Pannon Egyetem

Műszaki Informatikai Kar

Matematika Tanszék  
Programtervezező Informatikus

**SZAKDOLGOZAT  
United Unies**

**Rentler Ferenc**

Témavezető: Pozsgai Tamás

Külső/belső konzulens: Pozsgai Tamás

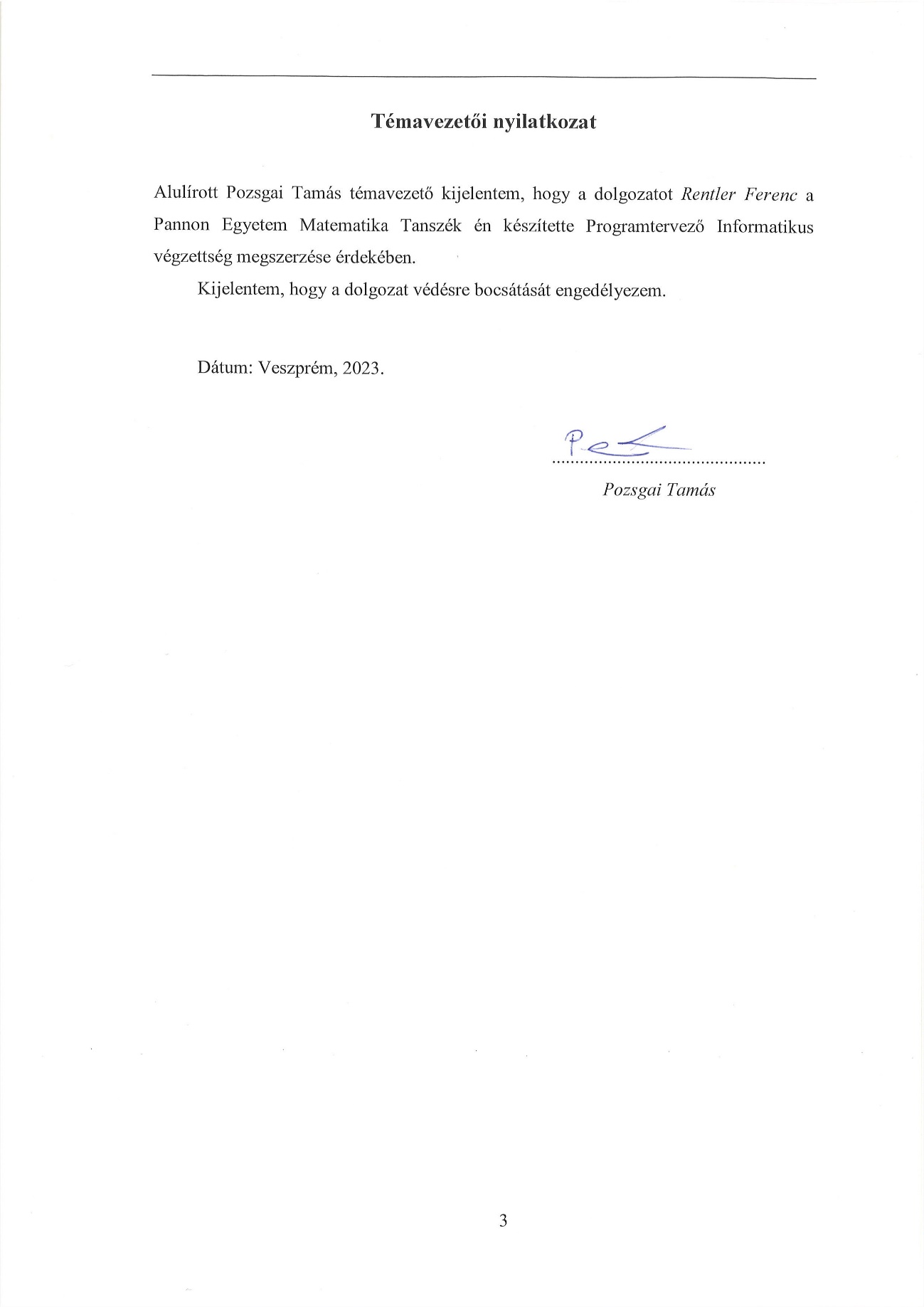
2023

Témakiírás



*A képen szöveg, levél látható

Automatikusan generált leírás*



Köszönetnyilvánítás

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

Azt a feladatot tűztem ki magam elé, hogy olyan weblapot készítsek, amely magába foglalja a köszöntő oldalán a karok történeté. Érdekes információkkal mutat az egyetem után érdeklődőket, segíti az egyetemistákat az egyetem területén tájékozódni térkép és útvonal tervezéssel. A weblapon megtaláljuk a különböző fő egyetemi épületek bejáratának helymegjelölését, illetve rövid tájékoztatóval az adott épületről. Ezen kívül bemutatja Veszprém és környéke érdekességeit szintén térképes helymegközelítéssel. Innovatív lehetőséget nyújt a diákoknak regisztráció után bejelentkezésre, ami lehetővé teszi az órarend felöltését, ezt a Neptun rendszeréből lehet importálni, valamint a törlésig megtekintheti a felhasználó. A weblapon regisztráció után lehetőséget is biztosít a későbbi adat módosításra, ha változnának felhasználó adatai.

**Kulcsszavak:** [weblap, elhelyezkedés, elérhető, felhasználók] (nem tudom ide mit kéne még írni, ezt így nem értem, bocsi xd)

Abstract

The task is to create a website that includes the history of the university on its welcome page, interesting information, and helps users navigate the university campus with a map and route planner. It also marks the entrances of different main university buildings with brief descriptions of each building. In addition, it showcases the interesting sights in Veszprém and its surroundings with a map for location accessibility. The website provides an opportunity for students to register and log in, allowing them to upload their schedules downloaded from Neptun and view them on page until deletion. After registration, users can also modify their data if their personal information changes.

**Keywords:** [webpage,location, available, users]

Tartalomjegyzék

[Jelölésjegyzék 10](#_Toc134518551)

[1. Bevezetés 11](#_Toc134518552)

[1.1. Összehasonlítás 11](#_Toc134518553)

[1.2. Feladat: 12](#_Toc134518554)

[1.2.1. Környezet: 13](#_Toc134518555)

[2. Adatbázisok kialakulása: 13](#_Toc134518556)

[2.1. Az adatbázisok története: 14](#_Toc134518557)

[2.2. Az adatbázis definíciója: 14](#_Toc134518558)

[2.3. Adatbáziskezelő rendszerekkel szemben támasztott követelmények: 15](#_Toc134518559)

[2.4. Fajtái: Adatbázis rendszerek az adatkapcsolati modell alapján lehetnek: 15](#_Toc134518560)

[2.5. Hierarchikus modell: 15](#_Toc134518561)

[2.6. Hálós (Plex) struktúra definíciója: 15](#_Toc134518562)

[2.7. A relációs modell: 16](#_Toc134518563)

[2.8. Az SQL nyelvi elemei: 16](#_Toc134518564)

[2.8.1. Adatdefiníciós utasítások: 16](#_Toc134518565)

[2.8.2. Adat lekérdezési utasítások: 17](#_Toc134518566)

[2.8.3. Adatmanipulációs utasítások: 17](#_Toc134518567)

[2.8.4. Adatelérés vezérlő utasítások: 18](#_Toc134518568)

[2.9. Adatbázis technológiák: 18](#_Toc134518569)

[2.9.1. MySQL: 18](#_Toc134518570)

[2.9.2. Oracle: 19](#_Toc134518571)

[2.9.3. PostgreSQL: 19](#_Toc134518572)

[3. Fejlesztői dokumentáció 19](#_Toc134518573)

[3.1.1. Funkcionális követelmények: 19](#_Toc134518574)

[3.1.2. Nem funkcionális követelmények: 20](#_Toc134518575)

[3.2. Adatbázis megvalósítása: 20](#_Toc134518576)

[3.2.1. A táblák szerkezete: 22](#_Toc134518577)

[3.3. Általam felhasznált technológiák és alkalmazások: 24](#_Toc134518578)

[3.3.1. PHP: 24](#_Toc134518579)

[3.3.2. HTML/CSS: 24](#_Toc134518580)

[3.3.3. Bootstrap 25](#_Toc134518581)

[3.3.4. JavaScript: 25](#_Toc134518582)

[3.3.5. Google Cloud 25](#_Toc134518583)

[3.3.6. Google maps Javascript API 26](#_Toc134518584)

[3.3.7. Visual studio code: 26](#_Toc134518585)

[3.3.8. MySQL Workbench 26](#_Toc134518586)

[3.4. SQL Lekérdezések: 27](#_Toc134518587)

[3.5. A program működése: 27](#_Toc134518588)

[3.5.1. Az oldal szerkezetének felépítése: 27](#_Toc134518589)

[3.5.2. Tartalom frissítése a weblapon: 29](#_Toc134518590)

[3.5.3. Helyek és térkép megjelenítése a weblapon: 30](#_Toc134518591)

[3.5.4. Kapcsolódás az adatbázishoz: 32](#_Toc134518592)

[3.5.5. Regisztráció: 33](#_Toc134518593)

[3.5.6. Bejelentkezés: 36](#_Toc134518594)

[3.5.7. Profil: 37](#_Toc134518595)

[3.5.1. Órarend feltöltése: 39](#_Toc134518596)

[3.5.2. Órarend törlése. 44](#_Toc134518597)

[4. Használat 45](#_Toc134518598)

[4.1. Kezdőlap: 46](#_Toc134518599)

[4.2. Épületek 47](#_Toc134518600)

[4.3. Nevezetességek 48](#_Toc134518601)

[4.4. Regisztráció 48](#_Toc134518602)

[4.5. Bejelentkezés: 49](#_Toc134518603)

[4.6. Órarend: 50](#_Toc134518604)

[4.7. Profil: 51](#_Toc134518605)

[4.8. Kilépés: 52](#_Toc134518606)

[5. Továbbfejlesztési lehetőségek 52](#_Toc134518607)

[6. Összefoglaló 53](#_Toc134518608)

[Irodalomjegyzék 54](#_Toc134518609)

[Mellékletek 55](#_Toc134518610)

[Ábrajegyzék 57](#_Toc134518611)

[Táblázatjegyzék 58](#_Toc134518612)

Jelölésjegyzék

# Bevezetés

Feladat egy olyan weblap készítése, amelyek segítik az egyetemre érkező gólyák, illetve nyíltnapra látogató diákok életét.

„Dizájn nem az, ahogy valami kinéz vagy amilyen érzést kelt. A dizájn az, ahogyan működik.” (Steve Jobs)[[1]](#footnote-1)

Ez az idézet tükrözi a weblapom hitvallását. Nem csak a kinézettével, hanem a funkcióival is szerettem voltan egy olyan oldalt létrehozni, ami lehető legtöbb segítséget nyújt az egyetemistáknak, kreatív kinézettel, könnyen átlátható menükkel. Feladatomnak éreztem a letisztult felület létrehozását, ami tükrözi az egyetemünk arculatát.

## Összehasonlítás

A kutató munkám során utána néztem, milyen hasonló megoldások léteznek a térképekhez a jelenlegi Pannon Egyetem honlapjához kiegészítő weblapokon, mint ami az enyémben is szerepel. Nagyon hasonló tematikára épülnek fel ezek a weboldalak, egyformán meg lehet találni az épületek jelölését tértképen. Amiben különbözik a tértképeink azaz, hogy míg én kétfajta térképet találtam erről, addig az én weblapom magába foglalja mind két térképet. Tehát a weblapokon vannak fent egy térkép az épületek elhelyezkedéséről, és egy külön térkép az bejáratokról. Míg az én weblapomnál mind ezt egy térképen van jelölve. Ami még lényeges különbség az a térkép nézet. Az enyém Google térkép segítségével készült, így a felhasználó ki tudja nagyítani, így könnyebben tájékozódást biztosít a form. Ezen kívül a parkoló helyeket is megtudják tekinteni, ami a másik oldalakról hiányzik. Találtam olyan oldalt, ahol az épületekről fényképek vannak, ez későbbiekben egy fejlesztési lehetőség lehetne az én weblapomnál is. Mint a Veszprémi látványoságok megtekintésénél az oldalamon, itt szintén hozzá lehetne rendelni fényképeket.

Ami az én weblapomon teljesen különbözik a többitől az a látványoságok összegyűjtése. Ahogy szakdolgozatomba is látszik, térkék jelölővel, rövid leírással és képekkel gyűjtöttem össze a Veszprémi látványosságokat. Ami jól tükrözi a város infrastruktúráját és sajátos arculatát. Számomra fontos volt a város bemutatása is, mivel a felvételizők nem csak az egyetemi oktatás alapján választanak, hisz Magyarországon sok a nívósabb egyetem, mint a Pannon egyetem ugyancsak ide tartozik. Tehát a felvételizők a város lehetőségeit is figyelembe veszik. Ezért tartom fontosnak a weblapom ezen részét szintén.

A további különbség, azaz órarend lehetősége. Az egyetemi honlapunkon nincs ilyen lehetőség, hogy a diákok onnan is elérjék az aktuális órarendjüket, csak a Neptunhoz irányit az oldal tovább. Szerintem ez a lehetőség könnyebb megoldást biztosit a golyókánknak, mivel egy weboldalon keresztül megtudják tekinteni az órájukat, valamint rögtön meg tudják keresni az adott órához tartozó épületet úgyszintén.

Hasonlóságok szintén akadnak a két weboldal között, mint például a köszöntő oldal. Itt is megtalálható egy rövid bemutatkozás az egyetemről. Megtalálható mindkét weblapon az Egyetem Neptun oldala, valamint a karok moodle oldalának elérhetőségei.

Az egyetem honlapján megtalálható a *Hírek, aktualitások* fül, mi az enyémből hiányzik. Ezzel tovább lehetne fejleszteni az én platformomat. Így még innovatívabba tenni a későbbiekben a saját weboldalamat.

## Feladat:

Olyan felületet szeretnénk létrehozni, amely a Pannon Egyetem hallgatói számára segítséget biztosít az egyetemi épületek közötti eligazodásban, valamint megtekinthetik, hogy éppen hol tartózkodnak hallgatótársaik. Egy regisztrációhoz kötött multiplatform oldal tervezése, majd kivitelezése a Pannon Egyetemen tanuló diákok számára. Az egyetem épületeinek, nevezetességeinek megjelenítése, valamint egyértelmű jelölése a térképen segítséget biztosít a tájékozódásban. Kiegészítésként feltölthetik majd a hallgatók az órarendjüket is. Ez a weboldal más szemszögből közelíti meg a diákok igényét, mint a jelenlegi egyetemi honlapunk. Kicsit letisztultabb platformot szerettem volna létrehozni, más fikciókkal, mi jól kiegészíti a jelenlegi iskolai weboldalt. Több segítséget nyújthat az egyetemistáknak, az érdekességek és a bejáratok megjelölésével, pedig a gólyáknak nyújthat asszisztálást a navigációban az első napokban.

### Környezet:

Akármilyen böngészőm fútatására alkalmas operációs rendszer vagy eszköz (mobil, asztali számítógép, tablet), valamilyen böngésző (Chrome, Safari, Firefox, Opera) valamint működő internet kapcsolat.

# Adatbázisok kialakulása:

„Az adatbázisok története szorosan összefügg az adatmodellek és ezzel együtt az adatbázis-kezelő rendszerekkel is. Mindenki ismeri a kialakulásukhoz vezető utat, a CODASYL ajánlást, a fejlődésük fontosabb állomásait, ami valóban végigkövette az adatmodellek hierarchikus, hálós és relációs útját. Mára pedig magukon hordozzák a '90-es évek és az ezredforduló vívmányainak a jegyeit.

Fejlődésük ugyan lassult, de meg nem állt. Az objektum-orientált szemlélet átvételét követően a következő legnagyobb lépés az XML nyelv és technológiák befogadása lett. (…) Ellenben egy új XML dokumentum szerkezeti kialakításának és az egyes megközelítések előnyeinek és hátrányainak áttekintése a jegyzetnek része, mert ez az a pont ahol célszerű megismerni ezen módszerek által kínált különféle lehetőségeket annak érdekében, hogy hatékony dokumentum struktúrával tudjunk dolgozni. Emellett kitérünk a validálásra is, mert egy adatbázis nem pusztán csak adatok gyűjtőhelye, hanem segítségükkel komplexebb szabályok is megfogalmazhatóak amelyek felügyelik a tárolandó adatokra vonatkozó megszorításokat. Ehhez szintén szabványos technológiákat fogunk alkalmazni.” (I. rész - Fejlett Adatbázis Technológiák – Jegyzet- 1. fejezet: Bevezetés) [[2]](#footnote-2)

Tehát 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 adatbázis (DB: Database) számítógépen (általában háttértárakon) tárolt adatok összessége. Az adatbázist egy adatbázis kezelő rendszer (DBMS : Database Management System) segítségével használhatjuk. Nem minden (számítógépen tárolt) adathalmazt tekintünk adatbázisnak” [[3]](#footnote-3)

Azaz 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 újra szervezni.  
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.
* Adat lekérdezés.
* 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;*

### Adat lekérdezési 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.

# Fejlesztői dokumentáció

### Funkcionális követelmények:

Egyik fő feladatom az hogy a felhasználók számára egy olyan felületet biztosítsak amely könnyen navigálható valamint elérhető. A tervezés során fő szempont volt a felhasználók segítése ezért elérhető egy navigációs menü ahol elérhető az Egyetem neptun oldala valami valamint a karok moodle oldalinak elérhetősége. Megtekinthetőek leszek az Egyetem épületei helyjelölővel valamint parkoló és épület bejárat jelölőkkel, valamint Veszprém és környéki nevezeteségekkel szintén térképpel jelölve. Regisztrációs lehtőség biztosítása az Egyetemi hallgatók számára valamint órarend feltöltési lehetősége.

### Nem funkcionális követelmények:

A felhasználó adatai legyen megfelelő eltárolva megfelelő biztonsággal.

Folyamatos működés és elérhetőség biztosítása a felhasználók számára.

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, illetve a könnyebb átláthatóság érdekében. Az adatbázis modellje itt található:

A képen diagram látható

Automatikusan generált leírás

1. ábra: Adatbázis modell.. [1]

Az összekötő táblára azért van szükség mert, ha nem lenne, akkor sok String összehasonlítás, String műelet lenne a classes.id mezőre, hogy mely sorok tartalmazzák a 'XXXXXX' neptun kódot így meg csak egy indexelt mezőre (amely értékek abc szerinti növekvő sorrendben vannak tárolja az index-táblában) keresi konkrétan a 'XXXXXX' stringre egyező classes\_id mezőket a taken\_classes táblában, így gyorsabb kikeresni a user kódjához tartozó classes.ID id-kat.

Az alábbiakban egy példa található az összekötő tábla létrehozására:

*create table taken\_classes*

*(*

*id int auto\_increment not null,*

*neptun varchar(6) not null,*

*classes\_id varchar(9) not null,*

*`timestamp` timestamp default now(),*

*primary key (id),*

*index(neptun),*

*index(classes\_id),*

*foreign key (classes\_id) references classes(id),*

*foreign key (neptun) references registered\_users(neptun)*

*) char set utf8 collate utf8\_bin engine=InnoDB;*

A kódrészletben láthatjuk a taken\_classes tábla létrehozását:

* "id": egy auto\_increment szám, amely egyedi azonosítót ad minden rekordnak.
* "neptun": egy varchar típusú mező, amely hat karakter hosszúságú és nem lehet üres. Ez a mező hivatkozik a "registered\_users" tábla "neptun" mezőjére, ami azt jelenti, hogy a "taken\_classes" tábla minden rekordja kapcsolódik egy regisztrált felhasználóhoz.
* "classes\_id": egy varchar típusú mező, amely kilenc karakter hosszú és nem lehet üres.
* Ez a mező hivatkozik a "classes" tábla "id" mezőjére, ami azt jelenti, hogy a "taken\_classes" tábla minden rekordja kapcsolódik egy tantárgyhoz.
* "timestamp":egy timestamp típusú mező, amely az aktuális időpontot veszi fel, ha nem adunk meg más értéket.

Az utolsó sor a karakterkódolást és a táblamotor típusát határozza meg. Ebben az esetben a karakterkódolás UTF-8, a táblamotor pedig az InnoDB. A tábla létrehozása során két indexet is létrehozunk, az egyik a "neptun" mezőn, a másik pedig a "classes\_id" mezőn, ami segíti a keresést a táblában.

Emellett két idegen kulcsot is definiálunk, az egyik a "classes\_id" mezőre, amely hivatkozik a "classes" tábla "id" mezőjére, a másik pedig a "neptun" mezőre, amely hivatkozik a "registered\_users" tábla "neptun" mezőjére. A foreign key (idegen kulcs) segítségével a táblák közötti kapcsolatot definiáljuk, és biztosítjuk, hogy csak olyan rekordok kerüljenek be a "taken\_classes" táblába, amelyek megfelelnek a kapcsolódó rekordoknak a másik két táblában.

### A táblák szerkezete:

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

* id (Varchar(9)) - Egy szöveg típusú azontosítót. Ez az elsődleges kulcs. A változóban egy Neptunkód és számok kimbinációját fogjuk eltárolni azonosítóként a mezőben.
* 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 (Varchar(9))) – 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 és alkalmazások:

### 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.

### Google Cloud

A Google Cloud Platform (röviden GCP néven is ismert) felhőalapú számítási szolgáltatások és eszközök összessége, amelyet természetesen a Google kínál. A Google nem csak a szerverterületet biztosítja, hanem lehetővé teszi saját infrastruktúra kiépítését is, a cég igényeinek megfelelően. Jelenleg (2020 júniusában) a GCP több mint 170 felhőszolgáltatást, eszközt és alkatrészt kínál, amelyek a jelenlegi és a jövőbeli igényektől függően szabadon hozzáadhatók az infrastruktúrához vagy eltávolíthatók abból.

A Google Cloud Platform szolgáltatásai többek között támogatják:

* az egyszerű alkalmazásindítást, függetlenül attól, hogy milyen technológiával készültek,
* az új funkciók bevezetését a meglévők módosítása anélkül, így a szerveren nincs szükség nagyobb változtatásokra,
* a biztonságot – védelem a külső támadások ellen és a szervezeten belüli veszélyek azonosítása,
* a fájlok, objektumok és tartalmak biztonságos és méretezhető tárolását,
* sokféle adatbázis létrehozását és használatát,
* az azonnali adatfeldolgozást és valós idejű szinkronizálást,

### Google maps Javascript API

A Maps JavaScript API lehetővé teszi, hogy testreszabott tartalmakat és képeket használj a térképek megjelenítéséhez weboldalakon és mobil eszközökön. A Maps JavaScript API négy alap térképtípust (útvonalterv, műholdas, hibrid és terep) tartalmaz, amelyeket rétegekkel és stílusokkal, vezérlőkkel és eseményekkel, valamint különböző szolgáltatásokkal és könyvtárakkal lehet módosítani.

### Visual studio code:

Visual Studio Code egy könnyű de erőteljes forráskód szerkesztő, mely asztali környezetben fut, és elérhető Windows, macOS és Linux rendszerekre is. Beépített támogatást nyújt JavaScript, TypeScript és Node.js nyelvekhez, valamint számos kiegészítővel bővíthető más programozási nyelvekhez és futtatási környezetekhez (például C++, C#, Java, Python, PHP, Go, .NET). Az alapok elsajátításához tekintsd meg ezeket az introdutkciós videókat a VS Code-hoz.

### MySQL Workbench

A MySQL Workbench egy grafikus adatbázis-tervező eszköz, amelyet a MySQL adatbázis-kezelő rendszerhez fejlesztettek ki. Ez az eszköz lehetővé teszi az adatbázisok tervezését, létrehozását, karbantartását és vizualizációját.

Az eszköz segítségével lehetőség van az adatbázisok tervezésére UML diagramok használatával, a kapcsolatok és az adatmodell kialakításával. Emellett lehetőség van az adatbázisokhoz kapcsolódó számos művelet végrehajtására, mint például táblák létrehozása, módosítása, törlése, adatok importálása és exportálása.

A MySQL Workbench rendelkezik egy integrált lekérdezés szerkesztővel, amely lehetővé teszi az SQL lekérdezések megírását és futtatását, valamint az eredmények megjelenítését. Az eszköz támogatja a többfelhasználós környezetet, így lehetővé teszi a kollaboratív munkát az adatbázisokon.

Az eszköz elérhető Windows, macOS és Linux rendszerekre is, és ingyenesen letölthető a MySQL hivatalos weboldaláról.

## 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.

### Az oldal szerkezetének felépítése:

A fejrész (head) tartalmát definiálja egy weboldalon. Az itt található elemek többek között a következők:

* A cím (title) az oldal címét határozza meg, amit a böngésző fülén lehet látni. A karakterkódolás (charset) megadja, hogy az oldal milyen karakterkódolást használjon. Az UTF-8 a legelterjedtebb karakterkódolás.
* Stíluslapok (link rel="stylesheet") tartalmazzák az oldalhoz tartozó CSS fájlokat, amik meghatározzák az oldal megjelenését.
* A Bootstrap egy nyílt forráskódú CSS keretrendszer, amely segít az oldal gyors és egyszerű tervezésében és fejlesztésében. Ebben a kódban Bootstrap stíluslapokat és JavaScript fájlokat is találhatunk, amelyek a Bootstrap alapú weboldalak megfelelő működéséhez szükségesek.
* A Font Awesome egy ikon és betűtípusokat tartalmazó csomag, amely lehetővé teszi a webfejlesztőknek, hogy könnyen használjanak ikonokat és betűtípusokat a weboldalukon. Itt megtalálható a Font Awesome CSS fájl, amely a betűtípusokat és ikonokat teszi lehetővé.
* A jQuery egy népszerű JavaScript könyvtár, amely lehetővé teszi a webfejlesztők számára, hogy könnyen manipulálják a HTML dokumentumot, és az interaktív weboldalak fejlesztéséhez használják. Itt a jQuery JavaScript fájlokat találjuk, amelyek a weboldalhoz szükségesek lehetnek.
* Az utolsó két JavaScript fájl a popper.js és a Bootstrap JavaScript fájlok, amelyek az interaktív elemek és a moduláris ablakok működéséhez szükségesek.

Headerben található elemek és kódok többek között a következők:

A kód egy navigációs menüt tartalmaz, amelynek elemei a "Kezdőlap", "Épületek" és "Nevezetességek" oldalakra mutató linkek. A PHP kód része az oldal felhasználói munkamenetkezelésének. Ha a felhasználó még nem jelentkezett be, akkor a regisztráció és bejelentkezés lehetőségek jelennek meg a menüben. Ha a felhasználó már bejelentkezett, akkor az "Órarend", "Profil" és "Kilépés" linkek jelennek meg, valamint egy üzenet köszönti a felhasználót. Valamint található egy kép az oldal fejlécében megjelenő logó, amelyet a "../Images/log.png" útvonalon található fájlból tölt be

.

Body-ban található elemek és kódók többek között a következők:

<body>

    <div id="content">

        <?php

        if (isset($\_GET["page"])) {

            $o = $\_GET["page"];

            $o .= ".php";

            include($o);

        } else

            include("main.php");

        ?>

    </div>

</body>

Ez a kód a weboldal fő tartalmát tartalmazza. Az oldal törzsében található egy div azonosítóval "content". Ha az URL-ben a "page" paraméter meg van adva, akkor az oldal ezt a paramétert kiegészíti a ".php" kiterjesztéssel, majd betölti az így kapott fájlt. Ha a "page" paraméter nincs megadva, akkor a "main.php" fájlt tölti be az oldal tartalmaként. Ezáltal a kód segítségével dinamikusan lehet változtatni az oldal tartalmát.[[4]](#footnote-4)

<a class="btn text-white roundeed-circle " style="background-color: #ffac44;" title="Hivatalos weboldal"

                href="https://uni-pannon.hu/hu/" target="https://uni-pannon.hu/hu/" role="button"><i

                    class="fab fa-fly"></i></a>

Footer-ben található elemek és kódók többek között a következők:

Ez egy HTML kód, amely egy weboldal láblécét hozza létre. A footer elemet definiálja, amelynek osztályai (class) a bg-dark, text-center, text-white és fixed-bottom stílusokat adják hozzá. A container osztály egy tárolót definiál, amely tartalmaz egy section elemet a közösségi média gombokkal és egy másik div elemet a szerzői jogi információkkal. A közösségi média gombokat az ikonok (<i>) együttes használatával egy a (link) elembe csomagolják. A href attribútum az egyes linkek céloldalaira mutat, míg a target attribútum meghatározza, hogy az adott link új ablakban (\_blank) vagy azonos ablakban (\_self) nyílik meg. A szöveg (title) a gombok felett jelenik meg, és az adott közösségi média platform nevét tartalmazza. A szerzői jogi információk a text-center osztály segítségével középre igazítják, és az oldal láblécének alján állnak. A script elem az aktuális évet írja ki a document.write metódus segítségével. A fas és fab osztályok a Font Awesome ikonok használatát jelzik.

### 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.

### Helyek és térkép megjelenítése a weblapon:

A kód egy HTML dokumentumot ír le, amely tartalmazza az oldal tartalmát és a struktúráját. Az oldal tartalma magyar nyelvű, és több szekciót tartalmaz, amelyek a kollégiumok és az épületek bemutatására szolgálnak.

A kód tartalmazza a HTML struktúráját, mint például a <html> és a <body> tag-eket, valamint több HTML elemet, mint például a <h2>, <h3>, <p>, <div>, <container> és <p> tag-eket, amelyek a címeket, szövegeket, képeket és más elemeket tartalmazzák, amelyek az oldal struktúráját és tartalmát meghatározzák.

Az oldal tartalmaz szöveges információkat, például a kollégiumok és az épületek bemutatását, valamint több képet és térképet is, amelyek megkönnyítik az információk megértését és bemutatását az olvasók számára.

A térképek az oldalon a map id-val ellátott divekben jelennek meg.

#### Térképek megjelenítése:

A térképek megjelenítéséhez Google Maps JavaScript API-t használom v3 ez egy olyan API (Application Programming Interface), amely lehetővé teszi, hogy a Google Maps-t integrálja webhelyébe vagy alkalmazásába a JavaScript nyelv segítségével. Valamint lehetővé teszi hogy az oldalon egyszerre több különböző térkép is megjelenjen egyidejűen a többivel.

Ez az API lehetővé teszi a felhasználók számára, hogy interaktív térképeket jelenítsenek meg weboldalukon vagy alkalmazásukban, valamint különféle helyszínek, útvonalak és térképes adatok lekérdezésére. Az API-t számos funkcióval és beállítással látják el, amelyek közül az alábbiakat használom:

* Térkép megjelenítése és testre szabása, beleértve a középpontot, a zoom szintjét és a térkép típusát
* Helyszínek és jelölők hozzáadása a térképhez, amelyekre kattintva további információk jelennek meg.

[[5]](#footnote-5)

JavaScript kód (maps.js):

let map;

var enterance = "../Images/door.png"

var car = "../Images/car.jpg"

//kk

const KKBuilding = { lat: 47.085661211826306, lng: 17.909818088728127 };

const KKEnterance = { lat: 47.08577263491504, lng: 17.9093277029896 };

const KKVCar = { lat: 47.0858504184038, lng: 17.908935327482457 };

A kód egy Google Maps térkép inicializálására és a térképen történő jelölők (marker-ek) elhelyezésére szolgáló JavaScript kód. A kód két fő részből áll. Az első részben definiáljuk a térkép különböző helyszíneit, mint például épületek, bejáratok vagy parkolók, és azok koordinátáit.

async function initMap() {

  var mapOption1 = {

    center: KKBuilding,

    zoom: 18,

    mapId: "7a17f3fb28447a5e",

    draggable: true,

  }

  map1 = new google.maps.Map(document.getElementById("map1"), mapOption1);

  const kk = new google.maps.Marker({

    position: KKBuilding,

    map: map1,

    animation: google.maps.Animation.DROP

  });

  const kkenterance = new google.maps.Marker({

    position: KKEnterance,

    title: "Központi Kollégium Épület bejárat",

    label: {

      text: "Központi Kollégium Épület bejárat",

      className: 'marker-label',

    },

    map: map1,

    icon: {

      url: enterance,

      scaledSize: new google.maps.Size(35, 35)

    },

    animation: google.maps.Animation.DROP

  });

A második részben inicializáljuk a Google Maps API-t, létrehozzuk a térképeket és hozzáadjuk a jelölőket a térképekhez. A kódban használunk aszinkron függvényeket is, amelyek lehetővé teszik az adatok aszinkron betöltését és a felhasználói felület zavartalan működését.

### Kapcsolódás az adatbázishoz:

Az adatbázis kapcsolódás olyan folyamat, amely lehetővé teszi, hogy egy program kapcsolatot építsen ki egy adatbázissal, és elérje az adatbázisban tárolt adatokat.

Az adatbázis kapcsolódás használata általában a következő lépésekből áll:

* Az adatbázis driver (illesztőprogram) betöltése: Az adatbázis kapcsolódásához szükség van egy driverre, amely lehetővé teszi a program számára, hogy kommunikáljon az adatbázissal. Az adatbázis típusától függően a megfelelő driver betöltése szükséges.
* Kapcsolat létrehozása: Miután a driver betöltődött, a programnak kapcsolatot kell létrehoznia az adatbázissal. A kapcsolat létrehozásához szükség van az adatbázis elérési útjára, a felhasználónévre és a jelszóra. Ez az információ általában a program konfigurációs fájljában vagy kódban található.
* Adatbázis műveletek végrehajtása: Miután a kapcsolat létrejött, a program végrehajthatja az adatbázis műveleteket, például adatok lekérdezése, beszúrása, frissítése vagy törlése az adatbázisból.
* Kapcsolat lezárása: Amikor a program befejezte az adatbázis műveleteket, fontos, hogy a kapcsolatot lezárjuk, hogy felszabaduljon az erőforrás és ne fogyasszon feleslegesen rendszer erőforrásokat.

Az adatbázis kapcsolódás általában szükséges, ha egy program adatokat szeretne elérni vagy manipulálni egy adatbázisban. Az adatbázis kapcsolat létrehozása és lezárása nagyon fontos a program hatékony és biztonságos működése szempontjából.

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);

A későbbiekben az adatbázis műveletek során mindig ezt hívom meg.

### Regisztráció:

A weboldalra történő regisztráció a következőképpen történik:

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 reguláris kifejezések (regexek) nagyon hasznosak lehetnek az HTML elemzése során, mivel lehetővé teszik, hogy kiválasszuk és manipuláljuk az HTML kód egyes elemeit. Az HTML regex kifejezések általában a HTML tagok, attribútumok és tartalmak kiválasztására szolgálnak.

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

                        pattern="([A-Z]([a-záéúőóüö.]{1,50}\s?)){1,}" required

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

A fenti kód egy reguláris kifejezés mintát határoz meg. Ez a minta egy szövegmezőben bevitt szöveg érvényességét ellenőrzi, amelynek követelménye, hogy a szöveg csak betűkből álljon, és az első betű nagybetű legyen, a többi kisbetű, a szavak között egy vagy több szóköz legyen, és a szöveg összesen 1 és 40 szóból álljon.

A minta leírása részletesen:

* ([A-Z] - A minta azzal kezdődik, hogy egy nagybetűvel kell kezdődnie a szövegnek
* ([a-záéúőóüö.]{1,}\s?)) - Ez a rész az első betű utáni részre vonatkozik. Itt az a feltétel van, hogy a szöveg csak betűket (kis és nagybetűket) tartalmazhat, illetve a magyar ékezetes betűket is, és lehet benne egy vagy több szóköz, illetve az utolsó szó után nem kell szóköznek lennie. A .{1,50} azt jelenti, hogy legalább 1-50 karakternek karakternek kell lennie a szóban, a \s? pedig azt jelenti, hogy ha van szóköz, akkor lehet, hogy csak 1 vagy nincs is.
* {1,} - A minta azt határozza meg, hogy a szövegben egy szó lehet csak.

Az required attribútum azt jelenti, hogy a szövegmező nem lehet üres, tehát a felhasználónak legalább 1 szót be kell írnia a mezőbe, hogy az érvényes legyen.

Tehát összefoglalva, ez a minta egy olyan szövegrészletre illeszkedik, amely legalább egy nagybetűvel kezdődik, majd kisbetűs karakterek, ékezetes betűk, vagy pontok követik, amelyek között lehet szóköz is, és a szövegnek legalább egy karakterből kell állnia. Ez a minta általában a nevek, címek, vagy hasonló szöveges adatok ellenőrzésére használatos, amelyek bizonyos formai szabályoknak kell megfelelniük.

Az adatok POST metódussal történnek elküldésre. A POST metódus az HTTP protokoll egyik módja a szerverrel történő kommunikációra, amely lehetővé teszi az adatok küldését a kliensből a szerverre. A PHP nyelvben a POST metódus használatához a $\_POST tömböt kell használni. Ez a tömb a POST kérésben küldött adatokat tartalmazza. A POST metódus használatakor a küldött adatok a HTTP kérés testében (request body) kerülnek elküldésre, így a kérés URL-je nem tartalmazza a küldött adatokat, mint a GET metódus esetében. A POST metódus használatakor általában űrlapok (form) küldése történik, amelyekben a felhasználó kitölti az űrlap mezőit, majd azokat elküldi a szervernek a "regisztráció" gomb megnyomásával. A szerver általában feldolgozza a küldött adatokat, majd valamilyen választ küld vissza a kliensnek. A következő kódrészlet bemutatja, hogyan lehet használni a POST metódus:

HTML kód (regrom.php):

<form name="registartionform" class="registration" method="POST">

    <label>Vezetéknév</label>

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

     pattern="([A-Z]([a-záéúőóüö.]{1,50}\s?)){1,}" required

   oninvalid="this.setCustomValidity('Vezétéknéknév megadásakötelező')">

     <button type="submit" name="reg">Regisztráció</button>

</form>

A fenti űrlap elküldi a „firstname”,”lastname”,”email”,”neptun” és „password” mezőket a registration.php fájlnak a POST metódus segítségével.

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

    if($sql)

    {

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

    }

    else{

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

    }

Az alábbi kód a regisztrációs űrlap feldolgozására szolgál. Az űrlap POST metódus segítségével küldi el a megadott adatokat. A kód az űrlapból beolvas néhány változót, például a felhasználó nevét, e-mail címét, jelszavát stb. Ezután ellenőrzi, hogy az adatok megfelelő formátumban vannak-e, például a neptunkód hossza 6 karakter, a jelszó legalább 8 karakter stb. Ha minden adat megfelelő, akkor az adatokat beszúrja az adatbázisba. Ha az adatok nem megfelelőek, akkor a kód hibaüzeneteket jelenít meg a felhasználónak, amelyek közlik, hogy mi a probléma felhasználó által megadott adatokkal. A kód használ egy SQL SELECT utasítást is, amely megkeresi az adatbázisban azonos neptunkóddal rendelkező felhasználókat. Mert ha talál azonos neptunkóddal rendelkező felhasználót akkor egy értesítést küld a felhasználó felé hogy a egy ilyen felhasználó már létezik az adatbázisban.

A jelszavak megfelelő titkosításának érdekében, a jelszavak MD5 ös titkosítással vannak eltárolva az adatbázisban. Az md5 egy hash függvény, amely matematikai algoritmusokat használ arra, hogy bármilyen hosszúságú szöveget vagy bájtsorozatot átalakítson egy rövidebb, fix hosszúságú karakterláncba. Az md5 algoritmus 128-bites hash értéket generál. Az md5 függvényt gyakran használják jelszavak titkosítására. Weboldal által tárolt jelszó helyett a weboldal az md5 hash értékét tárolja el a felhasználó jelszaváról.

### Bejelentkezés:

A weboldalra történő bejelentkezés nagyon hasonlóan fog megtörténni a regisztrációhoz. A felhasználó neptunkód és jelszó páros megfelelő megadása után tud majd belépni az oldalra a plusz funkciók elérése érdekében. Ha a felhasználó belép a weboldalra, a weboldal az md5 hash értékét számolja ki a felhasználó által megadott jelszóról, majd összehasonlítja az adatbázisban tárolt md5 hash értékkel. Ha a két hash érték megegyezik, akkor a felhasználó sikeresen beléphet a weboldalra.

A regisztrált felhasználóknak, akik beléptek már a felületre, megjelenik három új menüpont: kilépés, profil és órarend feltöltése címmel. Az órarend feltöltése a menüpontra kattintva a felhasználónak lehetősége lesz a neptunból kiimportált xlsx-file típusu órarend feltöltésére. A profil menüpontban lehetőségük lesz jelszó, Email és a regisztráláskor megadott nevük módosítására, valamint felhasználói fiókjuk törlésére. A kilépés menüpontra kattintva a felhasználó ki tud jelentkezni fiókjából

A PHP kód itt láttható: (login.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");

}

Ez a kód egy bejelentkezési űrlap kezelőscript, amely egy adatbázisból ellenőrzi a felhasználó által megadott Neptun-kódot és jelszót.

Az alábbiakat teszi:

* Az űrlapból kapott "neptuncode" és "password" adatokat kinyeri a POST tömbből, majd eltárolja őket a $nc és $pw változókban.
* A $pw jelszót MD5 hash értékre titkosítja a md5() függvény segítségével, majd eltárolja a $titkositott változóban.
* Az $nc változóban lévő Neptun-kódot nagybetűsre alakítja a strtoupper() függvény segítségével, majd eltárolja a $ncfel változóban.
* Az adatbázisban keres egy olyan sort a registered\_users táblában, amelyben a "neptun" oszlop értéke megegyezik a $ncfel értékével, és a "password" oszlop értéke megegyezik a $titkositott értékével. A lekérdezés eredményeit az $sql változóban tárolja.
* Megszámolja, hogy hány sor került visszaadásra a $sql lekérdezésből, majd az eredményt eltárolja az $exists változóban.
* Ellenőrzi, hogy az $exists változó értéke 1-e, ha nem, akkor kiírja a "Hibás neptunkód vagy jelszó!" üzenetet. Ha igen, akkor az "id" nevű SESSION változóba eltárolja a felhasználó Neptun-kódját, majd átirányítja a felhasználót a főoldalra (?page=main) a header() függvény segítségével.

### Profil:

Az oldal egy űrlapot jelenít meg a felhasználói adatok módosításához, amelyeket a felhasználó a szövegmezők kitöltésével módosíthat. Az adatokat a 'registered\_users' táblából tölti be, és a 'personal.php' fájlt is beleértve, ha az űrlapot elküldték. A PHP kód végrehajtja a felhasználó által beküldött adatok ellenőrzését, majd frissíti a 'registered\_users' táblát az új adatokkal. Az oldal tartalmaz még egy "Fiók törlése" gombot is, amely törli a felhasználói fiókot, amikor rákattintanak.

A PHP kód itt láttható: (personal.php):

$$surname = $\_POST['newsurename'];

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

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

$passw = $\_POST['newpassword'];

$passowrdagain = $\_POST['newpasswordagain'];

$id = $\_SESSION["id"];

if ($passw != "" && $passw == $passowrdagain) {

    $titkositott = md5($passw);

}

$sql = false;

if (isset($\_SESSION["id"])) {

    if($surname!="")

    $sql = $conn->query("UPDATE registered\_users set firstName = '$surname' WHERE neptun LIKE '$id'");

    if($lastname!="")

    $sql = $conn->query("UPDATE registered\_users set lastName = '$lastname' WHERE neptun LIKE '$id'");

    if($email!="")

    $sql = $conn->query("UPDATE registered\_users set email = '$email' WHERE neptun LIKE '$id'");

    if($passw!="")

    $sql = $conn->query("UPDATE registered\_users set `password` = '$titkositott' WHERE neptun LIKE '$id'");

    if ($sql) {

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

    } else {

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

    }

}

Ez a PHP kód az űrlap által elküldött adatok feldolgozásáért felelős. Az űrlap azon mezőinek értékeit, amelyeknek a neve a "newsurename", "newlastname", "newemail", "newpassword" és "newpasswordagain", az $\_POST szuperglobális tömbön keresztül kapja meg. A kód először eltárolja a bejelentkezett felhasználó azonosítóját az $\_SESSION szuperglobális tömbből.

A következő if utasítás ellenőrzi, hogy a két jelszómező (newpassword és newpasswordagain) tartalma megegyezik-e, és ha igen, akkor titkosítja a jelszót a PHP beépített md5() függvényével.

A következő if utasítás ellenőrzi, hogy a felhasználó be van-e jelentkezve (azaz van-e értéke az $\_SESSION["id"] változónak), majd frissíti az adatbázisban a megfelelő mezőket a megfelelő értékekkel.

Ha a frissítés sikeres, a kód "Sikeres adatmódósítás!" szöveget jelenít meg a felhasználónak, ellenkező esetben pedig "Sikertelen adatmódósítás" szöveget jelenít meg. A kód nem végez semmilyen adatellenőrzést, mint például a mezők kitöltésének ellenőrzését, mert a felhasználónak nem muszály megadnia minden adatot az adatok frissítésekor, ugyanis változtathat meg csak jelszót, nevet vagy emailcímet..

### Órarend feltöltése:

A PHP kódrészlet itt láttható: (callendar.php):

ini\_set('display\_errors', 1);

ini\_set('display\_startup\_errors', 1);

error\_reporting(E\_ALL);

include "../connections/connection.php";

if (isset($\_POST["save\_callendar"])) {

    include\_once("../connections/callendarupload.php");

}

Ez egy PHP kód, amely egy órarend feltöltési és megjelenítési funkcionalitást valósít meg. A kód elején az "ini\_set" függvények beállítják a hibakeresést a képernyőn, majd a "connection.php" fájl betölti az adatbázis kapcsolatot.

A PHP kódrészlet itt láttható: (callendar.php):

 <div class="col-md-12 mt4">

        <form class="form-inline" action="" method="POST" name="callendar" enctype="multipart/form-data"

            action="<?php echo $\_SERVER['PHP\_SELF']; ?>">

            <div class="form-group mb-2">

                <input class="upload" type="file" name="import\_file" id="import\_file">Exel file (xlsx)

            </div>

            <div class="form-group mx-sm-3 mb-2">

                <button type="submit" name="save\_callendar" class="btn btn-primary mt-3" <?php echo $disabled; ?>>Feltöltés</button>

            </div>

            <button type="submit" onclick="deleteCallendar()" name="delete" class="btn btn-danger" <?php echo $disabled2 ?>>órarend törlése</button>

        </form>

    </div>

<?php

$result = $conn->query($sql);

echo $conn->error;

// HTML táblázat létrehozása

echo "<table class='table table-dark'>";

echo "<tr><th scope='col'>Név</th><th scope='col'>Dátum</th scope='col'><th>Időpont</th><th scope='col'>Helyszín</th><th scope='col'>Rövid összefoglaló</th></tr>";

if ($result->num\_rows > 0) {

    while ($row = $result->fetch\_assoc()) {

        // Dátumok és időpontok formázása

        $start = date("Y-m-d H:i", strtotime($row["start"]));

        $end = date("Y-m-d H:i", strtotime($row["end"]));

        // HTML táblázat sor létrehozása

        echo "<tr>";

        echo "<td>" . $row["name"] . "</td>";

        echo "<td>" . $start . " - " . $end . " </td>";

        echo "<td>" . date("H:i", strtotime($row["start"])) . " - " . date("H:i", strtotime($row["end"])) . " </td>";

        echo "<td>" . $row["place"] . " </td>";

        echo "<td>" . $row["summary"] . " </td>";

        echo "</tr>";

    }

A fenti kódrészlet egy HTML oldalt ír le, amely tartalmaz egy űrlapot, amely lehetővé teszi az órarend feltöltését és törlését. Az oldal a következőket tartalmazza: Az oldal betöltésekor a dokumentum.callendar.reset () függvény meghívódik, amely visszaállítja az űrlap tartalmát. Egy h2 címsor a "Órarend" szöveggel, majd egy rövid leírás található a div.description osztályban. Ezután egy "Töltsd fel az órarended itt" szöveggel ellátott h3 címsor következik. Az űrlap tartalmaz egy exel fájl feltöltésére szolgáló mezőt és egy feltöltés gombot, amely az űrlap adatait az oldalhoz visszaküldi a POST módszer segítségével. Az oldal azt is megvizsgálja, hogy az adott felhasználónak van-e már órarendje az adatbázisban, és ha van, akkor letiltja a feltöltés gombját és engedélyezi a törlés gombját, és fordítva, ha nincs órarendje. Az oldal a MySQL adatbázisból lekérdezi az órarendhez szükséges adatokat, majd ezeket egy HTML táblázatba helyezi, amelyet a felhasználó megtekinthet. Végül a kód lezárja a HTML dokumentumot és lecsatlakozik az adatbázisról.

A PHP kódrészlet itt láttható: (callendarupload.php):

ob\_start();

require '../vendor/autoload.php';

ini\_set('display\_errors', 1);

ini\_set('display\_startup\_errors', 1);

error\_reporting(E\_ALL);

use PhpOffice\PhpSpreadsheet\Spreadsheet;

use PhpOffice\PhpSpreadsheet\Writer\Xlsx;

$id = $\_SESSION['id'];

if (isset($\_POST['save\_callendar'])) {

    $fileName = basename($\_FILES["import\_file"]["name"]);

    $file\_ext = pathinfo($fileName, PATHINFO\_EXTENSION);

    $allowed\_ext = ['xls', 'xlsx', 'csv'];

A fenti PHP kód a következőképpen működik: Először elindít egy buffer-t (ob\_start()), ami azután a végén elhagyja az összes kimeneti adatot.

Ezután betölti a PhpSpreadsheet nevű PHP könyvtárat a Composer autoload.php fájlon keresztül, ami lehetővé teszi a Microsoft Excel XLSX, XLS és CSV fájlok kezelését a PHP-ban. Beállítja a hibajelentési szintet, hogy minden hiba és figyelmeztetés megjelenjen a képernyőn. A PHP kód ellenőrzi, hogy az $\_POST változóban a 'save\_callendar' kulcs értéke be lett-e állítva. Ha igen, akkor folytatja a kód futtatását, ha nem, akkor a kód végrehajtása befejeződik. A kód beolvassa a feltöltött fájl nevét, majd meghatározza a fájl kiterjesztését. Az engedélyezett fájl kiterjesztéseket (xls, xlsx, csv) ellenőrzi, hogy a feltöltött fájl engedélyezett formátumú-e.

A PHP kódrészlet itt láttható: (callendarupload.php):

 if (in\_array($file\_ext, $allowed\_ext)) {

        $idExtension = 0;

        $fileName = $\_FILES["import\_file"]["tmp\_name"];

        /\*\* Load $inputFileName to a Spreadsheet object \*\*/

        $spreadsheet = \PhpOffice\PhpSpreadsheet\IOFactory::load($fileName);

        $data = $spreadsheet->getActiveSheet()->toArray();

A fenti kódrészlet ellenőrzi, hogy a feltöltött fájl kiterjesztése engedélyezett-e az $allowed\_ext tömbben meghatározott engedélyezett kiterjesztések között. Ha igen, akkor beolvassa a fájl tartalmát egy PhpOffice\PhpSpreadsheet\Spreadsheet objektumba a load() metódus segítségével. Ezután az objektum aktuális munkalapjának adatait tömbként lekéri a toArray() metódus segítségével, és eltárolja a $data változóban. Az $idExtension változó kezdeti értéke 0 lesz. Ha a feltöltött fájl kiterjesztése nem engedélyezett, akkor a program a megfelelő hibaüzenettel kilép.

A PHP kódrészlet itt láttható: (callendarupload.php):

foreach ($data as $row) {

            if ($count > 0) {

                $start = $row['0'];

                $end = $row['1'];

                $num = 2;

                $temp\_start = str\_replace(". ", "-", $start, $num);

                $temp\_end = str\_replace(". ", "-", $end, $num);

                $start\_formatted = str\_replace(". ", " ", $temp\_start);

                $end\_formatted = str\_replace(". ", " ", $temp\_end);

                $name = $row['2'];

                $place = $row['3'];

                $summary = $row['4'];

                $id\_str = $id . $idExtension;

                try {

                    $conn->autocommit(FALSE);

                    $msg1 = $conn->query("INSERT INTO classes(`id`,`start`,`end`,`name`, `summary`,`place`) VALUES('$id\_str', '$start\_formatted','$end\_formatted','$name', '$summary','$place');");

                    $msg2 = $conn->query("INSERT INTO taken\_classes(`neptun`,`classes\_id`) VALUES ('$id','$id\_str');");

                    if ($msg1) {

                        if ($msg2) {

                            $conn->commit();

                            $idExtension++;

                            $msg=true;

                        } else {

                            throw new Exception($conn->error);

                        }

                    } else {

                        throw new Exception($conn->error);

                    }

                } catch (Exception $ex) {

                    $conn->rollback();

                    $conn->autocommit(TRUE);

                }

            } else {

                $count = 1;

            }

        }

        if ($msg != false) {

            $\_SESSION['message'] ="Sikeres órarend feltöltés!";

            echo "<script>window.location.href='index.php'</script>";

        } else {

            $\_SESSION['message'] = "Sikertelen órarend feltöltés!";

            exit(0);

        }

A kódrészlet egy PHP script része, amely feldolgozza egy adatfájl tartalmát és feltölti az adatokat egy adatbázisba. A feldolgozás a következőképpen történik:

A "foreach" ciklus végigiterál minden "data" változóban tárolt soron, amelyet az adatfájl tartalmaz. Az "if ($count > 0)" feltétellel ellenőrzi, hogy a ciklus az első sortól kezdődjön-e, vagy sem. Ha az első sortól kezdődik, akkor a ciklus folytatódik, ha nem, akkor a "count" változó értéke 1-re lesz állítva. A következő sorok a feldolgozás részét képezik. A "start" és "end" változók a kezdés és a vég dátumát tartalmazzák, amelyek az adatfájl első két oszlopában találhatók. A "str\_replace" függvények az időpontok formázását végzik, amelyek helyett a pontokat kötőjelekre cserélik, majd a szóközöket az időpontokban visszaállítják. Az így formázott időpontokat "start\_formatted" és "end\_formatted" változókban tárolják. A "name", "place" és "summary" változók a sor további oszlopaiban található adatokat tartalmazzák. Az "id\_str" változóban az "id" és "idExtension" változók összeillesztett értéke tárolódik. Ezután a script a "try-catch" struktúrában található. Az adatokat az "INSERT INTO" SQL utasításokkal kerülnek beillesztésre az adatbázisba. A "msg1" változóban tárolja, hogy sikerült-e beszúrni az adatokat a "classes" táblába. A "msg2" változóban tárolja, hogy sikerült-e beszúrni az adatokat a "taken\_classes" táblába. Ha mindkét beszúrás sikeres volt, akkor a script elköveti a módosításokat, és hozzáadja az "idExtension" változó értékét. Ha bármelyik beszúrás nem sikerült, akkor a "catch" blokk fut, ahol a "rollback" függvény visszaállítja a tranzakciót az eredeti állapotába, majd a "autocommit" függvénnyel beállítja a visszatérési értéket.

### Órarend törlése.

A PHP kód itt láttható: (callendarDelete.php):

include\_once "../connections/connection.php";

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

try {

    $resultSet = $conn->query("SELECT classes\_id FROM taken\_classes WHERE neptun LIKE '$id';");

    if ($resultSet->num\_rows > 0) {

        $conn->autocommit(FALSE);

        while ($row = ($resultSet->fetch\_assoc())) {

            if ($row) {

                $conn->query("DELETE FROM classes WHERE id like '$row';");

            }

        }

        $sql = $conn->query("DELETE FROM taken\_classes WHERE neptun LIKE '$id';");

        $conn->commit();

        $\_SESSION['message'] = "Sikeres órarend törlés!";

        $conn->autocommit(TRUE);

    } else {

        throw new Exception($conn->error);

    }

} catch (Exception $ex) {

    $conn->rollback();

    $conn->autocommit(TRUE);

    $\_SESSION['message'] = "Sikertelen órarend törlés!" . $ex.message;

}

A következő PHP kód a következőket hajtja végre: Beimportálja a connection.php fájlt, amelyben az adatbáziskapcsolat és az adatbázis objektum inicializálása található. Az id változóba eltárolja a $\_POST globális tömbön keresztül kapott azonosítót.

Lekéri az összes classes\_id értéket a taken\_classes táblából, amelynek neptun oszlopa megegyezik az adott id értékkel. Ha legalább egy classes\_id érték van, akkor elkezdi a tranzakciót, amely lehetővé teszi a többi lekérdezéshez és művelethez történő hozzáférést és azok egyidejű végrehajtását. A törlési lekérdezéseket hozzáadja a kapcsolat objektumhoz, de nem hajtja végre azokat azonnal, hanem vár a tranzakció véglegesítésére. Az összes olyan rekordot törli a taken\_classes táblából, amelynek neptun oszlopa megegyezik az adott id értékkel. Ezzel egyszerre több rekordot is töröl a táblából. A tranzakció végrehajtása. Ha nem történt kivétel, akkor beállítja a $\_SESSION['message'] változót, hogy sikeres volt a törlés. A kapcsolat objektum automatikus véglegesítésének visszaállítása. Ha kivétel keletkezik a tranzakció végrehajtása közben, akkor a rollback() függvény visszaállítja a változtatásokat, és a $\_SESSION['message'] változóban hibaüzenetet jelenít meg a felhasználónak. A kapcsolat objektum automatikus véglegesítését szintén visszaállítja a kivétel keletkezése előtti állapotra.

# Használat

A program használata felettébb egyszerű, a felhasználónak nem kell telepítenie semmit az eszközére, csak tudnia kell a weboldal elérhetőségét, valamint rendelkeznie kell az eszközén internet kapcsolattal. Az alkalmazás itt látható:

## Kezdőlap:

A képen szöveg, elektronika, képernyőkép, képernyő látható

Automatikusan generált leírás

2. ábra: A weblap kezdőoldala. [1]

A kezdőlapon található egy rövid bemutató az alkalmazásról. Az oldalon szintén megtalálható az Egyetem Neptun oldala, valamint a karok moodle oldalának elérhetőségei. Itt látható még egy rövid leírás az egyetemről és egy kis egyetemi történelem. Az lap tetején található a fő navigációs sáv, amik segítésével az oldal későbbi használata során tudunk navigálni.

Ebben a sávban többek között elérhetők el weboldal fő oldala is. Az oldal alján található navigációs sávon található az Egyetem hivatalos weboldala, Facebook profilja, YouTube csatornája, valamint Instagram oldala. A további két menü pont a készítő GitHub oldala, illetve Buy me a cofee oldala. Továbbá az oldalon kezdőoldalon megtalálható még az Egyetem neptun oldala, valamint a karok Moodle oldalai.

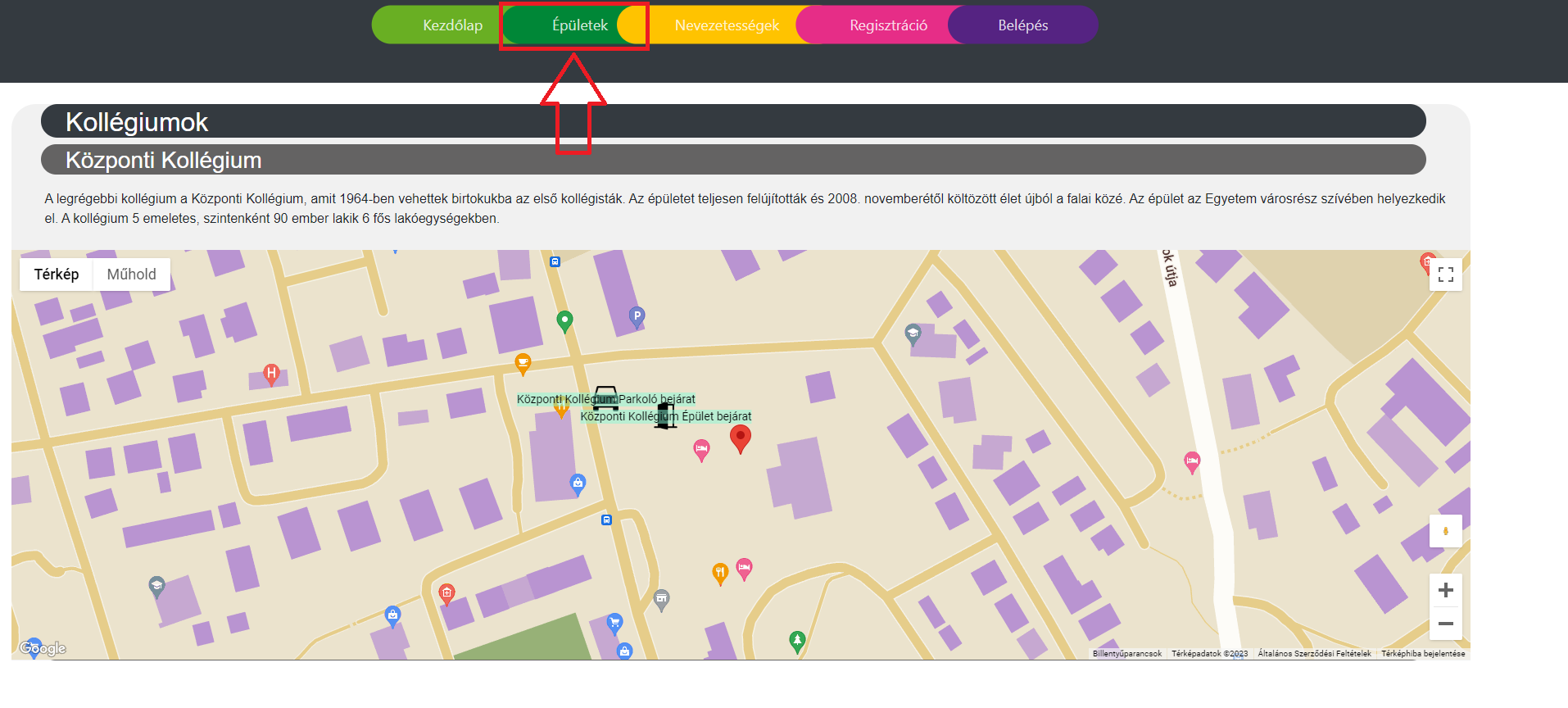


3. ábra: Neptun és Moodle jelzők.. [1]

A felhasználó a megfélő gomb lenyomása után átirányításra kerül a címnek megfelelő oldalra egy új lapon.

## Épületek

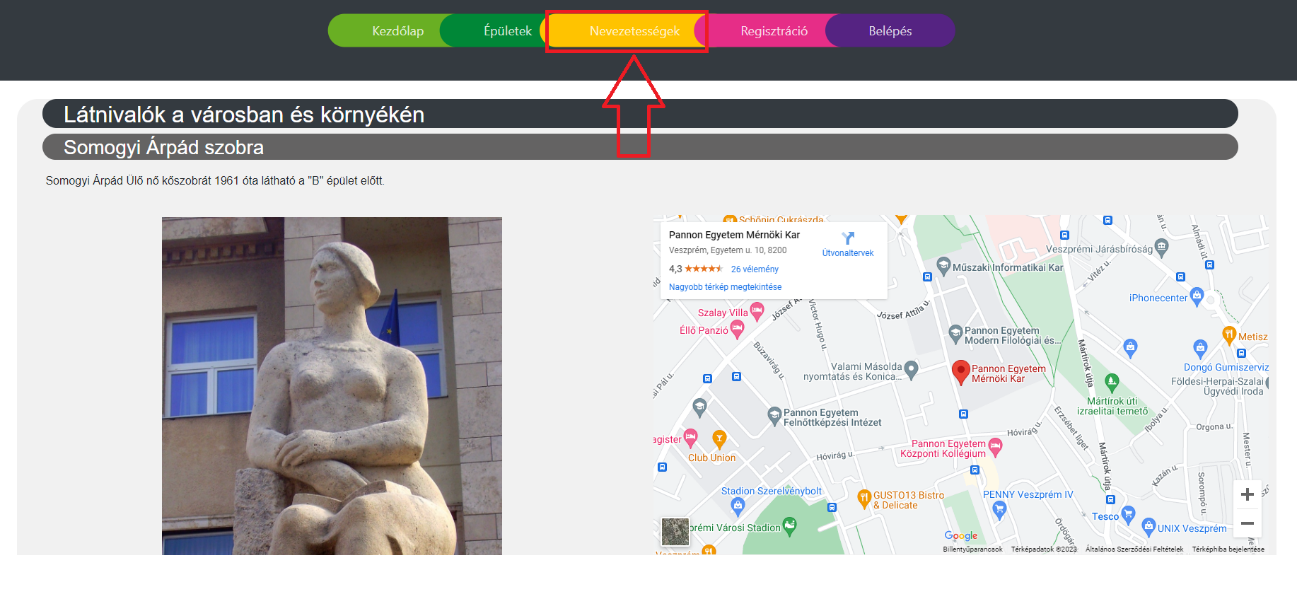
Az Épületek menüpontra a következő oldal jelenik meg:



4. ábra: Épületek. [1]

Az épületek menüpontban találhatóak meg az egyetem épületei, valamint a kollégiumok. Minden egyes épület rendelkezik egy rövid leírással, illetve egy térképpel. A térképen piros jelölővel vannak jelölve az épületek. A térképpen továbbá megtalálhatók az épület bejáratai, amelyek ajtóval és a parkolók bejáratai, amik egy autó ikonnal jelöltem. Ahogy már említettem, ezt a funkciót az újonnan érkező egyetemisták megsegítésére hoztam létre. Mivel az egyetem épületei nem csak egy helyen vannak, hanem karonként az épületek helyszíne is változik, így időmegtakarítás céljából szintén hasznos lehet.

## Nevezetességek

A Nevezetességek menüpontra a következő oldal jelenik meg:

. ábra: Nevezetességek. [1]

Nagyon hasonlóan az épületek menüponthoz itt is minden nevezetesség/látnivaló rendelkezik egy rövid leírással, valamint egy jelölővel a térképen, hogy hol található meg. Továbbá minden látnivalóhoz tartozik egy kép is. Ez a menüpont hasznos lehet azoknak, akik még csak gondoltodnak az egyetem kiválasztásán. Hiszen megmutatja nekik, milyen infrastruktúrával rendelkezik a város.

## Regisztráció

A Regisztráció menüpontra a következő oldal jelenik meg:

A képen Webhely látható

Automatikusan generált leírás

. ábra: Regisztrációs felület. [1]

A felhasználóknak a regisztráció során meg kell adniuk a Vezetéknevüket, Keresztnevüket, Emailcímüket, Neptun kódjukat, valamint a jelszavukat. A regisztráció során minden mező kitöltése kötelező a felhasználók számára. Ezen kívül regisztráció véglegesítése előtt bele kell egyezniük, hogy elfogadják a Felhasználási feltételek és az Adatkezelési tájékoztatót. Sikeres vagy sikertelen regisztráció esetén is a felhasználó tájékoztatást kap a regisztráció állapotáról. A jelszó megerősítő funkció, mint sok weboldalnál alkalmat biztosít arra, hogy a felhasználó kizárja az elgépelt jelszónak a lehetőségét.

## Bejelentkezés:

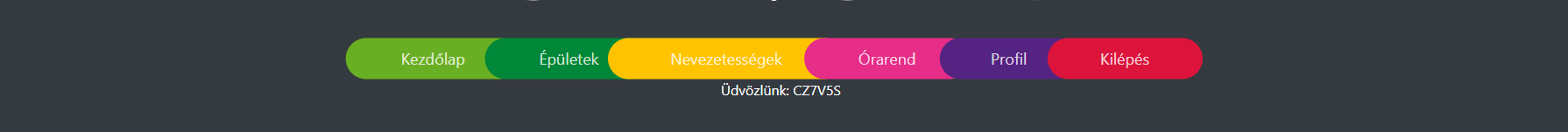
A Bejelentkezés menüpontra kattintva a következő oldal jelenik meg:

A képen szöveg látható

Automatikusan generált leírás

7. ábra: Bejelentkező felület. [1]

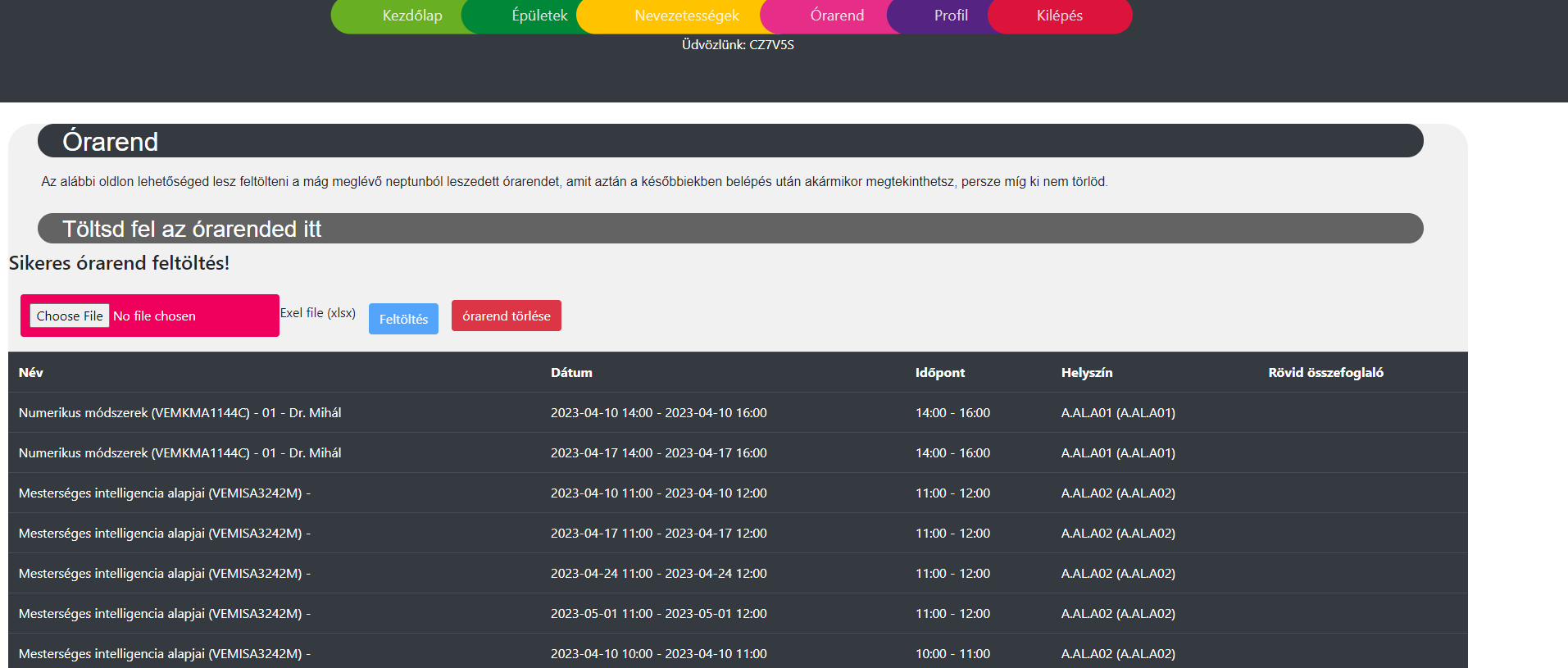
A sikeres regisztrációt követően a felhasználó itt léphet be a Neptun kódja, illetve megadott jelszava segítségével. A felhasználó sikeres bejelentkezés után már a lefrissült navigációs menüsort látja a plusz funkciókkal, valamint egy üdvözlő üzenetet.



8. ábra: Menüsor bejelentkezés után. [1]

## Órarend:

Az Órarend oldalra kattintva a következő oldal jelenik meg:



. ábra: Órarend feltöltési felület már egy feltöltött órarendel. [1]

Ez a weboldal az amely az órarendet jelenít meg, amelyet a user tud feltölteni, megtekinteni és törölni. A felhasználó a choose file gombra kattintva feltölteni kívánt órarend file-t amik a következő típusok lehetnek (xlsx, xls, csv). Ha a user ennél különböző típusú file-t próbál feltölteni a rendszer értesíti őt hogy a feltöltött file típusa nem megfelelő és hogy az feltöltés sikertelen. Sikeres órarend feltöltés után megjelenik az adott felhasználóhoz tartozó órarend törlésig. Az órarend törlése gombra kattintva a feltöltött órarendjét tudja kiüresíteni. A gombra kattintás után egy megerősítő üzenet jelenik meg ami a következő::

A képen szöveg látható

Automatikusan generált leírás

. ábra: Órarend törlése megerősítő üzenet. [1]

Az ok-gombra kattintva a felhasználó véglegesíti az órarendje törlését. Az oldalon a sikeres törlés után az órarend táblázat üresen fog megjelenni, újabb feltöltésig.

## Profil:

A Profil oldalra kattintva a következő oldal jelenik meg:

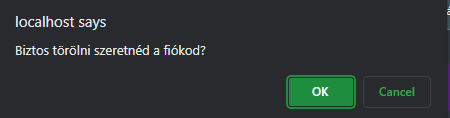
A képen Webhely látható

Automatikusan generált leírás

11. ábra: Profil. [1]

A felhasználónak a későbbiekben ezen az oldalon lesz lehetősége megváltoztatni a regisztráció során megadott adatait módosítani, valamint törölni a fiókját. A from bal oldalán találhatók meg a felhasználó megadott adatai (vezetéknév, keresztnév, emailcím). A jobb oldalon találhatók a mezők, ahol a felhasználó meg tudja változtatni megadott adatait. A megváltoztatáshoz nem szükséges minden egyes alkalommal az összes adatot megadni, elég csak a változtatni kívánt adat mezőjét kitölteni.

Ez lehetővé teszi a komfortosabb adatváltoztatást. A felhasználónak regisztráció után nincsen lehetősége a Neptun kódját megváltoztatni, mert ez a felhasználók egyedi azonosítója. A változtatások mentése gombra kattintva a felhasználó elmenti a megadott adatait, a mégse gomra kattintva pedig visszavonhatja a változatin kívánt adatok módosítását. A fiók törlésre kattintva meg a fiókját tudja törölni egy megerősítő üzenet után, ami a következő:



12. ábra: Fiók törlése megerősítő üzenet. [1]

Az ok-gombra kattintva a felhasználó véglegesíti a fiókra törlését. A felhasználó fiókja törlése után már többet nem fog tudni belépni, csak újabb regisztráció után.

A megszakítás gombra kattintva a fiók törlése nem történik meg. A megerősítő gomb meggátolja, hogy a felhasználó véletlenül kitörölje a fiókját, így egy fajta biztonságot nyújt a kliensnek.

## Kilépés:

A kilépés gombra kattintva a felhasználó kilép a fiókjából és visszatér a kezdőlapra. Itt újra a bejelentkezés előtti formokat találja az felhasználó.

# Továbbfejlesztési lehetőségek

A fejlesztési folyamat közeben számos ötlettel gazdagodott a projekt, melyek közül néhány megvalósítását a későbbiekre tűztük ki célul, miután az alapprogram végleges verziója elkészült. Ezek egyike például a Google fiókkal történő bejelentkezés, illetve megerősítő email küldése a felhasználóknak. Valamint felmerült még olyan, hogy az órarend feltöltése felületre kerülhet egy beépített naptár ami mindig mutatja az aktuális dátumot.

# Összefoglaló

Összegesében szerinem sikerült egy könnyen használható de mégis kicsit az átlagostól eltérő egyedi weblapot létrehozni, ami másmilyen szempontból közelíti meg kicsit az egyetemet. Egy olyan projekt került megvalósításra ami majdnem teljesen magába foglalja és összevonja a „konkurens” Pannon Egyetemes weblapokon található funkciókat. Valamint sikerült egy színes felületet létrehoznom amit szívesen használnának a diákok.

A funkciókat amiket terveztem sikerült belerakni plusz sikerült implementálni a profil funkciókat is, de mindig van lehetőség a jövőben a bővítésre. A térképek beszúrásakor ütköztem nagyobb nehézségekbe ugyanis ezt a technológiát még nem használtam korábban. A fő problémát az okozta hogy csak az GoogleMaps API V3-as verziója engedélyezi hogy egy felületen egyszerre több térképet is megjeleníthess egyidejűleg. Ez a megoldás azért hasznos mert a felhasználóknak egy mindig központilag frissített és valós és friss információ áll rendelkezésére. Valamint kisebb nagyobb nehézségekbe ütköztem az adatbázisok tervezése és megvalósítása közben, ugyanis előszőr nem is gondoltam bele hogy mennyire fontos az adatok megfelelő és hibamentes kezelése, de szerencsére sikerült túljutni a nehézségeken. Szerencsére regex függvényekkel és ellenőrzésekkel sikerült megoldani hogy a felhasználó csak helyes információkat tudjon megadni. A file feltöltéseket és törléseket pedig Tranzakciók segítségivel sikerült megoldani, tehát ha valami hiba keletkezik vagy a felhasználó vagy a szerver oldaláról, akkor nem kerülnek fel hiányos vagy sérült adatok az adatbázisba.

Irodalomjegyzék

|  |  |
| --- | --- |
| [1] | “https://www.php.net/docs.php”. |
| [2] | „https://phpspreadsheet.readthedocs.io/en/latest/”. |
| [3] | „https://developers.google.com/maps/documentation/javascript”. |
| [4] | „https://gyires.inf.unideb.hu/GyBITT/12/ch01.html”. |
| [5] | „http://vbence.web.elte.hu/regex\_leiras.html”. |
| [6] | „https://getbootstrap.com/docs/5.3/getting-started/introduction/”. |
| [7] | „https://fontawesome.com/”. |
| [8] | „http://info.berzsenyi.hu/adatbazisok/az-adatbazis-fogalma”. |
| [9] | Kreanilla, „https://kreanilla.hu/jo-design-kedvenc-dizajn-idezeteim/”. |

Mellékletek

Mappaszerkezet

+-connections

| callendarupload.php

| deleteCallendar.php

| delete.php

| login.php

| login.php

| linkek.json

| personal.php

| registartion.php

|

+-Images

| cannyon.jpg

| car.jpg

| diploma.jpg

| door.png

| geszti.jpg

| Github.png

| gizellakilato.jpeg

| google.png

| hm.jpg

| hosok.jpg

| kk1.jfif|

| kk2.jfif

| log.png

| login.jpg

| log.png

| ovaros.jpg

| registration.jpg

| slideshow1.jpg

| slideshow2.jpg|

| slideshow3.jpg

| szentistv.jpg|

| szobor1.jpeg

| szobor2.jpeg

| szobor3.jpg

| tuztorony.jpg

| varhegy.jpg

|

+-javascript

| callendar.js

| maps.js

| menu.js

|

+-Pages

| building.php

| callendar.php

| index.php

| login\_field.php

| logout.php

| main.php

| places.php

| profile.php

| regform.php

|

+-style

| building.css

| callendar.css

| login.css

| main.css

| menu.css

| page.css

| profile.css

| registration.css

|

+-vendor

|-------------------

| composer.json

| composer.lock

| uniteduniversities.sql

Ábrajegyzék

[1. ábra: Adatbázis modell.. [1] 21](#_Toc134367697)

[2. ábra: A weblap kezdőoldala. [1] 46](#_Toc134367698)

[3. ábra: Neptun és Moodle jelzők.. [1] 46](#_Toc134367699)

[4. ábra: Épületek. [1] 47](#_Toc134367700)

[5. ábra: Nevezetességek. [1] 48](#_Toc134367701)

[6. ábra: Regisztrációs felület. [1] 49](#_Toc134367702)

[7. ábra: Bejelentkező felület. [1] 50](#_Toc134367703)

[8. ábra: Menüsor bejelentkezés után. [1] 50](#_Toc134367704)

[9. ábra: Órarend feltöltési felület már egy feltöltött órarendel. [1] 50](#_Toc134367705)

[10. ábra: Órarend törlése megerősítő üzenet. [1] 51](#_Toc134367706)

[11. ábra: Profil. [1] 51](#_Toc134367707)

[12. ábra: Fiók törlése megerősítő üzenet. [1] 52](#_Toc134367708)

Táblázatjegyzék

[1. táblázat (forrás megjelölésével) [2] **Hiba! A könyvjelző nem létezik.**](#_Toc97890942)

1. <https://kreanilla.hu/jo-design-kedvenc-dizajn-idezeteim/> (2023.04.29.) [↑](#footnote-ref-1)
2. <https://gyires.inf.unideb.hu/GyBITT/12/ch01.html> (I. rész - Fejlett Adatbázis Technológiák – Jegyzet- 1. fejezet: Bevezetés) 2023.04.29. [↑](#footnote-ref-2)
3. <http://info.berzsenyi.hu/adatbazisok/az-adatbazis-fogalma> 2023.04.29. [↑](#footnote-ref-3)
4. https://www.php.net/docs.php [↑](#footnote-ref-4)
5. https://developers.google.com/maps/documentation/javascript [↑](#footnote-ref-5)