PÉCSI TUDOMÁNYEGYETEM
TERMÉSZETTUDOMÁNYI KAR
Matematikai és Informatikai Intézet

A CSILLAGÁSZAT ATLASZA

Témavezető: DR. NAGYVÁRADI LÁSZLÓ
egyetemi docens

Készítette: CZIKY PÁL
Programtervező Informatikus

PÉCS, 2019

„Minél többet tudsz, annál kevesebbet értesz.”

(Lao-Ce)

Tartalom

I. Téma Összefoglaló / Absztrakt

Ez a rendszer a középiskolás tanulók számára teszi elérhetőbbé a csillagászat tudományát oktatási segédanyagként.

„A csillagászat atlasza” című könyvet dolgozza fel. Ez a kiadvány a csillagászat alapjaival foglalkozik és érdekességekkel, színes képekkel bővíti minden fejezetét. A szakdolgozatban létrehozott rendszer a könyv által lefektetett tematikát használja fel. Megőrizve annak képi világát, átlátható tagolt fejezeteit egy marandó élményt nyújt a naprendszerünk égitesteinek bemutatásában. Minden fejezet végén egy játékos kvíz teszi próbára a tanulók frissen szerzett tudását.

A szakdolgozat Python (3.7.2) nyelven íródott a Django keretrendszer használatával. A weboldal tartalmait egy PostgreSQL adatbázisból szolgáltatjuk. Ezek segítségével, egy webes felületen, a hallgatók tudnak regisztrálni / belépni és a könyv fejezeteit egyenként végig venni. Majd mindegyik fejezet anyagából akár többször vizsgázni.

Az oldalhoz tartozó „admin” felületen teljes uralmat kapunk a tartalmak felett. A könyv fejezetei tetszés szerint bővíthetők, és hozzájuk akár több külön álló kvíz játék is létrehozható. A kvíz játékban variálható ponthatárokkal, kérdés sorral, válaszaik és azok pontszámaival játszhatunk szabadon. Egy külön részlegben az oldalon regisztrált tanulók adatait is menedzselhetjük. Hozhatunk létre új fiókot vagy törölhetünk már meglévőket.

II. Bevezetés

A szakdolgozati témám választását több tényező is befolyásolta.

Mindig is érdekelt hobbi szinten a csillagászat és a hozzá kapcsolódó más tudományágak, mint például az asztrológia. Mindig nagy élmény volt a várostól távol az éjszakai égboltot kémlelni és bár komoly csillagvizsgáló eszközhöz nem volt szerencsém, egy két csillagkép felismerése nagy boldogsággal töltött el. Joachim Hermann „És ez melyik csillag" című könyvében évszakokra bontva segít felfedezni az égi együtt állásokat és nagy segítségemre volt ebben a témában.

Kihívásként valami olyan programozási nyelvet szerettem volna választani, melyet az egyetemi éveim alatt egyáltalán nem vagy csak kis mértékben érintettünk. A Java és a C# nyelvek népszerűségét mára utolérte a Python trendje. Sok média fórumon és a munkahelyemen is lépten, nyomon találkozok azzal, hogy fejlesztés iránt érdeklődő vagy pálya kezdő fejlesztőknek ajánlják könnyen érthetősége miatt. Kisebb munkánkat elősegítő eszközöket lehet vele rövid idő alatt létrehozni.

Ezzel a mentalitással kerestem fel Dr. Nagyváradi Lászlót a Térképészeti és Geo informatikai Tanszék docensét, hogy ajánljon egy az elképzeléseimhez közel álló témát és témavezetőt. Nagy megtiszteltetés volt, hogy Ő maga vállalta el a témavezetői szerepet és egy érdekes és aktuális témát is kaptam tőle. „A csillagászat atlasza" című könyvet – melynek készítésében a témavezetőm saját munkássága is benne van – szerettük volna valamilyen digitális formában feldolgozni oktatási segédanyagként középiskolások részére. A könyv anyagát egy kvízjátékkal kérdeznénk vissza a tanulóktól.

1. Célkitűzés, a program rendszerterve

1. A rendszer rendelkezzen egy beléptető rendszerrel
    a. Az oldal felhasználói könnyedén tudjanak regisztrálni, belépni és kilépni
    b. Minden felhasználó rendelkezzen egy „könyvjelzővel" mely azt jelzi hány fejezet kvíz játékát oldotta már meg sikeresen
    c. Legyen egy kiemelt adminisztrátor, aki plusz jogosultságokkal bír és menedzselni tudja az oldal tartalmát és annak felhasználóit
2. A rendszer egy jól átlátható webes felületen jelenítse meg a könyvet fejezetenként
    a. Használja fel a könyv tagolását
    b. Legyen képes a könyv ábráinak, táblázatainak megjelenítésére
    c. Felülete ne legyen zavaró bonyolult stílus elemekkel
    d. Csak azok a fejezetek jelenjenek meg a könyvből ahányhoz az adott felhasználó még nem végezte el sikeresen a kvízt
3. A rendszer egy elkülönített részben szolgáltasson hozzáférést a kvízjátékhoz
    a. Lehessen akár több kvíz sort rendelni egy fejezethez
    b. Sokféle kérdést lehessen használni, mint pl.: igaz, hamis; kakukktojás; ismerd fel a képet stb.
    c. Változó mennyiségű választ lehessen a kérdésekhez írni

d. Legyen egy állítható ponthatár minden kérdés csoporthoz
e. Mindig legyenek könnyedén érthetők és játékosak, de biztosítsanak elég kihívást

Megoldásra váró problémáink nem csak a könyv feldolgozására vonatkozó akadályokra korlátozódnak, hanem azokkal az általános feladatokkal is foglalkoznunk kell, amik a webes szolgáltatások és az adatbázisok biztonság technikájának vele járói. Mint például, hogy az oldal rosszindulatú használói ne tudjanak kárt tenni az oldalban vagy a mögötte futó web szerverben. Ne lehessen hozzáférni az adatbázishoz olyan módszerekkel, mint például az SQL injektálás. Mi történjen, ha valami oknál fogva működés közben mégis valamiért megszűnik a kapcsolat az adatbázissal?

A fenti célkitűzésekre már megoldást kínáló programok:

Hasonló programok léteznek, főleg kvíz alkalmazások széles skálájával lehet találkozni az interneten. Így például direktben kereshetünk akár már létező Django csomagokra is a weboldalukon. ([1])

„Online quiz system"az egyik legösszetettebb ám fizetős – 200 USD- verzió. Ez is ugyanúgy a Python Django keretrendszerét használja fel és professzionális kinézetű saját igényeknek megfelelően alakítható kvízeket lehet létrehozni és azokat kiértékelni. Eredmény táblái tökéletes átláthatóságot biztosítanak a különböző kvízek között. ([2])

„Tomwalker quiz"a Django kereső oldal szerint legtöbbet használt megoldás. Többen is lemásolták és tovább fejlesztették vagy le egyszerűsítették a megoldását. Twitterre hajazó stílusa rögtön ismerős környezetet tár a felhasználók elé. ([3])

„Funkybob" szinte semmilyen leírás nem tartozik hozzá és 2013-ban nyúltak hozzá legutoljára. Egyszerűen a legelavultabb, legkevésbé megbízható az összes opció közül. ([4])

Mindegyik rendszernek az a problémája, hogy a könyv feldolgozására nem ad lehetőséget. A Django rendszer egyik különlegessége, hogy különböző alkalmazások könnyedén hozzá építhetők a programunkhoz. De mindegyik, itt felsorolt lehetőség Python 2.7-es vagy még régebbi verzióval készült, ezt tetőzve a Dajngo verzió különbségekkel együttesen, az integrációt vagy bármilyen iránymutatást teljesen ellehetetlenít.

[1] https://djangopackages.org/packages/p/django-quiz-app/
[2] https://www.freeprojectz.com/paid-projects/django-python-mysql/employee-online-quiz-system
[3] https://github.com/tomwalker/django_quiz
[4] https://pypi.org/project/simple-quiz/

III. Eszközök/anyagok és módszerek ismertetése

1. Python

Először is el kellett döntenem, hogy melyik Python verziót használjam. A régebbi 2.7-es verzióhoz elvileg több leírás és különböző kiegészítők elérhetőek, de a támogatása ebben az évben megszűnik és mindenki fokozatosan tér át a 3.x-es verziókra. ([5])

Ezért a python 3-ban terveztem elkészíteni a szakdolgozatom.

2. Pip

A python csomag telepítését követően elérhetővé válik egy beépített modul is, ez a pip. A parancssorból használható pip utasításokkal telepíthetünk sokféle python kiegészítőt a számítógépünkre. Használatáról még később a telepítés fejezetben esik szó.

3. A Django keretrendszer rövid bemutatása

Rövid keresgélés után találtam rá, hogy a Python nyelvben a webes fejlesztést többféle keretrendszer is támogatja. Ilyenek például a Django, Flask és a Pyramid, Tornado, Bottle stb. Ezek közül a kettő legelterjedtebb a Django, a Flask és a Pyramid. Míg a Flask kisebb alkalmazásokra lett kitalálva a Django és Pyramid bonyolultabb alkalmazásokat is támogat. ([6])

A Django mellett döntöttem végül, mert beépített SQLite adatbázis, és egy fejlesztői web szerver is járt hozzá alapból az installálást követően. Ezzel a környezettel könnyű volt rögtön elindulni és megtanulni a működését. Beépített admin felülettel rendelkezik, illetve nagyon jól érthető, kereshető dokumentációja van. A dokumentáció kiemelten fontos szempont volt, mivel a hátránya ennek a keretrendszernek, hogy sok előre megírt kód és modulja van, amik egy bizonyos szabvány / logika szerint lettek megtervezve és ezeket a szabályosságokat követnem kellett nekem is a fejlesztés során.

4. Django és az MVN

A Django az MVN – Modell Nézet Vezérlő (angolul: MVC - Model View Controller) program tervezési mintát követi.

Leírása:

[5] https://pythonclock.org/
[6] https://www.airpair.com/python/posts/django-flask-pyramid

„Gyakori egy alkalmazás több rétegre való felbontása: megjelenítés (felhasználói felület), tartománylogika és adatelérés. Az MNV-ben a megjelenítés tovább bomlik nézetre és vezérlőre. Az MNV sokkal inkább meghatározza egy alkalmazás szerkezetét, mint az egy programtervezési mintára jellemző.

4.1 Modell

Az alkalmazás által kezelt információk tartomány-specifikus ábrázolása. A tartománylogika jelentést ad a puszta adatnak (pl. kiszámolja, hogy a mai nap a felhasználó születésnapja-e, vagy az összeget, adókat és szállítási költségeket egy vásárlói kosár elemeihez).

Sok alkalmazás használ állandó tároló eljárásokat (mint mondjuk egy adatbázis) adatok tárolásához. Az MNV nem említi külön az adatelérési réteget, mert ezt beleérti a modellbe.

4.2 Nézet

Megjeleníti a modellt egy megfelelő alakban, mely alkalmas a felhasználói interakcióra, jellemzően egy felhasználói felületi elem képében. Különböző célokra különböző nézetek létezhetnek ugyanahhoz a modellhez.

4.3 Vezérlő

Az eseményeket, jellemzően felhasználói műveleteket dolgozza fel és válaszol rájuk, illetve a modellben történő változásokat is kiválthat.

Habár az MNV-nek sok értelmezése létezik, a vezérlés menete általánosságban a következőképp működik:

A felhasználó valamilyen hatást gyakorol a felhasználói felületre (pl. megnyom egy gombot).

A vezérlő átveszi a bejövő eseményt a felhasználói felülettől, gyakran egy bejegyzett eseménykezelő vagy visszahívás útján.

A vezérlő kapcsolatot teremt a modellel, esetleg frissíti azt a felhasználó tevékenységének megfelelő módon (pl. a vezérlő frissíti a felhasználó kosarát). Az összetett vezérlőket gyakran alakítják ki az utasítás mintának megfelelően, a műveletek egységbezárásáért és a bővítés egyszerűsítéséért.

A nézet (közvetve) a modell alapján megfelelő felhasználói felületet hoz létre (pl. a nézet hozza létre a kosár tartalmát felsoroló képernyőt). A nézet a modellből nyeri az adatait. A modellnek nincs közvetlen tudomása a nézetről.

A felhasználói felület újabb eseményre vár, mely az elejéről kezdi a kört.

A modell és a nézet kettéválasztásával az MNV csökkenti a szerkezeti bonyolultságot, és megnöveli a rugalmasságot és a felhasználhatóságot.” (7)

A Django-ban a következő párokat állíthatjuk fel ez alapján:

Django adatbázisa – Modell

Django view – Nézet

Django URL-ek – Vezérlő

5. Django projekt felépítése

Első Django projektünk létrehozásakor a főmappában létrejön egy manage.py file, amivel a fő funkcionalitások irányíthatóak. Például web szerver elindítása, adatbázis változások migrálása vagy új „app” létrehozása. Ezt a fájlt nem ajánlott szerkeszteni.

Illetve a projektünk nevével azonos mappa, amiben csak pár fájl jön létre. Ezek közül a kettő legfontosabb:

settings.py: itt találhatjuk az „app”-ok listáját, adatbázis kapcsolathoz szükséges információk, időzóna, nyelvi beállítások, statikus fájljaink elérési útjai stb.

urls.py: a projektünkből elérhető oldalak listája. A gyakorlatban használatos szabály, hogy nem minden elérési utat adunk itt meg, csak azokat, amik elvezetnek a különböző „app”-okhoz.

5.1 Django App

A Django projektek több különböző feladatokat ellátó részekre bonthatók ezeket „app”-oknak nevezzük. Hasonlatosan egy Linux operációs rendszerhez a projektünk – mint mag – köré rengeteg különböző „app”-ot tudunk építeni, amik mind jól definiált funkcionalitást látnak el. Ezeket a rendszer integritásának megsértése nélkül ugyanígy le is tudjuk választani. Ilyen formában egy rendkívül jól skálázható rendszerről beszélünk.

Minden „app” regisztrálásra kerül létrehozása után a projekt settings.py fájljában és az urls.py-ban megkapja a hozzá vezető url stringet.

Minden különálló „app” a következő képen épül fel:

migrations mappa – az adatbázis létrehozásakor, vagy a tábláink bármi változása, ami a models.py fájlt érinti, ezeket migrálni kell a tényleges adatbázisunkba. Ilyenkor a Django a

[7] https://hu.wikipedia.org/wiki/Modell-n%C3%A9zet-vez%C3%A9rl%C5%91

kiválasztott adatbázis számára érthető leíró nyelven legenerált kódot ad át. Így például a mi esetünkben a postgreSQL számára fordított SQL parancsok.

templates mappa – újrahasználható html lapok. Szerepük, hogy a Django logikát és a html kódot különválasszák egymástól. A lapokat mindig szabvány szerint úgy helyezzük el, hogy a templates mappán belül az app neve alá. Futási időben ezeket a lapokat a Django egy helyre húzza össze, és ha egy másik app-ban is lenne, mondjuk home.html, index.html vagy login.html, akkor is meg tudja különböztetni melyik-melyikhez, tartozik.

admin.py – azt állíthatjuk itt be, hogy az admin oldalon hogyan és milyen tartalmakat lehet majd szerkeszteni. Tipikusan ilyen például az „app" adatbázis táblái

apps.py – az „app" neve, ahogyan a rendszer többi részében hivatkozhatunk rá

forms.py – adatlapojat tartamaz. Ha a felhasználótól adatokat kérünk be, amiket az adatbázisba szeretnénk elmenteni, akkor azokat itt érdemes létrehozni. Tipikusan ilyen például a felhasználó regisztrálásához szükséges adatlap.

models.py – adatbázis struktúrák és kapcsolataik

tests.py – teszt függvények

urls.py – vezérli, mit prezentálunk a weboldalon url minták alapján

views.py – függvényeket tartalmaz. Ezek kezelik a felhasználóval való interakciót

5.2 A projekt követelményeit leíró fájl

A fő mappánkban elhelyezhetünk egy requirements.txt fájlt, amibe feltölthető az összes olyan python -django kiegészítő, amit a felhasználó számítógépére telepíteni kell a projekt megfelelő működéséhez.

Ezt megtehetjük akár kézzel is ha fejből tudjuk mit használtunk, de nagyobb projektek esetén biztonságosabb ezt egy már létező szoftveres eszközre bízni.

Álltalában érdemes minden ilyen python fejlesztéshez egy virtuális környezetet létrehozni, de ez csak egy ajánlás. Abban az esetben, ha létrehoztuk a projekt indulásakor ezt a környezetet használhatnánk a pip modulunk „freeze" parancsát. De ha nincs virtuális környezetünk a számítógépen létező összes telepített modult összegyűjtené, nem csak azokat melyeket a projektünk ténylegesen használ. Erre egy jobb megoldást ajánl a Pipreqs.

5.3 Pipreqs ([8])

Telepítését követően, ezzel a szoftveres eszközzel nagyon könnyedén összegyűjthető az összes projekthez szükséges modul egy „requirements.txt” nevű fájlba. Ennek álltalános menete a következő:

pip install pipreqs

pipreqs /project/mappanév

5.4 Django nézetek személyre szabása ([9])

A Django-ban lehetőségünk van osztály típusú nézetet (Class Based View) vagy függvény típusú nézetet (Function Based View) használni.

Az osztály típusú nézetek egyszerű feladatokat látnak el minimális kód megírásával. A web fejlesztésben felmerülő álltalános szituációkra adnak gyors megoldást például: új objektumok létrehozása, űrlapok kezelésére, lista nézetek, egy adott objektum részletes nézetére stb. Cserébe sok rejtett kód lehet a szülő osztályokban, amiket nem látunk, nehezen olvashatóak ezek a kód részletek és sokféle osztály metódust kell felülírni.

A függvény típusú nézetek könnyen implementálhatók, mert nincsenek előre lefektetett szabályosságok amik az osztályokból adódnak, könnyen olvasható kódot kapunk. Cserébe nem egyszerű újra hasznosítani a kódunkat.

Az egyszerűségük és könnyű személyre szabhatóságuk miatt mindenhol függvény típusú nézetet használtam.

5.5 Django biztonsági intézkedései

A Django keretrendszere nagyszerű megoldásokat nyújt az ártó szándékú felhasználók, programok ellen. ([10])

XSS (Cross site scripting)

Az XSS-támadásoknál a rossz szándékú felhasználók futtatható állományait más felhasználók böngészőibe futtatja. Ezt általában úgy érik el, hogy a rosszindulatú parancsfájlokat eltárolja az adatbázisba, ahol azt más felhasználóknak fogják letölteni és

[8] https://github.com/bndr/pipreqs
[9] https://simpleisbetterthancomplex.com/article/2017/03/21/class-based-views-vs-function-based-views.html
[10] https://docs.djangoproject.com/en/2.2/topics/security/

megjeleníteni, vagy ha a felhasználók rákattintanak egy olyan linkre, amely a felhasználó böngészőjének a támadó JavaScript-jét fogja végrehajtani. Azonban az XSS támadások bármilyen bizalmatlan adatforrásból, mint például a „sütik”-ből vagy a webszolgáltatásokból származnak, ha az adatok nincsenek megfelelően megtisztítva az oldalra való belépés előtt.

A Django sablonok használata megvédi a rendszerünket az XSS támadások többsége ellen.

CSRF (Cross Site Reference Forgery) védelme

A CSRF támadások lehetővé teszik, hogy egy rosszindulatú felhasználó végrehajthassa a műveleteket egy másik felhasználó hitelesítő adataival anélkül, hogy a felhasználó ennek tudatában lenne vagy beleegyezett volna.

A Django beépített védelmet nyújt a legtöbb CSRF-támadás ellen, feltéve, hogy engedélyezve van és használtuk az adatbekérő lapokon a projektben.

SQL injekciós védelem ¶

Az SQL befecskendezés olyan támadás típusa, ahol egy rosszindulatú felhasználó képes tetszőleges SQL-kódot végrehajtani az adatbázisban. Ez azt eredményezheti, hogy a rekordok törlődnek vagy adatszivárgás következik be.

A Django querysettjei védettek az SQL injekciótól, mivel lekérdezéseiket lekérdezési paraméterezéssel állították össze. A lekérdezés SQL-kódja a lekérdezés paramétereitől elkülönítve kerül meghatározásra. Mivel a paraméterek felhasználó által biztosítottak és ezért nem biztonságosak, az alapul szolgáló adatbázis-illesztőprogram eldobja őket.

És még sok más csak felsorolás szintjén: Clickjacking védelem, SSL helyett HTTPS használata, Host fejléc érvényesítése, felhasználó álltal feltöltött tartalmak stb.

5.6 Hogyan reagál a Django, ha működés közben megszűnik a kapcsolat az adatbázissal? ([11])

Ha a kérések feldolgozása során bármilyen adatbázishiba történt, a Django ellenőrzi, hogy a kapcsolat még mindig működik-e, és bezárja, ha nem. Így az adatbázis hibák legfeljebb egy kérést érintenek; ha a kapcsolat használhatatlanná válik, a következő kérés új kapcsolatot kap.

[11] https://docs.djangoproject.com/en/2.2/ref/databases/

6. SQLite, MySQL és a PostgreSQL adatbázis

6.1 SQLite

A Django keretrendszer alapértelmezetten biztosít egy SQLite nevű adatbázist. ([12])

Egyszerű, ám nem alábecsülendő, tökéletesen megfelel egyszerűbb adatbázis modellel rendelkező applikációk kiszolgálására.

Előnyei:

- Külön adatbázis szerver nélkül is működik
- Kis hardware igényű ezért sokféle eszközön használható
- Applikációi könnyen mozgathatók más operációs rendszerekre is, mivel egy fájlban tárolódik
- Opensource license miatt teljesen ingyenes a használata, nincs rajta szerzői jog

Hátrányai:

- Az adatbázis mérete 2GB-ra korlátozott
- Nem támogatja jól a több felhasználós környezeteket, a teljesítménye gyorsan degradálódik, ha ugyanaz az adatbázis egyszerre több klienst is kiszolgál egy időben
- Nincs semmilyen authentikációs rendszere

Ismert applikációk, cégek, amik SQLite adatbázist használnak:

- Adobe
- Bosch
- McAfee
- Skype

6.2 MySQL

A MySQL az egyik legismertebb adatbázis, amit nemrégiben megvett az Oracle. ([13])

Előnyei:

- Opensource license miatt teljesen ingyenes a használata, nincs rajta szerzői jog
- Cross platform, így többféle operációs rendszeren is működik
- Rengeteg kiegészítő alkalmazást használhatunk hozzá az adattáblák kezeléséhez (például: Adminer, HeidiSQL, MySQL Workbench)
- Minimális finomhangolással nagy teljesítményt érhetünk el vele
- Nagyon jó dokumentáció

[12] https://www.linkedin.com/pulse/w.https://www.linkedin.com/pulse/what-difference-between-mysql-sqlite-mahesh-sharma/
[13] https://hackr.io/blog/postgresql-vs-mysql#Summary,

Ismert applikációk, cégek, amik MySQL adatbázist használnak:

- Joomla
- YouTube
- Google
- Twitter

6.3 PostgreSQL

PostgreSQL a másik nagyon ismert relációs adatbázis. ([14])

Előnyei:

- Opensource license miatt teljesen ingyenes a használata, nincs rajta szerzői jog
- Cross platform, így többféle operációs rendszeren is működik
- Nagyon jó dokumentáció
- Sok feladat, kérés hatásos kiszolgálása egyidőben

Ismert applikációk, cégek, amik PostgreSQL adatbázist használnak:

- Apple
- Instagram
- Yahoo!
- Facebook

Általánosságban elmondható, hogy a MySQL ideális lesz egy projekthez, ha nagy biztonságú relációs adatbázist kívánunk használni a webes alkalmazásokhoz vagy egyéni megoldásokhoz, de ha nem szükséges egy teljesen SQL-kompatibilis relációs adatbázis, amely képes gyorsan komplex feladatokat ellátni. Eközben a PostgreSQL ideális lesz a projekthez, ha az igényeink között szerepelnek az összetett eljárások, integráció, bonyolult tervek és adatok integritása köré épülnek, nem pedig a nagy sebesség és könnyű üzembe helyezés. ([15])

A PostgreSQL mellett döntöttem, amiért a támogatott programozási nyelvei között nem csak szerepelt a Python, de a Python fejlesztők közkedvelt adatbázisa.

A Django keretrendszerrel való kommunikációhoz előkövetelmény, hogy a „psycopg2” nevű modul telepítve legyen a számítógépre.

[14] https://hackr.io/blog/postgresql-vs-mysql#Summary,
[15] https://hackr.io/blog/postgresql-vs-mysql#Summary,

7. DBeaver, pgAdmin, TablePlus ([16])

7.1 Álltalános

A DBeaver egy ingyenes univerzális adatbázis eszköz és SQL kliens, amely az Eclipse RCP platformon alapul.

A pgAdmin az egyik legnépszerűbb PostgreSQL adatbázis tervezési és kezelési eszköz.

A TablePlus egy modern, natív GUI kliens, amely lehetővé teszi a fejlesztők és a DBA-k számára, hogy egyszerre több adatbázist is elérhessenek, lekérdezzenek, szerkesszenek és kezelhessenek.

7.2 Operációs rendszerek támogatása

A DBeaver sok olyan platformon működött, amelyeket korábban az Eclipse keretrendszer támogatott (Windows, Linux, MacOS, Solaris, AIX, HP-UX). A 4.2-es verziótól kezdve azonban csak a Windows, MacOS és Linux támogatására korlátozta támogatását.

A legutóbbi pgAdmin 4-frissítési csomagja óta, a pgAdmin webes alkalmazássá vált, amely bármilyen számítógépen futtatható egy webböngészővel.

Kezdetben a TablePlus egy Mac natív verzióval kezdődött. A népszerű keresletnek köszönhetően a közelmúltban bevezetésre került egy natív Windows verzió.

7.3 Árazás

A DBeaver szabad és nyílt forráskódú közösségi verzióval rendelkezik (DBeaver CE). Van egy vállalati kiadás (DBeaver EE) a vállalati ügyfelek számára, akik több driver támogatást, fejlettebb funkciókat és dedikált ügyféltámogatást kapnak.

A pgAdmin egy ingyenes és nyílt forráskódú eszköz. Bárki letöltheti a pgAdmin-t a webhelyéről és ingyenesen használhatja.

A TablePlus alapvető verziója ingyenes és nincs időbeli korlátozása. Teljes funkciókkal rendelkezik. Az egyetlen korlátozás az, hogy egyszerre legfeljebb 2 nyitó fül / csatlakozás / szűrő használható. A teljes licensz 49 USD-be kerül, ami eltávolítja az összes korlátozást.

[16] https://tableplus.io/blog/2018/10/dbeaver-vs-pgadmin-vs-tableplus.html

7.4 Összegzés

DBeaver

Előnyök:

- Cross-platform
- Több illesztőprogram támogatása
- Színes kapcsolat ábrák
- Az entitás-kapcsolatok diagramjai rendelkezésre állnak
- Beépített SQL formázás

Hátrányok:

- Ez egy Java virtuális gépen fut, rengeteg memóriát használ futás közben.
- Zavaros ikontervezés és nem intuitív. Néha nem tudja az ember, hogy hol keresse meg, amire szüksége van.

pgAdmin

Előnyök:

- Ingyenes.
- Könnyen telepíthető bármely kiszolgálóra, és távolról is hozzáférhető bármely webböngésző segítségével.
- A felhasználói felület rugalmas, és újrarendezhető. A felhasználói felület elemei leválasztható panelek, amelyeket át lehet húzni és újra elhelyezni, hogy önállóan vagy a lapozott böngészőben megjelenjenek.

Hátrányok:

- Web - alapú.
- Csak a PostgreSQL-t támogatja.
- Lassan válaszol parancsokra.
- Lassú elindítani.
- Sok erőforrást használ

TablePlus

Előnyök:

- Natívan Mac operációs rendszerre fejlesztve. Gyors, könnyű és stabil. Kevesebb, mint fél másodperc alatt el tud indulni.
- Tiszta és egyszerű felület.
- Több feltételes adatszűrő
- A lekérdezések előzményei elérhetőek egy egyszerű listából
- Az aszinkron betöltések a lekérdezések eredményeinek gyorsabb megjelenítéséhez vezetnek anélkül, hogy blokkolná a felhasználói felületet.
- A táblázatok és struktúrák nagyon gyors szerkesztése.
- Intelligens lekérdezés szerkesztő kiemelt szintaxissal, azonnali automatikus kiegészítéssel, SQL formázással.
- A lekérdezések szerkesztése közvetlenül.
- Egy plugin rendszer, amely kiterjeszti az alkalmazást az alapfunkciókon túl.
- Minden funkcióhoz hozzárendelt gyorsbillentyű tartozik.
- Gyors fejlődés. Új verzió kiadása hetente.

Hátrányok:

- Nincs Linux verzió.
- A felhasználó kezelés jelenleg nem támogatott
- Nincs ERD (Prosa Entity Relationship Diagram)-diagram, ami azonos az adatbázis modellek UML diagramjával.

Szakdolgozatom elkészítéséhez a pgAdmin mellett döntöttem. Egyértelmű döntés volt annak alapján, hogy a PostgreSQL relációs adatbázist választottam az oldal adatainak tárolására. Habár kipróbáltam a DBeaver-t is a pgAdminnal mindig egyszerűbb volt dolgozni.

8. Pycharm és Visual Studio Code ([17])

Programozáshoz elegendő lenne akár egy notepad++ is, de azért ez összetettebb feladatok esetén puritán megoldásnak számít. Sokban elősegíti a fejlesztést egy komolyabb IDE.

8.1 Pycharm

Támogatja a Python, JavaScript, CSS nyelveket.

Létezik ingyenes „közösségi" és fizetős „professzionális" kiadása is. A közösségi verzió is rendelkezik mindazokkal a kényelmi funkciókkal, amik nagyban megkönnyítik a fejlesztési munkálatokat, mint például a kód kiegészítés, hiba felismerés, intelligens kód újraírás.

Pycharm előnyei:

- Python fejlesztők első számú választása
- Sokkal jobb Git integrációval rendelkezik, lehet alap Git parancsokat kiadni vagy akár fájlok korábbi commit előtti változatait megtekinteni
- Pip integráció. Amikor benyitunk egy projektet, ellenőrzi, hogy a „requirements.txt" fájlban megadott kiegészítők közül hiányzik e bármi és felajánlja a telepítésüket is.
- Kétféle automatikus szövegkiegészítője is van: strukturális és szó kiegészítő. A strukturális a megfelelő python objektum szerkezetek alapján tesz ajánlásokat azok részeinek kiegészítésére. A szó kiegészítő pedig a már előbb felhasznált változók, függvények, osztályok saját függvényei vagy változói stb. neveit tudja kiegészíteni.

8.2 Visual Studio Code

A Visual Studio Code az egyik legnépszerűbb fejlesztői környezet ha Javascript programozással foglalkozik valaki, de sok más nyelvet is támogat.

Visual Studio Code előnyei:

- JavaScript fejlesztők első számú választása
- Teljesen ingyenes
- Tökéletesen felszerelt mindenféle funkcióval telepítés után, még ha nincs is túl sok később telepíthető plugin hozzá
- Havonta érkezik új verzió frissítés hozzá

A szakdolgozathoz a Pycharm fejlesztői környezetet választottam, főleg azért is mert a jetbrains cég termékeit jól ismerem és python fejlesztéshez mindenhol csak ajánlották.

[17] https://www.slant.co/versus/1240/5982/~pycharm_vs_visual-studio-code

9. Github és GitKraken

9.1 Github([18])

A GitHub megértéséhez először meg kell értenünk a Git-et. A Git egy nyílt forráskódú verzióvezérlő rendszer, amelyet Linus Trovalds indított - ugyanaz a személy, aki a Linuxot hozta létre. A Git hasonlít a többi verzióvezérlő rendszerhez – mint például a Subversion, a CVS és a Mercurial.

Szóval, a Git egy verzióvezérlő rendszer, de mit jelent ez? Amikor a fejlesztők létrehoznak valamit (például egy alkalmazást), folyamatosan változtatják a kódot, új verziókat bocsátanak ki az első hivatalos (nem béta) kiadásig és után.

A verziók vezérlőrendszerei ezeket a módosításokat tisztán tartják, és a módosításokat egy központi tárolóban / repóban tárolják. Ez lehetővé teszi a fejlesztők számára, hogy könnyen együttműködhessenek, mivel letölthetik a szoftver új verzióit, módosíthatják és feltölthetik a legújabb verziót. Minden fejlesztő láthatja ezeket az új változásokat, letöltheti és hozzáadhat újakat.

Hasonlóképpen, azok, akiknek semmi köze a projekt fejlesztéséhez, még mindig letölthetik a fájlokat és használhatják azokat. A legtöbb Linux-felhasználónak ismernie kell ezt a folyamatot, mivel a Git, a Subversion vagy más hasonló módszerek használata eléggé gyakori a szükséges fájlok letöltéséhez - különösen a program forráskódból történő összeállításának előkészítése során.

A Git a legtöbb fejlesztő előnyben részesített verziószabályozó rendszere, mivel több előnye van a többi rendelkezésre álló rendszerrel szemben. A fájlmódosításokat hatékonyabban tárolja, és a fájl integritását jobban biztosítja.

9.2 A „Hub” a GitHub-ban

Megállapítottuk, hogy a Git egy verzió követő rendszer, amely hasonló, de jobb, mint a sok rendelkezésre álló alternatíva. Szóval, mi teszi különlegesnek a GitHubot? A Git egy parancssori eszköz, de az a centrum, amely minden Git-et összefog, a hub - GitHub.com -, ahol a fejlesztők tárolják projektjeiket és hasonló gondolkodású emberek hálózataival találkozhat.

[18] https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/

9.3 Mi az a Repository (röviden Repó)

Ez egy tároló, mely egy olyan hely, ahol az adott projekt összes fájlja tárolódik. Minden projektnek saját repója van, és egyedi URL-címmel férhet hozzá.

9.4 Mi az a Clone

Amikor egy repót első alkalommal lementünk a szerverről azt Clone-nak / klónozásnak nevezzük.

9.5 Mi az a Pull / Push

Amikor a meglévő repóba frissítéseket szeretnénk a szerverről lekérni akkor Pull, ha pedig a saját változtatásainkat szeretnénk feltölteni Push parancsokat adhatunk ki.

Mindenképp szükségesnek láttam egy verzió követő rendszer használatát. A fejlesztést nagyban elősegíti, ha mindig van, egy mentési pontja az embernek ahova akármikor visszatérhet, ha egy új fejlesztési ág nem sikerül.

Lehet parancssorból is kezelni, de az átláthatóság kedvéért valamilyen git klienst szokás telepíteni. A munkahelyemen a smartGIT –et használjuk. ezért a legfőbb verseny társát a GitKrakent választottam.

GitKraken rendelkezik ingyenesen használható és fizetős verzióval is. Az ingyenes változat minden elvárást kielégít egy ilyen kis projekt esetében.

10. Materialize és a Bootstrap ([19])

Mind a Bootstrap, mind a Materialize egy Frontend keretrendszer;

A Bootstrap egy HTML, CSS és JavaScript keretrendszer, míg a Materialize a Google Material dizájnján alapuló CSS keretrendszer.

A fő különbség az, hogy a Bootstrap több szabadságot és irányítást biztosít az UX elemek felett; A Materialize inkább arról szól, hogy az UX elemek hogyan viselkedjenek és nézzenek ki.

Miért választják a fejlesztők a Bootstrap vagy a Materialize világát:

[19] https://www.quora.com/Which-is-the-best-option-Bootstrap-or-Materialize-CSS

- A Bootstrap közismerten gyors, következetes és rugalmas. Széles körben használják és jól dokumentáltak.
- A Materialize –ot a Google Material dizájnjának rajongói értékelik inkább. Közismerten gyors, könnyen használható és jól dokumentált.

Weboldalak, akik a Bootstrap-et használják:

Spotify.com, twitter.com, udemy.com, docker.com

Weboldalak, amik a Materialize-ot használják:

Nimbusnine.co, Angular.io, Android.com

Amikor a Materialize-ot választottam pusztán az egyszerűsége a színes dizájnja fogott meg.

11. CKeditor ([20])

Ez a kiegészítő a fájl feltöltés és különböző richtext beviteli eszközök támogatására ad megoldást. Django admin felületébe a CKEditor integrációja a következő lehetőségeket nyitja meg a projektben:

RichTextField – lehetővé teszi, hogy formázást is elmenthessünk nem csak sima szöveget

RichTextUploadingField – fájl feltöltő mező

CKEditorWidget – gombok szöveg formázásához, táblázatok létrehozásához stb.

CKEditorUploadingWidget - fájl feltöltő modul

12. Graphviz([21])

A Django MVN – Modell Nézet Vezérlő tervezési mintája miatt az adatbázis terve nagyon fontos és elsődleges. Mivel a Django keretrendszere alapból tartalmaz egy admin és a hozzá tartozó kiegészítőinek adatbázisát fontosnak tartottam, hogy egy az egész rendszert bemutató UML leírást kapjunk.

Használatának előkövetelménye, hogy a „django extensions” nevű modul is fel legyen telepítve a számítógépre.

[20] https://django-ckeditor.readthedocs.io/en/latest/index.html?highlight=image#
[21] https://www.graphviz.org/

Ebben a Graphviz nevű python modul állt rendelkezésre. Miután integráltam a projektbe, mint installált kiegészítőt, gyorsan el is készült a teljes UML ábra, amit az adatbázis tervben prezentálok. Az eredeti kép pedig megtalálható a projekten belül az \atlas\my_project_visualized.png néven.

13. Felhasznált eszközök és verzióik

Python 3.7.2

Django 2.1.5

django_ckeditor 5.6.1

Pillow 5.4.1

psycopg2 2.7.7

django-extensions 2.1.6

Pycharm 2018.2.1 community edition

GitKraken 5.0.4

graphviz 2.38

IV. Eredmények bemutatása, elemzése és megvitatása

1. Adatbázis bemutatása

Hűen a Django álltál használt MVN – Modell Nézet Vezérlő tervezési mintához az adatbázis terve volt a legfontosabb, mivel ez lett a szíve – lelke a projektnek.

1.1 A könyvet leíró adattáblák

Chapter tábla: a könyv fejezeteire vonatkozó adatokat tartja nyilván.

id: a fejezet azonosítója. Számtípus, kulcs, nem lehet azonos, kitöltése kötelező.

title: a fejezet címe. Karakter típusú, lehet azonos, kitöltése kötelező.

summary: a fejezet rövid leírása. Karakter típusú, lehet azonos, kitöltése nem kötelező.

Subheading tábla: a könyv fejezeteinek alcímei és azok tartalma

id: a fejezet azonosítója. Számtípus, kulcs, nem lehet azonos, kitöltése kötelező.

chapter_id: a hozzátartozó fejezet azonosítója. Számtípus, kulcs, nem lehet azonos, kitöltése kötelező.

title: az alcím. Karakter típusú, lehet azonos, kitöltése kötelező

content: az alcím tartalma. Karakter típusú, lehet azonos, kitöltése nem kötelező.

1.2 A felhasználókat leíró adattábla

Django álltalános user tábla: a felhasználók alap adatait tartja nyilván. A keretrendszer már meglévő felhasználó adatbázisának lehetséges mezőiből válogatott tulajdonságok a következők:

id: a felhasználó azonosítója. Számtípus, kulcs, nem lehet azonos, kitöltése kötelező.

password: a jelszó. Karakter típusú, lehet azonos, kitöltése kötelező.

username: felhasználónév. Karakter típusú, lehet azonos, kitöltése kötelező.

first_name: a felhasználó keresztneve. Karakter típusú, lehet azonos, kitöltése kötelező.

last_name: a felhasználó utóneve. Karakter típusú, lehet azonos, kitöltése kötelező.

email: a felhasználó e-mail címe. Karakter típusú, nem lehet azonos, kitöltése kötelező.

automatikusan töltött és használt adatmezők minden felhasználóhoz:

last_login: utolsó belépés időpontja. Időbélyeg típusú.

is_superuser: a felhasználó admin jogát határozza meg. Bool típusú.

is_staff: azt mutatja meg, hogy a felhasználó elérheti e az admin oldalt. Bool típusú.

is_active: azt mutatja meg, hogy aktív e még a felhasználó fiók. Felhasználók törlése helyett az ajánlás az, hogy ezt a flag-et False értékre állítjuk. Így nem fog problémát okozni, ha vannak olyan adatbázis kulcsok, amik erre a felhasználóra mutatnak. Bool típusú.

date_joined: azt mutatja meg mikor jött létre a felhasználó. Időbélyeg típusú.

Profile tábla: a felhasználókat tartja nyilván. Csak kiegészítő tábla egy az egyhez kapcsolattal. A Django keretrendszer eredeti user tábláját egészítjük csak ki olyan információkkal, ami nem szükséges a felhasználó beléptetéséhez.

id: a profil azonosítója. Számtípus, kulcs, nem lehet azonos, kitöltése kötelező.

user_id: a django user azonosítója, akihez a profil tartozik. Számtípus, kulcs, nem lehet azonos, kitöltése kötelező.

birth_date: a felhasználó születési dátuma. Dátum típusú, lehet azonos, kitöltése nem kötelező.

division: a felhasználó osztályának azonosítója. Karakter típusú, lehet azonos, kitöltése nem kötelező.

bookmark: könyvjelző, ami meghatározza hányadik fejezet kvíz játékát oldotta már meg a felhasználó. Számtípus, lehet azonos, kitöltése kötelező.

1.3 A kvízjátékhoz tartozó adattáblák

Exam tábla: a kvízjátékok összefoglaló táblája egy adott fejezethez.

id: a kvíz azonosítója. Számtípus, kulcs, nem lehet azonos, kitöltése kötelező.

chapter_id: a hozzátartozó fejezet azonosítója. Számtípus, kulcs, nem lehet azonos, kitöltése kötelező.

name: a kvíz neve. Karakter típusú, lehet azonos, kitöltése kötelező.

summary: a kvíz rövid leírása. Karakter típusú, lehet azonos, kitöltése kötelező.

goal: az a ponthatár aminél már sikeresnek számít a megoldás. Számtípus, lehet azonos, kitöltése kötelező

Question tábla: a kvíz kérdések összefoglaló táblája egy adott kvízhez

id: a kérdés azonosítója. Számtípus, kulcs, nem lehet azonos, kitöltése kötelező.

exam_id: a hozzátartozó kvízjáték azonosítója. Számtípus, kulcs, nem lehet azonos, kitöltése kötelező.

text: a kérdés szövege. Karakter típusú, lehet azonos, kitöltése kötelező.

amount: a kérdés pontszám értéke. Számtípus, kulcs, nem lehet azonos, kitöltése kötelező.

Answer tábla: a kvíz válaszok összefoglaló táblája egy adott kérdéshez

id: a válasz azonosítója. Számtípus, kulcs, nem lehet azonos, kitöltése kötelező.

question_id: a hozzátartozó kérdés azonosítója. Számtípus, kulcs, nem lehet azonos, kitöltése kötelező.

text: a válasz szövege. Karakter típusú, lehet azonos, kitöltése kötelező.

is_valid: a válasz helyességét határozza meg. Bool típusú, lehet azonos, kitöltése nem kötelező

Examlog tábla: a kvíz résztvevőinek naplózó táblája

id: a napló azonosítója. Számtípus, kulcs, nem lehet azonos, kitöltése kötelező.

exam_id: a hozzátartozó kvízjáték azonosítója. Számtípus, kulcs, nem lehet azonos, kitöltése kötelező.

participant: a résztvevő felhasználóneve. Karakter típusú, lehet azonos, kitöltése kötelező.

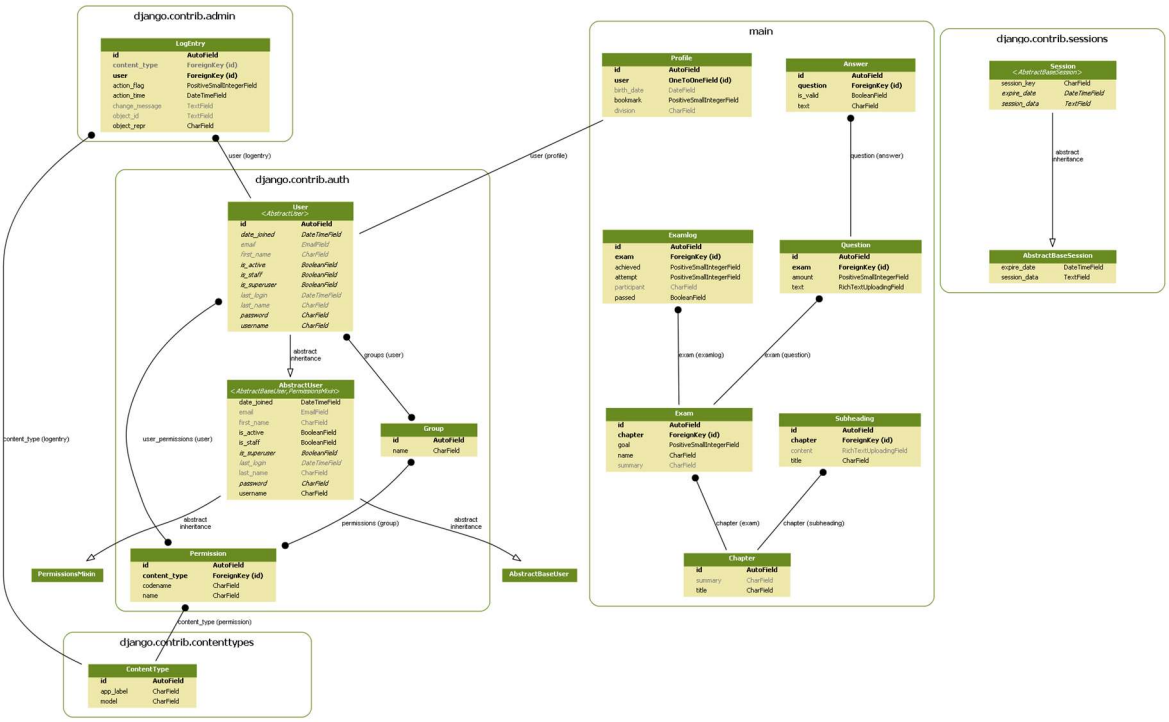
achieved: az elért pontok száma. Számtípus, lehet azonos, kitöltése kötelező.

passed: a felhasználó eredményességét határozza meg. Bool típusú, lehet azonos, kitöltése nem kötelező

attempt: a próbálkozások számát határozza meg. Számtípus, lehet azonos, kitöltése kötelező.

Ezek a táblák a \main\ models.py fájlban találhatók. A kapcsolatok úgy lettek beállítva, hogy ha egy szülő objektum törlésre kerül az összes vele kapcsolatban levő elem is törlődik. Ilyen folyamatot idézünk elő, ha például kitörlünk egy fejezetet. A hozzá tartozó összes alcím, kvízjáték (kvíz kérdések, kvíz válaszok) és a naplózó is törlődni fog.

Az adatbázis táblákhoz tartozó UML ábra:



2. Nézetek

A Django álltál használt MVN – Modell Nézet Vezérlő tervezési mintája szerint a következő pont a nézetek, amikkel a felhasználó interakcióba lép.

homepage

Ez a főoldal szolgáltatásáért felelős nézet. Itt listázzuk ki a könyv fejezeteit és a navigációhoz szükséges gombokat és információval szolgálunk arról is melyik felhasználó van éppen belépve az oldalon.

A hozzá tartozó oldal a \templates\main\home.html.

detail

Ez a nézet főoldalon kiválasztott fejezet tartalmát jeleníti meg.

A hozzá tartozó oldal a \templates\main\detail.html.

quiz

Ez a nézet a fejezetekhez tartozó összes kvízt kilistázza. Abban az esetben, ha a felhasználó még nem jelentkezett be, felhívjuk erre a figyelmét és átirányítjuk a bejelentkező oldalra.

A hozzá tartozó oldal a \templates\quiz.html.

examdetail

Ez a nézet a kiválasztott kvíz kérdéseit és azok válaszait jeleníti meg. A válaszok között rádió gombokkal tippelhetjük meg a helyes választ. Minden kérdésnél egy helyes válasz létezik. A kérdés sor végén a „Bead" gombra kattintva lehet jóváhagyni a választásainkat.

A hozzá tartozó oldal a \templates\examdetail.html.

quizdone

Ez a nézet dolgozza fel a beadott kvíz feladványokat. Abban az esetben, ha legalább annyi pontot sikerült szereznie az illetőnek, mint a kvízhez megjelölt ponthatár sikeresnek minősítjük és elmentjük a naplóba.

Ha nem sikerült elérni a megadott pontszámot, akkor is elmentjük csak sikertelen minősítéssel.

Ennek a nézetnek nincs dedikált megjelenítendő oldala, inkább csak a logikai számítást és az adatbázis frissítését végzi. Végül átirányítja a felhasználót a \templates\ results.html oldalra.

results

A quizdone nézetben kiszámolt adatokkal megjeleníti a kvíz ponthatárát, az ebből elért pontokat és azt, hogy sikeres volt e vagy sem a játék.

A hozzá tartozó oldal a \templates\results.html.

account

Ez a nézet a már bejelentkezett felhasználó adatait jeleníti meg.

A hozzá tartozó oldal a \templates\account.html.

register

Ez a nézet a felhasználók regisztrációjáért felelős. Mivel ez egy biztonság kritikus eljárás felhasználásra került a django regisztrációs sablon a CSRF (Cross Site Reference Forgery) védelem és a beírt adatok megtisztítása is.

A hozzá tartozó oldal a \templates\register.html.

logout_request

Ennek a nézetnek nincs dedikált megjelenítendő oldala, csak azt a célt szolgálja, hogy kijelentkeztesse az aktuális felhasználót. Majd átirányítja a fő oldalra: \templates\homepage.html.

login_request

Ez a nézet a felhasználó beléptetéséért felel. Mivel ez egy biztonság kritikus eljárás felhasználásra került a django regisztrációs sablon a CSRF (Cross Site Reference Forgery) védelem és a beírt adatok megtisztítása is. Ha sikerül megfeleltetni egy felhasználót a megadott adatok alapján akkor belépteti a rendszerbe és átirányítja a fő oldalra: \templates\homepage.html.

A hozzá tartozó oldal a \templates\login.html.

Ezek a nézetek a \main\views.py fájlban találhatók.

3. URL-ek

A Django álltál használt MVN – Modell Nézet Vezérlő tervezési mintája szerint a következő pont a vezérlők, amikkel a felhasználó bejárhatja a teljes oldalt.

Elérési út: "", meghívja a homepage nézetet.

Elérési út: "quiz/", meghívja a quiz nézetet.

Elérési út: "register/", meghívja a register nézetet.

Elérési út: "logout/", meghívja a logout_request nézetet.

Elérési út: "login/", meghívja a login_request nézetet.

Elérési út: "account/", meghívja a account nézetet.

Elérési út: "<int:chapter_id>/", meghívja a detail nézetet.

Elérési út: "quiz/<int:exam_id>/", meghívja a examdetail nézetet.

Elérési út: "quiz/<int:exam_id>/quizdone", meghívja a quizdone nézetet.

Elérési út: "quiz/<int:exam_id>/results/", meghívja a results nézetet.

Ezek az url címek a \main\urls.py fájlban találhatók.

4. Admin oldal

A beépített admin oldal elérésére a következő url címet használhatjuk: „/admin”

Ezen az oldalon minden eddig felsorolt tartalom szabadon szerkeszthető a belépés után.

5. Elért eredmények

A felhasználókat biztonsággal lehet regisztrálni, ki- és beléptetni.

Az admin oldalon szabad kézzel szerkeszthetjük az oldal tartalmát és felhasználóit. Az oldal könnyen modulálható. A csillagászatban még mindig fejlődő tudomány, javíthatunk mostani kijelentéseken vagy akár későbbi felfedezéseket, képeket, érdekességeket adhatunk hozzá a tananyaghoz.

A weboldal sikeresen meg tudja jeleníteni a könyv fejezeteit és azokhoz rendelt kvíz játékait. A Kvíz játékokat amennyiben a felhasználó bejelentkezett el lehet végezni és erről eredmény visszajelzést is kap.

A weboldal megjelenítése sok tekintetben dinamikus. Az ablak újra méretezése esetén a szövegek is átméreteződnek. A navigációs sáv menüpontjai egy mobil alkalmazásoknál már megszokott gombba csoportosulnak, melyet meg kattintva egy oldal menüből navigálhatunk tovább.

6. Telepítési útmutató

1. Windows környezet esetén futtassuk le az alábbi telepítőket a mellékelt CD meghajtó \install\ könyvtárából, ebben a sorrendben:
   - python-3.7.2-amd64.exe
   - postgresql-11.1-1-windows-x64.exe
     a. telepítéskor a következő adatokat adjuk meg az adatbázis szerverének: 'USER': 'postgres',

        'PASSWORD': 'postgres',

        'HOST': '127.0.0.1', (alapértelmezett)

        'PORT': 5432 (alapértelmezett)
   - pgadmin4-4.3-x86.exe
2. Másoljuk le a számítógépünkre a mellékelt CD meghajtóról a teljes \atlas könyvtárát. Semmiképp ne változtassunk a nevén, tartsuk azt meg „atlas”-nak.
3. Alternatív megoldásként, ha van Git telepítve a számítógépre ellátogathatunk a Github -ra is és klónozhatjuk vagy direkt módon egy csomagolt fájlban letölthetjük a projektet erről a linkről:

gitrepo: https://github.com/paliccolo01/astronomy_atlas_django

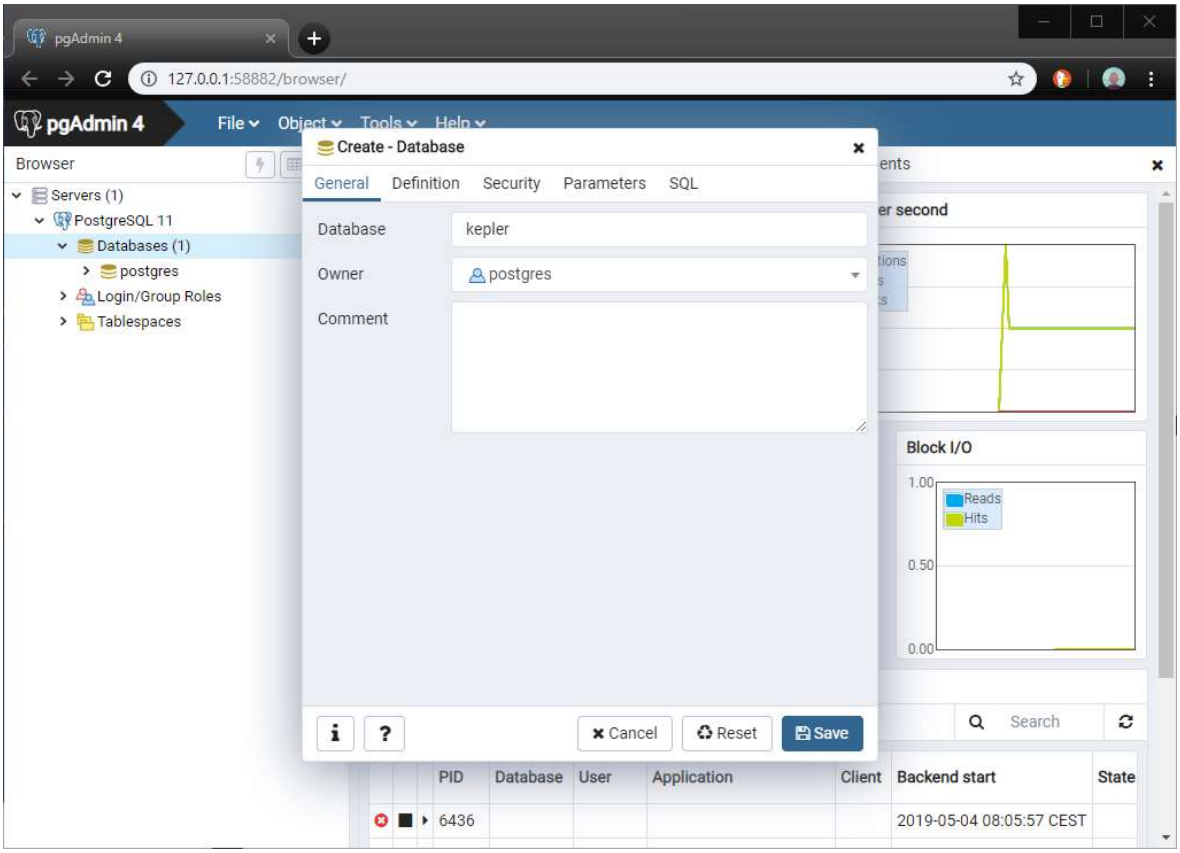
4. Ha ezzel megvagyunk nyissunk egy parancssort és váltsunk az \atlas\ mappában, ahol a „manage.py” fájl is található. Ugyan a python-t frissen telepítettük, de a pip modult be kell frissítenünk ezzel a paranccsal:

python -m pip install --upgrade pip

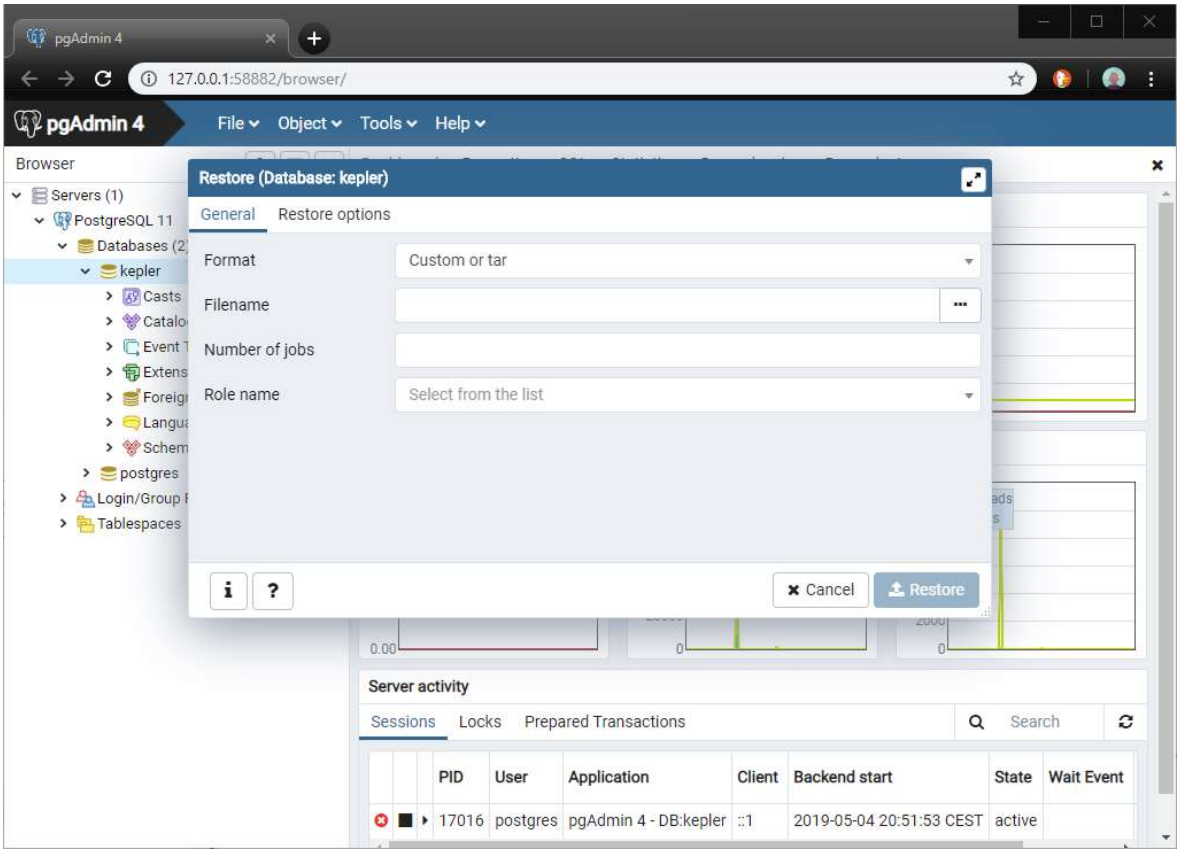
5. Majd telepítsük fel az összes projekthez szükséges python modult ezzel a paranccsal: pip install -r requirements.txt

Hogyan telepítsük a példa adatbázist a CD \test database\ mappából:

6. Indítsuk el a startmenüből a pgAdmin4 v4-et
7. A databases -re jobb egér gombbal kattintva a create database opciót válasszuk és adjuk meg a nevét: „kepler”.



8. Kattintsunk a Save gombra.
9. Most a kepler adatbázisra kattintsunk jobb egér gombbal és válasszuk a restore opciót



10. A filename-re kattintva tallózzuk be a CD meghajtóról a \test database\kepler_backup nevű fájlt
11. Kattintsunk a Restore gombra.

Készen is vagyunk! A parancssorban az \atlas mappában állva, ahol a „manage.py” fájl is létezik adjuk ki a következő parancsot:

python manage.py runserver

látogassunk el a kedvenc web böngészőnkkel erre a címre: 127.0.0.1:8000/

Ha befejeztük a futtatást vagy valami hiba lépett fel és szeretnénk újra kezdeni a parancssorban nyomjuk le a CTRL+C billentyű kombinációt. Ez leállítja a webservert.

A példa adatbázis admin felhasználó adatai:

Felhasználónév: admin

Jelszó: Subaruwrxsti01

Az adatbázisban szereplő összes felhasználónak ugyanez a jelszava. Belépéshez a felhasználóneveik:

moni, pal, ksriszti.

A kvízjátékok könnyű tesztelése érdekében az összes esetében az első a helyes válasz.

7. Eltávolítási útmutató

A windows (control panel -> add/remove programs) vagy linux(octopi / pamac / yast) rendszerünkben megfelelő helyen távolítsuk el a következőket:

- python. Ezzel együtt az összes Django releváns kiegészítők is távozni fognak a számítógépről
- pgadmin4
- postgreSQL

V. Következtetések

A programot nem sikerült teljesen befejeznem. A felhasználók account oldalán meg szerettem volna jeleníteni, a kvíz naplózó rák eső részét.

A felhasználók könyvjelző és a kvíz naplózó próbálkozási számát nem használom jól, vagy semmilyen hatása nincs jelenleg a projektben.

Felhasználói fiók létrehozása esetén, ha valaki a profil mezőiben elgépel valamit, a rendszer akkor is azt a hibaüzenetet küldi, mintha a két megadott jelszó nem egyezne meg. Ez egyszerűen azért van mert még nem írtam meg rá a hibakezelést.

Az admin oldal külön fejlesztést igényel, szinte ugyanannyi idő lenne a megjelenítését átalakítani, és a funkcionalitását kibővíteni mint a jelenlegi oldal elkészítése. Ezen az oldalon sem tudunk egyértelmű lekérdezéssel megnézni a tanulók hogyan teljesítettek a kvízjátékokon.

A weboldal sok helyen angol maradt. Ez annak köszönhető, hogy megszokásból mindent angolul fejlesztettem és csak később kezdtem el átírni a megjelenítésben szereplő szövegeket.

1. Néhány fejlesztési lehetőség

A weblap képeinek feltöltése már nem fért bele az időbe, de mellékeltként hivatkozásokkal együtt megtalálható a CD-n ebben a mappában: \documentation\pictures

Django admin felületét ki lehet bővíteni funkciókkal és át is lehet alakítani a hivatalos dokumentáció alapján

Kvíz kérdések kibővítése több jó válasz lehetőséggel. Checkbox-okkal könnyedén megoldható.

Test fájlok írása, hogy biztosak lehessünk benne jól működnek a funkcióink.

VI. Hivatkozott irodalmak jegyzéke

https://django-ckeditor.readthedocs.io/en/latest/index.html?highlight=image#

https://djangopackages.org/packages/p/django-quiz-app/

https://docs.djangoproject.com/en/2.2/ref/databases/

https://docs.djangoproject.com/en/2.2/topics/security/

https://github.com/bndr/pipreqs

https://github.com/tomwalker/django_quiz

https://hackr.io/blog/postgresql-vs-mysql#Summary,

https://hu.wikipedia.org/wiki/Modell-n%C3%A9zet-vez%C3%A9rl%C5%91

https://pypi.org/project/simple-quiz/

https://pythonclock.org/

https://simpleisbetterthancomplex.com/article/2017/03/21/class-based-views-vs-function-based-views.html

https://tableplus.io/blog/2018/10/dbeaver-vs-pgadmin-vs-tableplus.html

https://www.airpair.com/python/posts/django-flask-pyramid

https://www.freeprojectz.com/paid-projects/django-python-mysql/employee-online-quiz-system

https://www.graphviz.org/

https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/

https://www.linkedin.com/pulse/w.https://www.linkedin.com/pulse/what-difference-between-mysql-sqlite-mahesh-sharma/

https://www.quora.com/Which-is-the-best-option-Bootstrap-or-Materialize-CSS

https://www.slant.co/versus/1240/5982/~pycharm_vs_visual-studio-code

VII. Mellékletek

1 db CD a program telepítőjével

1 db Összefoglaló leírás

NYILATKOZAT

az írásmű eredetiségéről

(*PTE SZMSZ 5.sz. mellékletének 14/1. számú melléklete alapján*)

Alulírott

…….(név)

…….(NEPTUN kód), büntetőjogi felelősségem tudatában kijelentem, hogy .....

.....

.....

című írásomban foglaltak saját, önálló munkám eredményei, ennek elkészítéséhez kizárólag a hivatkozott forrásokat (szakirodalom, eszközök stb.) használtam fel, írásomat a Pécsi Tudományegyetem vonatkozó szabályzatainak betartásával készítettem. Tudomásul veszem, hogy a szerzői jogi szabályok betartását a Pécsi Tudományegyetem plágiumkereső rendszeren keresztül ellenőrizheti.

Pécs, 201……év……hó .....nap

…….……

Hallgató aláírása