Pannon Egyetem

Műszaki Informatikai Kar

Matematika Tanszék  
Programtervezező Informatikus

**SZAKDOLGOZAT  
United Unies**

**Rentler Ferenc**

Témavezető: Pozsgai Tamás

Külső/belső konzulens: Pozsgai Tamás

2023

Témakiírás



Hallgatói nyilatkozat

Alulírott Rentler Ferenc hallgató kijelentem, hogy a dolgozatot a Pannon Egyetem Matematika Tanszékén készítettem a Programtervező Informatikus végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozatban lévő érdemi rész saját munkám eredménye, az érdemi részen kívül csak a hivatkozott forrásokat (szakirodalom, eszközök stb.) használtam fel.

Tudomásul veszem, hogy a dolgozatban foglalt eredményeket a Pannon Egyetem, valamint a feladatot kiíró szervezeti egység saját céljaira szabadon felhasználhatja.

Dátum: Szántód, 2023.04.21

*Rentler Ferenc*

Témavezetői nyilatkozat

Alulírott Pozsgai Tamás témavezető kijelentem, hogy a dolgozatot *Rentler Ferenc* a Pannon Egyetem Matematika Tanszék én készítette Programtervező Informatikus végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozat védésre bocsátását engedélyezem.

Dátum: Veszprém, 2023.

*Pozsgai Tamás*

Köszönetnyilvánítás

Szeretném megköszönni témavezetőmnek és konzulensemnek, Pozsgai Tamásnak az hasznos tanácsokat és ötleteket amiket a félév folyamán adott a siker szakdolgozat elkésztítése érdekében.

Továbbá köszönetet szeretnék mondani a szüleimnek és a páromnak, akik tanulmányaim során végig segítettek és ösztönöztek az egyetemi tanulmányaim és a szakdolgozatom elkészítése folyamán.

Végül szeretném megköszönni barátaimnak és hallgató társaimnak a sok hasznos tanácsot és élményt amit az egyetemi éveim alatt szereztem.

Tartalmi összefoglaló

A feladat:

Egy weboldal készítése amely magába foglalja a köszöntő oldalán a karok történetét érdekesebb információkat, segít az egyetem területén tájékozódni térkép es útvonal tervezéssel illetve a különböző fő egyetemi épületek bejáratának megjelölésével illetve rövid tájékoztatóval az adott épületről és ezenfelül bemutatja Veszprém és környéke érdekességeit szintén térképes helymegközelítéssel, lehetőséget nyújt a diákoknak regisztrációra majd bejelentkezésre ezzel lehetővé teszi az órarend felöltését amit neptunból letöltve tud importálni és a törlésig megtekintheti google callendar segítségével, illetve regisztráció után lehetőséget is biztosít a későbbi adat módosításra ha változnának felhasználó adatai.

Tartalmi összefoglaló magyarul. Az összefoglalónak tartalmaznia kell (rövid, velős és összefüggő megfogalmazásban) a következőket:

* téma megnevezése,
* megoldott feladat megfogalmazása,
* megoldási mód,
* elért eredmények,
* kulcsszavak (4-6 darab)
* terjedelme nem lehet több 1 A4-es oldalnál.

Az összefoglalót magyar és angol nyelven kell készíteni. Sorrendben a dolgozat nyelvével megegyező kerül előrébb. A cím Title stílusú, formázása: Times New Roman, nagybetű, 14 pt, félkövér, középre igazított; az összefoglaló Normál stílusú, formázása: Times New Roman, 12 pt, sorkizárt, 1.5-ös sortávolság.

**Kulcsszavak:** [weblap, elhelyezkedés, elérhető, felhasználók]

Abstract

Abstract in English

**Keywords:** [list 4-6 keywords]

Tartalomjegyzék

[Jelölésjegyzék 9](#_Toc130553983)

[Adatbázisok kialakulása: 10](#_Toc130553984)

[1.1. Az adatbázisok története : 10](#_Toc130553985)

[1.2. Az adatbázis definíciója: 10](#_Toc130553986)

[1.3. Adatbáziskezelő rendszerekkel szemben támasztott követelmények: 11](#_Toc130553987)

[1.4. Fajtái: Adatbázis rendszerek az adatkapcsolati modell alapján lehetnek: 11](#_Toc130553988)

[1.5. Hierarchikus modell: 11](#_Toc130553989)

[1.6. Hálós (Plex) struktúra definíciója: 12](#_Toc130553990)

[1.7. A relációs modell: 12](#_Toc130553991)

[1.8. Az SQL nyelvi elemei: 12](#_Toc130553992)

[1.8.1. Adatdefiníciós utasítások: 13](#_Toc130553993)

[1.8.2. Adatlekérdezési utasítások: 13](#_Toc130553994)

[1.8.3. Adatmanipulációs utasítások: 13](#_Toc130553995)

[1.8.4. Adatelérés vezérlő utasítások: 14](#_Toc130553996)

[1.9. Adatbázis technológiák: 14](#_Toc130553997)

[1.9.1. MySQL: 14](#_Toc130553998)

[1.9.2. Oracle: 15](#_Toc130553999)

[1.9.3. PostgreSQL: 15](#_Toc130554000)

[2. Általam választott megoldások: 15](#_Toc130554001)

[2.1. Adatbázis megvalósítása: 16](#_Toc130554002)

[2.2. Általam felhasznált technológiák: 18](#_Toc130554003)

[2.2.1. PHP: 18](#_Toc130554004)

[2.2.2. HTML/CSS: 18](#_Toc130554005)

[2.2.3. JavaScript: 19](#_Toc130554006)

[2.3. SQL Lekérdezések: 19](#_Toc130554007)

[2.4. A program működése: 19](#_Toc130554008)

[2.4.1. Adatbázis kapcsolat: 20](#_Toc130554009)

[2.4.2. Tartalom frissítése a weblapon: 20](#_Toc130554010)

[2.4.3. Bejelentkezés: 20](#_Toc130554011)

[Irodalomjegyzék 22](#_Toc130554012)

[Mellékletek 23](#_Toc130554013)

[Ábrajegyzék 24](#_Toc130554014)

[Táblázatjegyzék 25](#_Toc130554015)

Jelölésjegyzék

|  |  |
| --- | --- |
| AI: | Artificial Intelligence (Mesterséges Intelligencia) |
| GPU: | Graphical Processing Unit (Grafikus Processzor / Grafikus Feldolgozó Egység) |
| API: | Application Programming Interface (Alkalmazásprogramozási Felület) |
| CPU: | Central Processing Unit (Központi Feldolgozó Egység / Processzor) |
| GUI: | Graphical User Interface (Grafikus Felhasználói Felület) |
| HCI: | Human Computer Interaction (Ember-gép kapcsolat) |

# Adatbázisok kialakulása:

A programozás egyik alapvető eleme az adatbázis-kezelés, amely nélkülözhetetlen a fejlesztések során. Rengeteg különféle adattal dolgozunk, melyek eltárolásra kerülnek, manipuláljuk, illetve le is kérdezzük azokat. Fontosnak tartom, hogy a használt technológiának ismerjük a múltját is, ezért szeretném bemutatni az SQL történetét, szintaxisát a következő néhány bekezdésben.

## Az adatbázisok története :

Azóta rendelkezünk adatbázisokkal, mióta írásban vagyunk képesek rögzíteni adatokat. Már az ókortól. Később a kartoték-rendszerek lettek az adatbázisok fejlettebb formái. Majd megjelentek a számítógépek. Kezdetben lyukszalagon, lyukkártyán tárolták az adatokat, ezekhez a számítógép nem tudott közvetlenül hozzáférni.

Értelemszerűen az adatmodellek és az adatbázis-kezelő rendszerek történetével szoros összefüggésben áll. Az adatok gyors, gépesített, tárolásának és visszakeresésének igénye már az 1900-as évek elején is felmerült, amikor az első népességnyilvántartást végző (valójában halálozási statisztikákat készítő) lyukszalagos számítógépek megjelentek. Mindazonáltal a lyukkártyás technológia a jelenlegi rekordokban gondolkodó szemléletben éppúgy jelen van, mint a kezdetek kezdetén. Az adatbázisok jelenlegi, korszerű formái csak az 1960-as évek közepén kezdtek el kialakulni, elsősorban intézmény bérszámfejtési, adatsorelemzési problémáinak kezelésére. A rutinszerű feladatokat először egy hálós adatbázis segítségével igyekeztek lerövidíteni. E korszak terméke pl. a CODASYL.

## Az adatbázis definíciója:

Az információ valamely jelenségre vonatkozó értelmes közlést jelent, melynek a felhasználó számára újdonságszerű tartalma van. Az adat az információnak a továbbító, vagy tároló rend­szereken történő konkrét megjelenítési formája. Az entitás azon elemeknek az összessége, amelyekről információt tárolunk. A rájuk jellemző tulajdonságokat rekordokba foglalhatjuk.  
Az adatbázis az adott feladatok megoldásához szükséges adatok összekapcsolt halmaza, amelyek együttesen a vonatkozó szervezet elvi, fogalmi modelljét képezik.

## Adatbáziskezelő rendszerekkel szemben támasztott követelmények:

* Programfüggetlenség
* Adatkapcsolatok kezelése
* Adatsérthetetlenség / integritás / konzisztencia
* Adatvédelem
* Redundancia minimalizálása
* Többszörös keresési stratégiák
* Adatkezelés és felhasználás központi irányítása
* Egyidejű osztott adatelérés (sharing)
* „ad hoc” igény-kielégítés (azok is könnyen megtanulhassák, akik nem profik)
* Igény szerinti átalakítás lehetősége, vagyis automatikus adatkonvertálás.

## Fajtái: Adatbázis rendszerek az adatkapcsolati modell alapján lehetnek:

1. **Hierarchikus szerkezetű** (IMS, IMS-VS, DL/1, DL/1 ENTRY, VANDL/1)
2. **Hálós szerkezetű** (IDMS, DENNIS, CODASYL, SÁMÁN, BANK)
3. **Relációs elvű** (ORACLE, SQL, SYBASE, DB2, SYSTEM-R, Paradox)

## Hierarchikus modell:

A hierarchikus modell tekinthető a hálós modell szűkítésének, ahol a logikai rekordtípusok és kapcsok egy erdőt alkotnak (fák egy halmazát). Azaz, ha minden kapcsot úgy tekintünk, hogy a tulajdonos őse a tagnak, akkor a logikai rekordtípusok egy er­dőt alkotnak. A probléma ezzel az, hogy ez nem minden hálóra készíthető el. A hierarchikus modellben a sok-sok kapcsolat reprezentálható a kapcsolódó típusok virtuálismásolata segít­ségével. A virtuális típus tekinthető úgy, mint egy mutató a valós típus egy rekordjára.

## Hálós (Plex) struktúra definíciója**:**

A hálós struktúrában bármelyik csomópontot egy másik csomóponttal össze lehet kapcsolni. Lehet, de nincs értelme szinteket megkülönböztetni.  
Egyes elemek között egyszerű, mások között pedig komplex leképezés van. Ezen kapcsolatok azo­nos hálón belül egyesítve adják a hálós struktúrákat általánosan jellemző N:M kapcsolati viszonyt. Komplex plex-struktúráknak nevezzük azokat, melyeknél a séma egyik vonalán mindkét irányban 2 nyíl van. Azt, ahol 1 vonalnak sincs 2 nyila mindkét irányban egy­szerű plex-struktúrának hívjuk.

## A relációs modell:

E.F.Codd dolgozta ki 1971-72-ben. Az volt a célja, hogy kiküszöbölje a hálós struktúra hátrányait. A relációs modellnél a logikai adatszerkezet független a fizikai tárolástól, en­nek következtében a logikai kapcsolatok változtatása után nem kell a tárolást újraszervezni.  
A relációs adatmodell lényege, hogy az adatokat logikailag relációkban ábrázoljuk. Ez az áb­rázolás olyan, mintha az adataink kétdimenziós táblázatba lennének foglalva. Ezért lehet a relációs modell esetében táblázatokról is beszélni. A táblázatos ábrázolásmód legnagyobb előnye, a könnyű áttekinthetőség, a felhasználói szemlélethez közelálló modell-megjelenítés. A modell alapja a relációelmélet. Az alkalmazott műveleteket kifogástalan matematikai esz­közökkel írják le. Hátránya, hogy a számítástechnikai szakembereknek új kifejezéseket kell megtanulni, illetve a szokásostól eltérő gondolkodásmódot kíván.

## Az SQL nyelvi elemei:

Az SQL nyelv alapvetően 4 részre osztható:

* Adatdefiníció.
* Adatlekérdezés.
* Adatmanipuláció.
* Adatelérés vezérlése.

### Adatdefiníciós utasítások:

Ezek segítségével az adatbázisok sémáját definiáljuk. Ilyenek a CREATE, ALTER, DROP utasítások. A következő példával szemléltetem a használatát:

Létrehozunk egy UnitedUniversities nevű utf-8 kódólású adatbázist:

*create database UnitedUniversities default character set UTF8 collate UTF8\_bin;*

### Adatlekérdezési utasítások:

A lekérdezési nyelv egyetlen egy utasításból áll (SELECT), amely rengeteg másik alparancsot tartalmas és foglal magába, és a lekérdező utasítások többszörös mélységig egymásba ágyazhatók így összetettebb és bonyolultabb lekérdezések létrehozva. Célja az hogy több vagy esetlegesen egy adatbázisból relációt állítson össze. A felhasználó a bemeneti adatokon különböző algebrai műveleteket hajthat végre, amelynek következményeként egy eredmény táblát fog visszakapni.

Itt látható egy példa, mely a regisztrált felhasználókból lekérdezi azt a személyt akinek a jelszava és neptun kódja egyezik a bevitt értekkel:

*select count(neptun) from registered\_users*

*where neptun = @neptun and `password` = @pass;*

### Adatmanipulációs utasítások:

Az adatmanipulációs parancsok akkor használhatók, amikor egy meglévő mezőnek az értékeit szeretnénk módosítani (UPDATE), vagy amikor új adatot szeretnénk beszúrni az adatbázisba (ISERT INTO), vagy rekordokat szeretnénk törölni az adatbázisból (DELETE).

Itt látható egy példa ahogy a regisztrált felhasználók táblába veszünk fel egy új felhasználót:

*insert into registered\_users(neptun,fistName,lastName,email,`password`)*

*values(@nep,@fname,@lname,@email,@hasshedPasswordString)*

### Adatelérés vezérlő utasítások:

Az adatelérést vezérlő utasítások szolgálnak a jogosultságok kezelésére az adatbázisok kezelése esetén.  
A felhasználóktól meg lehet vonni az adatbázishoz való hozzáférést (REVOKE), vagy hozzáférést biztosítani a (GRANT) parancs segítségével. Az alábbi parancsok segítségével bizonyos műveletekre vonatkozóan (például írási jog megvonása) vagy különböző tábálákhoz vagy sémához való hozzáférési jogot lehet megadni/ elvenni.

Például elvesszük egy basic felhasználótól a táblakészítési jogosultságot egy felhasználótól:

*revoke create table from user;*

## Adatbázis technológiák:

Az adatbázisok megvalósításakor különböző technológiák és megoldások állnak rendelkezésre, ezekből szeretnék most párat bemutatni.

### MySQL:

A MySQL egy nyílt forráskódú adatbázis-kezelő rendszer. Nagyon népszerű, mert nagy megbízhatósága, könnyű kezelése és nagy teljesítmény. A MySQL-t számos, az Apache, Linux, Perl / PHP stb. Alapuló alkalmazásaira használják. Sok népszerű szervezet, mint a Google, az Alcatel Lucent, a Facebook, a Zappos és az Adobe is.

### Oracle:

Az Oracle legújabb verziója RDBMS (Object Relational database Management System). Ezt az Oracle Corporation fejlesztette ki. Az Oracle adatbázis legfrissebb verziója 11g, amely magas színvonalú szolgáltatásokat biztosít:

* Párhuzamos DBA termelékenység
* Megszünteti az adatközpont redundanciáját és maximalizálja a rendelkezésre állást.
* Konszolidálja és klasztereli a vállalati alkalmazásokat skálázható, gyors és megbízható magánfelhőkre.
* Csökkenti a változás kockázatát a DBS termelékenység megduplázásával.

### PostgreSQL:

A progress egy erős nyílt forráskodú objektum-relációs adatbázis kezelő rendszer ami hasonló a relációs adatbázishoz, de az objektum orientált adatbázis-modellel rendelkezik. Az objektumokat, osztályokat és az öröklődést közvetlenül támogatják az adatbázis-sémák és lekérdezések. Ez a rendszer több mint 30 év aktív fejlesztés alatt nagy hírnevet szerzett magának a megbízhatóságával, teljesítményével és rengeteg funkciója miatt.

# Általam választott megoldások:

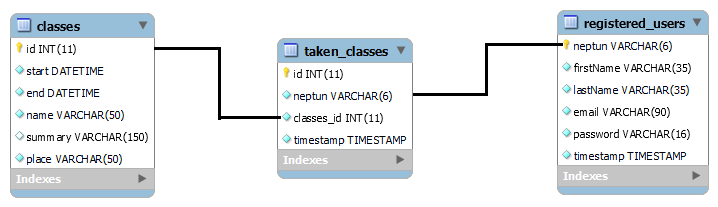
Mivel a weboldal felületén majd lehetőség lesz a felhasználóknak regisztrálni és feltölteni az órarendjüket a rendszerbe ezért adatbázis használata elengedhetetlen lesz a készítési folyamatok során. Az adatbázis szerver megvalósításához MySQL használata mellett döntöttem, azért mert nem lesz szükség a weblap megvalósítása során hatalmas adatbázisra.

Az adatbázis létrehozásához, modell megvalósításához és a lekérdezések teszteléséhez a MySQL Workbenchet használom. A szerverhez való kapcsolódáshoz a WAMP szervert használom. A MySQL workbanch használata mellett azért döntöttem mert, nagyban megkönnyíti a fejlesztési folyamatokat. A workbanch megkönnyíti az adatbázis szerkezetének átlátását és annak modelljét könnyen elérhetővé illetve importálhatóvá teszi.

## Adatbázis megvalósítása:

Az adatbázisomban jelenleg a 3 következő tábla található meg.

* classes
* taken\_classes
* registered\_users

A szerkezet relatíve egyszerű, de könnyen átlátható is. Ez azét van mert nincsen szükség hatalmas mennyiségű adatok eltárolására így rengeteg helyet meg spórolva ezzel az adatbázisban. Az adatbázis modellje itt található:

* 1. *Az adatbázis modellje.*

A classes tábla fogja majd eltárolni a feltöltése kerülő órarendekben található órák adatait.

* id (Int) - Egy azontosítót tárol egész számként. Ez az elsődleges kulcs. Változó típusa azért int mert az órákat azonosító szerint fogjuk eltárolni 1 től kezdve
* start (Datetime) – Az órák kezdési időpontját tárolja el. A változó típusa azért Datetime mivel az órák az ics file-ban is dátum típusuként vannak eltárolva amiből a kolvasás történik az órarend feltöltése során.
* end (Datetime) – Az órák befejezési időpontját tárolja el. A változó típusa azért Datetime mivel az órák az ics file-ban is dátum típusuként vannak eltárolva amiből a kolvasás történik az órarend feltöltése során.
* name (Varchar(50)) – A feltöltött órák nevét tárolja el. A változó típusa azért Varchar(50) hogy minden létező óra neve könnyen elvérjen.
* summary (Varchar(50)) – Az órákhoz tartozó leírást tárolja el. A váltózó típusa azért Varchar(50) hogy az esetlegesen beírt leírásokat az órához eltárolja.
* place (Varchar(50)) – A az órák megtartási helyét tárolja el. A típus azért Varchar(50) hogy az egyetemen található összes terem beleférjen az adatbázisba.

A registered\_users tábla tárolja el a beregisztrált felhasználóknak az adatait.

* neptun (Varchar(6)) – Neptun kódot tárolja el, ami a felhasználók egyedi azonosítója ként is szolgál, elsődleges kulcs. A változó azért Varchar(6) mert az azonosítóban szöveg és szám is lehet egyaránt ami pontosan 6 karakter hosszú.
* firstName (Varchar(35)) - A felhasználók vezetéknevét tárolja. A változó típusa azért Varchar (35) mert így minden Magyarországon található vezetéknevet el lehet tárolni.
* lastName (Varchar(35)) – A felhasználók keresztnevét tárolja el. A típus szintén Varchar (35) mert így minden Magyarországon található vezetéknév eltárolható.
* email (Varchar(90)) - A felhasználók emailcímét tárolja el. A típusa Varchar(90) hogy, minden típusu email eltárolható legyen, pl(gmail, hotmail, freemail).
* password (Varchar(16)) – A felhasználók jel szavait tárolja titkosított formában. Varchar (16) típusu mert a jelszóban betű, szám és különleges karakter is szerepelhet aminek hossza maximum 16 karakter lehet majd. Ezek a jelszók majd titkosítva kerülnek be az adatbázisba.
* timestamp (Timestamp) – Eltárolja a regisztráció időponjtát.  
  Timestamp típusu, hogy rögzítse hogy a regisztráció mikor történt pontosan az adatbázisba.

A taken\_classes táblra biztosítja a kommunikációt a fent említett két tábla között.A táblában két idegen kulcs található: neptun és a classes\_id A táblában találhatók:

* Id (INT(11)) – Egy azonosítót tárol el egész szám típusban.
* neptun (Varchar(6)) – Egy idegen kulcs ami hivatkozik a registerd\_users táblában található neptun elsődleges kulcsra. Varchar(6) típusu szintén a betűk és számok szereplése miatt.
* classes\_id (Int(11)) – Egy idegen kulcs ami hivatkozik a classes táblában található id elsődleges kulcsra.
* timestamp (Timestamp) – Eltárolja az órarend felhasználóhoz történt csatolás időpontját. Timestamp típusu mert egy módósítás időpontot tárol el.

## Általam felhasznált technológiák:

### PHP:

A PHP egy szerveroldali szkriptnyelv, mely segítségével dinamikus weblapokat készíthetünk a kódot a webszerver PHP feldolgozómodulja értelmezi. Az első szkriptnyelvek egyike, amely külső fájl használata helyett HTML oldalba ágyazható. A PHP nyelven írt kódokat a webszerver PHP feldolgozómodulja értelmezi. A PHP egy olyan programozási nyelv, mely segítségével képesek lehetünk elkészíteni egy adatbázisalapú weboldalt is. Rasmus Lerdorf 1995-ben indította útjára.

### HTML/CSS:

A HTML egy leíró nyelv olyan nyelv, melyet weboldalak elkészítésére használhatunk és mára már internetes szabvánnyá vált a W3C támogatásával.. Minden weboldal mögött HTML található (de nem csak weboldalaknál lehet használni, hanem például e-könyveknél, vagy más dokumentumoknál is). A kifejezés a Hypertext Markup Language (hiperszöveges jelölőnyelv) rövidítése, és egy utasításkészlet a webböngésződ számára. Az utasítások betartásával a böngésző úgy jeleníti meg a weboldalad, ahogy azt a tervező szeretné. Az aktuális változata az 5, mely az SGML általános jelölőnyelv egy konkrét alkalmazása.

A CSS (Cascading Style Sheets), mely azt határozza meg, hogyan nézzen ki például egy szöveg a weboldalon. A CSS egy stíluslap, mely leírja egy weboldal megjelenését. A CSS azt mondja meg a böngészőnek, hogy egy HTML dokumentum egyes elemei miként jelenjenek meg a felhasználók számára. Legfontosabb jellemzője, hogy képes különválasztani a weboldal-tartalmat a designelemektől. A CSS a számítástechnikában egy stílusleíró nyelv, mely a HTML vagy XHTML típusú strukturált dokumentumok megjelenését írja le. Ezenkívül használható bármilyen XML alapú dokumentum stílusának leírására is, mint például az SVG, XUL stb. A CSS specifikációját a World Wide Web Consortium felügyeli.

### Bootstrap

A Bootstrap egy nyílt forráskódú keretrendszer (framework), mely **HTML, CSS, JavaScript technológiákat használ**. Alapvetően arra jó, hogy nagyon könnyedén, és minimális energia befektetéssel tudjon valaki jól kinéző, bármilyen képernyőméreten szépen megjelenő weboldalakat készíteni.

### JavaScript:

A JavaScript programozási nyelv egy objektumorientált, prototípus-alapú szkriptnyelv, amelyet weboldalakon elterjedten használnak. A javaScript egy kliensoldali programozási nyelv. A JavaScriptek, tehát, a böngészőben futnak, a HTML lap szövegébe épülnek be, és a felhasználó számára teszik élvezetessé az interneten való böngészést: a statikus weboldalakba dinamikát visznek, színesítik-, élvezetessé teszik a felhasználók számára: videók, 2D-s és 3D-s grafikák, interaktív térképekkel színesítik.

## SQL Lekérdezések:

Az alább látható kódrészlettel történik a megfelelő felhasználó neptun kódjának és jelszavának lekérdezeése. Megfelelően megadott adatok után a felhasználó sikeresen bejelenetkezik.

*select count(neptun) from registered\_users*

*where neptun = @neptun and `password` = @pass;*

*select count(neptun) from registered\_users*

*where neptun = 'CGSHJS' and `password` = 'asdasdasdased';*

## A program működése:

A weblapon történő regisztráció után az registered\_users táblában létre jön egy új rekord ami a következő adatokat fogja tárolni: neptunkód(ez az elsődleges kulcs), vezetéknév, keresztnév, email, jelszó és a regisztráció időpontja. A regisztrált felhasználóknak akik beléptek már a felületre, megjelenik egy új menüpont órarend feltöltése címmel. Erre a menüpontra kattintva a felhasználó egy amit majd a neptunból kiimportált ics-file ból fognak majd tudni feltölteni.

### Adatbázis kapcsolat:

Az adatbázis kapcsolat létrejötte a következőképpen történik meg:

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "uniteduniversities";

// Kapcsolat létrehozása

$conn = new mysqli($servername, $username, $password, $dbname);

// Kapcsolat ellenőrzése

if ($conn->connect\_error)

    die("Sikertelen kapcsolódás: " . $conn->connect\_error);

?>

1.

A későbbiekben a különbőző adatbázisműeletek során

### Tartalom frissítése a weblapon:

A weboldalalon a tartalom megjelenítését egy session segítésével valósítom meg. Ami azt jelenti hogy az oldalon a tartalmak és a fülek közötti lepegetésekor nem minden alkalommal frissül a lap hanem csak mindig az adott oldalnak megfelelő tartalmat frissíti le.

### Regisztráció:

A weboldalra történő regisztráció a következőképpen törtnéik:

<input class="input-field" type="text" placeholder="Vezetéknév" name="firstname"

                        pattern="([A-Z]([a-záéúőóüö.]{1,}\s?)){1,40}" required

                        oninvalid="this.setCustomValidity('Vezétéknéknév megadása kötelező')">

A felhasználó egy form segítségével megadja a felhasználó adatait amiknek helyeségét és meglétet regex pattern segítésével ellenörzöm pl:

pattern="([A-Z]([a-záéúőóüö.]{1,}\s?)){1,40}"

A-z karakterig plussz a (a-záéúőóüö.) magyar abc ékezetes beüti

<?php

$firstname = $\_POST['firstname'];

$lastname = $\_POST['lastname'];

$email = $\_POST['mail'];

$neptun = $\_POST['neptun'];

$jelszo = $\_POST['psw'];

$jelszo2 = $\_POST['pswag'];

$sql = $conn->query("SELECT \* FROM registered\_users WHERE neptun LIKE '$neptun'");

$first = strlen($firstname);

$vane = $sql->num\_rows;

$last = strlen($lastname);

$kodhossz = strlen($neptun);

$jelszohossz = strlen($jelszo);

$titkositott = md5($jelszo);

$neptunfel = strtoupper($neptun);

if ($first > 0 && $last > 0 && $vane == 0 && $kodhossz == 6 && $jelszo == $jelszo2 && $jelszohossz >= 8) {

    $sql = $conn->query("INSERT INTO registered\_users VALUES('$neptun','$firstname','$lastname','$email','$titkositott', now()) ");

    if($sql)

    {

        print " <div class='not'>Sikeres regisztráció! </div>";

    }

    else{

        print " <div class='not'>Sikertelen regisztráció! </div>";

    }

### Bejelentkezés:

A weboldalra történő bejelentkezés a következő módon fog megtörténni:

<?php

$nc = $\_POST['neptuncode'];

$pw = $\_POST['password'];

$titkositott = md5($pw);

$ncfel = strtoupper($nc);

$sql = $conn->query("SELECT \* FROM registered\_users WHERE neptun LIKE '$ncfel' && password like '$titkositott'");

$exists = $sql->num\_rows;

$name = $conn->query("SELECT lastName FROM registered\_users WHERE neptun LIKE '$ncfel'");

//ellenörzés

if ($exists != 1)

    print "Hibás neptunkód vagy jelszó!";

    else{

        $\_SESSION["id"] = $ncfel;

    header("Location:?page=main");

}

Irodalomjegyzék

|  |  |
| --- | --- |
| [1] | T. De Smedt and W. Daelemans, “Pattern for python.,” *The Journal of Machine Learning Research,* vol. 13, no. 1, pp. 2063-2067, 2012. |
| [2] | „Záróvizsga információk,” [Online]. Available: https://mik.uni-pannon.hu/index.php/hu/oktatas/zarovizsga.html. [Hozzáférés dátuma: 04 03 2022]. |
| [3] | D. J. Wetherall és A. S. Tanenbaum, Computer networks, Pearson Education, 2013. |

Mellékletek

Mappaszerkezet

+chatbot

| backen.bat

| backend.py

| files.doc

| fixedlinks.json

| ipcheck.py

| linkek.json

| linkfix.py

| log.txt

| sqlwriter.py

| textprocessor.py

|

+---backend

| pyvenv.cfg

+---lara

| | entities.py

| | nlp.py

| | parser.py

| | stemmer.py

+---static

| +---css

| | chat.css

| |

| \---js

| chat.js

|

+---templates

| index.html

|

[PÉLDA!!! Megjegyzés: A Python csomagkezelője által telepített fájlok, illetve a különböző cache fájlok a fenti listából kimaradtak, mivel ezekkel indokolatlanul és aránytalanul hosszú lenne a fenti felsorolás. A beadott fájlok között azonban a teljesség kedvéért szerepelnek ezek a fájlok is.]

Ábrajegyzék

Táblázatjegyzék

[1. táblázat (forrás megjelölésével) [2] 1](#_Toc97890942)