár csomagot (lásd [6.2.4](#) fejezet).

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'allamvizsga_1',
        'USER': '*****',
        'PASSWORD': '*****',
        'HOST': 'database-1.*****.us-east-2.rds.amazonaws.com',
        'PORT': '5432'
    }
}
```

Entity Relationship Diagram (ERD)

Az adatbázis sémája a 6.5 ábrán látható. A diagramot a pgAdmin segítségével generáltam ki, majd finomításokat végeztem rajta, a jobb láthatóság érdekében. Összességében 18 táblát használtam, amiből 8 táblát én hoztam létre és 10-táblát a Django által használt beépített modulokkal generáltattam ki és használtam.

- *django_session*: lehetővé teszi tetszőleges adatok tárolását és lekérését webhelyenkénti látogatóként [19].
- Az *auth_group* és *auth_group_permissions* táblákban tárolom a felhasználói csoportokat: Diák, Tanár, Admin és azok hozzáférési jogait.
- Az *auth_user* táblába regisztrálom be az új felhasználókat, amelyek össze vannak kapcsolva ID-jük alapján a *users_profile* táblával, amely egy felhasználó regisztrációja esetén automatikusan kigenerálok.
- *users_profile*: Ebben a táblában mentem el a felhasználó adatait, amelyeket akár a felhasználó maga is megváltoztathat a profil oldalán. Eltárolom az eddig elért pontszámait, amely regisztráció során alapvetően nullára van beállítva. Ez annak alapján fog nőni, hogy hány pontot szerez feladatok megoldása során. Továbbá elmentődik a felhasználó neve, email címe, profiljának létrehozási dátuma, illetve egy default.jpg profilképe. Ezt később lehetősége lesz megváltoztatni a profilján.
- *auth_permissions*: Ebben a táblában van eltárolva, hogy milyen felhasználói jogosultságok adhatóak a felhasználók számára.
- *auth_user_groups*: Itt tárolom, hogy melyik felhasználó, mely csoporthoz tartozik.
- *auth_user_user_permissions*: Itt tárolom, hogy egy felhasználónak milyen jogosultságai vannak.
- *django_admin_log*: Az Admin műveleteit, bejelentkezését tartalmazza.
- *django_content_type*: A Django tartalmaz egy *contenttypes* alkalmazást, amely képes nyomon követni a Django által működtetett projektbe telepített összes modellt, magas szintű, általános felületet biztosítva a modelljeivel való együttműködéshez. Ezzel én nem kellett foglalkozzak, a rendszer automatikusan létrehozta kérés nélkül. [20].
- *blog_post*: Ebben a táblában tárolom az Adminok által készített bejegyzéseket. Egy Admin több bejegyzést is írhat, így egy a sokhoz típusú kapcsolatban áll az *auth_user* táblával.

- *classroom_classroom_user_profile*: Ebben a táblában tárolom el, hogy mely diákok tartoznak egy osztályba. Egy osztályba több felhasználó is csatlakozhat, így itt is egy a többhöz kapcsolat áll fenn.
- *practices_exercise*: Itt tárolom el a feladatok információt: feladat címe, szintje, leírását, típusát, pontszámát, létrehozási dátumát, és a végső választ rá.
- *practices_solving_exercise*: Ebben a táblában tárolom, hogy melyik feladatra milyen megoldások jöttek be. Egy feladatra több megoldás is bejöhethet be egy személytől.
- *practices_solving*: a megoldások információt tárol itt: elért pontszámot, hozzászólást, magát a választ, és a létrehozás dátumát is. Fontos tudnunk, hogy ki küldte be a választ, így ez a tábla is össze van csatolva a *users_profile* táblával.
- *practices_solving_user_profile*: ezzel a táblában tárolom, hogy melyik felhasználó melyik megoldást küldte be.

Modellek létrehozása

Minden modell egy Python osztály, amely a `django.db.models.Model` hívja meg. Általában minden modell egyetlen adatbázis táblát hoz létre és minden attribútuma egy adatbázis mezőt képvisel. [21]

Exercise modell:

Ennek a modellnek a segítségével hoztam létre *classroom_exercise* táblát és adtam meg a mezők számára megszorításokat. A kódrészlet megtekinthető 6.5 ábrán.

- *title*: Itt tárolom a feladat címét. `null=True` beállítás arra szolgál, ha vannak olyan feladatok az adatbázisban, amelyek számára nem volt *title* megadva, akkor ezek ne okozzanak hibákat.
- *level*: Ugyan úgy, mint a *title* ez egy `CharField` kiegészítve egy `choices=LEVEL` megkötéssel, ami annyit tesz, hogy megadjuk, hogy kizárólag, csak milyen mezők adhatóak meg. Ezeket a mezőket a `LEVEL`-ben adjuk meg.
- *description*: Tartalmazza a feladat teljes leírását. `RichTextField`-nek hála lehet személyre szabni a leírást és lehet korlátlan számú karaktert beírni.
- *exercise_type*: Itt adjuk meg a feladatok típusait, felépítése hasonló a *level* mezőjéhez.
- *score*: A feladatra elérhető maximális pontszám, amely egész típusú szám kell legyen.
- *date_created*: A feladat létrehozásának dátuma, mely automatikusan hozzá adódik a táblához egy feladat létrehozatala során.
- *final_answer*: Ebben a mezőben adhatja meg egy Tanár vagy Admin típusú felhasználó a feladat megoldását.

```

class Exercise(models.Model):
    title = models.CharField(max_length=200, null = True)
    LEVEL =(
        ('Konnyu','Konnyu'),
        ('Kozepes','Kozepes'),
        ('Nehez','Nehez')
    )
    level = models.CharField(max_length=200,null = True, choices=LEVEL)
    description = RichTextField(blank=True, null=True)
    EXERCISE_TYPE =(
        ('Verem','Verem'),
        ('Sor','Sor'),
        ('Csatolt Lista','Csatolt Lista'),
        ('Moho algoritmus','Moho algoritmus'),
        ('Rekurzio','Rekurzio'),
        ('Dinamikus programozas','Dinamikus programozas'),
        ('Visszalepeses kereses','Visszalepeses kereses'),
        ('Grafok','Grafok'),
        ('Binaris kereses','Binaris kereses')
    )
    exercise_type = models.CharField(max_length=200,null =
        True,choices=EXERCISE_TYPE)
    score = models.IntegerField(default=10)
    date_created = models.DateTimeField(default=timezone.now)
    final_answer = RichTextField(blank=True, null=True)

    def __str__(self):
        return self.title

```

6.5. kódrészlet. Exercise modell

Profile modell:

A *create_profile* függvény arra szolgál, hogy amikor egy felhasználó létrehozva a felhasználói fiókját, akkor azzal egyszerre létrehozza profilját is. Ahhoz, hogy véglegesítsük és feltöltsük az adatokat az adatbázisba szükségünk van a változtatásaink mentésére, erre szolgál a *save_profile* függvény (lásd 6.7 ábra).

A Profile modell, User mezője egy-egy (OneToOneField) [22] kapcsolatban áll a *auth_user* táblával, melyet User néven. Fogalmilag ez hasonló az ForeignKey-hez, amelynek *unique=True*, de a reláció másik oldala közvetlenül egyetlen objektumot ad vissza. Az eddig tárgyaltakhoz képest csak az image mező az újdonság itt. A *default='default.jpg'*-ben megadjuk, hogy minden létrehozott fiók számára ez legyen az alap profilkép. Az *upload_to='profile_pics'*-ban pedig megmondjuk, hogy hova töltsse fel a az újonnan hozzáadott fényképeket.

Az *__str__* arra szolgál, hogy az Admin oldalon milyen név alatt, hogyan láthassuk a rekordokat.

```

@receiver(post_save, sender=User)
def create_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(
            user=instance,
            name=instance.username,
            email=instance.email)

@receiver(post_save, sender=User)
def save_profile(sender, instance, **kwargs):
    instance.profile.save()

class Profile(models.Model):
    user = models.OneToOneField(User, null=True, blank=True,
        on_delete=models.CASCADE)
    name = models.CharField("Felhasznalonev",max_length=200, null=True)
    email = models.EmailField("E-mail cim",max_length=200, null=True)
    score = models.IntegerField("Pont",null=True, default=0)
    date_created = models.DateTimeField(auto_now_add=True, null=True)
    image = models.ImageField(null=True,default='default.jpg',
        upload_to='profile_pics')

    def __str__(self):
        return f'{self.user.username} Profile'

```

6.6. kódrészlet. Profile modell

Classroom modell:

Ezen modell esetén a creator ForeignKey-t tartalmaz, amelyen keresztül össze kapcsolódik a classroom tábla a Userok táblával. Az *on_delete=models.CASCADE* szerepe, hogy az objektum törlésekor törölje azokat az objektumokat is, amelyek hivatkoznak rá, jelen esetben ha töröljük a felhasználót az általa létrehozott osztályok is törlésre kerülnek. A code-ban tároljuk az osztályba belépéshez szükséges kódot. Ezt a view-ben egy véletlenszerű függvénnnyel generáljuk ki. A *user_profile* tartalmazza az összes olyan felhasználót, aki csatlakozott az adott osztályhoz.

```

class classroom(models.Model):
    classname = models.CharField(max_length=50, null=True)
    creator = models.ForeignKey(
        User, on_delete=models.CASCADE,
        related_name='creator', null=True)
    code = models.CharField(max_length=6, null=False, default="passwd")
    user_profile = models.ManyToManyField(Profile)
    image = models.ImageField(null=True,default='default_classroom.jpg',
        upload_to='classroom_pics')
    def __str__(self):
        return self.classname

```

6.7. kódrészlet. Classroom modell

Post modell:

Ez tartalmazza az Adminok által létrehozott bejegyzéseket. Jövőbeli fejlesztésként tervben van ennek bővítése az által, hogy a Tanárok is tudjanak írni bejegyzéseket külön classroomokba.

```
class Post(models.Model):
    title = models.CharField("Cím", max_length=100)
    content = RichTextField("Tartalom", blank=True, null=True)
    date_posted = models.DateTimeField("Datum", default=timezone.now)
    author = models.ForeignKey(User, on_delete=models.CASCADE)

    def __str__(self):
        return self.title
```

6.8. kódrészlet. Post modell

6.2.4. Felhasznált csomagok és könyvtárak

ASGI (Asynchronous Server Gateway Interface)

Az ASGI (Asynchronous Server Gateway Interface, v.3.3.4) a WSGI szellemi utódja, amelynek célja az aszinkronra képes Python webserverek, keretek és alkalmazások közötti szabványos interfész biztosítása.

Ahol a WSGI szabványt biztosított a szinkron Python-alkalmazásokhoz, az ASGI mind az aszinkron, mind a szinkron alkalmazásokhoz biztosít egy WSGI visszafelé kompatibilis megvalósítást, valamint több szervert és alkalmazáskeretet [23].

Boto3

Boto3 (v.1.17.94) segítségével olyan AWS-szolgáltatásokat hozhat létre, konfigurálhat és kezelhet, mint az Amazon Elastic Compute Cloud (Amazon EC2) és az Amazon Simple Storage Service (Amazon S3). Az SDK objektum-orientált API-t, valamint alacsony szintű hozzáférést biztosít az AWS-szolgáltatásokhoz [24].

Certifi

A Certifi (v.2018.10.15) biztosítja a Mozilla gondosan gondozott gyökértanúsítványgyűjteményét az SSL-tanúsítványok megbízhatóságának és a TLS-állomások személyazonosságának ellenőrzése érdekében [25].

Crispy forms

A Django DRY (v.1.8.1) űrlapok legjobb módja. Hozzon létre programozott, újra felhasználható elrendezéseket az összetevőkből, teljes ellenőrzése alatt a renderelt HTML-kódot anélkül, hogy HTML-t írna sablonokba. Mindez anélkül, hogy megsértené a Django

szokásos módját, így minden más űrlapalkalmazással jól játszik. Az alkalmazás elsősorban [26]:

- A | Crispy elnevezésű szűrő elegáns div alapú formákat jelenít meg. Gondoljon rá, mint a beépített módszerekre: `as_table`, `as_ul` és `as_p`. Nem hangolhatja a kimenetet, de könnyű elkezdni használni.
- A (z) % crispy% nevű címke, amely a konfiguráció és az adott elrendezés beállításai alapján jelenít meg űrlapot, ami rengeteg időt megtakarít.

Django-filter

A Django-filter (v.2.4.0) egy általános, újrafelhasználható alkalmazás, amely enyhíti a hétköznapibb nézetkódok írását. Pontosabban lehetővé teszi a felhasználók számára, hogy a modell mezei alapján kiszűrjék a lekérdezéseket, megjelenítve az űrlapot, hogy ezt megtehessek [27].

Font Awesome

Ezt a csomagot úgy tervezték, hogy könnyen használható legyen egy django projektben, hogy kiaknázhassa, hogyan kezeli a Django a statikus fájlokat, de telepíthető bármilyen Python projekthez. A Font Awesome (v.5.15.3) ingyen letöltésre kerül a webhely-csomagjaira, így ha a Django-tól eltérő keretrendszert használ, akkor módosítania kell ezeket az utasításokat, hogy működhessen azzal, amit használ [28].

Psycopg2 (Python-PostgreSQL Database Adapter)

A Psycopg (v.2.8.6) a legnépszerűbb PostgreSQL adatbázis-illesztő a Python programozási nyelv számára. Főbb jellemzői a Python DB API 2.0 specifikáció teljes megvalósítása és a szálbiztonság (több szál ugyanazt a kapcsolatot használhatja). Erősen többszálú alkalmazásokhoz tervezték, amelyek sok kurzort hoznak létre és pusztítanak el, és nagyszámú egyidejű „INSERT” vagy „UPDATE” sávot készítenek.

A Psycopg 2 többnyire C-ben van megvalósítva libpq csomagolásként, ami egyszerre hatékony és biztonságos. Ügyfél- és szerveroldali kurzorokkal, aszinkron kommunikációval és értesítésekkel, „COPY TO / COPY FROM” támogatással rendelkezik. Számos Python típust támogat a dobozon kívül, és a PostgreSQL adattípusokhoz igazítja; Az adaptáció kiterjeszthető és testre szabható a rugalmas objektumadaptációs rendszernek köszönhetően [29].

Pytz

A pytz (v.2018.5) behozza az Olson tz adatbázist a Pythonba. Ez a könyvtár pontos és platformon átívelő időzóna-számításokat tesz lehetővé a Python 2.4 vagy újabb verzió használatával. Megoldja a nyári időszámítás végén felmerülő kétértelmű idők kérdését is, amelyekről a Python Library Reference (datetime.tzinfo) oldalon olvashat bővebben [30].

urllib3

Az urllib3 (v.1.26.5) egy erőteljes, felhasználóbarát HTTP kliens a Python számára. A Python ökoszisztéma nagy része már használja az urllib3-at, és neked is kell. Az urllib3

számos kritikus funkciót tartalmaz, amelyek hiányoznak a Python szabványos könyvtáraiból [31]:

- Ügyféloldali SSL / TLS ellenőrzés.
- Fájlfeltöltés többrészes kódolással.
- Proxy támogatás a HTTP és a SOCKS számára.

Whitenoise

Gyökeresen egyszerűsített statikus fájl kiszolgálás Python webalkalmazásokhoz. Pár soros konfigurációval a WhiteNoise lehetővé teszi, hogy webalkalmazása saját statikus fájljait tudja kiszolgálni, önálló egységgé téve, amely bárhová telepíthető anélkül, hogy támaszkodna az nginx-re, az Amazon S3-ra vagy bármely más külső szolgáltatásra. (Különösen hasznos Heroku, OpenShift és más PaaS szolgáltatóknál). [32]

A WhiteNoise (v.5.2.0) bármely WSGI-kompatibilis alkalmazással működik, de van néhány speciális automatikus konfigurációs funkciója a Django számára.

6.2.5. Modulok bemutatása

Ebben a fejezetben a fontosabb modulok kerülnek bemutatásra, hogy a felhasználó egy általános képet kapjon webalkalmazásról.

Felhasználói típusok ellenőrzése

Vannak bizonyos html elemek, amelyeket csak bizonyos felhasználók láthatnak. Ehhez egy saját szűrőt kellett írnom, amely a 6.9 kódrészleten tekinthető meg. Lényegében annyit csinál ez a kódrészlet, hogy megvizsgálja, hogy egy felhasználó csoportjai közt a kérvényezett csoport megtalálható-e és ennek függvényében térít vissza True és False-ot

```
@register.filter(name='has_group')
def has_group(user, group_name):
    group = Group.objects.get(name=group_name)
    return True if group in user.groups.all() else False
```

6.9. kódrészlet. Felhasználói típusok ellenőrzése

Diákok válaszainak megtekintése

A webalkalmazás egyik legfontosabb szempontja, hogy a Tanár meg tudja tekinteni egy adott diák által megoldott feladatokat, és ezeket a válaszokat tudja pontozni, írásban értékelni. Az alábbi 6.10 kódrészlet szolgál arra a célra, hogy egy adott diák profiljára rákattintva a Tanár vagy Admin típusú felhasználó számára láthatóvá válik a diák által összes megoldott feladat. Ezek a feladatok egymás alatt, dátum szerint csökkenő sorrendben jelennek meg. A kódban látható első sor (@login_required) szolgál arra, hogy csupán bejelentkezett felhasználó vehesse igénybe ezt a funkciót. Az @allowed_users(allowed_roles=['admin','teacher']) részben pedig elmondjuk, hogy konkrétan

melyik típusú felhasználói csoporthoz kell tartozzon ez a felhasználó. Amikor rákattintok a diák felhasználó nevére, ekkor a diák user profilját tovább küldöm a pk-jen keresztül. Először ezen a PK-jen keresztül kiszűrjük a Profilok halmazából a megfelelő profilt, amely ehhez az ID-hez tartozik, majd pedig megkeressük, hogy ezzel a profillal rendelkező felhasználó melyik válaszokat küldte be. Végül pedig ezeket az adatokat továbbítjuk a html, számára, ahol majd megjelenítsük őket.

```
@login_required
@allowed_users(allowed_roles=['admin','teacher'])
def student_profile(request, pk):
    profile = Profile.objects.get(id=pk)
    answer = Solving.objects.filter(user_profile=profile)
    context = {'profile': profile, 'answer': answer}
    return render(request, "classroom/student_profile.html", context)
```

6.10. kódrészlet. Diákok profiljainak megtekintése

Jegyek és vélemények hozzáadása egy beküldött válasza

Mint már említettem a diákok leadott megoldásait nem egy automatikus tesztelésen futtatom le, hanem a Tanárok számára áll fenn ez a feladat, hogy megfelelő visszajelzést biztosítson a diák számára. Mint, ahogyan 6.11 ábrán jól látható ehhez a nézethez is, csak az Admin és Tanár típusú felhasználók férhetnek hozzá. A függvényben 3 paraméter adódik át: a request, amely a kérés információit tartalmazza, a pk, amely ebben az esetben a beérkező válasz azonosítóját tartalmazza, illetve a vizsgált felhasználó profilját. A függvény meghívja az egyik általam személyre szabott form-ot, melynek helyes kitöltése esetén meg tudja adni a diák számára elért pontszámot, illetve a Tanár megoldásról szóló véleményét. Miután ellenőrzésre került az űrlap a diák pontszámához hozzáadjuk, a most elért pontszámát és elmentsük azt.

```
@login_required
@allowed_users(allowed_roles=['admin','teacher'])
def AddScoreToAnswerTeacher(request, pk, profile):
    answer = Solving.objects.get(id=pk)
    form = AddScoreToAnswerTeacherForm(instance=answer)
    profile_user = Profile.objects.get(id=profile)
    if request.method == 'POST':
        form = AddScoreToAnswerTeacherForm(request.POST, instance=answer)
        if form.is_valid():
            form.save()
            profile_user = Profile.objects.get(id=profile)
            profile_user.score += answer.score
            profile_user.save()
            return redirect('practices')

    context = {'form':form, 'answer': answer, 'profile':profile_user}
    return render(request, "practices/add_score_to_answer_teacher.html",
        context)
```

6.11. kódrészlet. Jegyek és vélemények hozzáadása egy beküldött válaszra

Egy feladatra beküldött válaszok megtekintése

Fontos szempont, hogy csupán azoknak a diákoknak láthassák a Tanárok a megoldásait, és csak azokat pontozhassák le, amelyek diákokkal egy osztályban vannak. Ezt a 6.12 kódrészlettel sikerült megvalósítanom. A kódban ki kellett listáznom a kérést intéző Tanár osztályait egy listába, majd pedig meg kellett vizsgálnia, hogy a melyek azok a diákok, amelyek küldtek be választ az adott feladatra és egy osztályban vannak a Tanárral. Majd ezeket az adatokat továbbítottam a megjelenítést intéző sablonba.

```
@login_required
@allowed_users(allowed_roles=['admin','teacher'])
def AllAnswerExercise(request, pk):
    exercise = Exercise.objects.get(id=pk)
    answer = Solving.objects.filter(exercise=exercise)
    teacher_profile = Profile.objects.get(user=request.user.id)
    teacher_class = classroom.objects.filter(user_profile = teacher_profile)
    listA = []
    listB = []

    for i in teacher_class:
        for j in i.user_profile.all():
            listA.append(j.id)

    final_listA = list(dict.fromkeys(listA))
    for x in answer:
        for y in x.user_profile.all():
            listB.append(y.id)

    final_listB = list(dict.fromkeys(listB))
    final_listA_as_set = set(final_listA)
    intersection = final_listA_as_set.intersection(final_listB)
    intersection_as_list = list(intersection)

    answer = Solving.objects.filter(
        exercise=exercise,user_profile__in=intersection_as_list)
    context = {'answer':answer, 'exercise': exercise, 'intersection_as_list':
        intersection_as_list}
    return render(request,"practices/student_answers.html", context)
```

6.12. kódrészlet. Egy feladatra beküldött válaszok megtekintése

Tanár regisztráció

Mivel regisztráció során ki lehetett választani, hogy milyen típusú felhasználót szeretne létrehozni a látogató így két féle regisztrációs view-ot kellett létrehoznom. A 6.13 kódrészlet mutatja be a Tanár típusú felhasználó regisztrációját. Amikor a Tanár regisztrálni szeretne ki kell tölteni, egy űrlapot, amely a 7.2 ábrán látható. Amennyiben ez az

űrlap helyesen kerül kitöltésre elkészítjük a felhasználói fiókot, ezzel egyidőben létrejön a felhasználói profil is, majd a következő lépésben hozzáadjuk a felhasználót a kiválasztott felhasználói típusához, majd elmentjük és egy üzenetet továbbítunk a sikeres regisztrációról.

```
def teacher_register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            user = form.save()
            group = Group.objects.get(name='teacher')
            user.groups.add(group)
            messages.success(request, f'Felhasznaloi fiokod sikeresen
                                   létrejott!')
            return redirect('login')
        else:
            form = UserRegisterForm()
    return render(request, 'users/register.html', {'form': form})
```

6.13. kódrészlet. Tanár regisztráció

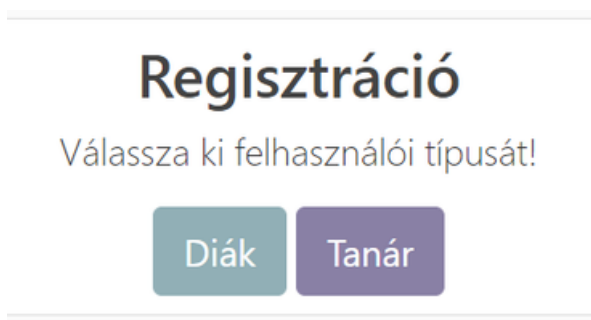
7. fejezet

A felhasználói felület bemutatása

Mivel a felhasználói felület számos nézetet tartalmaz, ezért itt csak pár elemet mutatok be. Az alábbi linken megtekinthető a projekt live-demo verziója: <https://molnar-allamvizsga-live.herokuapp.com/>

7.1. Regisztráció

A látogató típusú felhasználó a menüben ki tudja választani a regisztráció fület, melyre kattintva az alábbi két lehetőség közül választhat (lásd 7.1 ábra), annak függvényében milyen típusú felhasználót szeretne létrehozni.



The image shows a registration form with a light gray background. At the top, the word "Regisztráció" is written in a large, bold, dark blue font. Below it, the text "Válassza ki felhasználói típusát!" is written in a smaller, gray font. Underneath this text are two rounded rectangular buttons. The left button is light blue and contains the word "Diák" in white. The right button is light purple and contains the word "Tanár" in white.

7.1. ábra. Regisztráció típusának kiválasztása

A kiválasztott elem után a regisztrálásra szolgáló űrlapot láthatjuk, 7.2 ábra, ahol megjelenik 4 mező, amelyeket az alattuk lévő utasítások alapján szükséges kitöltenünk. Felhasználónév kitöltése esetén kötelező kevesebb, mint 150 karakter, csak betűk, számjegyek és @ / . / + / - / használata. A rendszer figyelembe veszi és visszajelez, amennyiben az általunk kívánt felhasználónév már foglalt. Az általunk megadott E-mail cím formátuma hiteles kell legyen, ami azt jelenti, hogy a következő alakban kell legyen legyen:

- név: felhasználó neve, a postafiók azonosítója
- @ (kukac): kötelező elválasztójel
- kiszolgáló: levelező szerver címe
- például: valaki@valahonnan.com

A jelszavunk megadásánál számos tényezőre kell figyelnünk, hogy megfelelő, biztonságos legyen fiókunk:

- Jelszava nem lehet túl hasonló a többi személyes adatához.
- Jelszavának legalább 8 karakterből kell állnia.
- Jelszava nem lehet gyakran használt jelszó.
- Jelszava nem lehet teljesen numerikus.

A jelszavunkat hitelesíteni azzal tudjuk, ha megadjuk ugyanazt a jelszót, mint az előbb, ezzel bizonyítva, hogy tudjuk az általunk megadott pontos jelszavót. Jelszó visszaállításra kattintva elmenthetjük az újonnan létrehozott jelszavunkat.

Csatlakozz hozzánk!

Felhasználónév*

Kötelező. Kevesebb, mint 150 karakter. Csak betűk, számjegyek és @ / . / + / - / _ .

Email*

Hiteles email cím megadása.

Jelszó*

- Jelszava nem lehet túl hasonló a többi személyes adatához.
- Jelszavának legalább 8 karakterből kell állnia.
- Jelszava nem lehet gyakran használt jelszó.
- Jelszava nem lehet teljesen numerikus.

Jelszó hitelesítés*

Az ellenőrzéshez írja be ugyanazt a jelszót, mint korábban.

[Regisztráció](#)

Van már felhasználói fiókja? [Bejelentkezés](#)

7.2. ábra. Regisztrációs űrlap

Sikertelen regisztráció esetén, a hibás mezőt kijelölve megkér minket a rendszer, hogy javítsuk ki hibáinkat. Sikeres regisztráció esetén pedig átirányít minket a bejelentkezéshez.

7.2. Bejelentkezés

Amennyiben most regisztráltunk és úgy kerültünk át a bejelentkezéshez, abban az esetben a 7.3 űrlapot látjuk. Ha a menüből kiválasztjuk a bejelentkezés fület, akkor ez az üzenet természetesen nem lesz látható számunkra. Bejelentkezéskor a regisztráció során megadott felhasználónevünket és a jelszavunkat kell megadnunk.

Felhasználói fiókod sikeresen létrejött!

Bejelentkezés

Felhasználónév*

Jelszó*

Belépés

[Elfelejtette a jelszavát?](#)

Felhasználói fiókra van szüksége? [Regisztráció most](#)

7.3. ábra. Bejelentkezés űrlap

Amennyiben elfelejtettük a jelszavunkat az "Elfelejtette jelszavát" elemre kattintva meg tudjuk adni E-mail címünket, amelyre a rendszer egy email-t küld, amelyben található egy link, amelyre kattintva megadhatjuk az általunk kívánt új jelszót (lásd 7.4 ábra). Itt is, mint regisztráció során meg kell adnunk az általunk kívánt új jelszót, az alapvető követelményeknek megfelelően és ezt hitelesítenünk kell.

Jelszó visszaállítása

Új jelszó*

- Jelszava nem lehet túl hasonló a többi személyes adatához.
- Jelszavának legalább 8 karakterből kell állnia.
- Jelszava nem lehet gyakran használt jelszó.
- Jelszava nem lehet teljesen numerikus.

Új jelszó hitelesítése*


Jelszó visszaállítása

7.4. ábra. Jelszó visszaállítása

7.3. Kezdőlap

A kezdőlapot ugyan úgy elérheti mindegyik felhasználói típus és ugyan azon elemeket fogják látni változtatások nélkül. Itt vannak az Adminok által felhívó szövegek, verseny és pályázat kiírások, karbantartásra figyelmeztető üzenetek (lásd 7.5 ábra). Az oldalon egyszerre öt bejegyzés tekinthető meg, ha ennél több bejegyzésünk van, akkor lapozni lehet a bejegyzések között. A bejegyzések létrehozási dátum alapján sorrendbe jelennek meg a felhasználók számára. Egy felhasználó képes kattintani a bejegyzést létrehozó Ad-

min nevére, vagy a bejegyzés nevére, ezáltal megtekintheti az Admin által eddigi összes létrehozott bejegyzést vagy konkrétan egy bejegyzést.

Admin 2021/06/21


Felhívás

Kódolj! csapata a 2020/2021-es tanévre meghirdeti a 12. Kódolj! programozási versenyt.

A versenyt **három fordulóban** rendezzük meg, a három fordulóban szerzett összpontszám alapján hirdetünk nyertest!

A verseny tárgya, követelményei:

- programozási alapismeretek (programozási tételek),
- a C++, Pascal, C# vagy Java programozási nyelv ismerete,
- rendszerszemléletű feladatmegoldás, algoritmusok kidolgozása, programok fejlesztése, tesztelése.

Admin 2021/06/21

Karbantartás

Kedves Felhasználó! Oldalunkon rövid karbantartást fogunk végezni.

Az alábbi dátumokon nem lesz elérhető az oldal a felhasználók számára: **2021.03.12 - 2021.03.15**

Megértésed köszönjük!

1

2

Következő

Utolsó


7.5. ábra. Kezdőlap

Admin:

Jelenleg csak az Admin képes bejegyzések írására, frissítésére és törlésére (lásd 7.6 ábra). A bejegyzések írása során a tartalom szerkesztéséhez számos lehetőség biztosítva van:

- A szöveg számára stílust adni.
- Bekezdéseket formázni.
- Szöveget kiemelni, aláhúzni, áthúzni, megdönteni.
- Linkeket elhelyezni.
- Fényképeket hozzáadni.
- Táblázatokat készíteni.
- Továbbá számos más lehetőség is elérhető.

Amennyiben a szöveget frissíteni szeretnénk, a frissítés gombra kattintva betölti nekünk az eddig létrehozott bejegyzést tartalmával együtt (lásd 7.7 ábra).



Admin 2021/06/21

FrissítésTörlés

Felhívás

Kódojl! csapata a 2020/2021-es tanévre meghirdeti a 12. Kódojl! programozási versenyt.

A versenyt **három fordulóban** rendezzük meg, a három fordulóban szerzett összpontszám alapján hirdetünk nyertest!

A verseny tárgya, követelményei:

- programozási alapismeretek (programozási tételek),
- a C++, Pascal, C# vagy Java programozási nyelv ismerete,
- rendszerszemléletű feladatmegoldás, algoritmusok kidolgozása, programok fejlesztése, tesztelése.

7.6. ábra. Bejegyzés frissítése és törlése

Bejegyzés írása

Cím*

Felhívás

Tartalom

Styles | Normal | **B** *I* U ~~S~~ ↶ ↷ | | Source

Kódojl! csapata a 2020/2021-es tanévre meghirdeti a 12. Kódojl! programozási versenyt.

A versenyt **három fordulóban** rendezzük meg, a három fordulóban szerzett összpontszám alapján hirdetünk nyertest!

A verseny tárgya, követelményei:

- programozási alapismeretek (programozási tételek),
- a C++, Pascal, C# vagy Java programozási nyelv ismerete,
- rendszerszemléletű feladatmegoldás, algoritmusok kidolgozása, programok fejlesztése, tesztelése.

body p

Küldés

7.7. ábra. Bejegyzés írása

7.4. Feladatok

Feladatok Megtekintése:

A három típusú felhasználó három különböző nézetet lát a feladatok tekintetében. A feladatok egy táblázatban helyezkednek el. A feladatokról itt nyilván van tartva a sorszá-
muk, címük, nehézségi szintjük, típusuk, hány pontot érnek illetve minden felhasználói
típus számára más más lehetőséget. A Vendég felhasználó képes megtekinteni a feladato-

kat, de nem tudja őket megoldani. A Diáknak lehetősége van a feladatok kidolgozására és a feladatok megoldásának megtekintésére, amennyiben van ilyen feltöltve.

A Tanárnak lehetősége van a feladatok szerkesztésére, törlésére, feladat megoldásának megadására illetve meg tudják tekinteni azoknak a Diákoknak a válaszait, amelyek egy osztályban vannak a Tanárral (lásd 7.8 ábra) Mindegyik felhasználói típus tud keresni a feladatok közt. Keresni lehet a címek, szintek, feladat típus és elérhető maximum pontszám között. Továbbá a felhasználó képes az oldalanként megjelenő elemek számának megváltoztatására. A megjeleníthető elemek száma 10, 25, 50 vagy akár 100. Amennyiben nincsen elem a táblázatban, vagy keresés során nem található a keresési követelményeknek megfelelő elem abban az esetben értesítve vagyunk erről. A feladatok között képesek vagyunk lapozni illetve az első vagy akár az utolsó oldalra ugrani tetszés szerint.

Új feladat létrehozása:

A Tanár illetve Admin típusú felhasználónak jogában áll új feladatokat készíteni és hozzáadni azt a meglévő feladat állományhoz. Szükséges a feladatnak egy találó címet választani, kiválasztani egy legördülő menüből a feladat szintjét, szintén egy legördülő menüből ki kell választani, hogy milyen típusú feladatot hoztak létre. Továbbá meg kell adniuk magát a feladatnak a szövegét. Ugyanúgy, mint a Bejegyzések szerkesztése során számos formázási lehetőség áll rendelkezésre a felhasználók számára. Az utolsó kitöltésre váró mezőbe a feladat nehézségétől függően szükséges megadnunk egy maximum elérhető pontszámot, hogy mindenki számára legyen egyértelmű a feladatok hány pontot érnek.

Elemek száma oldalanként:

10

Keresés

Könnyű

Cím	Szint	Típus	Pont	Frissítés	Törlés	Megoldás	Válaszok
Fazekas	Könnyű	Dinamikus programozás	10	Frissítés	Törlés	Megoldás	Válaszok
Fotótás	Könnyű	Verem	10	Frissítés	Törlés	Megoldás	Válaszok
Fénykép	Könnyű	Mohó algoritmus	15	Frissítés	Törlés	Megoldás	Válaszok
Munka	Könnyű	Sor	15	Frissítés	Törlés	Megoldás	Válaszok
Raktár	Könnyű	Dinamikus programozás	10	Frissítés	Törlés	Megoldás	Válaszok
Szállítás	Könnyű	Gráfok	10	Frissítés	Törlés	Megoldás	Válaszok

Oldal: 1/1 (Keresés mind a 14 bejegyzés között)

Első

Előző

1

Következő


Utolsó

7.8. ábra. Feladatok közti keresés tanárként

7.5. Profil

A Profil menüpont alatt minden bejelentkezett felhasználó meg tudja tekinteni saját felhasználó nevét, e-mail címét, illetve az eddig elért összpontszámát, amit a feladatok

megoldása során szerzett (lásd 7.9 ábra). A felhasználó nem csak meg tudja nézni, hanem meg is tudja változtatni személyes adatait illetve tetszőleges profilképet állíthat be magának. A Felhasználónév illetve a E-mail cím szerkesztése során fontos, hogy megfelelőképpen töltsse ki a felhasználó az adatait. Amennyiben nem töltött ki egy mezőt figyelmeztető üzenetet kap, illetve nem engedi meg a rendszer a felhasználó számára a profil frissítését.



Felhasználónév: Tanar6
E-mail cím: ass@as.com21
Pontszám: 25

Profil információk

Felhasználónév *

Kötelező. Kevesebb, mint 150 karakter. Csak betűk, számjegyek és @ / . / +

E-mail cím *

Fénykép *

Jelenlegi profilkép: [profile_pics/en2_AINRWgr.jpg](#)
Megváltoztatás:

Nincs fájl kiválasztva

7.9. ábra. Felhasználói profil megtekintése

7.6. Osztályok megtekintése

A felhasználók az "Osztályok" menüpont alatt képesek megtekinteni, hogy melyik osztályokba tartoznak (lásd 7.10 ábra). Egy kiválasztott osztály "Megtekintés" gombjára rákattintva lehetőség nyílik látni, hogy kik vannak az adott osztályban. A Tanár itt ki tud választani Diákokat, és megtudja tekinteni az általuk eddig beküldött feladatokat.



7.10. ábra. Osztályok megtekintése

Összefoglaló

Összegezve a dolgozatomban leírtakat azt tudom mondani, hogy sikerült a kitűzött céljaimat megvalósítani, és egy olyan, mindenki számára könnyen hozzáférhető webalkalmazást létrehozni, amely megkönnyíti úgy a diákok, mint a tanárok munkáját.

- Az alábbi linken megtalálható a projekt GitHub Repository: <https://github.com/molnahr/allamvizsga>.
- Az alábbi linken megtekinthető a projekt live-demo verziója: <https://molnar-allamvizsga-live.herokuapp.com/>

7.7. Továbbfejlesztési lehetőségek

Mint említettem számos továbbfejlesztési lehetőség rejlik a projektben, hadd említsek meg párat ezek közül.

- Az oldalt ki lehetne bővíteni egy olyan funkcióval, hogy a felhasználók tudják szerkeszteni a programjaikat különböző programozási nyelveken a felületen: C, C++, Java, Python, JavaScript.
- Egy tanuló aktivitásának grafikonokon való ábrázolása.
- A tanárok számára osztály szinten összevont statisztika.
- A webalkalmazás front-endjének további fejlesztése, animációk készítése.
- Programozói versenyek megszervezésére való lehetőség.
- Tanárok számára lehetőséget, hogy feladatok jelöljenek ki a diákok számára, melyről a diákok értesítést kapnak, mind a profiljukon, mind E-mailben.
- Lehetősége legyen a felhasználóknak a feladatok, bejegyzések kedveslésére és hozzászólás hozzáfűzésére.

Ábrák jegyzéke

2.1. Nem és életkor	13
2.2. Diákok által felhasznált eszközök versenyre való készülés céljából	14
2.3. Nem megfelelő fordítás	14
2.4. Igényfelmérés	14
4.1. SQLite	20
4.2. PostgreSQL	21
4.3. Django MVT	24
5.1. A rendszer használati eset diagramja	29
6.1. A rendszer architektúrája	34
6.2. A projekt gyökérkönyvtárának tartalma	37
6.3. A projekt belső könyvtárának tartalma	37
6.4. Practices alkalmazás könyvtára	38
6.5. Entity Relationship Diagram	41
7.1. Regisztráció típusának kiválasztása	51
7.2. Regisztrációs űrlap	52
7.3. Bejelentkezés űrlap	53
7.4. Jelszó visszaállítása	53
7.5. Kezdőlap	54
7.6. Bejegyzés frissítése és törlése	55
7.7. Bejegyzés írása	55
7.8. Feladatok közti keresés tanárként	56
7.9. Felhasználói profil megtekintése	57
7.10. Osztályok megtekintése	58

Irodalomjegyzék

- [1] M. C. Craig S. Mullins, „Database management system (dbms).” <https://searchsqlserver.techtarget.com/definition/database-management-system>. Utolsó megtekintés 2021-03-02.
- [2] wikipedia.org, „Sqlite.” <https://hu.wikipedia.org/wiki/SQLite>. Utolsó megtekintés 2021-03-06.
- [3] sqlite.org, „What is sqlite?.” <https://www.sqlite.org/index.html>. Utolsó megtekintés 2021-04-16.
- [4] postgresql.org, „What is postgresql?.” <https://www.postgresql.org/about/>. Utolsó megtekintés 2021-01-11.
- [5] TablePlus, „Sqlite vs postgresql.” <https://tableplus.com/blog/2018/08/sqlite-vs-postgresql-which-database-to-use-and-why.html>. Utolsó megtekintés 2021-03-25.
- [6] A. Ryabtsev, „Web frameworks.” <https://djangostars.com/blog/what-is-a-web-framework/>. Utolsó megtekintés 2021-03-25.
- [7] C. Kopecky, „What is django python? build your first program from scratch.” <https://www.educative.io/blog/what-is-django-python>. Utolsó megtekintés 2021-02-11.
- [8] javatpoint.com, „Django mvt.” <https://www.javatpoint.com/django-mvt>. Utolsó megtekintés 2021-05-21.
- [9] K. Yadav, „What is aws and what can you do with it.” <https://blog.usejournal.com/what-is-aws-and-what-can-you-do-with-it-395b585b03c>. Utolsó megtekintés 2021-05-16.
- [10] A. W. Services, „Amazon relational database service.” <https://aws.amazon.com/rds/>. Utolsó megtekintés 2021-05-18.
- [11] A. W. Services, „Amazon s3.” <https://aws.amazon.com/s3/>. Utolsó megtekintés 2021-06-02.
- [12] stackshare.io, „Apache http server vs heroku.” <https://stackshare.io/stackups/apache-httpd-vs-heroku>. Utolsó megtekintés 2021-05-21.

- [13] tutorial.djangogirls.org, „Django urls.” https://tutorial.djangogirls.org/en/django_urls/. Utolsó megtekintés 2021-05-23.
- [14] frontendmasters.com, „What is a front-end developer?” <https://frontendmasters.com/books/front-end-handbook/2018/what-is-a-FD.html>. Utolsó megtekintés 2021-06-04.
- [15] javatpoint.com, „Front-end web development.” <https://www.javatpoint.com/what-is-bootstrap>. Utolsó megtekintés 2021-04-17.
- [16] docs.djangoproject.com, „The django template language.” <https://docs.djangoproject.com/en/3.2/ref/templates/language/>. Utolsó megtekintés 2021-05-23.
- [17] J. Matthews, „A non-magical introduction to pip and virtualenv for python beginners.” <https://www.dabapps.com/blog/introduction-to-pip-and-virtualenv-python/#:~:text=virtualenv%20is%20a%20tool%20for,time%20on%20the%20same%20machine>. Utolsó megtekintés 2021-04-17.
- [18] D. D. version: 3.2, „Writing your first django app, part 1.” <https://docs.djangoproject.com/en/3.2/intro/tutorial01/>. Utolsó megtekintés 2021-04-17.
- [19] D. D. version: 3.2, „Django session.” <https://docs.djangoproject.com/en/3.2/topics/http/sessions/>. Utolsó megtekintés 2021-05-22.
- [20] D. D. version: 3.2, „Django content types.” <https://docs.djangoproject.com/en/3.2/ref/contrib/contenttypes/>. Utolsó megtekintés 2021-05-22.
- [21] D. D. version: 3.2, „Models.” <https://docs.djangoproject.com/en/3.2/topics/db/models/>. Utolsó megtekintés 2021-05-22.
- [22] T. D. Book, „Model definition reference.” <https://web.archive.org/web/20190119091044/http://djangobook.com/model-definition-reference/>. Utolsó megtekintés 2021-05-27.
- [23] A. Team, „Asgi documentation.” <https://asgi.readthedocs.io/en/latest/>. Utolsó megtekintés 2021-05-27.
- [24] A. W. Services, „Boto3 documentation.” <https://boto3.amazonaws.com/v1/documentation/api/latest/index.html>. Utolsó megtekintés 2021-05-27.
- [25] pypi.org, „Certifi.” <https://pypi.org/project/certifi/>. Utolsó megtekintés 2021-06-01.
- [26] pypi.org, „django-crispy-forms.” <https://pypi.org/project/django-crispy-forms/>. Utolsó megtekintés 2021-06-01.
- [27] C. G. Alex Gaynor *et al.*, „django-filter.” <https://django-filter.readthedocs.io/en/stable/>. Utolsó megtekintés 2021-06-03.

- [28] pypi.org, „fontawesome-free.” <https://pypi.org/project/fontawesome-free/>. Utolsó megtekintés 2021-06-03.
- [29] pypi.org, „psycopg2.” <https://pypi.org/project/psycopg2/>. Utolsó megtekintés 2021-05-19.
- [30] pypi.org, „pytz.” <https://pypi.org/project/pytz/>. Utolsó megtekintés 2021-05-19.
- [31] pypi.org, „urllib3.” <https://pypi.org/project/urllib3/>. Utolsó megtekintés 2021-05-19.
- [32] pypi.org, „whitenoise.” <https://pypi.org/project/whitenoise/>. Utolsó megtekintés 2021-05-19.