**VÁCI SZAKKÉPZÉSI CENTRUM   
BORONKAY GYÖRGY   
MŰSZAKI TECHNIKUM ÉS GIMNÁZIUM**

**SZAKDOLGOZAT**

**Less Máté, Nemecz Krisztián**

**2023.**

**VÁCI SZAKKÉPZÉSI CENTRUM   
BORONKAY GYÖRGY   
MŰSZAKI TECHNIKUM ÉS GIMNÁZIUM**



**SZAKDOLGOZAT**

**Váczi várweb (ProjectX), Várkezelő alkalmazás**

|  |  |
| --- | --- |
| Konzulens:   * Wiezl Csaba * Kemenes Tamás * Blahut Lóránt | Készítette:   * Less Máté * Nemecz Krisztián |

# **Hallgatói nyilatkozat**

Alulírottak, ezúton kijelentjük, hogy a szakdolgozat saját, önálló munkánk, és korábban még sehol nem került publikálásra.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Less Máté |  | Nemecz Krisztián |

# **Konzultációs lap**

Vizsgázók neve: Less Máté, Nemecz Krisztián

Szakdolgozat címe: Váci várweb, Várkezlő alkalmazás

A Programok által nyújtott szolgáltatások:

Váci várweb:

* Webes felület biztosítása a várral kapcsolatosan
* Tájékoztatás várban szerveződő eseményekről
* Geolokációs műveletek egy térkép funkció segítségével
* Programokhoz tartozó képek megjelenítése
* Felhasználói fiók biztosítása regisztrációval és bejelentkezéssel
* Megjegyzés és email küldése felhasználói fiókon keresztül

Várkezelő alkalmazás:

* Várban szerveződő események hozzáadása, vagy törlése a weboldalhoz
* Az események tulajdonságainak módosítása és képeik feldolgozása
* Adminisztrátorok kezelése

|  |  |  |
| --- | --- | --- |
| Sorszám | A konzultáció időpontja | A konzulens aláírása |
| 1. | 2022.10.10. |  |
| 2. | 2022.11.14. |  |
| 3. | 2022.12.12. |  |
| 4. | 2023.01.16 |  |
| 5. | 2023.02.13. |  |
| 6. | 2023.03.13 |  |

|  |  |  |
| --- | --- | --- |
| A szakdolgozat beadható: |  | A szakdolgozatot átvettem: |
| Vác, 2023. |  | Vác, 2023. |
| Konzulens |  | A szakképzést folytató intézmény felelőse |

**Tartalomjegyzék**

[**Hallgatói nyilatkozat** 2](#_Toc132459001)

[**Konzultációs lap** 3](#_Toc132459002)

[**Tartalomjegyzék** 4](#_Toc132459003)

[**1** **Témaválasztás** 7](#_Toc132459004)

[1.1 Mit hozunk létre? 7](#_Toc132459005)

[Váczi várweb (weboldal) 7](#_Toc132459006)

[Várkezelő alkalmazás (Asztali alkalmazás) 7](#_Toc132459007)

[**2** **Fejlesztői dokumentáció** 8](#_Toc132459008)

[2.1 Technológiák 8](#_Toc132459009)

[Felhasznált programok és fejlesztői környezetek 8](#_Toc132459010)

[Felhasznált programnyelvek 9](#_Toc132459011)

[2.2 Adatbázis terv 12](#_Toc132459012)

[Az adatbázisunkról alapvető információk 12](#_Toc132459013)

[Táblák az adatbázisban 12](#_Toc132459014)

[Táblák közötti kapcsolatok 13](#_Toc132459015)

[Táblák és szerkezeteik 15](#_Toc132459016)

[2.3 Váczi várweb főbb funkcióinak dokumentációja 24](#_Toc132459017)

[A Váci várweb mappaszerkezete és főbb mappái 24](#_Toc132459018)

[A Váczi várweb weboldal osztályai 25](#_Toc132459019)

[OOP és MVC megvalósulása a projektben 27](#_Toc132459020)

[A regisztráció funkció megvalósulása MVC,OOP szerint 27](#_Toc132459021)

[A bement szűrése php-val a regisztrációnál 29](#_Toc132459022)

[A bemeneti hibák megjelenítése JavaScripttel a regisztráció oldalon 34](#_Toc132459023)

[A beépített térkép és funkciói 37](#_Toc132459024)

[Email küldése nekünk az elérhetőség oldalról 39](#_Toc132459025)

[A várprogramok listázása 41](#_Toc132459026)

[A kiválasztott program funkciója 45](#_Toc132459027)

[Megjegyzések a weboldalon 49](#_Toc132459028)

[A galéria oldal a weboldalon 52](#_Toc132459029)

[Sessionkezelés a weboldalon 53](#_Toc132459030)

[A composer működése a weboldalon 54](#_Toc132459031)

[Az ágyú animáció a weboldalon 56](#_Toc132459032)

[A weboldal adatbázis kapcsolódása 56](#_Toc132459033)

[Nem található az oldal felülete 57](#_Toc132459034)

[Stílus elemek az oldalakhoz 58](#_Toc132459035)

[2.4 A Várkezelő asztali alkalmazás főbb funkcióinak fejlesztői dokumentációja 60](#_Toc132459036)

[Alapvető információk a programról 60](#_Toc132459037)

[Az adatbázis kapcsolódás megvalósítása 60](#_Toc132459038)

[Bejelentkezés a várkezelőbe 61](#_Toc132459039)

[A várkezelő alkalmazás menüje 65](#_Toc132459040)

[Program hozzáadás, módosítás 67](#_Toc132459041)

[A geléria képek feltöltése funkció 76](#_Toc132459042)

[A programok törlése funkció 78](#_Toc132459043)

[Adminisztrátorok kezelése funkció 80](#_Toc132459044)

[A bementi mezők értékeinek a titkosítása az alkalmazásban 83](#_Toc132459045)

[A várkezelő program osztályainak ismertetése 84](#_Toc132459046)

[2.5 Továbbfejlesztési lehetőségek 87](#_Toc132459047)

[Fordítás 87](#_Toc132459048)

[3d-s bejárható asztali alkalmazás 87](#_Toc132459049)

[Főoldalon a kép slider helyett blender animáció 87](#_Toc132459050)

[Webshop 88](#_Toc132459051)

[Hírportál a főoldalon 88](#_Toc132459052)

[Elővételes belépő QR kód vásárlása felület 88](#_Toc132459053)

[Email létezésének a validációja 88](#_Toc132459054)

[Felhasználók kezelése a Várkezelő programban 88](#_Toc132459055)

[Trigger naplózási események generálása az adatbázishoz 88](#_Toc132459056)

[**3** **Felhasználói dokumentáció** 89](#_Toc132459057)

[3.1 A Váci várweb felhasználó dokumentációja 89](#_Toc132459058)

[Rendszerkövetelményei 89](#_Toc132459059)

[Alkalmazás célja a cél közönsége 89](#_Toc132459060)

[Fejléc (header) 89](#_Toc132459061)

[Lábléc (footer) 90](#_Toc132459062)

[Főoldal 90](#_Toc132459063)

[Regisztráció 90](#_Toc132459064)

[Bejelentkezés 91](#_Toc132459065)

[Elérhetőség 92](#_Toc132459066)

[Jövőbeli programjaink 93](#_Toc132459067)

[Kiválasztott program 94](#_Toc132459068)

[Megjegyzések 95](#_Toc132459069)

[Vártörténet 96](#_Toc132459070)

[Kijelentkezés 96](#_Toc132459071)

[A weboldal felállítása (telepítése) 97](#_Toc132459072)

[3.2 A várkezelő alkalmazás felhasználói dokumentációja 98](#_Toc132459073)

[Rendszerkövetelmények 98](#_Toc132459074)

[Ki használja ezt az alkalmazást és milyen célból 98](#_Toc132459075)

[Bejelentkezés az alkalmazásba 98](#_Toc132459076)

[Menü használata 99](#_Toc132459077)

[Programok hozzáadásának, módosításának felülete 99](#_Toc132459078)

[A galéria képek feltöltésének funkciója 102](#_Toc132459079)

[Programok törlése 103](#_Toc132459080)

[Adminisztrátorok kezelése 104](#_Toc132459081)

[Visszalépés a menübe 105](#_Toc132459082)

[**4** **Munkabeosztás a csapatban** 105](#_Toc132459083)

[4.1 Less Máté munkássága 105](#_Toc132459084)

[4.2 Nemecz Krisztián munkássága 106](#_Toc132459085)

[**5** **Irodalomjegyzék** 107](#_Toc132459086)

[**6** **Mellékletek** 109](#_Toc132459087)

# **Témaválasztás**

## Mit hozunk létre?

### Váczi várweb (weboldal)

Csapatunk elsősorban egy felhasználóbarát és középkori stílust megcélzó webes felületet alkot meg Váczi várweb néven, amely a Vác várának a weboldalát takarja és tartalmazza a várról kapcsolatos hasznos funkciókat. Azért készítjük el ezt a weboldalt, mivel a várba látogatók száma évről évre növekszik Vácon, és a turistáknak szüksége van egy weboldalra, ahol felhasználóbarát felületet találhatnak a számítógépen, vagy bármilyen okos eszközön keresztül. Tehát segítséget nyújtunk számukra azzal, hogy nyomon követhetik a várban szerveződő, illetve megszervezett programokat, olvashatnak a vár történetével kapcsolatban, megtudhatják a várral kapcsolatos híreket, tájékoztatásokat például hogy, hol található a komplexum és mi a nyitvatartási idő, valamint, hogy milyenek voltak a vár benyomásai a korábbi látogatók számára.

Továbbá, a weboldal segíti a vár megismertetését a felhasználóval és megismerteti a várban rejlő lehetőségeket azokkal is, akik jobban szeretnek úgy látogatni turista célpontokat, hogy részletesen meg vannak győződve arról, hogy milyen szolgáltatásokat kaphatnak a helyen. Egyszóval a mi célunk is ezzel a webes szolgáltatással, hogy stílusos és áttekinthető felületet nyújtunk a Váczi várral kapcsolatban. A vár fikcióra épül, tehát nem létezik.

**Miért ezt a témát választottuk?**

Azért választottuk ezt a témát, mivel jelentős mértékben foglalkoztat minket a várak és a történelem, mint téma. A középkori vár, mint témában, mind a funkcionalitást és mind a stílust nézve sok fantáziát láttunk.

### Várkezelő alkalmazás (Asztali alkalmazás)

Mivel a webes felület adatbázisra támaszkodik, így létrehozunk egy „Várkezlő” névre hallgató Windows Forms alapú asztali segédprogramot a weboldalunkhoz. A lényeg a Várkezelő appnak, hogy segítségével az adminisztrátorok gyorsan tudják kezelni, a várhoz fűződő adatbázis adatokat, amelyek lehetnek programok, képek elérési útvonalai is. Ezeket akár törölni, vagy módosítani tudjuk, vagy jó esteben azok idejét és egyéb tulajdonságait tudjuk manipulálni az alkalmazás segítségével. Ezzel a programmal kezelni tudjuk az adminisztrátorokat. Vagyis törölni tudunk amint, módosítani, és hozzáadni is akár. A célunk még a Várkezelővel, hogy a legtöbbször előforduló a weboldal által használt adatbázis rekordokat kényelmesen egy asztali felület segítségével lehessen szerkeszteni SQL parancsok begépelgetése nélkül. Valamint, hogy a karbantartó mobilisan tudja használni a programot egyszerű telepítéssel.

# **Fejlesztői dokumentáció**

## Technológiák

### Felhasznált programok és fejlesztői környezetek

#### Programok és fejlesztési környezetek felsorolása

* Visual Studio Code
* Visual Studio
* Composer
* Xampp
* Adobe photoshop

#### Néhány program ismertetése

**Composer:** A Composer egy alkalmazásszintű függőségkezelő a PHP programozási nyelvhez, amely szabványos formátumot biztosít a PHP szoftverek és a szükséges könyvtárak függőségének kezelésére. Vagyis webfejlesztésben használt technológia, ami elősegíti, hogy a programozó ne találja ki újra és újra a kereket, hiszen az teljesen felesleges időpocséklás lenne, hanem leakaszt egy már meglévő library-t a polcról és azt használja, megkímélve magát a sok redundáns munkától. A composer egyfajta "csomagkezelő", amely kezeli a különböző csomagjainkat, azok függőségeit. A program a parancssorból fut és automatikus betöltési fuknciókat is rendel a webalkalmazásokhoz. A composer segítségével végezzük el a phpmailer technológián keresztül az email rendszerünk felépítését és autoloader részének köszönhetően automatkus kommunikációt értünk el az oszályaink között.1

**Visual Studio Code:** A webes alkalmazásunk fejlesztésést a Visual Studio Code környezetében végezzük, mert ingyenes, megbízható, sok nyelvet támogat és átlátható a szintaxis rajta keresztül. Ez a programkód szerkesztő Támogatja a hibakeresőket, valamint beépített Git támogatással rendelkezik, továbbá képes az intelligens kódkiegészítésre az IntelliSense segítségével. A VSCode-ban a felhasználók megváltoztathatják a kinézetet (témát), megváltoztathatják a szerkesztő gyorsbillentyű-kiosztását, az alapértelmezett beállításokat és még sok egyebet. Támogatja a kiegészítőket, melyek segítségével további funkciók, testre szabási lehetőségek érhetőek el. A VSCode az Electron nevű keretrendszeren alapszik, amellyel asztali környezetben futtatható Node.js alkalmazások fejleszthetőek.2

**Xampp:** egy szabad és nyílt forrású platformfüggetlen webszerver-szoftvercsomag, amelynek legfőbb alkotóelemei az Apache webszerver, a MariaDB (korábban a MySQL) adatbázis-kezelő, valamint a PHP és a Perl programozási nyelvek értelmezői (végrehajtó rendszerei). Ez a szoftvercsomag egy integrált rendszert alkot, amely webes alkalmazások készítését, tesztelését és futtatását célozza, és ehhez egy csomagban minden szükséges összetevőt tartalmaz. A rendszer egyik nagy előnye az összehangolt elemek könnyű telepíthetősége, és a weboldalak egyszerűen tesztelhető[[1]](#footnote-1)ek rajta kockázat nélkül.3

### Felhasznált programnyelvek

#### Felhasznált programnyelvek és egyéb technológiák felsorolása

* HTML
* PHP
* JavaScript
* C#
* JSON
* Google APIs
* Bootsrtap
* MySQL Connector
* Media (css)
* .htaccess
* SHA256

#### Néhány Technológia ismertetése

**Bootstrap:** Bootstrap egy nyílt forráskódú CSS és JavaScript keretrendszer, amely lehetővé teszi a webfejlesztők számára, hogy gyorsan és hatékonyan készítsenek reszponzív, mobilbarát weboldalakat és alkalmazásokat. A Bootstrap tartalmaz előre elkészített komponenseket, mint például gombok, űrlapok, navigációs menük és modális ablakok, amelyek egyszerűen alkalmazhatók a weboldalakon. A Bootstrap 6 azonban több szempontból eltér elődjétől, a Bootstrap 5-től, és javítja a szemantikus HTML használatot és a működési megközelítést, amelyek még inkább javítják a teljesítményt és a webfejlesztők hatékonyságát. 1

**MySQL Connector:** A MySQL Connector/ODBC, korábban MyODBC néven ismert, az Oracle Corporation számítógépes szoftvere. Ez egy ODBC interfész, és lehetővé teszi, hogy az ODBC interfészt támogató programozási nyelvek kommunikáljanak egy MySQL adatbázissal. A MySQL Connector/ODBC-t eredetileg a MySQL AB hozta létre.2

**CSS:** rövidítése Cascading Style Sheets, egy webfejlesztési technológia, amely lehetővé teszi a weboldalak stílusának és megjelenésének testre szabását. A CSS segítségével a webfejlesztők képesek megadni az elemek méretét, elrendezését, betűtípusát, betűméretét, színeit, háttérképeit, és sok más formázási lehetőséget. A CSS lehetővé teszi a weboldalak egyszerűbbé és átláthatóbbá tételét, csökkentve a HTML kód mennyiségét, ami javítja a weboldalak sebességét és hatékonyságát. A CSS lehetővé teszi a weboldalak reszponzív kialakítását, ami azt jelenti, hogy a weboldalak megjelenése önműködően alkalmazkodik az eszközök képernyőméretéhez, így azok megfelelően jelennek meg mobilokon, tableteken és asztali számítógépeken egyaránt. A CSS további funkciói közé tartozik a szintaxisöröklés, a stíluskasztni és az időzített animációk, amelyek lehetővé teszik a weboldalak még kreatívabb és dinamikusabb kialakítását.3

**CSS Media Query:**  A CSS Media Query egy olyan technológia, amely lehetővé teszi a webfejlesztők számára, hogy különböző stílusokat alkalmazzanak a weboldalak különböző eszközökön történő megjelenítéséhez. A Media Query lehetővé teszi az alkalmazkodást a különböző képernyőméretekhez, például a mobiltelefonok, tabletek és számítógépek képernyőihez. Használatával a webfejlesztők felülírhatják a globális stílusokat és specifikus stílusokat határozhatnak meg az egyes képernyőméretekhez. A Media Query kulcsfontosságú szerepet játszik a reszponzív webtervezésben, és lehetővé teszi a weboldalak jobb használhatóságát és felhasználói élményét. Ez úgy működik, hogy a stílusnak, amit használni akarunk, egy maximális szélességet állítunk be. És ha az oldal szélessége a megadott érték alá megy, akkor applikálja a stílust az elemre. 4

**Google APIs:** A Google API-k a Google által kifejlesztett alkalmazásprogramozási felületek (API-k), amelyek lehetővé teszik a Google-szolgáltatásokkal való kommunikációt és azok integrációját más szolgáltatásokkal. Ilyen például a Keresés, a Gmail, a Fordító vagy a Google Térkép. A harmadik féltől származó alkalmazások használhatják ezeket az API-kat, hogy kihasználják vagy bővítsék a meglévő szolgáltatások funkcionalitását. 5

Az API-k olyan funkciókat biztosítanak, mint az analitika, a gépi tanulás szolgáltatásként (a Prediction API) vagy a felhasználói adatokhoz való hozzáférés (ha engedélyt adnak az adatok olvasására). Egy másik fontos példa egy webhelyen található beágyazott Google-térkép, amely a Static Maps API Places API vagy a Google Earth API használatával érhető el.5

**Google Map API:** A Google Térkép a Google által fejlesztett ingyenes internetes térképszolgáltatás. Az szolgáltatás térképek, és műholdfelvételeken alapszik, amelyek az egész Földet lefedik. A többi Google internetes alkalmazáshoz hasonlóan nagy mennyiségű JavaScriptre épül. A térkép mozgatása során a kis térképrészletek folyamatosan töltődnek le a térképet kiszolgáló szerverről, illetve jelennek meg a böngészőablakban. Ha a felhasználó valamilyen kereskedelmi objektumra keres, annak helyét egy gombostűszerű jel mutatja, ami lényegében egy a térkép fölé helyezett áttetsző PNG kép. Az újabb fajta technológia, AJAX hálózati kérések segítségével nagyobb interaktivitást tesz lehetővé. Ezt a technológiát be lehet ágyazni egy alkalmazásba.6

**JSON:** A JSON a JavaScript Object Notation rövidítése

,ami egy könnyű adatcsere-formátum, amely lehetőséget ad programnyelvek és struktúrák közötti adatátvitelre és közvetítésre.

A JSON nyelvfüggetlen.7

Szintakszis:

A JSON-adatok név/érték párokként (más néven kulcs/érték párok) vannak írva.

A név/érték pár egy mezőnévből áll (idézőjelben), amit kettőspont követ, amit egy érték követ. {"name":"John"}

**.htaccess:** A .htaccess egy könyvtárszintű konfigurációs fájl, amelyet megtalálhatunk többnyire minden webes alapú szerver állományába és a benne létrehozott utasítások érvényesek az egész szerverre. Segítségével állományelérési jelszót állíthatunk be, hitelesíthetünk, megadhatjuk a szerver, hogyan kezeljen különböző állománytípusokat (MIME) továbbá, megadhatjuk az alapértelmezett kezdő oldalt, blokkolhatunk különböző ip címeket. Ezen kívül, a technológiával gyorsítótárazást hajthatunk végre és megadhatjuk a szerver válaszának módját, ha a beképelt URL nem megfelelő, vagy nem található a tartalom. Ez a konfigurációs fájl úgy lép működésbe, hogy utasításokat intéz a webszerver különböző állományaihoz.8

**SHA titkosítás:** Az SHA (Secure Hash Algorithm) az Egyesült Államok Nemzeti Szabvány és Technológia Hivatala (NIST) által kibocsátott szabványos eljárások összefoglaló elnevezése. (SHA-1, SHA-224, SHA-256, SHA-384 és SHA-512). Az első változatát 1993-ban fejlesztették az NSA felügyelete alatt. Ez 160 bit hosszúságú üzenetkivonatot (message digest) képez, amelyet ezt követően a DSA elektronikus aláírás algoritmusban használhatunk. Az SHA-512 hasonló elvek alapján működik, mint az SHA-1 de lényegesen nagyobb adat mennyiséget képes kezelni. A bemenete 2128-1 bit hosszúságú lehet. Ezt egyébként, a feldolgozás során 1024 bites blokkokra osztja. A kapott hasító érték összességében 512 bit hosszúságú és 8 db 64 bites blokk alkotja. Az SHA alapjaiban bármilyen hosszúságú adatra adott hosszúságút képez le, ahol egy tetszőleges byte átírásásval, hozzáadásával a hash kulcs véletlenszerűen megváltozik. A nagyon kicsi hosszúságú kulcsoknál előfordulhat, hogy ütköznek az a hash értékek, de kellő hosszúságú SHA-val szinte lehetetlen az ugyan-olyan minta legenerálása. 9

**SHA256:** Az SHA256 egy 256 bites „biztonságos kivonatolási algoritmus” (Secure Hash Algorithm) rövidítése, és egy titkosítási módszert jelöli az SHA csoporton belül. A titkosítási kivonatoló algoritmusok visszafordíthatatlan és egyedi kivonatot készítenek. Minél nagyobb számú kivonat lehetséges, annál kisebb annak az esélye, hogy két érték ugyanazt a kivonatot hozza létre. Az SHA-256 esetében 2256 alehetséges kombinációk száma. 2256ez a hatalmas szám meghaladja a homok szemcséjének számát a földön. Alapvető elvárás, hogy a hash algoritmus csak a bemenő adat bájtjaitól és azok sorrendjétől függjön, tehát determinisztikus/előre meghatározott legyen. A bitcoin digitális fizetőeszköz és klónjai is az SHA-256 algoritmust használják. 10

[[2]](#footnote-2)

## Adatbázis terv

### Az adatbázisunkról alapvető információk

**Adatbázis neve:** vaczi\_var\_adatbazis

**Adatbáis motorja:** innoDB

**Protokoll verzió:** 10

**Kiszolgáló verziója:** 10.4.25-MariaDB

**PHP verzió:** 8.1.10

**Létrehozás ideje:** 2023. Jan 31. 14:10

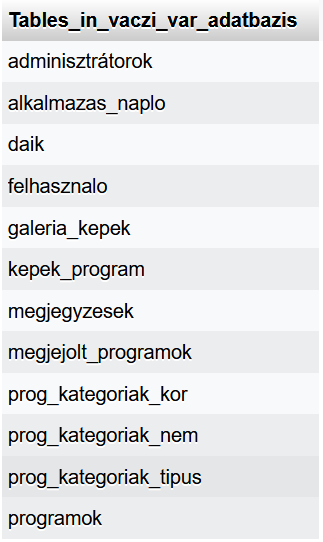
**Az adatbázis karakterkódolása:** UTF8, generális karakterkészlettel

**SQL mód beállítás:** Nincsen automatikus érték 0 után

SET SQL\_MODE = **"NO\_AUTO\_VALUE\_ON\_ZERO"**;

Ennek használata megmagyarázza azt, hogy van 0-s id-jű elem az adatbázisban.

### Táblák az adatbázisban

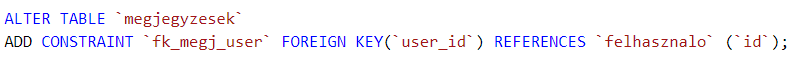


### Táblák közötti kapcsolatok

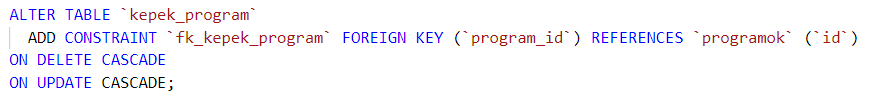
A kapcsolatokat idegenkulcsok segítségével határozzuk meg, ahol törlésre és módosításra is a ’CASCADE’ kulcsszót defináljuk. Tehát ha törlésre kerül a szülőtábla egy sora, akkor a hozzá tartozó rekordok is törlődnek, vagy esetleg módosulnak a gyerektáblában.



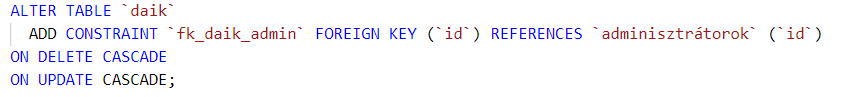
A megjegyzesek és programok táblák kapcsolata



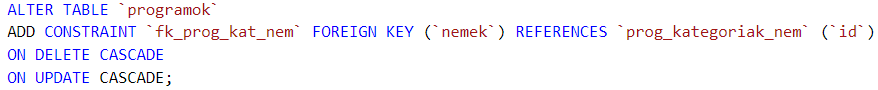
A megjegyzesek és felhasznalo táblák kapcsolata



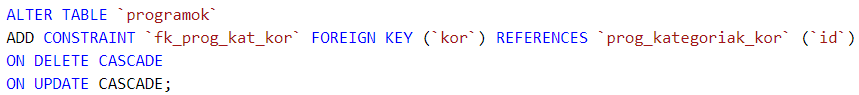
A kepek\_program és programok táblák kapcsolata



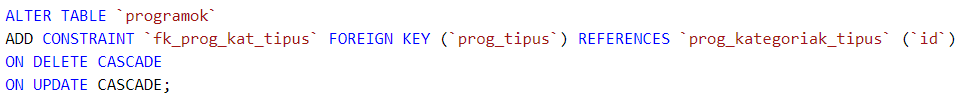
A daik és adminisztrátorok táblák kapcsolata



A programok és prog\_kat\_nem táblák kapcsolata



A programok és prog\_kat\_kor táblák kapcsolata



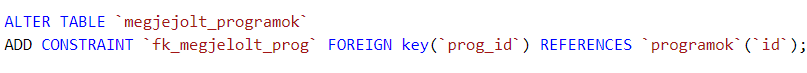
A programok és prog\_tipus táblák kapcsolata



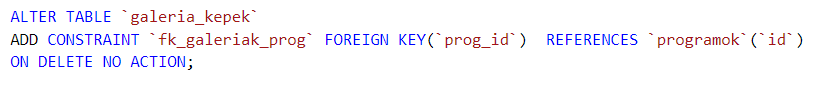
A megjegyzesek és programok táblák kapcsolata



A megjegyzesek és programok táblák kapcsolata



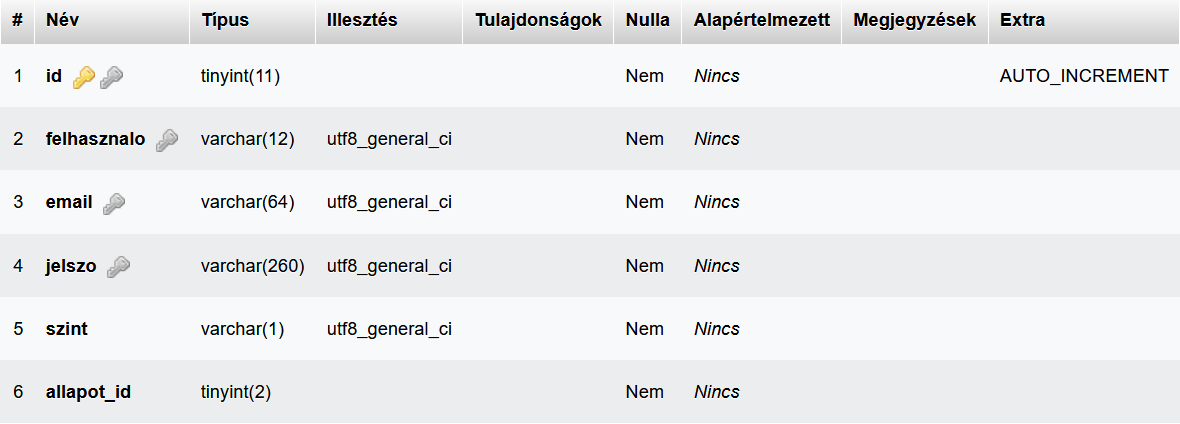
A megjelolt\_programok és programok kapcsolata

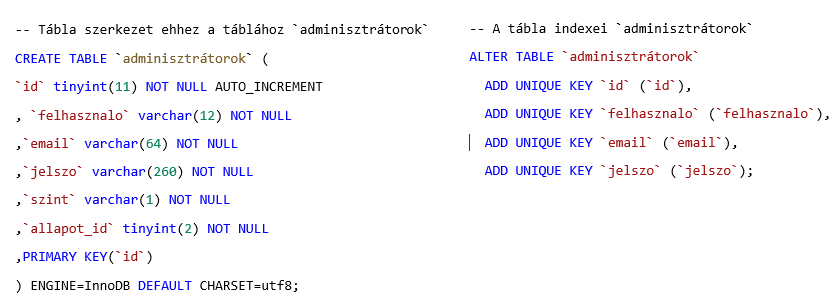


A galeria\_kepek és programok kapcsolata

### Táblák és szerkezeteik

#### Adminisztrátorok tábla





##### *Az adminisztrátorok tábla funkciója*

A tábla tárolja a rendszer kezelőit és azok adatait, akik a Várkezelő alkalmazással működtetni tudják a rendszert. A tábla az asztali program alapvető működéséhez szükséges, mivel kiszolgálja a programba való belépési funkciót megfelelő adatokkal, és a programon belüli validációnak is nyújt megfelelő rekordokat.

##### *Adminisztrátorok tábla szerkezete és rekordjai*

##### **id:** A tábla elsődleges kulcsa. Tinyint(11) típusa van, amely kevesebb rekordot képes tárolni, mint az int, de így elég a szükséges jövőbeli rekord tárolásához az adminisztrátoroknak. Egy meghatározott admin azonosítására szolgál. Egyedinek kell lennie.

**felhasznalo:** Az adminisztrátor neve. Varchar(12) es értéke van, amivel max 12 karakterű neve lehet egy adminnak, pont annyi, amennyi egy normál felhasználónak is. A név mezőkhöz hozzá van rendelve. Egyedinek kell lennie a táblán belül.

**email:** Az adminisztrátor email címe. Varchar(64) es értéke van, amivel eléri az emailek átlagos megengedett hosszát.Egyedinek kell lennie a táblán belül.

**jelszo:** Az adminisztátor jelszava. Varchar(260)-as értékű, mivel a jelszava titkosítva van SHA256-al,ami egy 255-bites kódolt szöveget hoz létre.

**szint:** Az adminisztrátor jogi szintje az adminokon belül. Varchar(1) es értéke van, amivel 1 karakter értéket vehet fel. A működés során ’1’ vagy ’2’ értékű lehet, azaz admin vagy főadmin. A Főadminisztátorok számát a kezelő programban limitáljuk.

**allapot\_id:** Az adminisztrátor állapotának id-ja. Tinyint(2)es értéke van, mivel 1-3-ig skálázzuk az értékét, vagyis 1 az aktív, 2 az a törölt és a 3 tiltott státuszt jelenti. Ezen értékek alapján lehet módosítani az adminisztrátort.

#### A daik tábla

##### 

##### 

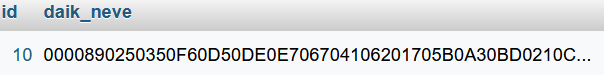
##### *A daik tábla funkciója*

A programon belül az adminisztátoroknak nemcsak a jelszóval kell hitelesíteni magukat, hanem hozzávetőlegesen szükséges egy daik kulcs is ami a Dedikált Adminisztációs Információs Kulcs nevet kapta. A daik ugyan-úgy tikosítva van Sha256-tal, mitn a jelszó. A fő lényege az, hogy ez az érték egy külön táblába van elhelyezve, ami a daik nevű tábla.

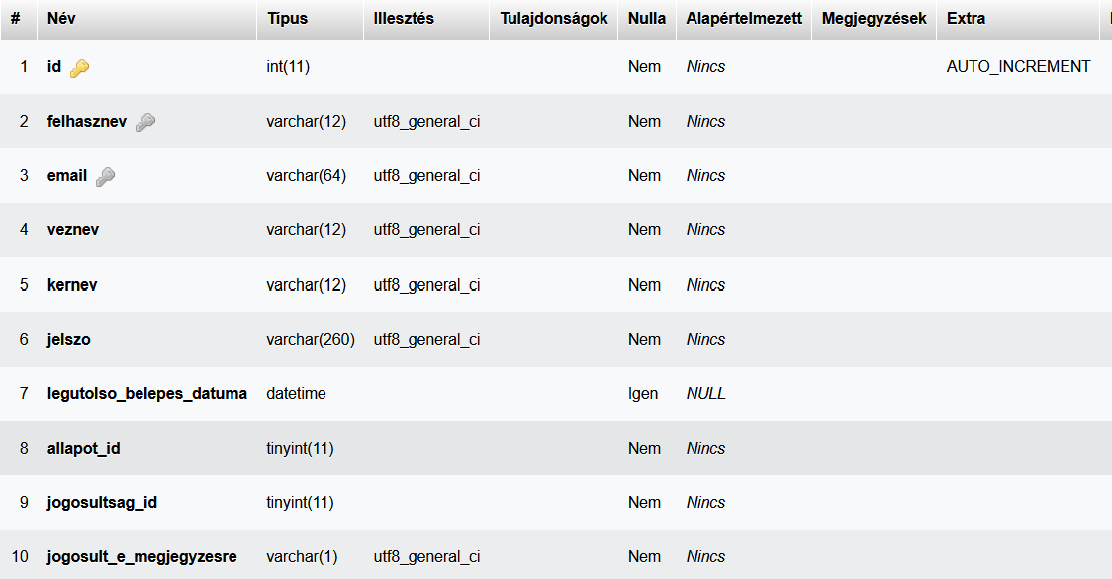
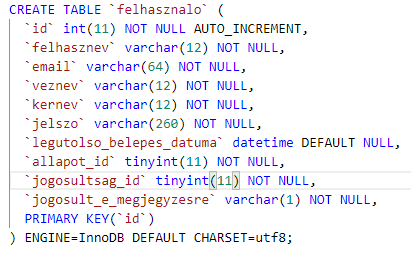
##### *A daik tábla szerkezete és rekordjai*

**id:** A tábla elsődleges kulcsa, a rekordok azonosítására van fenntartva. Mivel össze van kapcsolva az adminisztátorok tábla id-val, így ez az id is tinyint típust kap, mivel nincs szükség sokat generálni belőle.

**daik\_neve:** A daik érték, ami egyben a 2. jelszó. Mivel SHA256-al van titkosítva így a típusa varchar(260). Az alábbi képen láthatjuk, hogyan néz ki a titkosított értéke:



#### A felhasználók tábla



##### *A felhasználók tábla funkciója*

A felhasználók nevű táblában tároljuk a regisztrált felhasználókat, és a hozzájuk fűződő összes szükséges adatot, amelyek alapján a weboldalunk tudja szolgáltatni a bejelentkezést igénylő tevékenységeket.

##### *A felhasználók tábla szerkezete és rekordjai*

**id:** Egy felhasználó egyedi azonosítója. Int(11)-es értékű, amivel már jóval több felhasználó lehet, mint adminisztrátor. A tábla egyedi kulcsa.

**felhasznev:** Egy felhasználó neve**.** Varchar(12)-es értékű mivel a maximum névhossz 12 karakter lehet a weboldalunkon. A név szigorúan egyedi a táblában.

**email:** A felhasználó email címét tárolja varchar(64)-es értékkel.

**veznev:** A felhasználó vezetékneve. varchar(12)-es a maximum értéke.

**kernev:** A felhasználó keresztneve. varchar(12)-es a maximum értéke.

**jelszo:** A felhasználó jelszavát tárolja, varchar(260)-as értékkel bír, mivel sha256-os titkosítással van eltárolva, ami 255 biten dolgozik.

**legutolso\_belepes\_datuma:** A felhasználó legutolsó, jelenlegi weboldalra való bejelentkezésének napját mutatja. Ezért használjuk ezt az oszlopot, mert megkönnyíti azoknak a felhasználóknak a törlését, akik például már 5 éve nem léptek be. datetime változója van. Itt megengedett, hogy az értéke NULL legyen.

**allapot\_id:** A felhasználó állapotának jelzésére van fenntartva. Tinyint(11)–es értéke van. A felhasználó állapota lehet: -aktív(1) –törölt(2) –tiltott(3). A felhasználó könnyebb és hatékonyabb kezelését kívánja elősegíteni ez az oszlop, illetve tiltani tudjuk az erkölcstelen felhasználókat.

**jogosult\_e\_megjegyzesre:** A felhasználónak beállítja, hogy tud-e megjegyzést tenni, vagy sem. Mivel 1 vagy 0 lehet az értéke így mi varchar(1)-es értéket állítottunk be neki.

#### A programok tábla



##### *A programok tábla szerepe*

A programok tábla tartalmazza azokat a programokat és azok rekordjait, amelyek a várban találhatóak és ott kerülnek megrendezésre. A tábla kiszolgálja mind a webes és az asztali alkalmazást.

##### *A programok tábla szerkezete és rekordjai*

**id:** A program egyedi azonosítására használjuk. Int(11)-es típusa van, mivel rengeteg programot lehet hozzáadni. A tábla egyedi kulcsa.

**megnevezes:** A program nevét tárolja, mint attribútum. varchar(30) a típusa.

**leiras:** A program részletes és hosszabb leírását tartalmazza. Varchar(2000) típussal már hosszabb programkifejtést tudunk vele felmutatni a weboldalon.

**kezd\_datum:** A program kezdő dátuma, azt a dátumot mutatja(év, hó,nap) amikor elkezdődik a program. Date típusú.

**kezd\_ido:** A program kezdési ideje órában és percben. Time típusa van.

**befejez\_datum:** A program befejező dátuma, azt a dátumot mutatja(év,hó,nap) amikor végetér a program. Date típussal rendelkezik.

**befejez\_ido:** A program befejezési ideje órában és percben. Time típusa van.

**nemek:** A program szűrésénél használatos. Megadja, hogy fiúknak, lányoknak, vagymindennem számára ajánlott-e a program. A típus varchar(1), mivel 1 karaktert kell csak tárolni. A mező másodlagos kulcs.

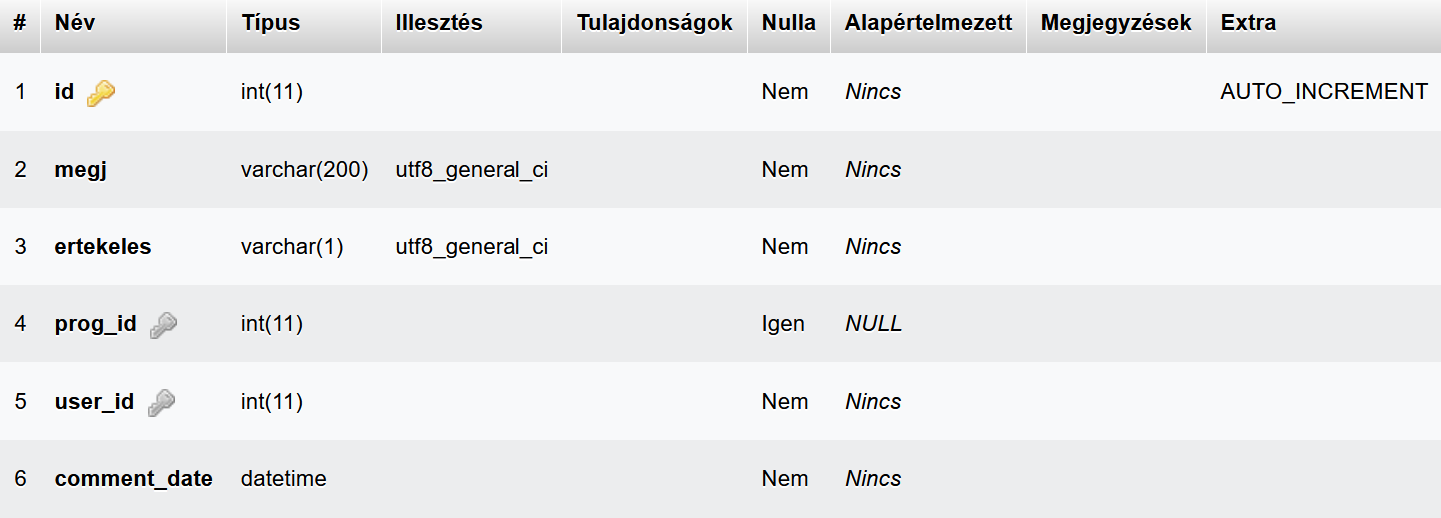
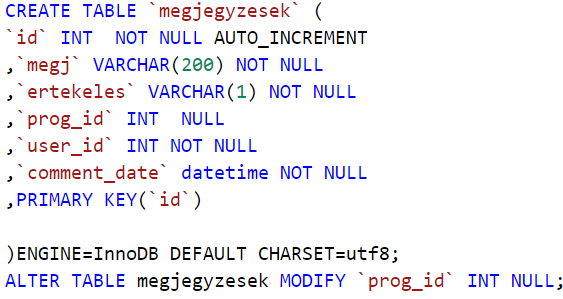
**prog\_tipus:** A program típusát adja meg: kaland, kulturális, zene, esztétikus A típusa ugyan-úgy varchar(1). A mező másodlagos kulcs.

**kor:** Az ajánlott korosztályt mutatja : gyermek(1),serdülő (2),serdülő és felnőtt (3), felnőtt(4), minden korosztály(5).Az értékeVarchar(1)**-**esés egykarakteres számokat tárolunk itt is. A mező másodlagos kulcs.

**resztveteles:** Itt egy karakter szám helyezkedik el 0 vagy 1 formájában, Megadja, hogy a felhasználó részt tud e venni, azaz ki tudja e próbálni a programot a részvételével. varchar(1) típussal rendelkezik.

**megtartva:** Megadja, hogy a program meg lesz e tartva**.** Az adattípusa: varhcar(1).

#### A megjegyzések tábla



##### *A megjegyzések tábla funkciója*

A megjegyzések tábla tárolja a kommenteket, amelyeket a felhasználók írnak a weboldalon. Továbbá ehhez tárolja az olyan értékeket, amelyekkel a weboldal azonosítani tudja, hogy ki írta a kommentet, és hogy milyen programmal kapcsolatos az adott kommentáló írása.

##### *A megjegyzések tábla szerkezete és rekordjai*

**id:** A megjegyzés azonosítójára fenntartott elsődleges kulcs. A típusa int(11).

**megj:** A megjegyzés maga, aminek a típusa varchar(200), tehát maximum 200 karakteres megjegyzéseket tud tárolni.

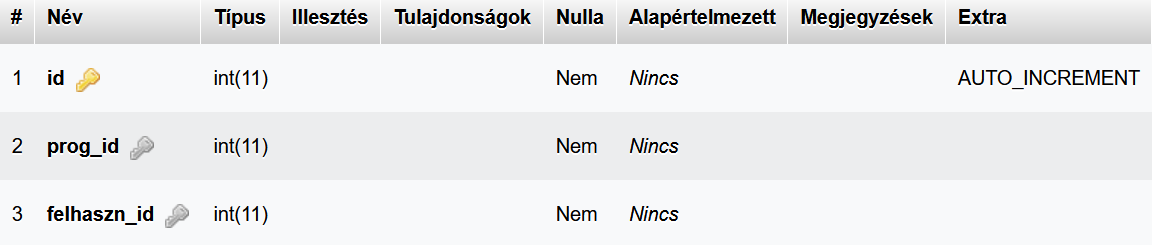
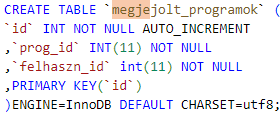
**ertekeles:** A tábla tartalmaz egy olyan mezőt, amely az értékeléshez szükséges csillagok formájában. Ebben az oszlopban csak a számokat tároljuk 1 től 5-ig, amelyek leképzik a csillagokat a programban. Varchar(1)-es a típusa.

**prog\_id**: Ez a mező másodlagos kulcsként van definiálva. A megjegyzéshez fűződő program id-ját mutatja, Int(11) típússal. Mivel a programok id-ja is int(11).

**user\_id:** Ez az oszlop is másodlagos kulcs. A megjegyzéshez fűződő felhasználó id-ját mutatja, Int(11) típussal.

**comment\_date:** A komment idejét mutatja date formátummal.

#### A megjelelolt\_programok tábla



##### *A „megjelolt\_programok” tábla szerepe*

A megjelölt programok táblája tárolja annak a webes funkcióhoz az adatokat, amellyel ki tudjuk jelölni a számunkra érdekes programokat, eseményeket, amelyeket érdeklődési listára helyezünk. A tábla biztosítja a felhasználónként programok iránti érdeklődés különbözőségét, és azt, hogy ki és visszalépés után is be legyen jelölve.

##### *A megjelolt\_programok tábla szerkezete*

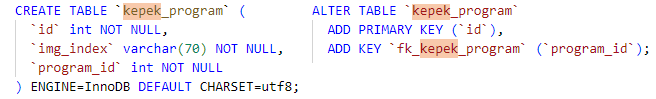
**id:** A megjelölések azonosítója. Int(11)-es típussal rengeteg ilyen megjelölést lehet elvégezni, és kell is, mivel egy felhasználó sok programot megjelölhet. A mező a tábla elsődleges kulcsa.

**prog\_id:** Ez a mező a megjelölt programot tárolja másodlagos kulcsként. Int(11) típussal bír.

**felhaszn\_id:** Ez a mező a megjelölt programhoz tartozó felhasználót tárolja, másodlagos kulcsként Int(11) típussal.

#### A kepek\_program tábla





##### *A kepék program tábla funkciója*

Ez a tábla tárolja a különböző programokhoz tartozó képeket, amelyeket így programonként meg tudunk jeleníteni. Vagyis szorosan össze van kapcsolva a programokkal és lényegében a tábla léte a programok tábla mezőinek számát hivatott csökkenteni azáltal, hogy nem kell az kép elérési útját a programok táblában definiálni.

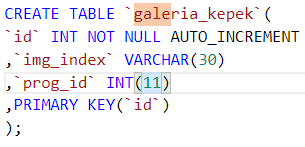
##### *A „kepek\_program” tábla szerkezete:*

**id:** A képek egyedi azonosítására szolgál. int(11) a típusa és be van állítva az AUTO\_INCREMENT automatikus azonosító érték hozzáadása.

**img\_index:** A kép elérését tárolja varchar(70)-es típussal a lehetséges név és elérési hosszára tekintettel

**program\_id:** A mező a képhez tartozó program id-t tárolja, int(11)-es típussal. A mező másodlagos kulcs a táblában.

#### A galéria képek tábla



##### *A galéria képek tábla funkciója*

A tábla azokhoz a programokhoz tartalmazza a képeket, amelyek már befejezettek így a galéria oldal a tábla segítségével megjeleníti a képeket és a hozzá tartozó befejezett programot.

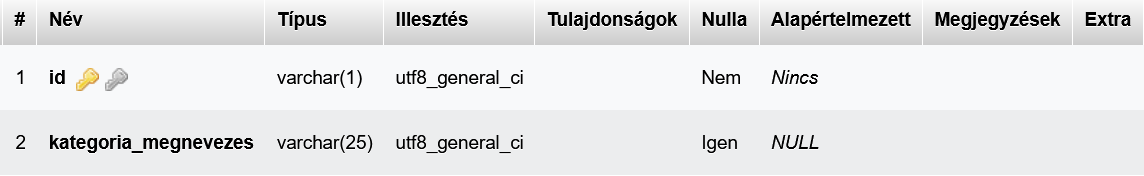
##### *A galéria képek tábla szerkezete*

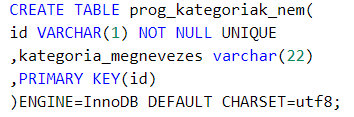
**id:** A galériához tartozó kép egyedi azonosítója int(11) típussal. Az AUTO\_ICNREMENT be van állítva az oszlopban.

**img\_index:** A képhez tartozó elérési útvonal varchar(30)-as típussal.

**prog\_id:** A képhez tartozó befejezett program azonosítója, amely másodlagos kulcs.

#### A [prog\_kategoriak\_kor](http://localhost/phpmyadmin/index.php?route=/sql&pos=0&db=vaczi_var_adatbazis&table=prog_kategoriak_kor) tábla





##### *A „prog\_kategoriak\_kor” tábla szerepe*

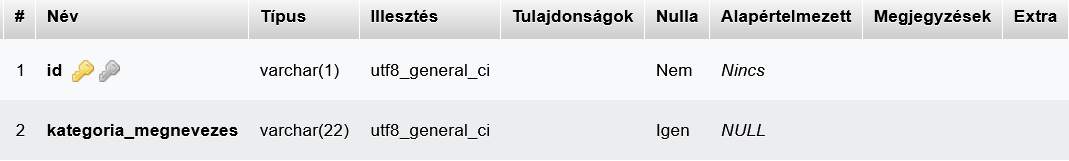
A tábla tárolja azokat a rekordokat, amelyeket a programok tábla használ az idegen kulccsal való összekötésre. A programok „kor” nevű mező hivatkozik a táblára. A tábla adatait listák feltöltésére használjuk, amelyek az programok hozzáadását, módosítását, keresését segítik elő a weboldalon és az asztali alkalmazásban.

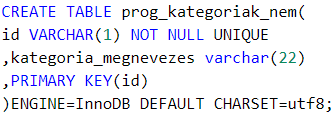
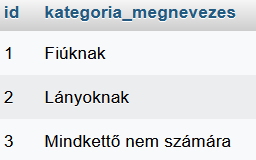
##### *A „prog\_kategoriak\_kor” tábla szerkezete*

**id:** a kor azonosítását szolgálja. Egyedi kulcs AUTO\_INCREMENT használata nélkül. A mező varchar(1) típusú, mert karakter formájában elég egytől ötig tárolni a számokat.

**kategoria\_megnevezes**: a kategóriának a megnevezésére szolgál varchar(25) típussal.

#### A „prog\_kategoriak\_nem” tábla



****

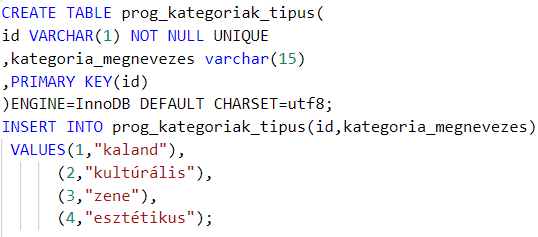
##### *A „prog\_kategoriak\_nem” tábla szerkezete*

**id**: A kategórián belüli rekord azonosítására szolgál ez az egyedi kulcs. Az AUTO\_INCREMENT nincs beállítva, mivel nem szükséges automatikus azonosító hozzáadás a tábla esetén.

**kategoria\_megnevezes:** A kategória megnevezésére szolgáló varchar(22)-es típussal rendelkező mező.

#### A prog\_kategoriak\_tipus tábla

****

****

##### *A „prog\_kategoriak\_tipus” tábla szerkezete*

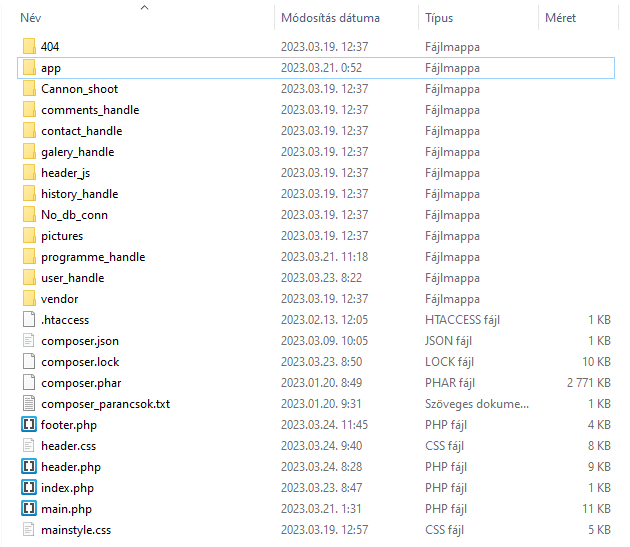
**id:** A típuson belüli rekord azonosítására szolgál ez az egyedi kulcs. Itt az AUTO\_INCREMENT nincs beállítva.

**kategoria\_megnevezes:** A kategória megnevezésesre szolgáló varchar(15)-ös típussal rendelkező mező.

## Váczi várweb főbb funkcióinak dokumentációja

### A Váci várweb mappaszerkezete és főbb mappái

#### A weboldal fő mappájának szerkezete



A projekt főmappája

**/app:** A weboldal php osztályait, függvényeit és metódusait tartalmazza

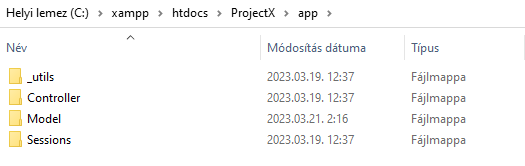
**/comments\_handle, contact\_handle, galeryhandle, programme\_handle, user handle:**

A kommentek, elérhetőség, galéria, programok, vártörténet, regisztráció, bejelentkezés oldalakhoz tartozó stílust meghatározó CSS és JavaScript állományokat tartalmazza.

**/pictures:** Itt tárolódnak a nagy felbontású képek az oldalhoz.

**/vendor:** a composer működését biztosító szerelvények és állományok találhatóak itt.

#### Az app mappa és tartalma alapjaiban



**\_utils:** Az adatbázishoz való csatlakozást megvalósító php állományok vannak itt.

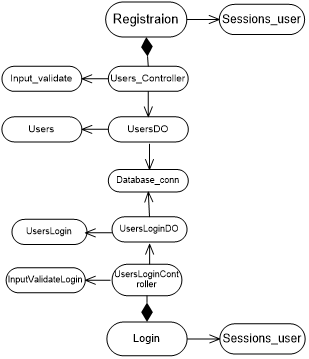
**Controller:** A Controller szerepű php osztályok helyeződnek ide, amelyek vezérlést biztosítanak az adatbázis és a megjelenítés között a weboldal oldalain.

**Model:** Az adatbázis adatokat lekérdező és azokat szerkezetben tároló php osztályok találhatóak itt. Másszóval, itt vannak a DAO állományok az MVC megvalósításához.

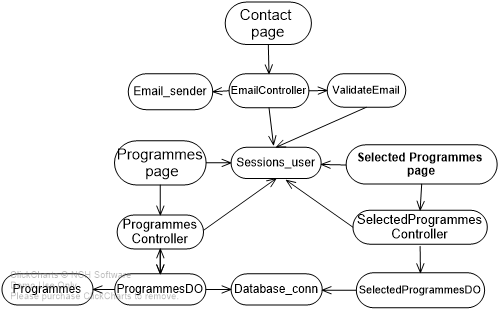
**Sessions:** Itt tárolódnak a weboldal munkameneteit kezelő php függvények.

### A Váczi várweb weboldal osztályai

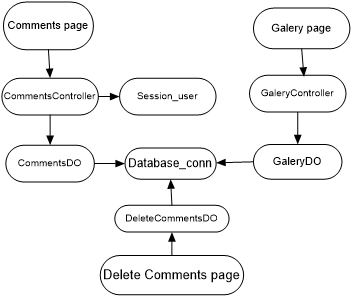
#### A regisztráció és a bejelentkezés osztályainak diagramja



**A bejelentkezés és regisztráció osztályainak kapcsolatai**



**Az elérhetőség, a programok és a kiválasztott programok oldal*akhoz tartozó osztályok kapcsolatai***



**A kommentek, galéria oldal, valamint a kommentek törlése funkcióhoz tartozó osztályok és kapcsolataik**

### OOP és MVC megvalósulása a projektben

#### OOP rövid ismertetése a projektünkben

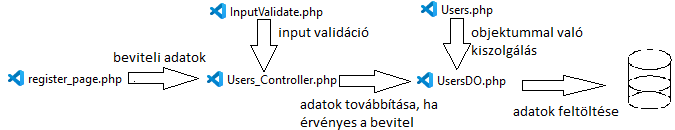
A weboldalunkat fejlesztés során igyekszünk az objektum orientált elvek alapján felépíteni. Ez azt jelenti, hogy a funkciók működéseshez különböző osztályokat és metódusokat használunk, amelyek különböző fájlokba vannak elhelyezve és a funkciók működéséhez meghívjuk a megfelelő osztályt, amit példányosítunk, és utána rá hivatkozva használunk és felhasználunk, mint objektum. Ez az elv a munkálatok során hatékonyabb kereshetőséget és rendezettséget biztosított számunkra, ami nagyban megkönnyítette a haladásunkat a munka során.

#### Az MVC jelenléte a weboldalon

A weboldalunk működésében nagyrészt szerepet vállalnak az adatbázis lekérdezések és műveletek. Nos, annak érdekében, hogy ez a rendszer dinamikus legyen és átlátható az MVC, magyarul a Modell Nézet Vezérlő elvét is felhasználtuk.

A lényege a szakdolgozatunkban, hogy léteznek az úgynevezett Dao(DO) fájlok, amelyek az app/Model mappában vannak elhelyezve és szerepük az, hogy lekérdezzék az adatbázis adatokat, amelyeket a Controller szerepű osztályoknak küldjenek, amelyek feldolgozzák az adatokat és elküldik a nézetnek, ami egy HTML állomány és ez megjeleníti a megfelelő formában a felhasználó számára. Egyébként más esetben a weboldalunk pont fordítva jár el. Vagyis a felhasználó megad valamit az input mezőben, amit továbbít a Controller php osztály a DO osztálynak, ami felhasználja a kapott adatokat az adatbázisban található értékek manipulálására.

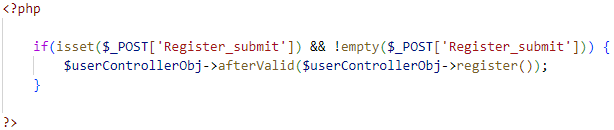
### A regisztráció funkció megvalósulása MVC,OOP szerint



A regisztráció azzal kezdődik, hogy a felhasználó megadja a jövőbeli fiókjának nevét, email címét, jelszavát, vezetéknevét és keresztnevét a „register\_page” oldalon. Ezután, ha a felhasználó rákattint a „Regisztráció” küldésgombra, akkor meghívja és elindítja a regisztrációs folyamatot, amit egy HTML kódba integrált php utasítással oldunk meg. A meghívás a Users\_Controller fájl egyik sikeres vizsgálat utáni feltöltésre hivatott metódusával történik.

Először persze el kell érni azt az adott osztályt, ahonnan példányosítani akarunk. A „use” kulcsszó segítségével adtuk meg az oldalnak, hogy mely elérési útvonalból érje el az adott osztály és csak utána hivatkoztunk rá:





Ezek után a Users\_Controller osztályban előhívott register nevű metódusában feldolgozza a kapott értékeket, amelyeket változókhoz társít és amiket átdolgozunk az adatbázis szempontjából biztonságos, bizalmas és kezelhetőbb formátumú adatokká:



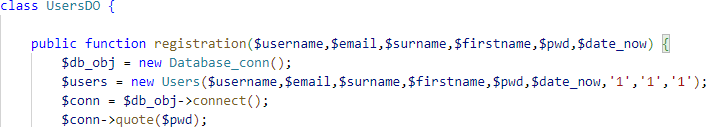
* **htmlspecialchars:** Eltávolítjuk a speciális karaktereket
* **hash:** titkosítjuk az érétket, mi a sha256-os módszert alkalmazzuk, így már titkosítottan kerül fel az adatbázisba
* **date(„Y-m-d h:i:s”):** Másodperce megadjuk, hogy mikor regisztrált a felhasználó

Ezután az OOP elmélete szerint a Users\_Controllerben létrehozunk objektum példányokat az UsersDO osztályból, amelyre később a hozzá társított változón keresztül hivatkozhatunk a példányosított osztályra és felhasználhatjuk a metódusait a kívánt funkcionalitásért:

Lényegében ezután a Users\_controller osztályban, ha a bemeneti érték átesik a szűrésen, akkor meghívjuk a UsersDoObjectpéldányból a „felhasznalok” tábla feltöltéséért felelős registration() metódust, úgy hogy megadjuk a Users\_Controller osztályban létrehozott változókat paraméterül, hogy velük dolgozzon és hajtson végre feltöltést a meghívott UserDO osztályból származó eljárás:

A „UsersDoObject” nevű osztály mielőtt megkezdené a feltöltést, felhasználja a példányosított adatbázis kapcsolódást végrehajtó osztályt, amiből meghívja azt a metódust, ami az adatbázis elérést hivatott megvalósítani.

Ezek után írunk egy SQL parancsot, amely egy utasítás az adatbázisnak. A változókat elhelyezzük a „Users.php” osztályának konstruktorába, tehát beágyazzuk őket egy objektumba.



Következőként továbbadjuk a konstruktor értékeit a hozzájuk tartozó visszatérési függvénnyel, amiket végül beágyazzuk az SQL parancsba úgy, hogy összekapcsoljuk a paraméteren keresztül kapott változókat az utasításan szereplő kettősponttal ellátott helyettesítő névvel. Ezek után az „execute” parancs segítségével végrehajtjuk az állapotot, amihez hozzárendeltük az adatbázis utasítást. Így megtörténik logikailag a regisztráció.



Végül elmondható, hogy teljesült az objektum orientált programozás elve, mivel már csak a regisztráció funkció 5 darab php fájlt és 5 darab osztályt használ fel úgy, hogy mindig csak a szükséges metódust hívjuk meg egy másik, vagy esetleg ugyan abból az osztályból, amibe paramétereken keresztül hivatkoztunk a felhasználandó változókra. Továbbá a felhasználótól kapott értékeket konstruktorba helyeztük és többnyire objektumként kezelve használtuk fel őket.

Azt MVC szabályok is többnyire teljesültek, mert mind a regisztrációs felület az ehhez tartozó „Users-Controller.php” állomány, és a „UsersDO.php” fájl más és más feladatokat lát el a funkción belül.

### A bement szűrése php-val a regisztrációnál

A regisztrációnál, ha a felhasználó rányom a regisztráció gombra, azzal nem csak a feltöltési folyamat indul el, hanem még az adatbázis folyamata elé beékelődik egyfajta bemeneti szűrés, annak az érdekében, hogy csak a megfelelő típusú értékeket engedjük be az adatbázisunkba, vagy, hogy a felhasználó akadály nélkül tudja majd a fiókját használni a weboldalon.

#### Az input szűrések létrehozása az InputValidate osztályban

Ha az input érték megfelel a követelmények 1-gyel, ha nem 0-val térnek vissza a szűrések.

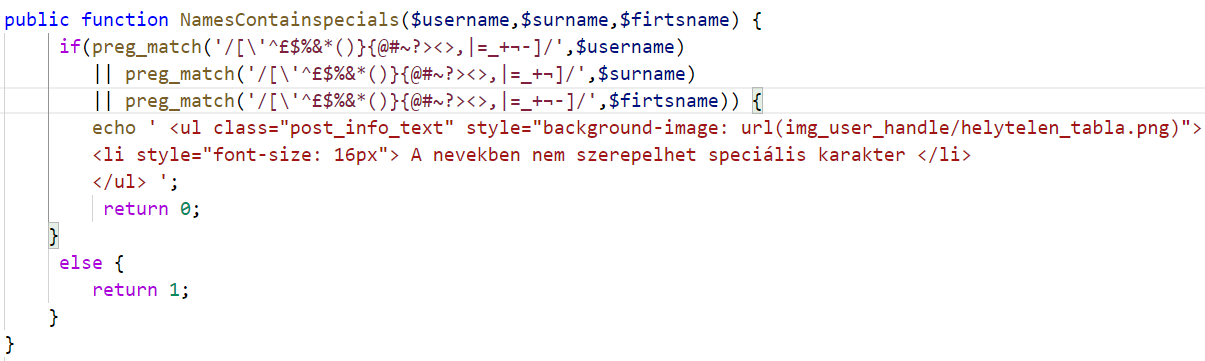
##### Felhasználónév hosszának szűrése

**

1. Meghatározzuk a minimumot és a maximumot
2. Betesszük egy filter\_var nevű php szűrő funkcióba, ami meghatározza, hogy a megfelelő intervallumban vannak-e a bementei értékek hossza.
3. Ha nincs benne akkor 0-t ad vissza és legenerál egy HTML alapú hibaüzenetet, ami megjelenítődik az osztály meghívásával és a megfelelő elágazásra téréssel. Ha nincs benne, akkor 1-et ad vissza és nincs hibaüzenet.

*A hibaüzenet a weboldalon*

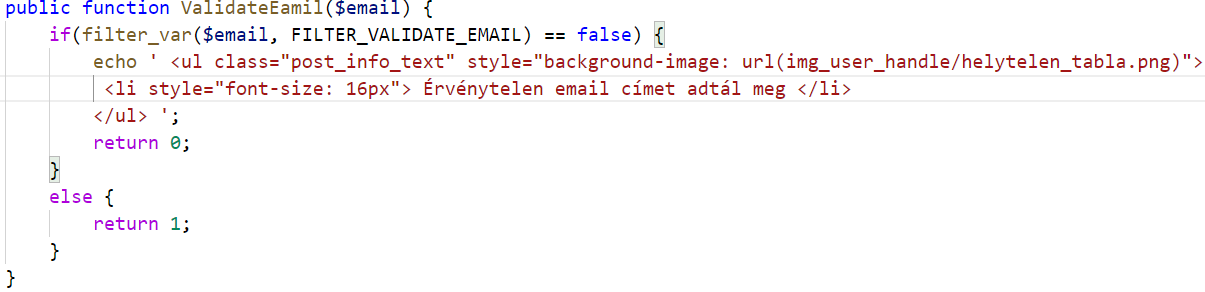
##### Felhasználónévnél, keresztnévnél és vezetéknévnél tiltott speciális karakterek

**Ha a felhasználónév, vezetéknév, vagy keresztnév speciális karaktereket tartalmaz, akkor szűrd ki, és térj vissza 0-val, különben 1-gyel.

1. A paraméteren keresztül megkapott input értékeket szűr, úgy, hogy mindegyik névhez elvégzi a regexen alapuló mintával való összehasonlítást a preg\_match() felhasználásával, ahol megadjuk mely karakterek tilottak.
2. Ha tartalmazza valamelyiket, akkor legenerálja a hibaüzenetet és 0-val tér vissza, különben 1-gyel.

##### Email helyességének validációja

Ha felhasználó helytelen email formátumot ad meg például: felhasznalo.com, akkor nem engedi regisztrálni. Ez arra készteti a regisztrálót, hogy adjon végül helyes email címet. A funkciót a FILTER\_VALIDATE\_EMAIL beépített php funkció oldja meg, amit egy feltételbe rakunk bele.

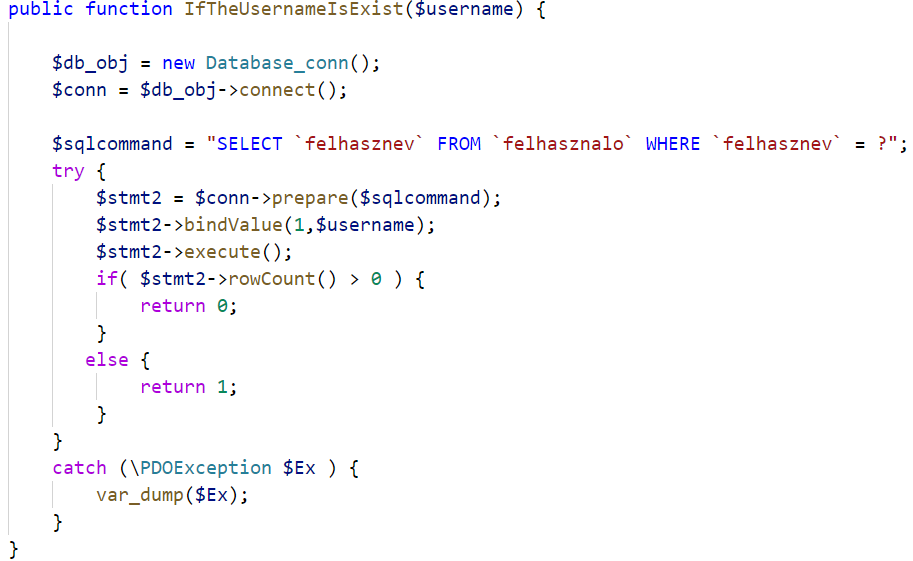


#### Ha a felhasználónév, email és jelszó már regisztrálva lett

A regisztrációnál természetesen szűrjük, ha a regisztráló olyan felhasználónevet és email címet ad meg, amely már létezik. Ha olyan jelszót írunk az bemenetbe, amely már el van tárolva, akkor ott azt jelenítjük meg, hogy helytelen jelszó. Ha az email cím, vagy a felhasználónév egyezik valamelyik adatbázisban tárolt értékkel, akkor értesítjük, hogy melyik lett már regisztrálva.

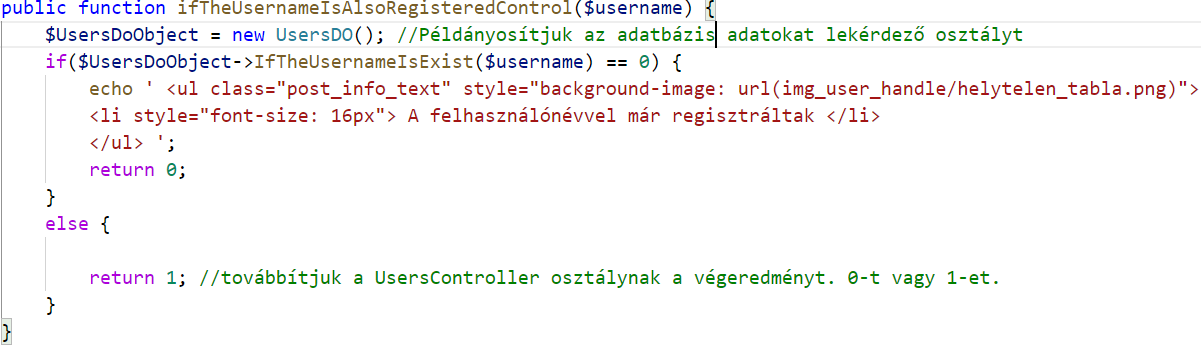
##### Ha a felhasználónév már regisztrálva lett

Első lépésként lekérdezzük az adatbázisból a regisztrált felhasználóneveket, amit a UserDO.php-n belüli osztályban a IfTheUsernameIsExist($username) függvényben teszünk meg.



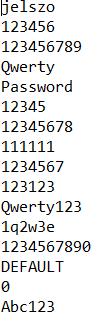
1. Először meghívjuk az adatbázis kapcsolatért felelős függvényt a conn nevű változóval.
2. Megírjuk az SQL utasítást, amellyel lekérdezzük a felhasználókat és ebbe a’?’ segítségével tudunk hivatkozni majd a paraméterként átadott felhasználónév értékre.
3. Összekötjük a kérdőjelet a username változóval a bindValue(1, $username) utasítás segítségével.
4. Végrehajtjuk az SQL parancsot az execute utasítással
5. Megvizsgálom, hogy a lekérdezés 0 sort adott e vissza a rowCount() > 0 használatával
6. Ha 0 sort adott vissza, akkor térj vissza 0-val, ellenben 1-gyel.

A visszatérési értéket ezek után továbbítjuk az InputValidate.php osztályába, ahol elvégezzük a szűrést stílussal a return érték alapján:



##### Híresen rossz jelszavak kiszűrése egy txt fájl alapján:

Ehhez a funkcióhoz kezdetben kell egy bad\_passwords.txt nevű állomány, amely tárolja az alábbi képen látható formában a rossz jelszavakat, amelyek feltöréséhez nem kell nagy tudomány:



Az állomány nagyjából 400 darab ilyen jelszót tartalmaz. Ezt az állományt az inputValidate osztály olvassa be és dolgozza fel a következőképpen:



1. Először megkeresi, hogy létezik-e a txt fájl.
2. Ha létezik, akkor a szöveget beolvassa a file\_get\_contents nevű eljárással és a szöveget egy values nevű változóhoz rendeli hozzá.
3. Leszűri, hogy a paraméterben kapott szó benne van e a szövegsorban és annak eredményeképpen tér vissza 0-val, vagy 1-gyel, amit majd egy másik metódus dolgoz fel a hibaüzenet legenerálására.

#### A szűrési eredmények érvényesítése a Users\_Controller osztályban

 A visszatérő értékeket az inputValidate osztály a Users\_Controller osztály dolgozza és használja fel mielőtt a regisztráció életbe lép.

A szűrési eredmények a következőképpen kerülnek érvényesítésre, ahogy a fenti képen is látható:

1. A szűrési eredményeket változókhoz rendeljük hozzá.
2. Ezeket elhelyezzük egy tömbbe.
3. Utána leszűrjük, hogy szerepel-e 0 érték a tömbben az in\_array() funkcióval. Ha igen, az azt jelenti, hogy valamelyik bemeneti érték nem felelt meg a kritériumoknak.
4. Ha szerepel 0 érték, akkor 0-val tér vissza, különben 1-gyel és végrehajtja a regisztrációt.

Ezután van egy metódusunk, ami elirányítja a felhasználót a bejelentkezésre, ha érték, amit megkap meghívásnál egyenlő 1-gyel. Azaz, ha minden mező megfelelt a követelményeknek.





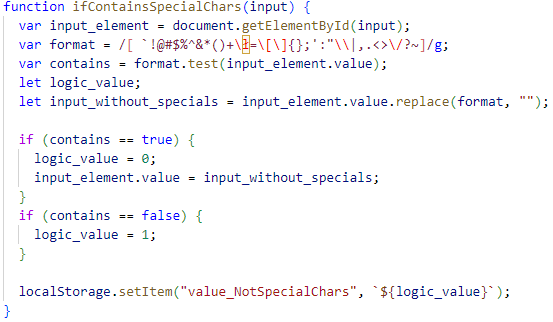
### A bemeneti hibák megjelenítése JavaScripttel a regisztráció oldalon

A weboldalunk regisztrációs felületén, ha a felhasználó rákattint a beviteli mezőre, akkor jelzi a program, hogy milyen neveket adhat meg a mezőben, továbbá, ha a regisztráció végrehajtása közben elkezdi kitölteni a mezőket, írás közben figyelmezteti egy dinamikusan megjelenő szöveg, hogyha rossz beviteli értéket adott meg, ami akkor jön elő, amikor a felhasználó elkattint a szövegdobozból. Mi azért találjuk ezt a funkciót hasznosnak, mert a felhasználó időt nyer azzal, hogyha a regisztráció gomb megnyomása előtt a rendszer eltávolítja a helytelen karaktereket és értesül a regisztráló mit írt be rosszul. A működés a HTML onchange eseményre kezdődik, amely meghívja a megfelelő JavaScript szerepeket.

A funkciót 2 JavaScript fájl biztosítja. Van egy input\_errors.js nevű fájl, amely elvégzi a szűrést függvények és REGEX segítségével. Továbbá van egy Write\_errors\_js.js nevű állomány, amely megjeleníti a megfelelő szöveget és stílust a bemenet alapján.

#### Felhasználónév bemenete

##### Speciális karakterek szűrése



Először van egy metódus, amely megkapja a felhasználónév beviteli mező értékét paraméterül és elvégzi a szűrést.

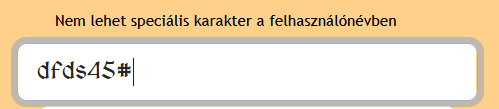
1. Van egy format nevű változó, amely deklarálja a lehető legtöbb speciális karaktert.
2. Létrehozunk egy logikai értéket contains néven, amely meghatározza, hogy az input mezőben van e speciális karakter
3. Ezután deklarálunk egy változót, ami egyenlő az input értékével, csak speciális karakterek nélkül.
4. Ha benne van valamelyik karakter, akkor cseréld le az ezen karterek nélküli szövegre az input szövegét és a logikai végeredményt állítsd 0-ra, különben 1-re.
5. Ezek után ezt a logikai értéket beágyazzuk egy globális localStorage tárolóba, amit fel tud használni a másik JavaScript fájl és a logikai érték szerint fogja formázni a stílust.

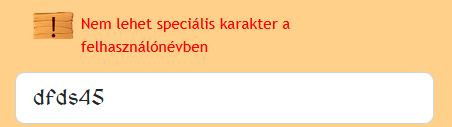
##### Legalább 2 karakter a szövegben

1. **Deklaráljuk a logikai értéket és az angol ABC betűit beletesszük egy tömbbe.
2. Ugyanúgy eljárunk a magyar betűkkel is.
3. A magyar és angol karaktereket tartalmazó tömböt összeolvasszuk egy all\_chars nevű tömbbe.
4. Létrehozunk egy username\_input\_text nevű változót, amely egy nagybetűssé teszi a beviteli szöveget a toUpperCase() függvénnyel
5. Bejárjuk a karaktereket egy ciklussal, amelyben megvizsgáljuk, hogy a nagybetűs szöveg indexein megtalálható-e az összeolvasztott tömb valamelyik eleme. Ha igen, akkor növeli c értékét.
6. Végül, ha a számított c érték nagyobb, mint egy akkor van 2 betű, vagyis akkor teljesül a feltétel és 1-et ad hozzá a stílus által felhasznált globális tárolóhoz. Különben 0-t ad át.

A Write\_errors\_js nevű állomány végül a localStorage tárolókban elhelyezett érték alapján dönti el, hogy melyik hibaüzenetet jelenítse meg és milyen stílusban.

***Előhívás a HTML kódban****.*

****Megmutatás példán keresztül:**



Ha elkattintok, akkor a piros fog megjelenni, úgy, hogy közben kitörli a „#” jelet.

### A beépített térkép és funkciói

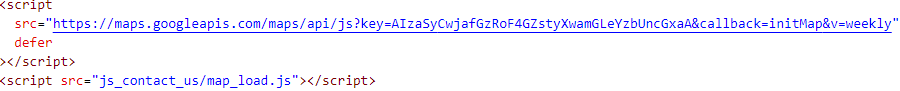
#### Röviden a funkcióról

Az elérhetőség oldalon a weboldalra látogatók használhatnak egy Google által származó integrált térképes alkalmazást a felületen belül. Egy ilyen funkció megvalósulása lesz a te helyzeted, illetve a vár elhelyezkedésének megmutatása a térképen, amelyeket a megfelelő gomb megnyomása után érhetnek el a felületet használók. Továbbá a funkció tud utca nézetet is, amelynél ugyan úgy mutatja a vár elhelyezkedésének a pontját. Ezek a programrészek a JavaScript programnyelv segítségével vannak megoldva.

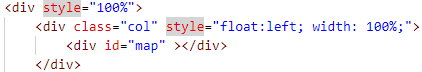
#### Térkép betöltése funkció a HTML oldalon:

A térképet egy divbe helyezzük a HTML kódon belül, amely divet a térkép tölti ki. Továbbá a HTML kód „head” részében van elhelyezve a JavaScript kód, amely előhívja a Google szerveréről a térkép adatait és az ahhoz tartozó kulcsot. A defer kulcsszóval a script párhuzamosan töltődik be a HTML-lel, de csak azután kerül feldolgozásra.

Ezután meghívjuk a map\_load.js JavaScript fájlt, ami felelős a térkép betöltéséért és az ehhez tartozó funkciókért.

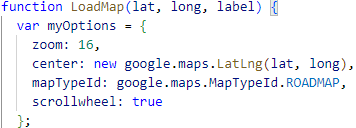


Továbbá, létrehozzuk a HTML divet, amely tartalmazza a képet, és ennek az id-ját a „map” -ot használjuk fel a JavaScript kódban a térkép paraméterek hozzáadásánál. A térkép mérete dinamikusan növekszik a div méretével és ezzel a módszerrel könnyen szegélyezhetővé válik a térkép.



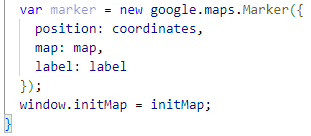
#### A térkép betöltésének megvalósítása a mapload.js fájlban

*Először map\_load.js állományban létrehozunk egy kulcs érték páron alapuló beállításértéket a térképhez a LoadMap metódusban, ahol a térkép típusát és a kezdeti értékeket állítjuk be.*

**

*Továbbiakban a LoadMap metódusban, következőképpen létrehozunk egy térképet, amelyhez hozzárendeljük a kijelölt divet id alapján és az előbb létrehozott beállításértékeket:*

*Utána, beállítjuk, a térképen hol legyen a* *jelölő ikon alapértelmezetten. Ugyanis itt beállítjuk a jel feliratát és pozícióját:*

**

Végül létrehozunk egy metódust, ami felhasználja a betöltést és ezt a metódust meghívjuk az elérhetőség HTML oldalon lévő <body onload="loadOriginalMapView()

eseményére, hogy az oldal betöltésével együtt töltsenek be a térképadatok a megadott értékekkel, amiket paraméterként adunk át:



#### A felhasználó pozíciójának megjelenítése geológiai funkciókkal

A felhasználónak lehetősége rejlik meghatározni a saját elhelyezkedését, amit a térkép mutat a számára.

**geo\_location():** a metódussal meghatározzuk a felhasználó koordinátáit és megjelenítjük a térképen. A getCurrentPosition() függvénybe ágyazzuk bele a koordinátákat és a hibaüzeneteket.

**showPosition():** Betölti a mapbetöltő funkciót az eszköz elhelyezkedésének adataival.

**showError():** Megjeleníti a hibaüzenetet, ha esetleg nem érhető el a pozíció, vagy ha megtagadták a helyadatokhoz való hozzáférést.

### Email küldése nekünk az elérhetőség oldalról

Az elérhetőségek oldalon lehetőség kínálkozik levelet küldeni a vár adminisztrációjának. Ezt azért tartjuk fontosnak, mivel a weboldalt használók ennek segítségével egyszerűbben tudnak jelezni tájékoztatásokat az adminisztrátorok felé. Esetlegesen tudjál őket hibákról tájékoztatni és kérdezni tőlük a platformon keresztül.

#### A levelező rendszer megvalósulása

Az email küldő rendszert a phpmailer segítségével oldottuk meg, amely az SMTP szervert használja fel emailek küldésére az alkalmazáson keresztül.

***Első lépés:*** *Az email küldő rendszer telepítése és hozzárendelése a weboldal állományához*

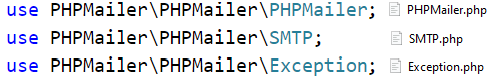
A phpmailer levelező funkciót először telepítettük egy cmd paranccsal a weboldal mappájába elnavigálva a parancssoron belül. A paranccsal legenerálható az email küldéséhez szükséges állományok és szerelvények:

**

***Második lépés:*** *A levélküldés megfelelő beállítása*

Annak érdekében, hogy a levélküldés sikeres legyen egy php állományban meg kell adnunk a szükséges paramétereket és a hitelesítést. Ezt az Email\_sender.php fájlban tesszük meg, ami felelős a beállításokért és magáért a levélküldésért.

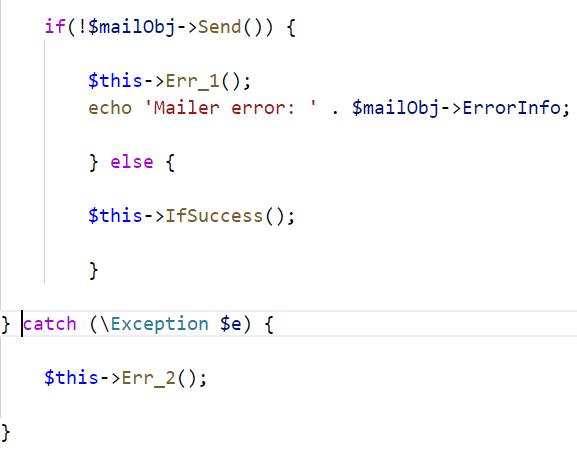
A beállításnál elsőként felhasználjuk a cmd parancs által leképzett a működést biztosító bázisállományokat, amelyek tartalmazzák az SMTP szerverhez való csatlakozási modulokat:

**

Ezután létrehozzuk az Email\_sender osztályt, amiben elvégezzük a beállításokat:

**

Ha érvényesek a beállítások és elérhető a levelezés, akkor elküldi a weboldal az üzenetet emailen keresztül, különben megjelenít egy hibaüzenetet.

**

Végezetül, ezt az Email\_Sender osztályt az EmailController osztály használja fel, úgy, hogy a sikeres validáció után, ha megfelelt, mind a megadott cím és az üzenet szövege, akkor meghívja az Email\_sender osztályt, amivel elküldjük a levelet. Ha sikeres a levélküldés, akkor a program olyan ágra fut, ami megjeleníti azt is, hogy sikeres volt a email kézbesítése.

#### Az email bemenetek szűrése

A levél küldése során, vagyis mielőtt elküldené a weboldal levelező szerver segítségével az üzenetet, egy szűrés hajtódik végre a ValidateEmail osztályban ahol létrehozzuk a szűrés eredményét kiváltó függvényeket, amik meghívásával az EmailController osztályban hajtunk végre validációt. Ha az egyik feltétel nem teljesül a felület hibát dob és nem küldi el a levelet, ha felhasználó a következőképpen nyomja meg a küldés gombot:

1. Ha a felhasználó nincsen bejelentkezve
2. Üres mező szerepel az emailnél, tárgynál, vagy a szövegnél
3. Helytelen email formátumot adott meg a weboldalt használó
4. Az üzenet tartalmaz linket, amely az alábbiak szerint működik és figyelmeztet



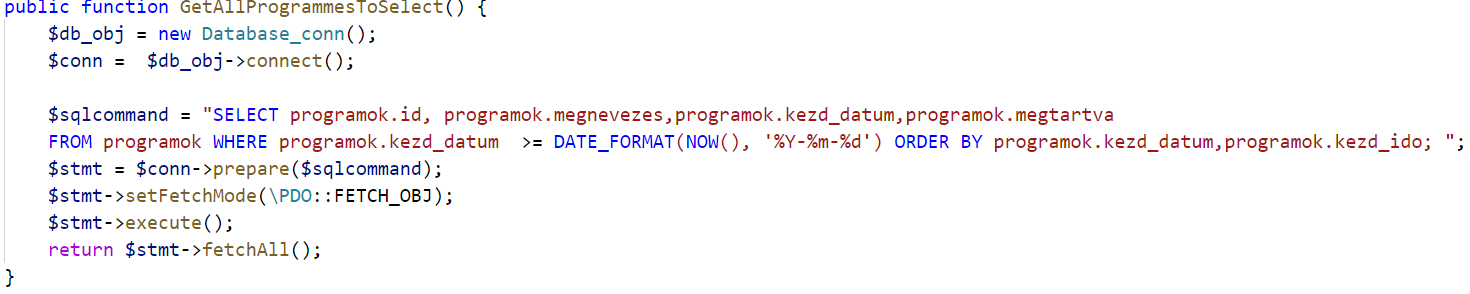
### A várprogramok listázása

A felhasználó megtekintheti a várban szerevezendő programokat, ha megnyitja a „jövőbeli programjaink” menüpontot. A listázás az MVC és OOP elvek szerint hajtódik végre. A felületen 3 féle listázás létezik: összes program, szűrt programok és kiválasztott programok lajstroma.

#### Összes jövőbeli program listázása:

Azokat a programokat listázzuk itt ki, amelyek még nem kezdődtek el, vagy még tartanak. Itt lényegében a programhoz tartozó egyik véletlen képet generáljuk le a program mellé, illetve a hozzá tartozó lényegesebb adatokat.

A funkció működéséhez legelőször az adatbázis adatokat kell megszereznünk így a ProgrammesDO osztályban a GetAllProgrammesToSelect() metódussal végrehajtjuk a lekérdezést.



Először, a már megszokott módon létrehozzuk az adatbázis kapcsolatért felelős változót. Ezután megalkotjuk az SQL utasítást, ahol a program kezdeti dátuma alapján listázunk, mégpedig úgy, hogy a kezdeti dátum legyen nagyobb, vagy egyenlő a mai nappal. Ezt a DATE\_FORMAT SQL függvény hatására tesszük meg, amibe beletesszük a NOW() kulcsszóval jelölt a mai napot tartalmazó időbéjeget és utána a '%Y-%m-%d' karakterlánc segítségével megadjuk a támogatott időformátumot. Végül rendezzük először a kezdeti időpont, majd az kezdő idő alapján a megjeleníteni kívánt adatsorokat az SQL alapú ORDER BY rendező-utasítás alapján.

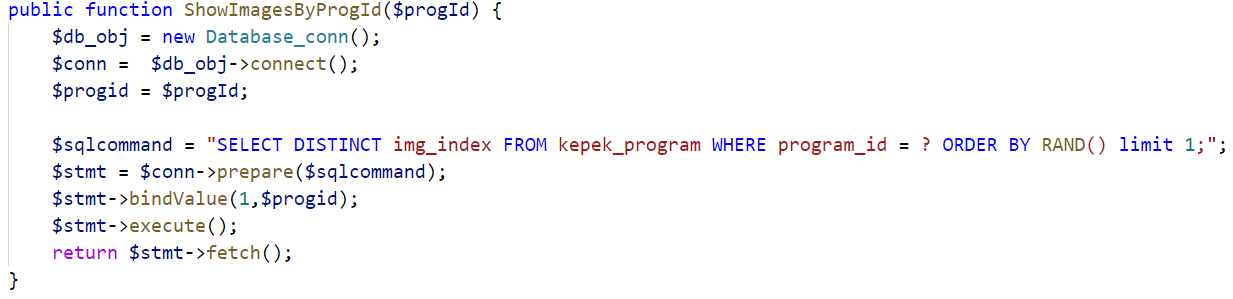
A következő lépésben végrehajtjuk az utasítást és a return $stmt->fetchAll() php paranccsal pedig továbbadjuk a lekérdezésből előhozott rekordokat a ProgrammesController osztálynak, amely osztály feldolgozza és legenerálja soronként és oszloponként az értékeket. Szóval, a kontroller szerepű osztály FillAllProgs() metódusa lenyomtat egy HTML táblát a programok felületen a megfelelő adatbázis adatokkal.

#### Kép megjelenítése a program alapján:

A képek megjelenítését az elérési útvonal alapján tesszük meg úgy, hogy elmentjük a programhoz tartozó képmappát és az abban lévő képeket és formátumaikat és a kép a szerver mappájából töltődik be. Láthatjuk a következő példák alapján:

**

A kép a program azonosítója alapján töltődik be a lista generálásánál véletlenszerűen, amely azt jelenti, hogy a programokhoz tartozó képek közül véletlenszerűen fogja valamelyiket megjeleníteni, legyen akármennyi kép is az adott programhoz rendelve.

**** Az alábbi kép pedig azt mutatja, hogy hogyan választ a program ki egy adott programhoz tartozó képelérést véletlenszerűen és hogyan tér vissza az értékével a paraméteren keresztül kapott program id-ja alapján.

A véletlen kiválasztást az SQL parancsban található ORDER BY RAND() limit 1 utasítás végzi el, ami a egy darab véletlenül lekérdezett értéket jelenít meg a „kepek\_program” táblából.

#### A listázott adatok megjelenítése

A listázott program adatokat, a program állapotával és az eseményhez tartozó random képet végül a PorgrammesController oszály jeleníti meg egy HTML táblázat formájában. A funkciót az alábbi képen tudjuk végig követni.



1. Meghívjuk a szelektált program adatait a programmes nevű változóba, ami így egy tömb lesz gyakorlatilag.
2. Végig járjuk a tömböt egy foreach ciklussal, amivel minden sor adatait legeneráljuk egymás után, így megjelennek a lekérdezésre és innentől tudunk az oszlopokra hivatkozni a foreach ($programes as $prog).. $prog->$oszlop\_neve beszúrásával.
3. Beletesszük a prog\_picture változóba az előhívott kép elérését, a fent említett ShowImagesByProgId($prog->id) alapján.
4. Leszűrjük az file\_exist paranccsal, hogy létezik-e a fájl, azaz a kép.
5. Ha nem, akkor töltse be az alapértelmezett képet, ha igen akkor add tovább a megfelelő képet, amelyhez illik az elérési útvonal az „img\_index” mező hivatkozásával.
6. Végezetül generáld le a táblába az összes jövőbeli, vagy aznapi programhoz fűződő információkat az oszlopértékek beszúrásával. Továbbá illeszd be a megfelelő azonosítóhoz tartozó képet véletlenszerűen. A program állapotát, amelyet feltételes formázással jelenítünk meg, hogy például „Elmarad” a program, vagy, hogy „Megváltozott a program időpontja” a SetStyleOfProgTimeInfo($state\_value) fügvénnyel generáljuk le.

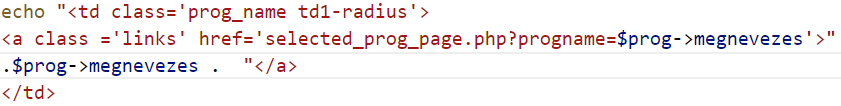
#### A programok listájának megjelenítése szűrés alapján

A jövőbeli programjaink felület olyan funkciót is tartalmaz, amely szűrési művelettel listázza a szűrési feltételeknek megfelelő program adatokat. A szűrési eredményeket egy tömbbe tesszük a ProgrammesController osztályban, amit tovább küldünk a ProgrammesDO osztálynak, ami a tömb elemei alapján lekérdezi a megfelelő sort, amit továbbít visszafelé a Contollernek, amely elküldi a megjelenítésnek a szűrt programot egy táblázat formájában. Ez a funkció nagyban hasonlít a korábban leírt összes program listázásával, annyi eltéréssel, hogy a csak azokat a programokat hívjuk elő az adatbázisból, amelyek megfelelnek a szűrési feltételeknek, mint például, hogy mikor kezdődik a program, milyen korosztálynak ajánlott és milyen a program típusa.

### A kiválasztott program funkciója

#### Az oldal betöltése program neve alapján

Ha a kilistázott programok közül valamelyiket kiválasztjuk, akkor azzal megnyitjuk a kiválasztott program felületet, úgy, hogy a linkbe beágyazzuk az előző oldalon (összes jövőbeli programjaink) oldalon megjelölt program nevét, amely így az új oldal megnyitásakor benne lesz a linkben. A hivatkozás létrehozására a ProgrammesController php osztály HTML kód kiiratásánál kerül sor, ahol legeneráljuk a program nevét a táblába és a hozzá tartozó linket. A program kiválasztása után a kiválasztott program id-ja session értékként fog elhelyeződni mindaddig, amíg felül nem írjuk egy másik programm kiválasztásával.



Például kiválasztjuk a „lángzsonglőr” programot, a kilistázott porgramok közül, akkor megjelenik a kiválasztott program oldal a megfelelő linkkel.



#### A program adatai a kiválasztott programok oldalon

#### A felhasználónak megjelenik, hogy mennyi ideig fog tartani a kiválasztott program. Ezt a SelectedProgrammes osztályban valósítjuk meg, úgy hogy leképezzük a DateTimeImmutable beépített php osztályt, mind a kezdésre és befejezésre és azután pedig a diff() függvénnyel visszaadjuk a 2 idő különbségét, amiből az órát és percet egy tömbbe pakoljuk. Ezt a tömböt elküldjük az időintervallum megjelenítésére, az oldal felületére.

#### A program felvétele a kedvencek közé

A kiválasztott program felületén a felhasználó képes felvenni a kiválasztott programot egy csillag ikon segítségével, amely program bekerül az adatbázis megjelölt programok táblájába, amint rányomott az illető arra a bizonyos ikonra. Ennek a funkciónak a működése nagyrészt JavaScript és nagyrészt PHP kódsorokból áll.

#### Megfelelő ikon leképzése a hozzáadás a kedvencekhez funkciónál

Ha a felhasználó már felvett egy programot a kedvencek közé, akkor a pipa jelű tábla jelenik meg számára, ha nem akkor az egyszerű csillag. Ahhoz hogy meg tudjuk állapítani, hogy az aktuális fiók hozzáadata e az adott programot, ahhoz meg kell állípítani, hogy melyik az éppen használt felhasználó és melyik programokra kattintott. Ezt a FilterifUserHasAlreadyInterest() nevű függvénnyel tesszük meg.

A metódusban meghívjuk a SelectedProgrammesDO osztályból a CheckIfRowOsNotEmptyProgInterested() nevű metódust, amely megállapítja az adatbázisban található „megjelolt\_programok” táblából, hogy létezik-e a felhasználónevet és kiválasztott program azonosítóját tartalmazó rekord, és ha igen 1-et ad vissza, vagyis már megjelölte. Különben 0-t továbbít, ha nem létezik a sor.

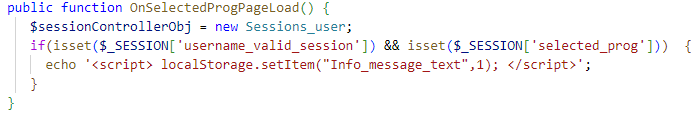
**

 Először látható, hogy van egy FilterIfUserHasAlreadyInterest() nevű függvényünk. Ebben meghívjuk azt a bizonyos CheckIfRowOsNotEmptyProgInterested() metódust, amely paramétere rendelkezik a bejelentkezés utáni munkamenetből(session) kapott felhasználónév és a kiválasztott program azonosítójával, vagyis ellenőrzi, hogy a felhasználó már felvette-e korábban az eseményt a kedvencek sorába. Vagyis ennek a metódusnak az eredményét továbbítja a munkamenetek felhasználásával.

A WriteInterestStar(), megkapja az eredményeket a fent említett függvénytől, és ha 0-t kap, azaz még nincs a program hozzáadva, akkor a hozzáadás csillag ikont mutassa, különben ha 1-et kap, akkor a törlő ikont jelenítse meg.

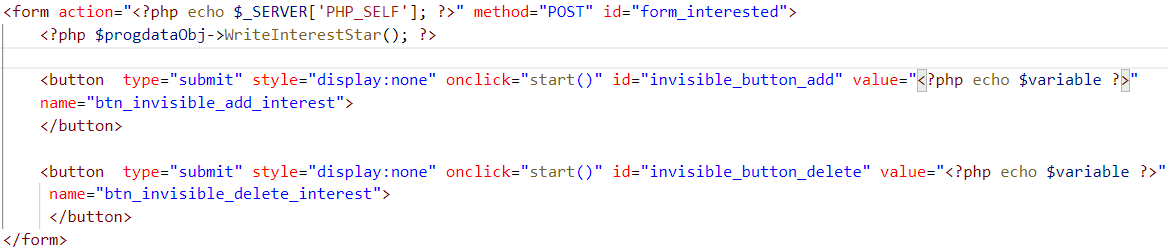
#### Bejelentkezésre irányuló figyelmeztetés megjelenítése

A felhasználó nem adhat programot a kedvencekhez, ha nincs bejelentkezve. Ennek a kimutatására php által leképzett HTML kód alapú tájékoztatást nyújt az oldalon. A funkcióban a bejelentkezés után létrehozott, a felhasználónevet képviselő „username\_valid\_session” SESSION változót használjuk fel, ami alapján létrehozunk egy JavaScript globális localStorage értéket a PHP kódban. Ezt az értéket minden JavaScript állomány használni tudja, amelyik meghívódik az aktuális HTML kódban.

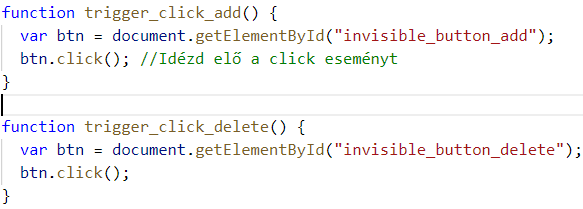


Az OnSelectedProgLoad() PHP metódus az oldal betöltése során kerül meghívásra. A működése során megvizsgálja, hogy a bejelentkezést és a kiválasztott program értékét képviselő SESSION érték létezik e, azaz be van e jelentkezve, és hogy kiválasztotta a programot az illető. Ha léteznek az értékek, akkor létrehozza az átfogó JavaScript változót, amelyet az info\_message.js fájlban használunk fel annak érdekében, hogy megjelenítsük, vagy elrejtsük a hibaüzenetet.

#### Felvétel a kedvencek listájára

A művelet működésének első részeként meghatározzuk, a szükséges bementi HTML vezérlőket. A csillag, mint kép még önmagában nem képes arra, hogy rákattintásra POST információkat küldjünk. Ezért úgy oldottuk meg, hogy érzékelje a posztolást a képkattintás, hogy létrehoztunk 2 láthatatlan gombot a kép alatt. Egyet, ami a hozzáadásra van fenntartva és egy másikat, amely a törlésre. A képek kattintására előidézzük JavaScript segítségével a gombok kattintási állapotát, ami már végrehajtja a POST műveletet.

A JavaScript metódusok működését, a hozzáadás vagy törlés képekre való kattintás idézi elő, amelyek beindítják a klikk eseményt így a POST php funkció is végrehajtódik, amely a bejelentkezés szűrése után a megjelölt program feltöltését esetleg törlését is elvégzi, attól függően, hogy melyik képre kattintunk.

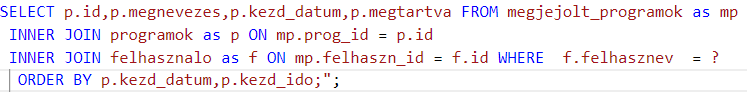




A hozzáadás, hozzáadott és törlés ikonjai

Ha beindítjuk a posztolást a kiváltó JavaScript művelet után, akkor meghívódik a SelectedProgrammesController osztályban létrehozott ByClickTheStar() vagy a megjelölt program eltávolítása során a ByClickDeleteIntereset() metódusok, a melyek először ellenőrzik, hogy a felhasználó be van-e jelentkezve, ha igen akkor végrehajtja az adatbázis műveletet a törléssel, feltöltéssel.

Így, ha felvesszük a programot és visszalépünk, akkor utána kilistázhatjuk a saját programjainkat, amely ugyan-úgy hajtódik végre, mint a szűrt és az összes program listázása, csak azokat a programokat mutatja a táblában, amely események azonosítója benne van a „megjelölt programok” prog\_id mezőben és a bejelentkezett felhasználó neve is benne található id alapján az összekapcsolás után.



A megjelölt programok kilistázásáért felelős SQL parancs a php kódban

### Megjegyzések a weboldalon

#### A megjegyzések oldalról általánosan

A weboldalunk egy mellékoldala a kommentekhez kapcsolódik. Azt kell tudni erről a felületről, hogy azért tettük bele a munkánkba, mert hasznosnak tartjuk, hogy a felhasználók meg tudják egymás között, az adminisztrátorokkal, esetleg a szervezőkkel osztani a megélt tapasztalatokat a várral és az abban szervezett programokkal kapcsolatban.

#### Megjegyzés írása

Ha a felhasználó, kommentet ír, akkor először a CommentsController nevű osztály végzi el a komment adatok feldolgozását és előkészítésést a CommentsDO adatbázis műveletek végző osztálynak. Az osztályban létezik egy UploadCommentsController metódus, amely megkapja a kommentből eredő értékeket.

A UploadCommentsController metódus először definiálja a HTML oldaltól kapott értékeket egy változóba és megállapítja az éppen bejelentkeztetett felhasználó azonosítóját GetuserId függvény meghívásával. Ezután elvégzi a szűrést miután a „send\_comment\_btn” gombot megnyomtuk. A szűrés során, található „$this->” kifejezéssel hivatkozunk a jelenlegi osztályra, amivel pedig megadjuk a validációs metódusokat, hogy eldöntsük, hogy az illető be van e jelentkezve, jogosult e a kommentre, vagy kitöltött e minden mezőt.

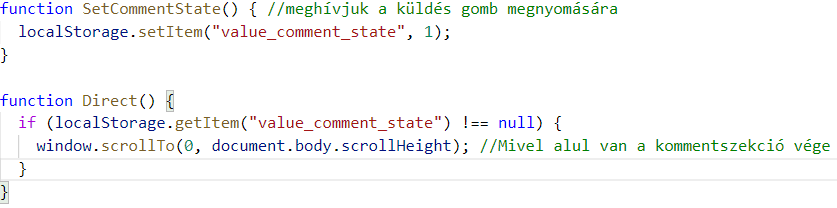


1. FilterIfcommentIsNotEmpty($comment\_value) metódussal validálhatjuk, ha üres kommentet akar a felhasználó küldeni.
2. A ValidateIfUserRegistered() szűri, ha a felületet használó be van jelentkezve, vagy nem.
3. FilterUserCanComment() pedig azt szűri ki, hogy a felhasználó meg van e tiltva a megjegyzések írására, amit az adatbázisból kérdez le a weboldal.

Ha az input megfelelt a kritériumoknak, akkor a program elvégzi a komment feltöltését a meghívott CommentsDO osztály UploadComment(input értékek….) metódusával.

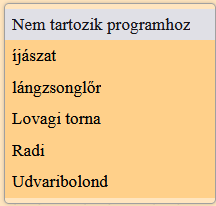
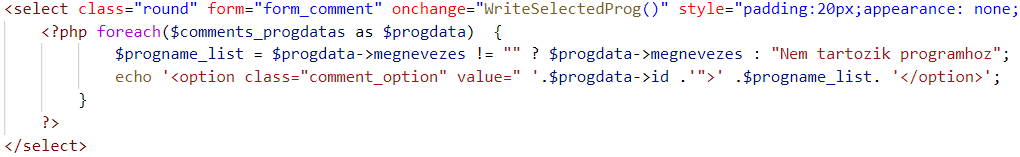
#### Scrolldown funkció az oldal betöltésekor:

Ha a felhasználó már írt kommentet, vagy ha akart korábban akkor a kommentek felölet betöltésekor egy az oldal betöltése eseményre meghívott JavaScript állomány segítségével legörgeti a kommentek aljára az oldalt, ahol lehetőség van új megjegyzést írni. Azt, hogy a felhasználó használta már előtte a megjegyzés küldését, egy globális localStorage változó jelzi, amit küldés során állítunk be, aminek a „value\_comment\_state nevet adtuk”. Ez az érték, ha nem null, vagyis, ha már létre hozta a felhasználó a küldés gommbal, akkor az oldal betöltődéskor, frissitésekor le fog görgetni az oldal.

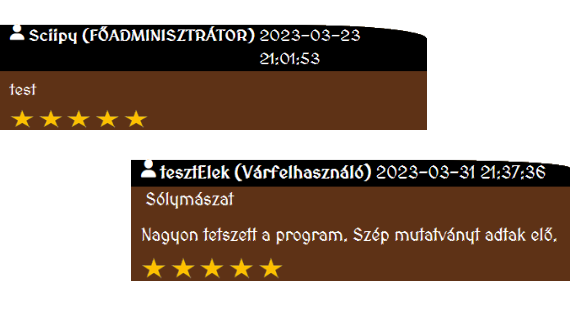


#### A kommentek eredménye

A komment elküldése előtt a felhasználó kiválaszthatja, hogy hány csillagot szeretne adni a kommentje mellé, és hogy melyik programhoz fűzze azt. Egy listából válogathat a felhasználó, amely listát adatbázisból nyert programnevekkel töltünk fel.

A comments\_progdatas egy tömb, amit feltöltöttünk a CommentsController által továbbküldött és az adatbázisból kiszelektált program adatokkal. Ezt a tömböt gyakorlatilag bejárjuk egy foreach ciklussal, amelyen belül kivesszük a bejárt elemből csak a programok nevét. Ezután egy feltételes névadás következik, ahol, ha a program neve üres, akkor a „Nem tartozik programhoz” mondatot írja ki, különben normálisan kiírja az adatbázisból előgyártott nevet. Ezután megjelenítjük az adatokat.

Ha a felhasználó elküldte a megjegyzést és a küldés sikeres volt, akkor megjelenik számára az új komment. Ha nem választott ki programot, amelyről írni szeretett volna akkor a komment keretben sem fog megjelenni semmilyen programnév. A megjegyzésben megjelenik a kommentáló joga, a neve, hogy milyen programmal kapcsolatosan írt és az exact ideje a programnak. Ezen adatok mind az adatbázisból hívódnak elő.



#### A megjegyzések törlése

A weboldalon belül erősen szükséges egy gyors megjegyzéstörlő funkció a felületen. A törlést csak adminisztrátorok és a fő adminisztrátor végezheti el. Azért fontos ez a művelet, mivel vannak trágár felhasználók és nem tenne jót a weboldalnak, ha sokáig ott lennének rosszindulatú hozzáfűzések az oldalon.

#### A megjegyzések törlése funkció működése

A kommentekhez szükséges lekérnünk a bejelentkezett felhasználó adatait, így a névvel együtt előhívjuk a felhasználó jogát is. A CommentsController osztályban úgy generáljuk le a megjegyzéseket, hogy ha a bejelentkezett felhasználó jogosultsága 1, vagy 2, azaz adminisztrátor, vagy fő admin, akkor a kommentekhez legenerálódik a weboldalt kezelők elé egy törlő ikon, amelyre rákattintva elirányítja a felhasználót a DeleteComment oldalra, úgy, hogy átadja a linken keresztül a törlendő hozzáfűzés azonosítóját.

A törlés oldalon ezután meghívásra kerül egy GET http változón keresztül a megkapott id és ennek az értékét a program eltárolja egy value\_id nevű változóban. Ezután meghívásra kerül a DeleteCommentDO osztály, ami törli az adatbázisból azt a megjegyzést, aminek az azonosítója egyenló a value\_id-val. Legvégül JavaScript alapú window.location.href kifejezéssel visszairányítja az adminisztátort a kommentek oldalra, ahol betöltődnek újból a hozzáfűzések a törölt komment nélkül.



### A galéria oldal a weboldalon

A galéria oldal lényege, hogy kilistázzuk azokat a programokhoz tartozó képeket, amelyek már véget értek, vagyis a befejezési dátuma kisebb értékű a mai jelenlegi napnál. Célja a felületnek, hogy megmutassa a felhasználóknak, hogy a várban bemutatott programok milyenek voltak. A képeket a szerver mappáján belül a galery\_handle/galery\_images almappában tároljuk a galériákhoz és az adatbázisban csak a képek neve tárolódik. Az oldalra csak azokat a képeket jeleníti meg a program, amelyek címe és formátuma benne van a „galeria\_kepek” táblában és egyben megtalálhatóak a kijelölt galéria képek mappában.

### Sessionkezelés a weboldalon

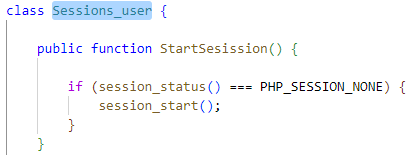
#### Session használatának indoklása

A bejelentkezés és ennek validációja, továbbá a kiválasztott program nevének felhasználása, mind a SESSION azaz munkamenet technológiájával van megoldva. Ennek a fő funkciója, hogy képes adatokat globálisan tárolni. Így, ennek az értékét használjuk más php osztályokban is, mint szűrési feltétel, vagy csak a felhasználói élmények fokozására, úgy, hogy nem kell még egyszer begépelni bizonyos értékeket a weboldalt használóknak.

#### Session indítása

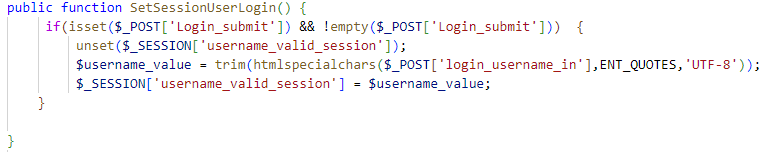
A munkamenetek létrehozását egy Sessions\_user nevű osztályba helyeztük el, ahol metódusok formájában valósulnak meg ezek a munkamenetek kezelése és értékük továbbítása.

A sessionok felhasználása előtt a munkamenetet el kell kezdeni a session\_start() függvénnyel, de mi előtte megadjuk azt a feltételt, hogy csak akkor kezdje el az új munkamenetet, ha a session státusza még nem aktív. A StartSession() metódust, minden munkamenetet igénylő funckiónál meghívjuk.



#### A felhasználó session értékének beállítása és eltárolása a bejelentkezésnél

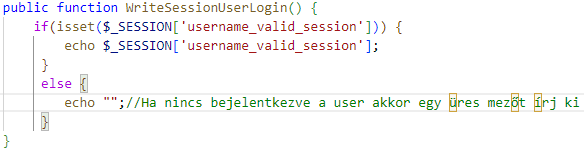
Ha sikeresen bejelentkezünk, akkor eltárolódik a felhasználónevünk munkamenet értékként, amit számos művelet során fel tudunk használni a weboldalon. A felhasználónév munkamentbe ágyazását a SetSessionInvalidUserLogin() metódus végzi el és a “username\_valid\_session” névre hallgat a nevet reprezentáló globális változó.

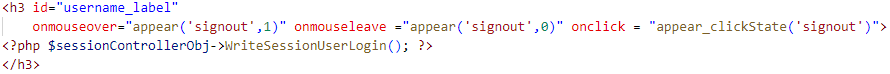


#### A felhasználónév kiiratása

A fejlécben kiírjuk a felhasználó nevét, ha be van jelentkezve. Ezt a nevet is a munkamenet segítségével oldjuk meg.

Először a létrehozzuk a Session\_user osztályban a GetSessionUserLogin() metódust, ami visszatér a felhasználónév session értékével, amelyet fejlécben iratunk ki.

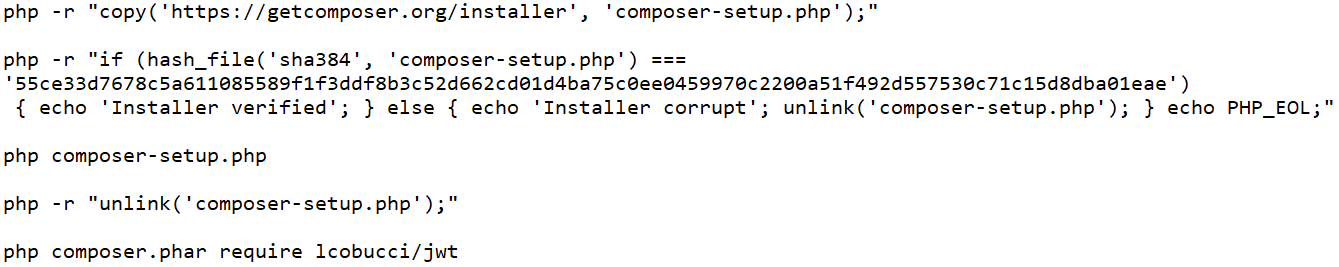


 Ezután a header.php állományban a Session\_user osztályt példányosítva kiiratjuk a nevet a metódus meghívásával, úgy, hogy a leképzett felhasználónéven keresztül lehet ki is jelentkezni egyben. A kijelentkezés úgy történik, hogy ha a felhasználó ráviszi az egeret, vagy rákattint, a fejlécben elhelyezett névre akkor meghívódik egy kijelentkezés opciót megjelenítő appear() nevű JavaScript metódus, amellyel előhívunk egy a kijelentkezést szolgáló menüpontot.

### A composer működése a weboldalon

A weboldalon a nagyobb dinamika elérése érdekében fehasznlájuk a composer technológiát. A weboldalon ez a segédprogram biztosítja az autoload funkciókat megvalósulását osztályok között és az emailezés is rajta keresztül valósul meg.

A composert letöltöttük, mint szoftvert és utána futtattuk a telepítését a parancssorban, a weboldal állományában egy szerverről elérhető online csomag meghívásával. Ezután a telepítés legenerálta a vendor mappát, ahol elhelyezte az automatikus betöltéséhez és a phpmailerhez szükséges fájlokat és osztályokat.



**A composer telepítése a cmd-ben**

#### A composer autoload konfigurálása

A composernek megadhatjuk, hogy mely könyvátarak között navigáljon, ezeket hogyan kapcsolja össze, és hogy milyen módon tegye azt. Továbbá, hogy milyen hozzáadott technológiákat vegyen figyelembe a programon belül. Ezt a composer által legenerált composer.json konfigurációs állományban tehetjük meg a következőképpen:



**Autoload:** Itt megadhatjuk az automatikus betöltés beállításait, hogy honnan indítsa a műveletet, és hogy mely mappákra használja azt.

**Requrie:** megadja a szükséges implementált technológiákat, amelyeket a composer felhasznál.

**optimize-autoloader:** beállítjuk, hogy optimalizálja automatikusan az újjonnan beszúrt osztályok elérését, így gyorsan adható új osztály a könyvtárhoz, amelyet már meg is tudunk hívni az autoloader segítségével.

Az autoloader funkció úgy működik az oldalakon, hogy meghívjuk az autoload osztályt, ami dinamikusan elintézi helyettünk az osztályok egyesével való meghívását. Még a Controller és Model osztályok között sem kell meghívni az állományokat, hanem csak azon az oldalon kell a vendor\autoload.php fájlt meghívni, amely feldolgozza a php osztályok metódusait. A meghívás egyszerűen a következőképpen zajlik minden php osztályt feldolgozó oldal php kódsoraiba:



### Az ágyú animáció a weboldalon

#### Ágyú animáció a fejlécben

Az ágyú animáció egy JavaScript alapú és időintervallumon működő képmozgás és képváltoztatás, amely a felhasználói élményt kívánja fokozni. Az animáció alkatrészei a Cannon\_shoot mappában találhatók, amely tartalmazza a mozgást előidéző JavaScript cannon\_shoot\_control.js állományt és a szükséges stílust és képeket.

Az animáció, úgy működik, hogy a fejléc HTML oldalon van egy képkeret, amire hivatkozunk a move\_cannon() metódusban, amelybe belehelyezzük az ágyú nyugodt állapotában lévő képét. A képkeretre eztuán meghívunk kezdetben egy intervallumot, amelyre beállítjuk a frame1() eljárást, amivel elindítjuk a képkeretet az X tengely mentén. Ha egy apró lépést tesz a keret hirtelen átváltjuk az ágyú tüzelési állapotát mutató képre és még egy kis lépés után pedig visszaváltjuk a nyugalmi állapotú képet, ezek a kis lépések az emberi szem számára hirtelen bekövetkeznek. Az ágyú kerete a képpel ezután tovább halad a tengely mentén szimulálva az ágyú hátralökődését. Ha eléri a maximum vízszintes pontot, akkor meghívjuk a frame2() eljárást, amviel visszafelé kezd haladni a kezdő pontig.

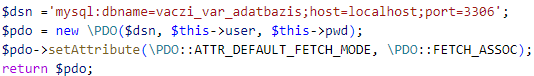
A SetIntervalX eljárásban az animációnak beállítunk egy körübelül 8 másodpercenkénti ismétlődést biztosító időintervallumot, amibe beágyazzuk a move\_cannon() animációt, így lejátszódik az animcáió minden 8-10 másodperc elteltével.

 Az SetIntervalX és a move\_cannon metódusokat meghívjuk az oldal betöltésekor egyaránt.



### A weboldal adatbázis kapcsolódása

Mivel a legtöbb PHP művelet a weboldalon valamilyen adatbázis műveletet végez el, fontos először a kapcsolatot definiálni. Az adatbázis elérést a PDO keretrendszer segítségével oldottuk meg. A kapcsolódást a weboldalon a Database\_conn osztályon belül a connect() nevű metódus végzi el. A függvény visszatér az adatbázis kapcsolódással, egyébként pedig, ha valamilyen hiba lép fel, akkor elirányítja a felhasználót egy hiba a szerverrel feliratú felületre.



### Nem található az oldal felülete

Ha a felhasználó a weboldalon belül olyan oldalra megy a címsorban, amely nem létezik a weboldalunkban, akkor a szerver egy kifejezeten erre a hibára fenntartott felületre dobja át a felhasználót jelezve neki, hogy rosszul adta meg az oldalt, amit meg akar tekitneni.

Az hibás keresés utáni átvezénylést a weboldal főmappájában található „.htaccess” állomány végzi el, amiben megadjuk, hogy melyik oldalra dobja át a weboldalt használót az érvénytelen URL keresése után. A fájl a weboldal szerverének fogja elküldeni a megfelelő irányítási utasítást.



A felhasználó a megjelenített odalon a vissza gombra kattintva visszaléphet egy helyzetet a böngészőben.



Az oldal nem található felület a weboldalunkon

### Stílus elemek az oldalakhoz

#### Reszponzivitás

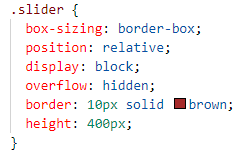
Ahhoz, hogy reszponzív legyen az oldal az elemek szélességének az oldal szélességéhez kell igazodnia. Továbbá azok az elemek, amelyek egymás mellett vannak, egy bizonyos oldal szélesség alatt az elrendezést oszloposra kell módosítani. Ezeket úgy oldhatjuk, meg hogy media queriket, százalék és vw (width of the viewport) mértékegységet használunk a css-ben. A betűméretet media query seggítségével különböző szélességek alatt alacsonyabb értéket állítunk be. A div-eknek százalékot vagy vw-t adunk így az alacsonyabb szélességen is az ablak méretnek megfelelő széles lesz a div. A nagyobb méretű képeknél a magasságot automatikusra állítjuk, a szélességre százalékot használunk. De maximális szélességet be kell állítani, mert különben a kép nagy szélességű ablakoknál túl nagyméretű lesz. Az oldalon nagy szerepet játszik a Bootstrap. A reszponzivitást a container, row, és col osztályok segítségével valósítottuk meg. A container osztály segítségével hoztunk létre egy konténert, amely tartalmazza a weboldal elemeit, míg a row osztály az adott konténeren belüli sorokat definiálja. A col osztályok pedig az egyes oszlopok méretét határozzák meg az adott soron belül, és lehetővé teszik az oszlopok egymás mellett történő elrendezését.

#### Főoldal

A betűtípus itt és a többi oldalon is a Medieval Sharp[[3]](#footnote-3). A színek kiválasztásához rgb, rga és hex modelleket használtunk. A szövegek igazítása sorkizárt.

A fáklya animációt 2 CSS animációval oldottuk meg. Ezt úgy, hogy a kép elé tettünk egy divet, aminek egy keyframes animációt adtunk és before, after pseudo osztályokat használtunk, amiknek ugyanúgy adtunk egy animációt. Az animáció CSS részekből áll: a "flare" és a "burn" mozgásokból. A "flare" mozgás által keltett gömbök által keltett intenzív fények ábrázolják, amelyek átjárják a fáklyát, és fokozatosan elhalványulnak, majd eltűnnek. A "burn" mozgás által ábrázolt folyamat a fáklyaláng színeinek változását és a fáklya körül látható árnyékok dinamikus mozgását mutatja be, hogy egy valóságosabb hatást érjen el.

A kép slidert bootstrap carousel osztályokkal oldottuk meg. Kapott egy slider osztályt az alábbi tulajdonságokkal.



#### Header és Footer

A header 2 nav részből áll. Mind a 2 rész egy listából áll. A listának adtunk egy list-style-type: none; és display: flex értéket. Az első nav kapott egy position: fixed –t hogy az oldal tetején maradjon görgetésnél. A hover animációkhoz transform: scale és transition elemeket használtunk. A z-index arra szolgál, hogy a többi elemhez képest előrébb legyen az adott elem, itt példaául az első nav és a dropdown. A dropdown úgy működik, hogy alapjáraton display: none értéket kap és a hozzá tartozó elemre hoverelve változik inline-blockra. A reszponzivitást úgy oldottuk meg itt, hogy az első navban néhány látvány elem eltűnik bizonyos width alatt. A második nav 767 px szélesség alatt eltűnik. És megjelenik egy hamburger icon az első navban, amire kattintva javascript segítésével megjelenik a 2. nav, de flex-direction: column értékkel.

A footerben és a kommentek oldalon is Font Awesome[[4]](#footnote-4) ikon könyvtárat használunk.

#### Regisztráció és Bejelentkezés

A formra Bootstrap classokat használtunk, így ezzel a stíluson kívül reszponzivitást is megoldottuk. Szintén a transform: scale és transition animációt használtuk az inputon.

#### Elérhetőség

A már főoldalon használt láng animációt felhasználtuk a küldés gombon is, csak más paramétereket használva. A hely mutatása térképen gomb animációjára transitiont használtunk cubic-bezier segítségével. Az animáció lényege, hogy ha ráviszed az egeret, akkor balról jobbra feketévé változik a gomb háttere. Ha a hoverelést abba hagyod, akkor pedig jobbról balra az eredeti színné változik a gomb, ezt a :not(:hover) segítségével. itt is felhasználtuk a már többször használt scale animációt.

#### Vártörténet

Ez az oldal az egyes századokhoz tartozó hidden tartalom megjelenítéséhez egy gombot használ javascript segítségével. Ezt simán a display property változtatásával oldottuk meg. Amint a gomb megnyomódik, egy animáció hajtódik végre. Ez az animáció úgy működik, hogy a width-et fokozatosan 0%-ról 100%-ra növeli. Az overflow: hidden property elengedhetetlen.

#### Programjaink

Szintén bootstrap osztályokat és transform: scale, transition animációt használtunk. A felsorolt programoknak border-bottom-left, és border-top-right értékeket adtunk.

#### Kiválasztott program:

A programhoz tartozó kép szegélyéhez külön képet használtunk.

#### Megjegyzések

A komment szekciót 2 diszítő oszlop övezi. Bootstrap row, és col osztállyal helyeztük 3 oszlopba. az oszlop egy kép, aminek adtunk bordert. A megjegyzés küldés gombnál itt is használtuk a flare és burn animációt. A program kiválasztás gombja background-image-t használva linear-gradient jellemzőt használva a színét, majd background-size-al a méretét állítjuk be.

#### Galéria

A képek alatti felirat megjelenítését javascript segítségével hívjuk elő.

#### Kijelentkezés

A dialóg hátterének adtunk egy backdrop-filter: blur() effectet így mosódott el a háttér. A gombok border-radius értékét beállítottuk 25px-re.

## A Várkezelő asztali alkalmazás főbb funkcióinak fejlesztői dokumentációja

### Alapvető információk a programról

**Fejlesztői környezet:** Visual Studio 2019, 2022

**Fejlesztői program:** Windows Froms Alkalmazás

**.NET Framework verziója:** 4.7.2

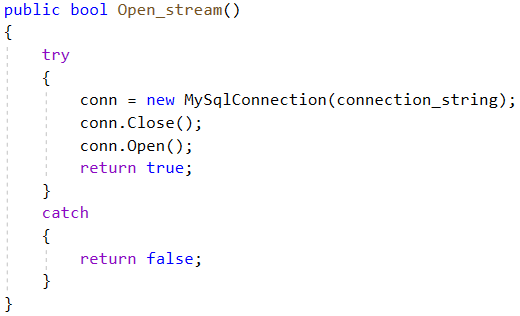
**MySQLConnector veziója:** 2.1.0.0

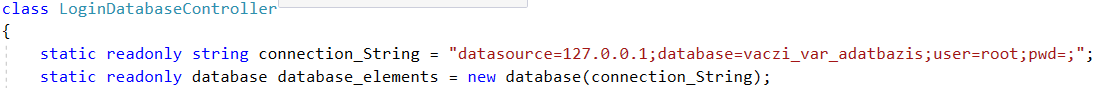
### Az adatbázis kapcsolódás megvalósítása

A program funkcióihoz elengedhetetlen adatbázis használata, ezért a Mysql.Data beimplementált dll fájlt referenciaként felhasználva csatlakozást hatjunk végre az adatbázis szerevere felé. Az adatbáis műveletek a database.cs állományban találhatók, azon belül a database nevű osztályban. A System.Data és a MySql.Data.MySqlClient névterek meghívása elengedhetelen az adatbázis adatok kezelése miatt.

A database osztályban lértehozunk egy példáynváltozót, amelyet beleépítünk egy konstruktorba. A connection\_string példányváltozóra hivatkozunk majd egy másik osztályban a végrehajtott adatbázis kapcsolódás metódus pédányosítása során.



Az osztályban az Open\_Stream() metódussal végrehajtjuk a kapcsolódást, a korábban létrehozott connection\_string példányváltozó felhasználásával. Ezt a metódust minden adatbázis művelet során fel tudjuk használni a kapcsolat megnyitása érdekében.

Egy másik osztályban, ami felhasználja az adatbázist a database meghívásával paraméteren keresztül megadjuk a kapcsolódás sémáját és ezzel létrejött a kapcsolat.

### Bejelentkezés a várkezelőbe

#### Alapvető célja a felületnek

Ha az adminisztrátor használni akarja az adatbázis adatokat manipuláló funkciókat a programban, akkor először be kell jelentkeznie. Szóval ez a felület garantálja a biztonságos és hitelesített belépést a programba.

Ha 3-szor rosszul írja be egy ember a jelszót, akkor kidobja a programból a bejelentkezőt és naplózva lesz, hogy nem tudott belépni. Ha egyszer rosszul írja be, akkor csak egy időzítő indul el, ami, ha letelik, ugyan-úgy kilép a program. Vagyis ezekkel a védelmi funkciókkal a brute-force támadásokat kívánjuk kivédeni az alkalmazásba való belépés során.

#### A bejelentkezés menete

A bejelentkezés a hitelesítés művelete úgy kezdődik, hogy meghatározzuk a bejelentkezésért felelős vezérlő gomb „Onclick” eseményét, ami metódus formájában elindul, ha a felhasználó rákattint a vezérlőre, vagyis a „Login\_button” nevű gombra. A bejelentkezés funkcióhoz szükséges a database osztály, amellyel a validációhoz szükséges adatosorokat nyerjük ki. A LoginDatabaseController osztály feldolgozza azon adatbázis rekrodokat és a felületért felelős Login.cs állománynak elküldi, hogy eldöntse, mit kezd a begépelt adatok alapján az alkalmazással, annak függvényében, hogy sikeres, vagy sikertelen volt a bejelentkezés. A funkció több lépésből áll:

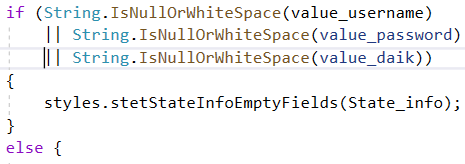
##### *Az adatbázis kapcsolat ellenőrése*

A database osztályban található Open\_Stream() metódus elküldi a kapcsolódást eredményét egy bool értékként. A következő lépésben a LoginDatabaseController CheckingDbConnection() függvénye továbbítja ezt az értéket a bejelentkezés Form osztályába, ahol a LoginDatabaseController példányosítása után megvizsgáljuk, hogy van-e adatbázis kapcsolat, ha van, akkor tovább halad a további feltételen, ha nincs, akkor nem hajtja végre a Login\_button\_Click esemény további műveletét, hanem megjeleníti a hibaüzenetet:

**

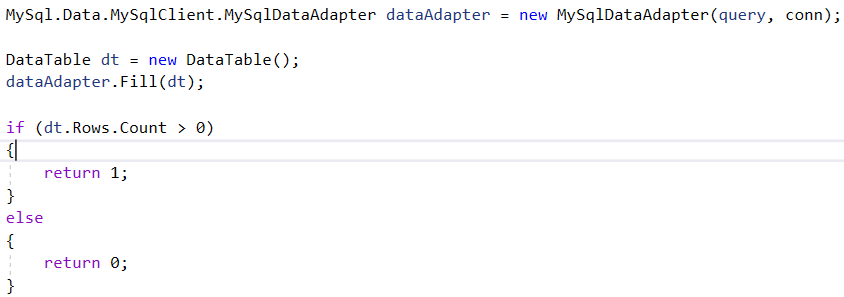
##### *Üres mezők:*

Ha megvalósult a kapcsolat, akkor a feltétel tovább ugrik és kiszűri a program, ha az egyik mező üresen van hagyva. Az IsNullOrWhiteSpace kifejezéssel az üres szóköz helyeket is üresnek tudjuk venni.

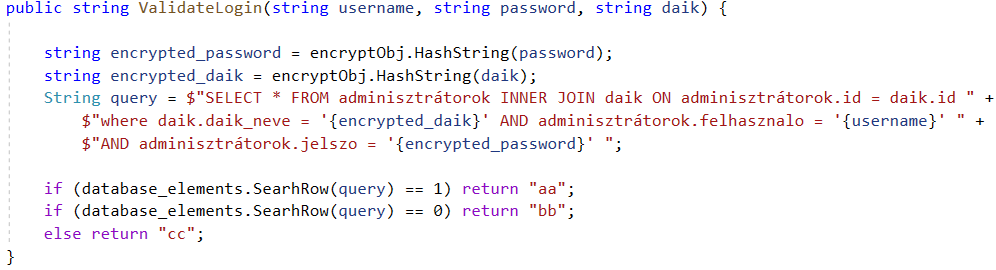
**

##### *Az bejelentkezés hitelesítése*

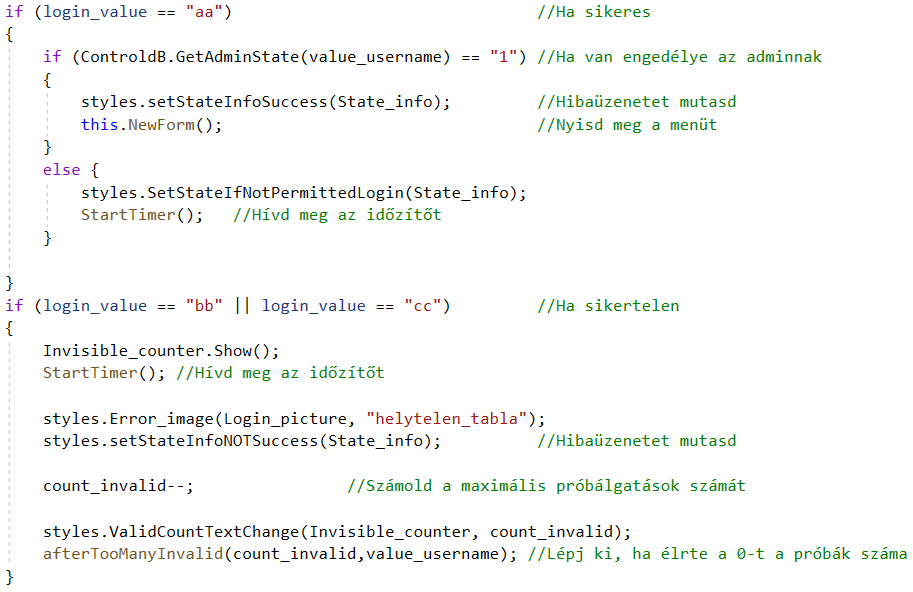
Ha van kapcsolat és nem üres egyik mező sem, akkor az alapján szűrűnk, hogy létezik-e az admin amit megad a felületet használó. Először van egy database osztályon belüli SearchRow(string lekérdezés) megnevezésű függvény, amely paraméteren keresztül kapja meg a lekrédezést ,ami alpján megjeleníti a szükséges rekordot. A metóduban létrehozunk egy MySqlDataAdaptert, amely tartalmazza a SQL utasítások halmazát, amelybe beleágyazzuk a kapcsolatot és a lekrédezést. Továbbá létrehozunk egy az adatbázisból kinyert adatokat tároló szerkezetet egy DataTable-t, amit összepárosítunk az adapterrel. Ezek után megvizsgáljuk, hogy nullánál több sort-adott –e vissza, jelen esetben azt, hogy az bemeneti adatok egyeznek-e egy admin adataival.

**

A ValidateLogin metódus a paraméteren keresztül kapott bemeneti értékek alapján legenerálja az adminisztrátort kilistázó SQL utasítást. Ezután a lekérdezést felhasználva meghívja a SearchRow() kereső metódust és annak függvényében továbbítja az „aa”, „bb”,”cc” eredményeket, a tényleges bejelentekzés szűréséhez, vagyis a Login.cs Form osztályba.



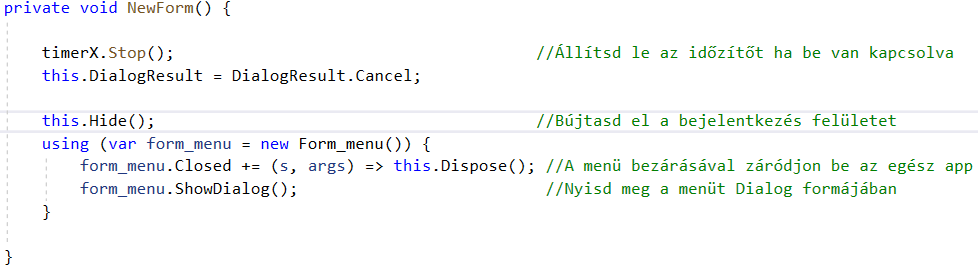
Ha nem felet meg az input a bejelentkezésnek, akkor kiírja a hibaüzenetet és elindul a visszaszámláló. Ha megfelelt, akkor még egy validálás van hátra, ami pontosan a lekérdezett admin állapota alapján szűr, úgyszintén az adatbázisból lekérve az értékét.



Ha minden feltételen átesnek a bemeneti értékek, akkor elbújtatja a bejelentkezés felületet az alkalmazás és megnyitja a menü Formját a NewForm() metódus segítségével. További esetekben, ha a programban többször helytelenül próbálnak meg bejelentkezni és ennek következtében a count\_valid számláló 0 értékű lesz, akkor leáll a program az Application.Exithtread() funkció segítségével, ami az afterTooManyInvalid() metódusban van definiálva.

#### A menü megnyitása a NewForm() metódussal

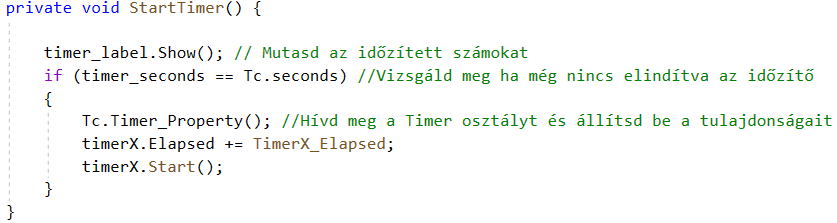
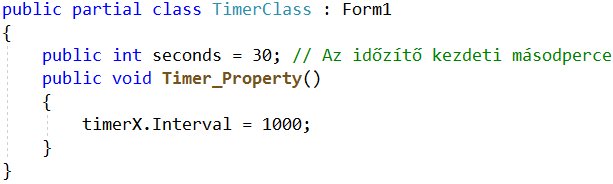
A sikeres bejelentkezés során a menü modális párbeszédpanel, vagyis dialógus formájában nyitódik meg a menú ablaka.

**

#### Időzítő a bejelentkezésnél

A bejelentkezés felület bal felső sarkában megjelenik egy szám, amely minden másodpercben egyre kisebb értékű lesz. A szám 30 –as értékre van alapértelmezetten állítva. Vagyis, ha a felhasználó rosszul üti be a valamelyik mezőbe az értéket, akkor fél perce van jól begépelni, mielőtt kilépteti a program. A működéséhez meghívjuk először a System.Timers névteret, hogy használni tudjuk a szükséges kifejezéseket.

A timer funkció indulását a StartTimer() metódus biztosítja. A metódus meghívja az időintervallum beállításáért felelős Timer\_Property() metódust és elindítja a timerX függvény folytonosságát, ezután pedig magát az időzítést is.



A Timer időhosszának beállítása

A timerX függvényben pedig definiáljuk az időértékek csökkenését és hozzárendelését egy címkéhez. Továbbá itt meghatározzuk azt, hogyha lejár az idő, lépjen ki az alkalmazás. A funkcióhoz elengedhetetlen volt a System.Timers névtér meghívása.

### A várkezelő alkalmazás menüje

#### A menü alapvető ismertetése

A bejelentkezés után a programban a menü jelenik meg, ahol az adminisztrátor tud navigálni a funkciók között. A menü reszponzív módon jelenik meg a menüpontok mérete és elhelyezkedése az alkalmazás nagyságához viszonyítva. A megjelenítés megváltoztatása a Form\_resize esemény hatására valósul meg.



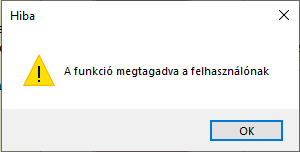


#### A bejelentkezett felhasználó nevének ismertetése

A belépésnél létrehozunk egy publikus változót, amelyet képesek vagyunk átadni egy másik Formnak. Szóval a Login.cs fájlban létrehozunk egy value\_username nevű szöveg típusú változót, amihez hozzáadjuk a név input értékét. Ezt a változót felhasználjuk a menünél is. Amit a bal felső sarokban lehet látni.

Ha valamilyen hiba folytán az alkalmazásba belépő személynek nincs meghatározva a neve, vagyis a beléptetett név nincs az adminisztrátorok listáján akkor hibát dob a program és jeleníti meg a kívánt Formot, amint valamelyik menügombot megnyitja. Azt, hogy benne van-e az adminok között azt a DbControllerMenu osztály és a database osztályon belüli SearchRow metódussal szűrjük. Ezzel a fukncióval a program esetleges meghibásodásából eredő visszaéléseket kívánjuk meggátolni.





#### Az adminisztárori lista feltöltése a menü betöltésénél

Ha betöltődik a főmenü, akkor végrehajtódik egy olyan adatbázis művelet, amely kilisázza az adminisztrátorokat és eltárolja egy listában, amely listát felhasználunk az adminisztrátorok kezelésesénél, mint globális adatszerkezet.

A database adatbázis műveleteket végző osztályban először létrehozunk egy GetColoumnsByQuery nevű metódust, ami paraméteren keresztül megkapja a lekrédezést. Ebben a metódusban létrehozunk egy adattáblát. Ebbe a táblába eltároljuk a lekérdezett adatokat és azután pedig továbbítjuk a táblát, mint visszatérési érték.



Ezután az AdministatorDatas osztályban felveszünk példányváltozókat, amelyek reprezentálják az adminisztrátorok tábla mezőit. Továbbá, definiálunk egy konstruktort, amellyel később beállíthatjuk e példányok kezdőértékét. Szóval van egy dbControllerMenu osztályban található FillListWithAdmins nevezetű lista típusú függvény, amiben deklarálunk egy li\_admins\_obj listát, amelynek megadja az AdministatorDatas objektum tulajdonságát, hogy tudjunk hivatkozni a listán belül a példányváltozókra.

A FillListWithAdmins metódusban meghívjuk az adattáblát, aminek a paraméterébe az SQL lekérdezést ágyazzuk be. Ezt az adattáblát bejárjuk egy for ciklussal. Eztuán a bejárásban feltöltjük a lista sorait a tábla rekordjaival úgy, hogy képviseltetjük az AdministatorDatas példányait az adattábla mezőivel. Végül visszatérünk az összetett adatszerkezettel, aminek az értékét hozzáadjuk a ténylegesen létrehozott globális listához, így meghívhatóak lesznek az elemei bárhol a programban.



A globális lista adminisztrátor nevekkel való feltöltéséért felelős metódus



Adminok listájának inicializálása



Adminok listájának deklarálása

### Program hozzáadás, módosítás

#### A program hozzáadása, módosítása ablak szerepe

Ezzel a felülettel az adminisztrátor egyszerűen és gyorsan adhatja hozzá az új várban szerveződő programokhoz az adatokat, amelyeket a Váczi várweb oldala megjeleníti a felhasználók számára. Nem csak hozzáadást képes az adatbázisba, hanem módosítani is tudja a program adatait, mind a program kezdetét, a befejezését, leírását, állapotát, tulajdonságát változtatni tudja, akár csak az egyik, több, vagy az összes értéket.

Az ablak segítségével a programhoz tartozó képeket is átmásolhatjuk egyszerűen a szerver mappájába, vagy könnyedén előkészíthetjük a képeket egy esetleges feltöltésre.

#### A felület betöltése

Az ablakban való navigálásra lett létrehozva egy TabControl típusú megjelenített menüvezérlés. Lényegében a vezérléssel ki lehet lépni a programból, vagyis 2 Tabpage lett megalkotva hozzá: a maradás az oldalon és kilépés a programból.

Továbbiakban programkezelő felületen ComboBox vezérlőket használunk, amelyek listaszerűen tárolnak neveket és ki lehet választani az elemei közül. A kiválasztandó programot módosításnál használjuk. A kezdeti, befejezési dátumot, a program típusát és állapotát is ComboBoxok segítségével választhatjuk ki, mind feltöltésnél és szerkesztésnél egyaránt.

A program hozzáadásánál szükséges 3 képet feltölteni a programhoz, ugyanis a weboldal megjeleníti a programokhoz a tartozó képeket is. Ezt dialógusok segítségével tesszük meg a felületen. Továbbá a program leképez 3 üres PictureBoxot is, amely a képek kiválasztása után megmutatják, hogy milyen képet választottunk.

Természetesen a program betölti a szükséges bemeneti TextBox mezőket is a program elnevezésének és leírásának meghatározása céljából.

#### A program hozzáadása funkció

Ha az adminisztrátor a hozzáadás CheckBoxot választja ki, és ha megnyomja a „Mentés” gombot akkor az Add\_btn\_Click() esemény indul el. Az esemény bekövetkezte után egy megerősítés üzenetdoboz megkérdezi a felhasználót, hogy biztosan végre akarja e hajtani a műveletet. Ha igen, elkezdődnek a szűrések az input alapján. A metódusban legelőször azt ellenőrzi az alkalmazás, hogy minden mező ki van-e töltve, utána, hogy a programot használó helyesen adta meg a dátumot, vagy sem. A legutolsó szűrés, ami pedig csak a hozzáadásnál van, hogy minden képet kiválasztottunk és azt is, hogy hova szeretnénk a képeket átmásolni.

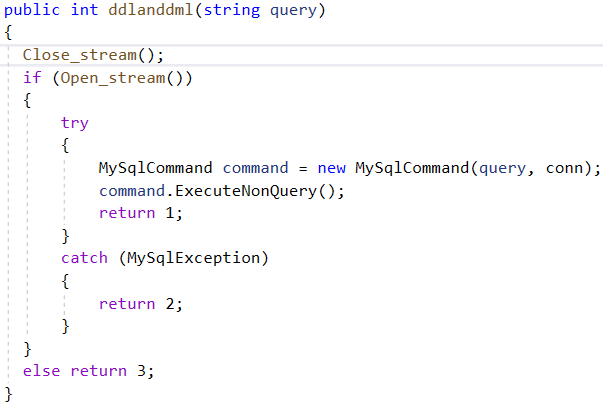
#### Hozzáadás és Módosítás adatbázis műveletei

A hozzáadás és módosítás funkciók csak abban különböznek egymástól, hogy más lekérdezést használnak. De ugyan azokat a bementi értékeket és metódusokat használják fel az adatbázis operációk megvalósításához.

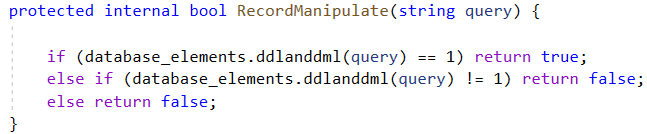
Először a database osztályban a ddlanddml függvényben megvalósul egy egyszerű DML parancs a paraméteren kapott lekérdezés felhasználásával.

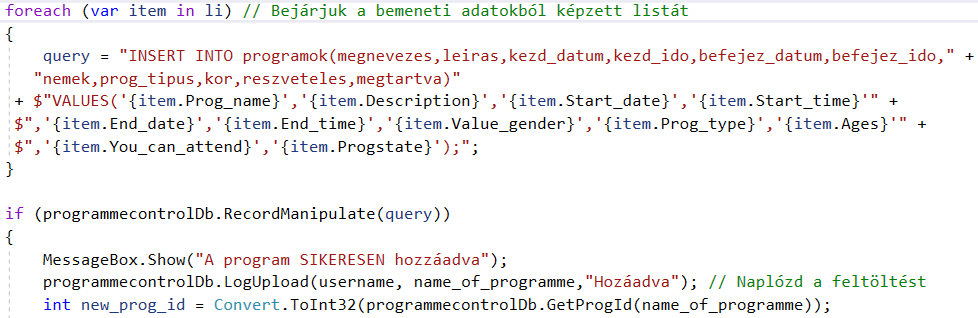
Az ExecuteNonQuery() paranccsal hajtódik végre a feltöltés, amit egy try-catch blokkba helyezünk , így visszatérhetünk az adatbázis művelet eredményével.

Ezek után az eredményt tovább küldi az osztály a ProgrammeHandleDbController osztály RecordManipulate metódusának, ami továbbítja a megkapott eredményeket logikai formában a programkezelő Form osztályába. A Programme\_handle osztályban pedig feldolgozásra kerül az adatbázis művelet a létrehozott lekérdezés és az ebbe ágyazott bementi értékek alapján, ha a szűrésen átmentek ezek az értékek.



A hozzáadás, módosításért felelős DML művelet a database osztályban



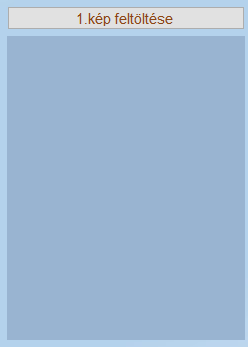
DML eredmény továbbítása bool típusban

Feltöltés meghívása az SQL parancs alapján

#### A képek elérésének feltöltése

A hozzáadásnál a kiválasztott képek fájlnevét és az átmásolás célmappájának nevét is feltöltjük úgy, hogy hozzáfűzzük a hozzátartozó program azonosítóját feltötött rekord azonosítására.

Ha a felhasználó megnyitja a “kép feltötése” gombot akkor elnavigálhat a fájlok között a kiválasztandó kép megnyitásához és kiválasztásához.

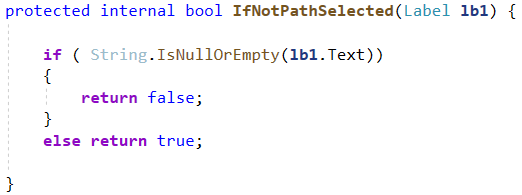


Ha a felhasználó már kijelölte a számára megfelelő képet, akkor a kép felett megjelenik egy elérés, amelyet egy címkének a szövegeként módosítunk.

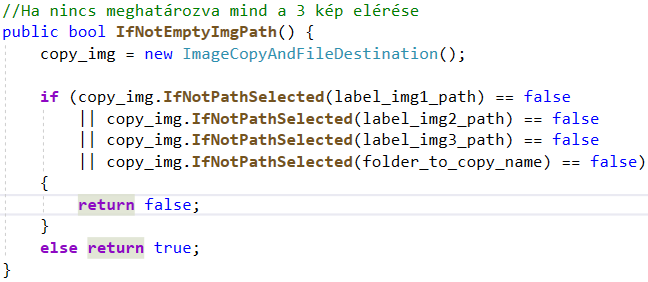
A program adatok feltöltése során, ha mind a három kép ki van jelölve, akkor feltöltésre kerülnek. A képek másolásának és kiválasztásának a folyamatát a ImageCopyAndFileDestination osztály biztosítja. A képek feldolgozása és feltöltése a következőképpen zajlik:

1. ***Lépés:*** *Szűrés ha, nincs három kép kijelölve*

Ha valamelyik kiválasztott kép feletti szöveg, üres akkor nem engedi a programot a további műveletek végrehajtására, amíg nincsen ez a probléma korrigálva. Ezt a validációt úgy érjük el, hogy egy függvényben formális paraméterben egy Címke típusú vezérlőt adunk meg és a meghívásnál ahol képek feletti elérésre hivatkozunk.

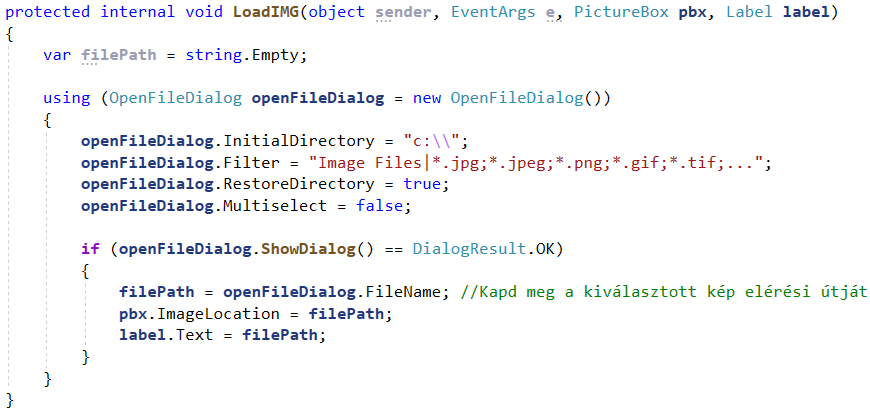
**

A Programme\_handle osztályban pedig meghívásra kerül. A feltöltés eseménynél pedig az IfNotEmptyPath() return eredménye is eldönti, hogy valóban lehetséges-e a feltöltés, vagy még ki kell jelölni további képet, hogy elérje a három darabot.



***2 .Lépés:*** *Megvalósítjuk a képek kijelölését, valamint az elérését megjelenítjük*

A képek kiválasztását a LoadIMG eljárás végzi. A metódus megnyitja dialógus formájában a kép kiválasztása párbeszéd ablakot és megjeleníti az adminisztrátor előtt.



1. Meghatározzuk a kép elérés címkéjét és a képdobozt a paraméteren keresztül.
2. Megjelenítjük a fájl kiválasztása ablakot az OpenFileDialog segítségével.
3. Definiáljuk a párbeszédablak kezdeti meghajtóját és a „Filter” kulcsszóval a megengedett kiterjesztést.
4. Ha egy kép kiválasztásra került, akkor a filePath változóban eltároljuk a kijelölés eredményét, mint szöveg alapú útvonal.
5. Végül a paraméteren keresztül megkapott PictureBox típusú vezérlővel megjelenítjük a kiválasztott képet az ImageLocation használatával. Továbbá, a Labelhez hozzáadjuk az eredményt, mint a kijelölt állomány elérésének nevét.

***3. Lépés:*** *Kijelöljük a másolás célmappáját és megjelenítjük*

Itt a mappát lehet kiválasztani, azt a cél könyvtárat, ahová másolni akarjuk a képeket.

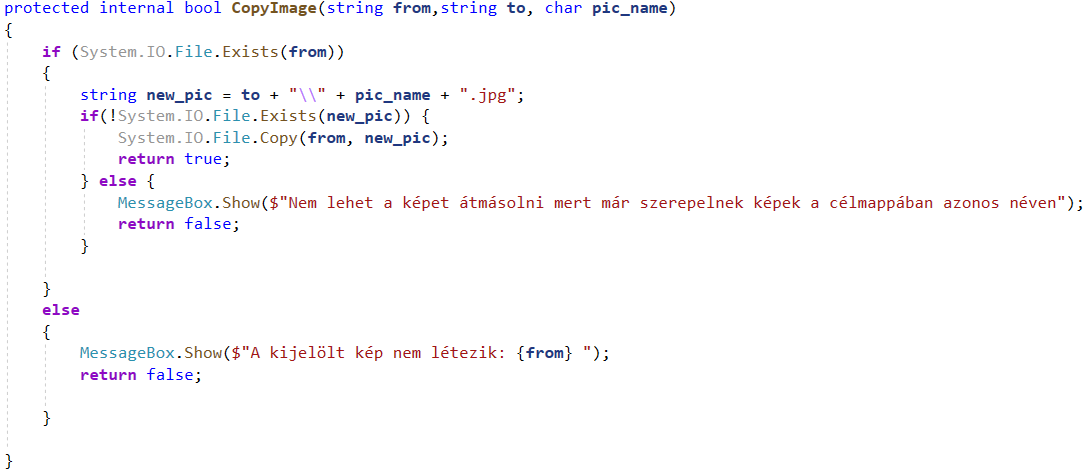


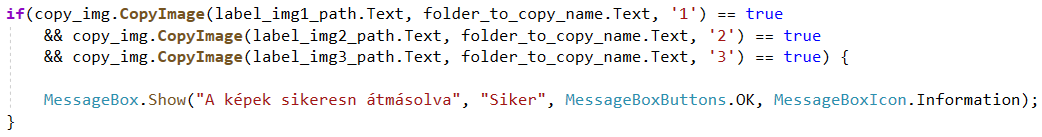
A SELECT\_path() függvényben a FolderBrowserDialog kifejezés segítségével kereshetünk és jelölhetünk ki mappákat, végül a kiválasztás után továbbítjuk a kiválasztott mappa elérésének értékét.

***4.Lépés:*** *A képek másolása az elérési útvonalak szerint*

Mivel a weboldalunk minden jövőbeli programhoz úgy tárolja a képeket, hogy a db\_prog\_images nevű mappába az adott programhoz tartozik egy mappa, amibe elhelyezünk 1.jpg, 2.jpg, 3.jpg nevű képeket, ezért a másolás során átnevezzük a képeket számozott nevűre.

1. Meghatározzuk a paramétert, amelybe beleiktatjuk a forrásfájl teljes helyét, a célmappát és a másolandó kép átmásolt nevét egy char változóként, amely az ’1’,’2’,’3’ típusaként bőven megfelel.
2. Megkeressük, hogy létezik-e a kijelölt kép, amit megadtunk. Ha igen tovább megy, ha nem, akkor hibaüzenetet dob.
3. Definiáljuk az új kép elérését jpg formátumban a célmappa és a szám alapú név kombinálásával a new\_pic nevű változóba
4. Megnézzük a new\_pic fájlnév alapján, hogy nem hoztak már e létre a cél helyén azonos néven képet, ha igen akkor hibaüzenetet mutat és nem hajtja tovább végre az utasításokat a függvényben.
5. Ha nem létezik az új kép, akkor másold át a File.Copy(forrás, új fájl) másoló funkció segítségével a képet.

**

****Végül a Formban meghívásra kerül háromszor a képmásolás, amellyel sikeresen átmásolja a kepéket 1,2, 3.jpg néven.

#### A képek elérésének feltöltése az adatbázisba

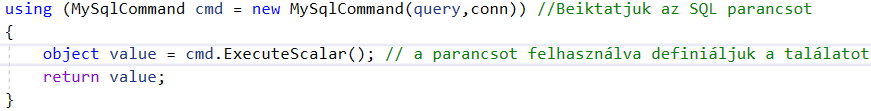
Az adatbázisban a kepek\_program táblában az átmásolt kép útvonalának mappáját és magát a fájl nevét tároljuk a következőképpen:



Ahhoz, hogy a képekhez tartozó program azonosítóját meghatározzuk, fel kell használnunk az újonnan feltöltött program id-ját. Mivel az utasítások sorban hajtódnak végre a programban, így először feltöltjük a programot, amelynek ezután elő tudjuk hívni az adatbázisból azonosítóját és utána a program elsődleges kulcsa alapján feltöltjük a képek elérését.

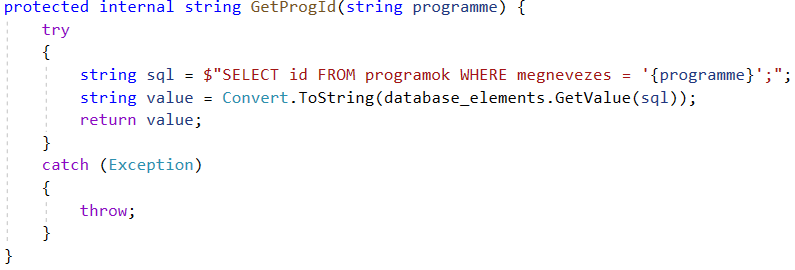
Az új program azonosítóját a ProgrammeHandleDbController osztályban a GetProgId metódus segítségével határozzuk meg, amely felhasználja a database osztály GetValue függvényét, ezáltal egy egyetlen azonosítót kérdez le a programok táblából. Végül a UpLoadImgPathToDBFromForm eljárással feltöltjük a képeket a “kepek\_program” nevű táblába, a lekérdezett program id-t és a kép eléréseket összevetve egy rekordba.

#### GetValue metódus szerepe a programban

** Ahogy az már említve lett a GetValue függvény egy adatot kérdez le az adatbázisból, amelyet egy skaláris szorzat segítségével pontosít és talál meg a parancs alapján, vagyis az ExecuteScalar kifejezés felhasználásával. A megtalált értéket továbbítja object típusként, amit a meghívásnál a talált adat a legmegfelelőbb típusába konvertálja a felhasználásra.

#### A GetProgId metódus szerepe a funkcióban

A metódus legenerálja az SQL lekérdezést, amely lekérdezi annak a programnak az azonosítóját, amely megfelel a paraméteren megkapott string típusú változónak. A program neve a hozzáadásnál, amelyet a formális paraméter képvisel, a megnevezés bementi mező értéke lesz.



Végül az id értékét visszatérítjük és felhasználjuk a hozzáadás eseménynél a példányosítás után, amit a new\_prog\_id változóba helyezünk.

#### A képek elérésének tényleges feltöltése

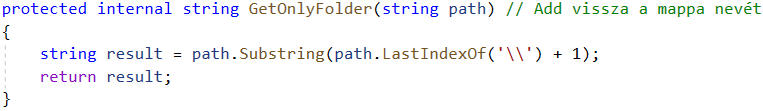
*A funkció az* UpLoadImgPathToDBFromForm metódus segítségével megy végbe. Ez a metódus a Programme\_handle Form osztályban lett definiálva. Működése során meghívja a képek másolására kijelölt forrásmappa elérési útvonalának az utolsó mappanevét a GetOnlyFolder metódussal, amely a ‘\\’ mappaelválasztó jel mentén feltördeli az elérés szövegét és az utolsó szekciót vissza adja a karaktertől jobbra.

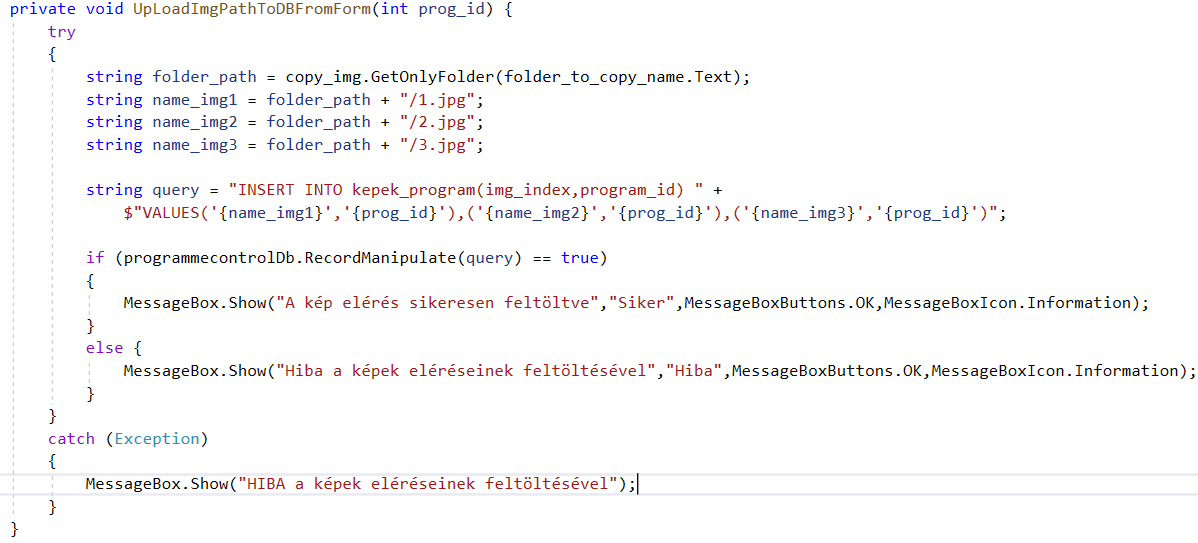
Ezután meghatározza a képek nevét úgy, hogy összefűzi az előbb feldolgozott mappa címét a képfájlok nevével.

Végül a SQL utasításban felhasználjuk a paraméterben kapott program azonósítóját és a megalkotott képeket, amelyet beágyazunk egy DML adatbázis utasításba. Legvégül pedig a RecordManipulate() metódus meghívásával végrehajtódik az útvonalak feltöltése a query lekérdezést felhasználva.



**Adatbázis feltöltés meghívása**



**Az elérés utolsó mappájának kiválasztása**

**Kép elérés feltöltése metódus**

### A geléria képek feltöltése funkció

#### A galéria képek feltötése funkciónak a célja

A felülettel a véget ért prorgamokhoz lehet gyorsan és hatékonyan feltölteni a kiválasztott képek nevét az adatbázis “galeria\_kepek” táblájába. Itt a kijelölt képeket képesek vagyunk a program által egyből átmásolni jpg és png formátumban a weboldal galéria képeket tároló mappájába, amely, könyvtárból a weboldalunk a “Galéria “oldalon megjeleníti, az adatbázisban szereplő kép nevek alapján a fotókat. Természetesen egyszerű képmásoló funkciókat is ellát, ha bárhova máshova szeretne a felhasználó képeket sokszorosítani. Ebben a formban, ellenben a programkezelő ablakkal, elég egy felvételt kiválasztani, hogy az adatait fel lehessen tölteni az adatbázisba és a képet átlehessen másolni.

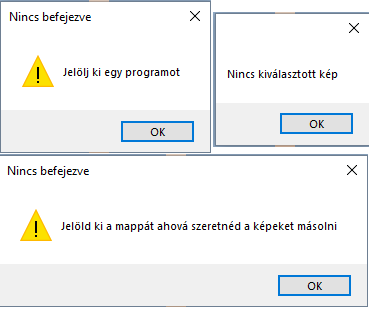
#### Galréia képek feltöltés ablak betöltődése

A felület betöltéskor kitölt egy comboboxot a befejezett programok nevével, amit úgy visz végbe, hogy egy adatbázis lekérdezéssel kiszelektálja azokat a programokat, amelyek már befejeződtek. Azután a lekérdezett program megnevezésekkel feltöltünk egy listát amelyet foreach ciklussal bejárva feltöltjük a comboboxot, ahonnan kiválaszthatjuk, hogy melyik programhoz tartozzon a feltöltött galéria fénykép(ek).

Az oldalon megtalálhatunk 4 képdobozt, amely a programkezlő alkalmazáshoz hasonlóan felhasználja az ImageCopyAndFileDestination osztályt a képek kezelésésre és másolásásra. Hasnonlóan itt is, ha kijelöltük a megnyíló ablakban a megfelelő képet a képdobozok megjelenítik a dialógus eredményét, vagyis a kiválasztott képet magát. A megmutatott kiválasztott kép felett egyezően megtalálható egy címke, amely tartalmazza a kijelült kép elérési útvonalát, persze csak akkor, ha megtörtént a kép szelektálása.

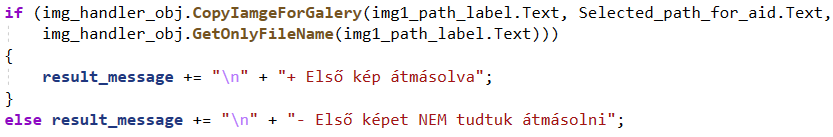
#### Galéria kép adatainak adatábzisba való feltöltése és képek átmásolása

Ha, a felhasználó kiválasztott egy programot a ComboBoxból és legalább egy képet a négy közül, valamint kijelölte a célmappát, akkor elvégezheti a feltöltést. Ha nincs valamelyik paraméter meghatározva, akkor a hiány szerint egy üzenetdoboz jelzi a hibát.



Ha a felületet használó minden szükséges műveletet elvégzett és rányom a feltöltésre, akkor elvégzi a program a képek másolását.

Először a Galery\_images Form osztályban található CopySpecifiedImages metódus fut le, amely felhasználja ImageCopyAndFileDestination osztályban elhelyezett CopyIamgeForGalery metódust, ami végrehajtja képek másolását a File.Copy parancs segítségével. Továbbá meghívásra kerül a CopyImageForGalery() függvény, amelynek paraméterébe beágyazzuk a kijelölt kép elérését és a célmappának a szövegét. Ezután kitölti a result\_message nevű értesítő szöveget a kép másolásának sikere alapján.



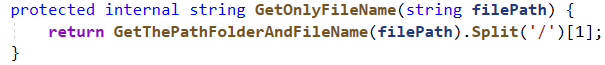
#### A kép fájlnevének meghatározása a másoláshoz

Létezik azImageCopyAndFileDestination osztályban egy metódus GetThePathFolderAndFileName néven, ami visszaadja a paraméterben megkapott teljes útvonalat egy string típusú formális paraméter értékeként. Itt a folder nevű változóban meghatározzuk a teljes elérésből a közvetlenül a képet tartalmazó mappa nevét. A következő lépésként pedig a file nevű változóhoz hozzárendeljük a kép nevét az útvonal elválasztó karakterrel való tördeléssel és ezután a LastIndexOf beépített metódust felhasználva határozunk meg, hogy nekünk csak a fájl neve kell az elérésből.

Ezután összefűzzük a mappát és a képnevet, amiket továbbítunk.

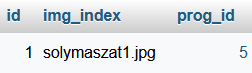
**

*A korábban felvázolt GetOnlyFilename függvény a szintén a* ImageCopyAndFileDestination osztályban lett definiálva. *A* GetThePathFolderAndFileName függvényből kapja meg a cél mappa és az abban található kép nevének együttes értékét. A megkapott „célmappa/kép” szöveget Split hasító funkcióval a’/’ mentén tördeli, és ezen beül, az első indexű elemet veszi ki, ami a jeltől jobbra lévő elem lesz, vagyis a kép neve. Végül a fent leírt módon felhasználjuk a képmásolásnál ezt a függvényt.

**

#### Galéria képek az adatbázisba

Ha megtörtént a képmásolás, akkor végrehajtódik a kijelölt kép nevének és a kiválasztott program azonosítójának feltöltése a „galeria\_kepek” táblába. Így a galéria oldal a weboldalon képes lesz megjeleníteni az új képet és az ahhoz kapcsolódó programot.

****

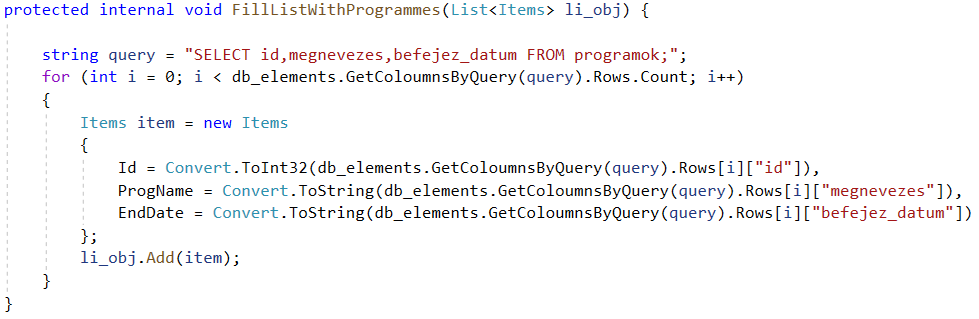
### A programok törlése funkció

#### A programok törlése funkció célja

Az ablak célja, hogy gyorsan és kezelhetően lehessen programokat törölni, ha valamelyik munkatárs esetleg poénból trágár megnevezésű és jellegű programot adott hozzá az adatbázishoz. Továbbá, a már nagyon régi programokat, amelyek már csak terhet jelentenek a rendszer számára is el lehet távolítani a felület segítségével.

#### A programok törlése funkció betöltése

A felület betöltésekor egy combobox jelenik meg, egy gomb, ami a végrehajtáshoz szükséges és egy címke, ami mutatja a kijelölt program befejezési időpontját. A comboboxot kitöltjük az adatbázisból lekérdezett összes program megnevezésével. A vezérlőt úgy töltjük ki nevekkel, hogy a Form betöltésekor egyből feltöltünk egy listát az adatbázisból lekérdezett program címekkel. Tehát már betöltéskor van egy listánk, ami nem csak a neveket tartalmazza, hanem a programokhoz fűződő azonosító számot és a program végének dátumát is, amelyekkel élve már nincs szükség külön adatbázis művelettel lekérdeznünk az adataikat, ha esetlegesen egy kiválasztott programhoz tartozó időpontot vagy id-t akarunk megjeleníteni.



Az előbbi képen látható a kódsorozat, amely bejárja a lekérdezéssel feltöltött adattáblát, amelyet teletöltjük adatbázisból lekérdezett programnevekkel a GetColoumnsByQuery metódus segítségével. Ezután példányosítjuk az Item osztályt, úgy, hogy az adattábla minden indexének értékét hozzárendeljük a megfelelő példányváltozóhoz. A következő lépésben pedig a feltöltött objektumot beágyazzuk egy listába, amelyre hivatkoztatjuk majd paraméteren keresztül.

Felület továbbá megjeleníti a program befejezésének dátumát vízszintes vonallal elválasztva, úgy, hogy mivel Datetime típusként kezeljük, őket le kell vágnunk az órát, percet, másodpercet, a Substring metódus felhasználásával, mert alapértelmezetten benne vannak a változóban és azokat nem akarjuk megjeleníteni.



Szóval a „Selected\_date” nevű Label vezérlő segítségével tudjuk, hogy melyek azok a programok, amelyek még nem értek véget, és amiket nem nagyon lenne ésszerű törölni.

#### A felület alapvető működése

Az felülethez szükséges vezérlők betöltését és a velük végzett eseményeket a DeleteProgrammesHandler osztály végzi el. A programok törléséhez megszokott módon a database osztály nyújt adatbázis műveleteket. Az adatbázis műveleteket a dbControllerDeleteProgram osztály dolgozza fel és teremt kapcsolatot a másik két osztály között. Ha a törlést végző ügyintéző elindítja az eltávolítás eseményt, akkor végrehajtódik a kijelölt program törlése az adatbázisból a combobox kiválasztott elemének szövege alapján egy törlő DML SQL parancs felhasználásával.

Ha megtörtént a törlési operáció, akkor utána pedig megszünteti a combobox elemeiből a törölt programot, hogy ne lehessen hibába futni, ha az eltávolított elemet még egyszer kiválasztjuk.

A program törlése során először az alkalmazás törli a kijelölt programhoz tartozó kommenteket, kép eléréseket és a rekordokat kedvencek közül, amelyek az eltávolított programhoz tartoznak. Ezek a táblák idegenkulccsal csatlakoztatva vannak a programok táblához kapcsolatot alkotva. Tehát az adatbázisunk beállítása miatt nem törölhetjük legelőször a programokat, mert azokhoz hozzá vannak kapcsolva különböző táblák, amelyek felhasználják a programok azonosítóját. Ehelyett a programokhoz kapcsolódó a törlendő program id-ját felhasználó táblák rekordjai távolítódnak el először és csak utána maga a kiválasztott program.

### Adminisztrátorok kezelése funkció

#### A felület célja

Az ügyintézők kezelésének a programban az a jelentősége elsősorban, hogy a fő adminisztrátor korlátozni tudja az a többi adminisztrátort és módosítani tudja az állapotukat gyorsan és rugalmasan, ha az adott admin visszaél a jogával. Továbbá, az ügyintéző képes a saját jelszavát és daik kulcsát módosítani. Esetlegesen törölni is képes a fő adminisztrátor az egyszerű ügyintézőket a programmal.

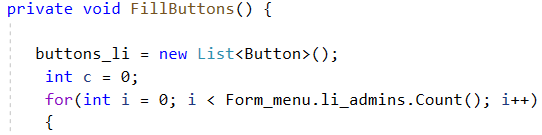
#### Az adminisztrátorok listájának felülete

Ha az ügyintéző belép a menüből az adminisztrátorok kezelése felületre, akkor először egy adminisztrátor nevekkel teli felsorolás fogadja, ahol minden név egy gomb típusú vezérlőben van. A programban létezik fő adminisztrátor és egyszerű adminisztrátor jogok. Az egyszerű admin csak magát tudja kiválasztani a listából, míg a főadmin másokat is. Szóval ezzel a formal kiválaszthatjuk az adminisztrátort, amivel átdob minket egy másik ablakra, ahol a kiválasztott adminisztrátor jelszavát, nevét, email címét és esetleg állapotát tudjuk módosítani.

A felület listát képez abból az adatszerkezetből, amelybe még a menü betöltésekor kitöltöttünk az adatbázisból kinyert adatokkal. A lista az ügyintézők nevét, jogát és állapotát tartalmazza.

##### Az adminisztrátorok kilistázása gombokkal

A nevek kilistázásához a FillButtons metódust használjuk fel. Először deklaráljuk a buttons\_li változót, amely egy lista, amely gombokat tárol. Ezután bejárjuk a karbantartó neveket tartalmazó listát egy for ciklussal.

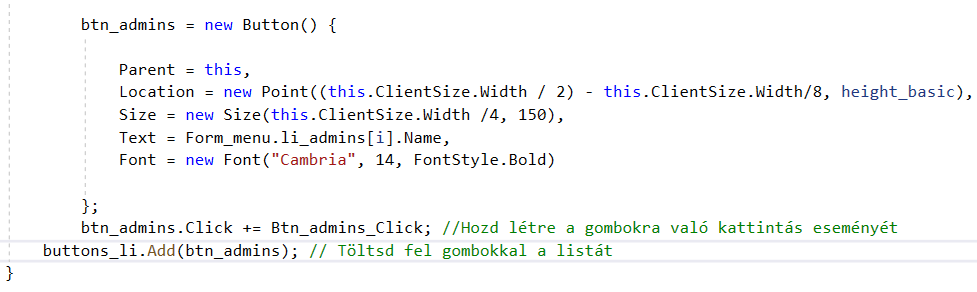


A bejárt globális listát menü betöltése során töltjük fel és frissítjük, ezért hivatkoznunk kell a menü osztályára is „Form\_menu” néven.

Ezután felvesszük a gombok távolságáért és pozíciójáért felelős növekményes értéket:



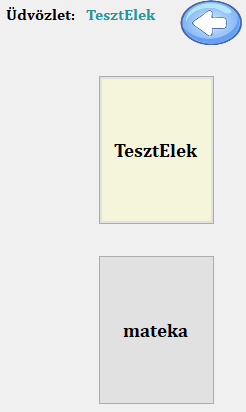
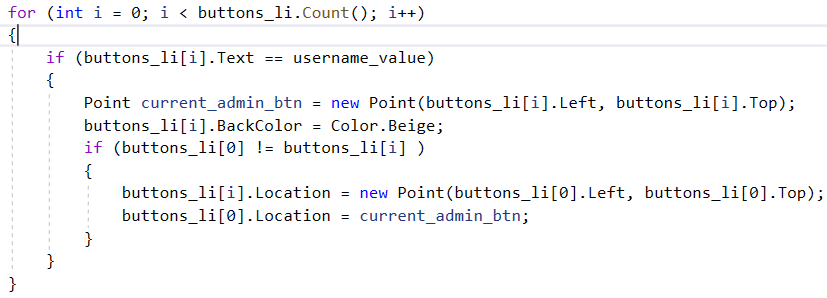
Végül legeneráljuk a vezérlőket a cikluson belül úgy, hogy az adminisztrátorokat tartalmazó lista miden indexéhez hozzárendelünk egy gombot és megadjuk a paramétereiket.



##### A saját fiókod megjelenítése a listában

Az ügyintézők lajstroma úgy lett megoldva, hogy a saját neved a lista tetején jelenik meg bézs színű háttérrel. Ezzel a funkcióval gyakorlatilag nem kell az adminisztrátornak keresgélnie a nevek között.

Ezt úgy oldottuk meg, hogy a gombokkal feltöltött listát bejárva meghatároztuk, hogy melyik a bejelentkezett felhasználóval megegyező szövegű gomb indexe. Ezután gyakorlatilag helyet cserél a név szerint megtalált és a nulladik indexű gomb, ha az éppen keresett nem az első elem az adatszerkezetben.

**

**A TesztElek név legtetejére helyezése**

Ha valamelyik adminisztrátorra rákattintunk, akkor, a selected\_admin globális változóhoz hozzáfűzzük a megnyomott gomb szövegét a selected\_admin = (sender as Button).Text parancs definiálásával. Ez alapján határozza meg az app, hogy melyik ügyintézőt kell kezelni a következő felületen, azaz, hogy melyik gombot nyomtuk meg.

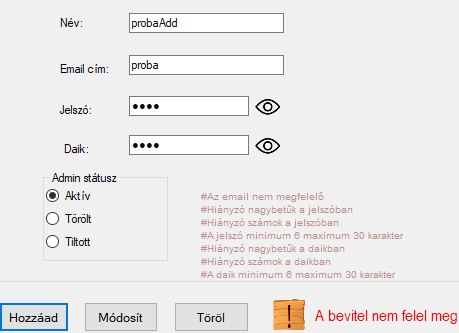
#### Adminisztrátorok kezelése felület

Ha ki lett választva egy adminisztrátor, akkor lehet az adott admint módosítani, olyan mértékben, amilyen a felületet használó joga éppen.

A funkcióban nagy szerepet vállal a database nevű az adatbázis műveletekkel dolgozó osztály, a DbControllerAdmins osztály, ami felhasználja és meghívja az adatbázisművelethez szükséges metódusokat, továbbá az Admins\_handle osztály, ami tartalmazza a vezérlőket és az ezekhez tartozó eseményeket és feldolgozza a másik 2 metódus műveleteit.

##### Input mező szűrése

*Az adminisztrátor hozzáadásánál, program hibaüzenetet ír, ha valamit üresen hagytál, vagy rosszul írtál be valamelyik bemeneti mezőbe. A következő kép azt állapotot mutatja, hogy a fő adminisztrátor mindenhol rosszul adta meg az értékeket:*

**

A ’#’ jellel ellátott hibaüzenetek a bemeneti mezők Text\_Changed eseményére jelennek meg és az InputAdminsFilter osztály validációs metódusait felhasználva. A hibaüzenetek egy listában vannk meghatárzova és mindig hozzáfűzi egy címkéhez a lista azon szövegét az alkalmazás, ami kapcsolódik az éppen hibát talált az input mezőhöz és a hibához. Ha kijavítjuk a helytelenséget, akkor eltávolítja a Labelből a megfelelő hibaüzenetet is.

#### Jogok érvényesítése az adminisztátorok kezelése felületen

A felületen a fő adminisztrátor tud ügyintézőt hozzáadni, és egy létező adminisztrátornak az adatait tudja módosítani vagy törölni. Egy fő adminisztrátor nem képes egy másik fő karbantartó fiókját kezelni, ha adódik olyan helyzet, hogy 2 fő karbantartó szükséges. Egy egyszerű admin csak a saját fiókját tudja kezelni a hozzáadás funkció és a saját név megváltoztatása nélkül. A nevét azért nem változtathatja meg, mivel a rosszindulatú ügyintéző mindig megváltoztatná a nevét, így nehéz lenne követni a tevékenységét. Azt hogy milyen joggal bír egy kiválasztott adminisztrátor és mit tehet a felületen és mit, nem azt a ValidateAdminOnLoad() metódus vizsgálja a kezelő felület betöltésekor a bejelenteztett és a kijelölt adminisztrátor név alapján. Ezt úgy teszi meg, hogy, úgy generálja le a vezérlőket az ügyintéző jogosultsági szintjének függvényében, hogy mivel egy normál adminisztrátor kevesebb művelet hajthat végre, ezért kevesebb vezérlő is képződik le a számára. Szóval nem tudja a számára nem megengedett funckiókat elérni.

#### Adminisztártorok hozzáadása, törlése módosítása

Ha az adminisztrátor rákattint az alcímben felsorolt funkciók valamelyikére, akkor meghívódnak a megfelelő a DbControllerAdmins osztályban létrehozott metódusok, amelyek pedig végrehajtják az adott DML parancsot a database osztályból előhívott eljárás segítségével.

Továbbá, ha hozzáadunk, akkor hozzáadódik, ha megváltoztatjuk, akkor megváltozik, és ha töröljük, akkor eltávolításra kerül a kiválasztott adminisztátor a nevüket tartalmazó globális listából, így a nevek újboli listázásánál már nem jelenik meg, vagy megváltozik az eltávolított név.

### A bementi mezők értékeinek a titkosítása az alkalmazásban

Az adminisztáror hozzáadása során SHA256-os titkosítással kerül feltöltésre mind a jelszó és a daik azonosító. A bejelentkezés során a bemeneti mezőbe begépelt szöveget ugyan-azzal a titkosítási eljárással kódoljuk és a titkosított sémát az adatbázisban található sha értékkel összehasonlítva megállapítjuk, hogy az adminisztrátor létező jelszavat adott e meg.

A titkosítás végrehajtása az Encryption osztályban valósul meg. A funkcióhoz szükséges a System.Security.Cryptography névtér meghívása.

A Makehash függvény a paraméterből megkapott értékből generál az SHA256 szabályai szerint a HashAlgorithm beépített osztály algoritmusa segítségével egy kulcsot, amit byte típusként továbbít, UTF-8 karakterkódolással. Ezt a kulcsot a HashString függvény megfelelő formátumú szövegre építi össze és végül visszatér a titkosítási eredménnyel, mint szöveg.

****

**A titkosítás kódsorozata**



**A titkosított érték az adatbázisban**

### A várkezelő program osztályainak ismertetése

#### Minden felület által felhasznált osztályok

**Database:** Definiálja az adatbázishoz tartozó műveleteket, amelyek metódusainak paramétereibe mindig az SQL utasítás kerül. Valamelyik függvényének vagy eljárásának meghívásánál azt a létrehozott adatbázis parancsot adjuk meg paraméterként, amellyel el akarunk valamilyen értéket érni, vagy éppen törölni, hozzáadni, vagy módosítani akarunk.

**StylesCommon:** Az alkalmazás általános stílusát határozza meg. Minden formban meghívásra kerül, mivel beállítja az ablak ikonját és a felhasználónevet mutató címkének stílusát. Az osztály biztosítja a jelszavak csillagozásáért felelős gombok ikonját is, amely több ablakban használatos.

Ezek az osztályok mivel minden feldolgozó osztályban meghívódnak, így az alábbiakban nem kerülnek felsorolásra.

#### A bejelentkezés funkcióhoz tartozó osztályok

1. **Form1:** A bejelentkezéshez tartozó funkciókat és vezérlők tulajdonságait tartalmazz, valamint garantálja és és feldolgozza az ide tartozó összes osztályt.
2. **LoginDatabaseController:** összeköttetést biztosít az adatbázis parancsok és a Form1 osztály között. És meghívja az adatbázis műveleteket.
3. **TimerClass:** Beállítja az időzítő intervallumát.
4. **StylesLogin:** Elvégzi a stílus hozzáadását a felület számára és ez által levesz néhány terhet a Form1 osztályról.
5. **Encryption:** A bejelentkezésnél lévő bemeneti mezőket is titkosítjuk az osztállyal.

#### A menühöz tartozó osztályok

1. **Form\_menu:** Legenerálja a menüben található vezérlőket és feldolgozza az ide tartozó osztályokat.
2. **DbControllerMenu:** Feltölti az adminisztrátorok listáját az adatbázisból kapott nevekkel és hitelesítési eljárást biztosít a Form\_menu osztálynak a bejelnetkeztetett név alapján.
3. **Styles\_menu:** A menü stílusát hivatott beállítani.
4. **AdministratorDatas**: Objektum példányt biztosít az adminisztrátorok listájának feltöltéséhez.

#### A porgramok kezelése funkcióhoz tartozó osztályok

1. **Programme\_handle**: Tartalmazza a programokat hozzáadó és módosító eseményeket és feldolgozza az ezen funkciókhoz tartozó osztályokat és biztosítja a felület működését.
2. **Controls\_programmes:** Feltölti a comboboxok listáit az adatbázisból lekérdezett nevekkel és tartalmazza a dátum szerinti szűrés eljárását, ha a felhasználó rosszul adta meg a dátumot. Továbbá a megerősítés metódus is itt van létre hozva, amit több ablak is felhasznál.
3. **ImageCopyAndFileDestination:** A képek másolására és kiválasztásuk során megkapott elérési eredményeket kezeli
4. **InputDataLoaderForUpdate:** A módosítás funkcióhoz garantálja adatbázis lekérdezés segítségével a módosításra kiválasztott program mezőit, az automatikus bementek kitöltése érdekében a program nevének kijelölése után.
5. **Styles\_programme\_handle:** A programok kezelése form számára állítja be a stílust és tartalmazza a hibaüzenetek metódusait.
6. **ProgrammeHandleDbController:** Hidat biztosít az adatbázis műveletekkel dolgozó database és a Programme\_handle osztály között és felkészíti a comboboxok adatbázis értékekkel való feltöltését.

#### Galréia képek funckióhoz kapcsolódó osztályok

1. **Galery\_images:** A funkciót garantáló form feldolgozó osztálya. Itt képződnek le a szükséges vezérlők, képdobozok és itt hajtódnak végre azok az események, amelyek kiváltják a képmásolást, kijelölést és feltöltést a meghívott metódusok segítségével.
2. **DbControllerGalery:** Tartalmazza azokat az eljárásokat és függvényeket, amelyek feltöltik a kiválasztott képek adatait az adatbázisba és megjelenítik a hibaüzenet, ha már létezik a kép a tábla rekordjai között. Az osztály felelős a felületen található combobox feltöltéséért a véget ért programok neveivel.
3. **ImageCopyAndFileDestination:** Ez a form is felhasználja az osztályt a kijelölt képek másolására és elérési útvonaluk megjelenítésére.

#### A programok törlése funkcióhoz tartozó osztályok

1. **DeleteProgrammesHandler:** Az események beindításához szükséges vezérlőket tartalmazza és feldolgozza a törlés műveletéhez szükséges osztályok metódusait.
2. **DbControllerDeleteProgram:** A törlés DML parancsot megvalósító osztály, ami eltávolítja a comboboxból elemi közül kijelölt programot és ahhoz a programhoz tartozó kiegészítőket az adatbázisból. Az osztály továbbá előkészíti a programok globális listájának feltöltését egy adatbázis művelet segítésével.
3. **Controls\_programmes:** Az osztályban található megerősítés funkció végett példányosítjuk.

#### Az adminisztrátorok kezeléséhez szükséges osztályok

1. **Administrators:** Biztosítja a listázáshoz szükséges vezérlőket és legenerálja a gombok listáját az adminisztrátorokkal teli globális litsa elemeivel, valamint biztosítja az adminisztátorok kezelésére fenntartott felület megnyitását a kiválasztott ügyintéző nevének továbbadásával.
2. **Admins\_handle:** A karbantartók törlését, módosítását, hozzáadását biztosító ablakhoz tartalmazza a vezérlőkkel végzett eseményeket. Feldolgozza az InputAdminsFilter, DbController\_Admins és a DbControllerMenu osztályokból származó metódusokat és elvégzi az adminisztrátorok tábla rekordjainak manipulálását a segítségükkel.
3. **InputAdminsFilter:** Az admin fiókjával végzett művelet előtti szűrésnél használjuk fel az osztályt. Tartalmazza az egyes bemeneti mezők értékének helyességét. Függvényei igaz, vagy hamis értéket adhatnak vissza a paraméterekben megadott input értékek alapján. Az eredményeivel az Admins\_handle osztályban validálunk.
4. **DbController\_Admins:** Itt találhatóak az SQL utasítás által végrehajtott hozzáadás, törlés, módosítás DML műveletek meghívása, feldolgozása. A metódusai az adminisztrátorokat tartalmazó globális lista elemeit és a felület bemeneti értékeit használják fel az adatbázis utasításokhoz.
5. **Encryption:** A bemeneti értéket titkosítjuk az osztály használatával.

## Továbbfejlesztési lehetőségek

### Fordítás

Tovább fejlesztési lehetőség az egész weboldal lefordítása különböző nyelvekre. Ez egy kiváló ötlet arra, hogy a weboldal minél szélesebb körű elérhetőségét biztosítsuk. Az egyértelmű fordítások lehetővé tennék, hogy az oldal látogatói különböző nyelveken használhassák az oldalt, ami hozzájárulna a weboldal globálisabb jellegéhez. Ezt egy legördülő menüvel lehetne megvalósítani JavaScript segítségével, ami a fejlécben lenne elhelyezve. Az integrált JavaScript fordítóalkalmazás lehetőséget biztosítana az oldal tartalmának dinamikus lefordítására, és a felhasználóknak nem kellene eltávolodniuk az oldaltól, hogy fordítást végezzenek. Ez nagyban hozzájárulna az oldal felhasználóbarát jellegéhez. Azonban az automatikus fordítóprogramok korlátai miatt a fordítás nem lesz mindig tökéletes, és néhány kifejezés vagy szó szerinti fordítása helytelen lehet. Ezért az oldal lefordítását nekünk is át kell néznünk.

### 3d-s bejárható asztali alkalmazás

Terveztük, hogy csinálunk egy 3d modellt a várhoz, csak nem volt elég időnk rá. A 3D modell elkészítése nagy kihívást jelenthet, de rengeteg lehetőséget kínál a vár és történelme élményes bemutatására. Ez úgy nézett volna ki, mint egy számítógépes játék egyik pályája, szóval ez lehetőséget adna arra, hogy a látogatók virtuális sétát tegyenek a várban és annak környékén, ahol megtekinthetik az épületeket, tárgyakat, berendezéseket és a környezetet. Az időjárást is lehetne szimulálni, így a látogatók az adott évszaknak megfelelő környezetben barangolhatnának. Ez lehet egy unity játék, ami vagy egy asztali alkalmazás vagy a weboldalba beépítve. Ha az asztali alkalmazást tovább fejlesztenénk játékká, akkor az adott időszakban élt emberek életét is bemutathatnánk, így a játékosok kipróbálhatnák, milyen lehetett az életük. A családalapítás mechanikája lehetőséget adna arra, hogy a játékosok családjukat is irányítsák és fejlesszék, míg a küldetések és munkák számos lehetőséget biztosítanának az interaktív játékmenet számára. Az események rekonstruálása szintén nagyszerű lehetőség lenne, hogy megmutassuk a várostromot és más történelmi eseményeket. A játékosok részt vehetnének ezekben az eseményekben, vagy megfigyelhetnék őket egy virtuális időutazás során. Az elkészült játékot akár ki is adhatnánk, hogy minél több emberrel megoszthassuk ezt az élményt. A játékban történő kereskedési rendszer lehetőséget adna arra, hogy a játékosok árukat vásároljanak és adjanak el egymásnak, míg a teljes értékű játékmenet további lehetőségeket nyitna az interakcióra és a történelmi időutazásra.

### Főoldalon a kép slider helyett blender animáció

A főoldalra lehetne csinálni egy egyszerűbb blender animációt, ami a vár 3d-s modelljét mutatná be a kép slider helyett. A várt felülről nézve lassan körbe mutatná be.

### Webshop

Habár a vár valójában nem létezik, de még lehetőség, hogy egy olyan webshopot hozzunk létre, amely a középkori életstílushoz és kultúrához kapcsolódó tárgyakat kínál. A vár által inspirált kiegészítők és fegyverek a látogatók számára érdekesek lehetnek, akik a középkori idők történelmi élményére vágynak. A webshop lehetőséget nyújtana a vár iránt érdeklődők számára, hogy közvetlenül megvásárolhassák a középkori életstílushoz és kultúrához kapcsolódó tárgyakat. Az áruk kínálata lehetne széles skálájú, beleértve a ruházatot, a fegyvereket, a kiegészítőket, az élelmiszereket, a könyveket és sok más tárgyat, amelyek kapcsolódnak a középkori időkhöz.

### Hírportál a főoldalon

Hírek felsorolása a főoldalon adatbázisból. Lehet egy egyszerű lista, amely a legújabb vagy legnépszerűbb híreket tartalmazza, vagy akár egy átfogóbb rendszer, amely a témakörök és a címek szerint csoportosítja a híreket. Az egyes hírekhez lehetne egy rövid összefoglaló és egy link, amelyre kattintva az olvasók elolvashatják az egész cikket.

### Elővételes belépő QR kód vásárlása felület

A belépőjegyet igénylő programokhoz tartozna ez a felület, ahol a felhasználó, ha bejelentkezett, akkor előre megvásárolhatja a várban zajló esemény belépéshez szükséges QR kódot a weboldalon, amit emailben kézbesítenénk a felhasználónak.

### Email létezésének a validációja

Az oldal meg tudná jeleníteni, ha a felhasználó létező email címet adott e meg, vagy sem. Ezt gyakorlatilag egy php osztályba integrált email címszűrő API segítségével oldanánk meg így gyakorlatilag az email küldő funkciónál és a regisztrációnál egyaránt.

### Felhasználók kezelése a Várkezelő programban

A weboldalon regisztrált felhasználók adatainak gyors módosítására lenne fenntartva, mint a megjegyzésre való jogosultsága. Az oldalon lehetséges lenne a felhasználók törlése és jelszóinak megváltoztatása.

### Trigger naplózási események generálása az adatbázishoz

Az alkalmazásaink által használt adatbázis tábláihoz készítenénk egy külön naplózásra fenntartott adatbázist, amely tárolná a másik adatbázisban található táblák és adatainak módosítását, törlését kiváltó eseményeket. Ezzel a funkcióval részletesen nyomon tudnánk követni, hogy mely tábla melyik sorában történt változás mikor, és milyen adatokkal, hogy gyorsan lehessen reagálni az eseményekre és utána nyomozni a rossz szándéknak. Ezt az SQL nyelvben található trigger parancsok segítségével oldanánk meg.

# **Felhasználói dokumentáció**

## A Váci várweb felhasználó dokumentációja

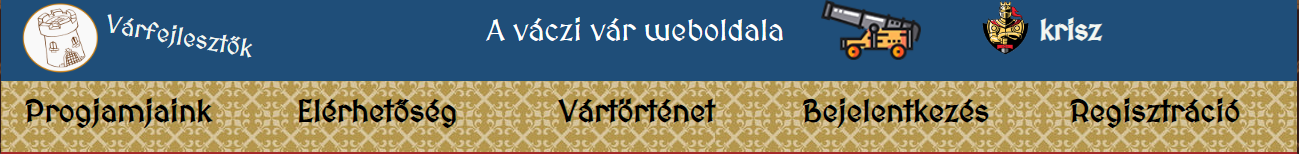
### Rendszerkövetelményei

* Chrome, Firefox, Microsoft Edge böngészők
* Internetkapcsolat
* Windows 7, Windows 10, Windows 11, MacOS, Linux operációs rendszerek

### Alkalmazás célja a cél közönsége

A programot mindenki képes használni, aki böngészni szeretne a váci vár weboldalán. A célja, hogy megfelelő információkat nyújtson a felhasználóknak a várról és az abban szervezendő programokról felhasználóbarát és interaktív módon.

### Fejléc (header)

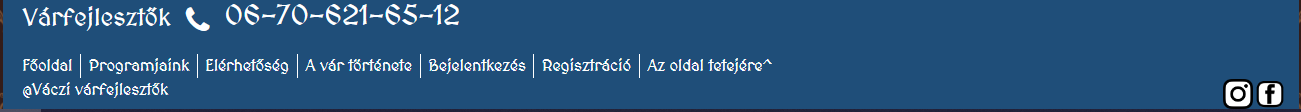
**Láthatóság:** Bejelentkezés nélküli felhasználó csak a bejelentkezett felhasználó részt nem látja, de a bejelentkezés után azt is látni fogja.

**Funkciók:** első sorban a főcímre kattintva az oldal a fő oldalra irányít át. Ha az egeret a belépett felhasználó nevére helyezi, megjelenik a kijelentkezés gomb.

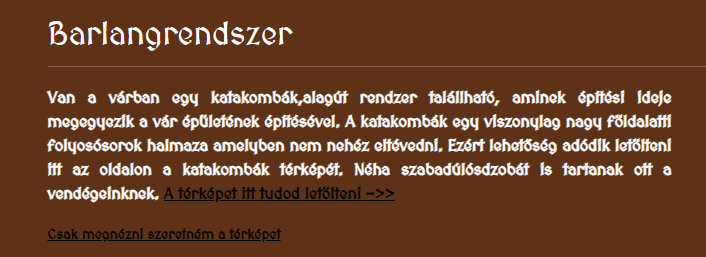
A 2. sorban a „Progjamjaink” link kivételével, ha a felhasználó a linkekre kattint az adott címnek megfelelő oldalra irányítódik át. Az első linkre emelve az egeret előjön az ablak ahol 3 további link található.



### Lábléc (footer)

A lábléc második sorában találhatóak a weboldal különböző oldalaihoz vezető linkek. Ha „Az oldal tetejére” szövegre kattint akkor a nézet az oldal tetejére kerül. A jobb alsó sarokban található 2 ikon, amik a vár szociális felületeikre vezetnek. Az első az Instagramra, a második a Facebookra.

### Főoldal

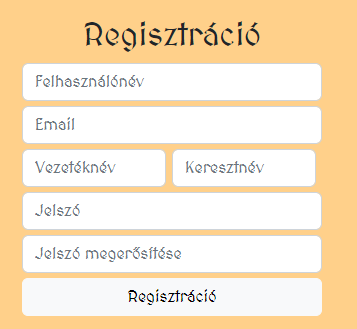
A főoldalon csak 2 link található a barlangrendszer részben fekete színnel és aláhúzással jelölve. Az első link letölt egy képet jpg formátumban, a második link megjeleníti a képet teljes képernyős formátumban.

### Regisztráció

A regisztráció oldalon 6 beviteli mező és egy gomb található.

***A beviteli mezők sorban:***

* **Felhasználónév:** betűket és számokat tartalmazhat, de speciális karaktereket nem, 2 karakter a minimális követelmény, és 12 karakter adható meg maximálisan.
* **Email:** email típusú szöveget tartalmazhat, kötelezően tartalmaznia kell a „@” szimbólumot és a levelezőszerver címét, példa: [felhasznalo@email.hu](mailto:felhasznalo@email.hu)
* **Vezetéknév és Keresztnév:** csak betűket tartalmazhat
* **Jelszó:** bármilyen karakter bevihető, minimum követelmény az 5 karakter és szóköszt nem tartalmazhat.
* **jelszó megerősítése:** Csak akkor fogadja el a bevitelt, ha megegyezik a jelszó mező szövegével.

Ha a mezőkbe a megfelelő értékeket ír be, akkor a „Regisztráció” gombra kattintva a fiókja elmentődik az adatbázisba és a bejelentkezés oldalra navigálódik.

### Bejelentkezés

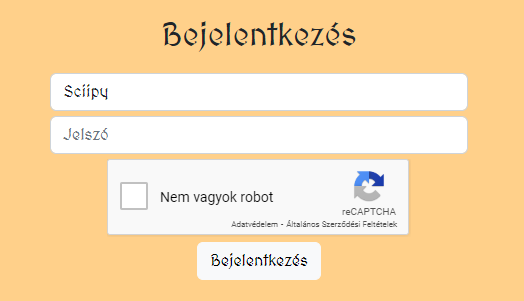
A bejelentkezés oldal 2 beviteli mezőt, egy ember hitelesítést, és egy gombot tartalmaz.

A beviteli mezőkbe a regisztráció oldal felhasználónév és jelszó mezője alapján kell bevinni az értékeket:

* **Felhasználónév:** betűket és számokat tartalmazhat, de speciális karaktereket nem, 2 karakter a minimális követelmény, és 12 karakter adható meg maximálisan.
* **Jelszó:** bármilyen karakter bevihető, minimum követelmény az 5 karakter és szóköszt nem tartalmazhat.

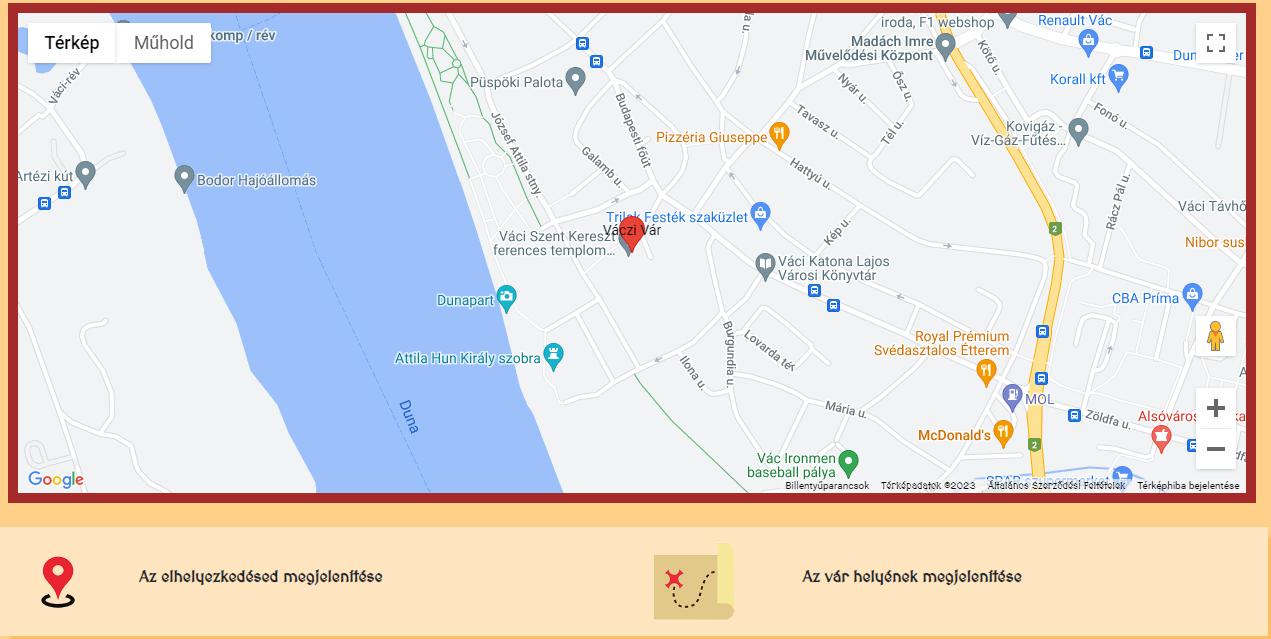
A validálás mező bepipálása nélkül nem lehetséges a bejelentkezés.

Ha megfelelő értékeket ír be, akkor a bejelentkezés gombra kattintva, ha egyezést talál a megadott fióknév és jelszó alapján, akkor bejelentkezik a fiókjába és megjelenik a fejlécen a fiókneve.



### Elérhetőség

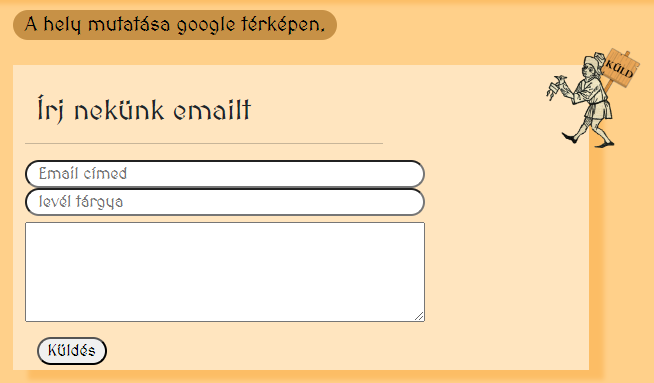
Az elérhetőség oldalon elsősorban egy térképet láthatunk, amely ugyanúgy funkcionál, mint a Google térkép. A térkép alatt található egy szekció, ami a térképen navigál. Az „Az elhelyezkedésed megjelenítése” szövegtől balra található ikon kattintva a böngésző megkérdezi, hogy az ön tartózkodási helyéhez hozzá férhet-e. Ha elutasítja az ikon nem fog funkcionálni, és nem fog történni semmi, ha utána rákattint. Ha elfogadja, akkor a térkép a helyzetére fog navigálni.

**A másik ikonra kattintva a vár elhelyezkedését láthatja a térképen.** 

A navigációs szekció alatt található egy gomb, amire rákattintva egy új lapra navigálva a Google térképen mutatja meg a vár elhelyezkedését.

A gomb alatt egy email küldő szekció található. Itt három beviteli mező és egy gomb található:

* **Email-cím:** a már meghatározott email forma.
* **levél tárgya:** bármilyen karakter használható, üresen nem maradhat
* **a levél tartalma:** 
  + bármilyen karakter használható
  + maximum 200 karakter
  + linket nem írhat a mezőbe
  + üresen nem küldheti be
  + Be kell jelentkeznie a felületre

Ha érvényes értéket írt be az email mezőbe, akkor elküld egy emailt nekünk és egy értesítést kap a szekció alján, amit a jobb oldalán található x-el eltüntetheti

### Jövőbeli programjaink

A programok oldalon egy szűrés található és a programok felsorolása.

A szűrés első sorában 3 legördülő menü található:

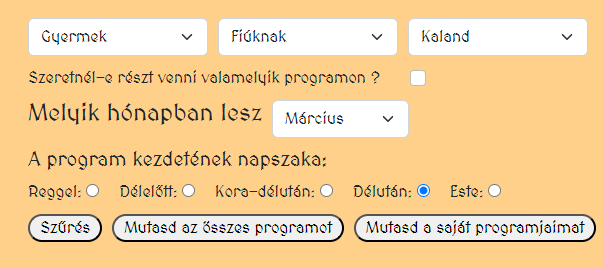
* milyen korosztálynak szól a program: gyermek, felnőtt, stb.
* melyik nemnek lehet érdekesebb a műsor: fiú/lány/mindkettő
* milyen típusú a műsor: kaland, kulturális, stb.

Ezek alatt egy bepipálható mező található.

A 3. sorban egy legördülő menü látható, amelyből azt választhatja ki, melyik hónapban szerepel.

Utána a program kezdetének napszakját választhatja ki az 5 lehetőség közül.

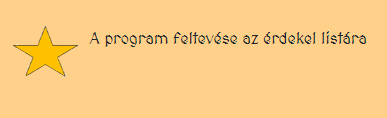
3 gomb található a végén:

* Szűrés: a kiválasztott opciók alapján, ha talál, a kritériumoknak megfelelő programot azt felsorolja a szűrés alatt.
* Mutasd az összes programot: Ez a gomb arra szolgál, hogy megmutatja az összes programot.
* Mutasd a saját programjaimat: Ez a gomb csak akkor funkcionál, ha be vagy jelentkezve, ha nem vagy, ugyanúgy rátudsz kattintani, de csak egy hiba üzenetet ír ki. A gomb arra szolgál, hogy ha a felhasználó egy programot felvett az érdeklődési listára, akkor itt tudja felsoroltatni.

A felsorolt program nevére tud a felhasználó kattintani és ezzel átirányítódik a program leírásának oldalára.

### Kiválasztott program

A kiválasztott program oldalon 2 gomb szerepel. Az első csak akkor működik, ha be van jelentkezve, különben egy hiba üzenetet dob ki. Ezzel a gombbal tudja felvenni a felhasználó a programot az érdekeltségi listájára, amelyet a programok oldalon tud felsoroltatni.

****

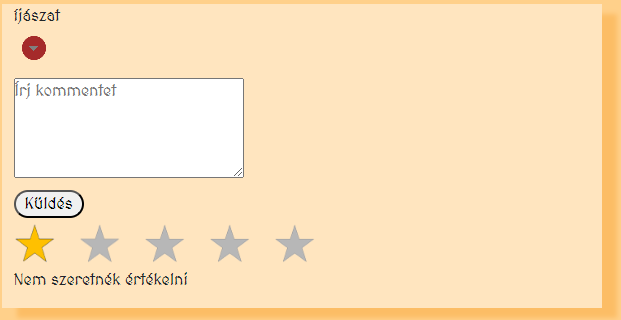
### Megjegyzések

A megjegyzések oldalon először a felsorolt kommenteket láthatja majd a megjegyzés küldő szekciót. A megjegyzés küldéséhez is be kell jelentkeznie.

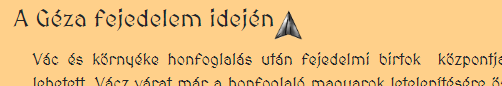
Először a program típusát lehet kiválasztani. Ez nem kötelező. Ha a szöveg alatti formára kattint, előugrik egy lista ahonnan a kiválasztott programot kell kiválasztania. Ezután a komment írására egy szövegdoboz található ahova bármennyi és bármilyen karakter beírható. Ez nem lehet üres.

Következőleg 5 csillagot láthat, balról jobbra, amelyik csillagra kattint, annyi értékelést ad a programnak vagy a várnak. Ez nem kötelező. Viszont ha már bejelölt egy értékelést, de meggondolta magát akkor a „Nem szeretnék értékelni” szövegre kattintva vissza vonhatja a csillag értékelést.

Ha be van jelentkezve és nem hagyta üresen a komment mezőt, akkor a gombra kattintva sikeresen elküldi a kommentet, és a szekció felett lévő felsorolásban látható lesz a kommentje.

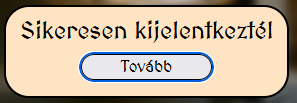
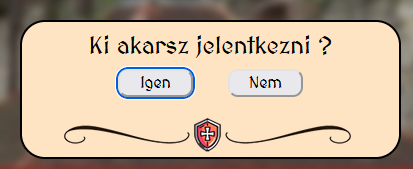


### Vártörténet

A vártörténet oldalon 5 korszak látható. mindegyik végén található egy nyíl, amire rákattintva megjelenik a korszakhoz tartozó történelem és egy kép. Újra a nyílra kattintva csukható vissza.

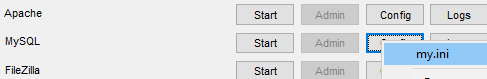
### Kijelentkezés

A kijelentkezést a fejlécben érheti el a bejelentkezett felhasználó nevére húzva az egeret, ekkor előjön a kijelentkezés gomb. Ha arra rákattint, előjön egy dialógus, amiben ha az „igen” szóra kattint, kijelentkezik a fiókjából és egy másik dialógus jön elő ahol csak a dialógust bezáró gombja található. Ha viszont az első dialógusban a „nem” szóra kattint, nem jelentkezik ki a fiókjából és bezáródik a dialógus.

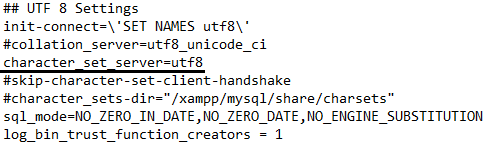


### A weboldal felállítása (telepítése)

Először lépjen be az XAMPP kezelő felületére és ott kattintson a MySQL sorában található Config fülre és azon belül a my.ini opcióra.



Megjelenik egy szöveges állomány, ahol keresse ki az UTF 8 beállításokat és távolítsa el a ’#’ jelet a „character\_set\_server=utf8” elejéről, hogy a weboldal az ékezetes karaktereket is meg tudja jeleníteni.



Ezután indítsa el a kezelőfelületen a MySQL-t és az Apachi szervert és menjen fel a localhost oldalára úgy, hogy beírja a címsorba, hogy **localhost/phpmyadmin** címet. Ha fellépett az adatbázis szerver felületére, akkor nyissa meg az importálás funkciót

Ha megnyitotta az importálást, akkor a „Tallózás” lehetőségnél válassza ki az SQL\_dump\_szakdolgozat.sql SQL parancsokat tartalmazó telepítő állományt és töltse fel.



Végül nem kell mást tennie, mint hogy a böngészője címsorába beírja, hogy: **localhost/ProjectX** és elindult a weboldal.

## A várkezelő alkalmazás felhasználói dokumentációja

### Rendszerkövetelmények

* Windows 10, Windows 11 operációs rendszerek
* 500 MB RAM
* Internetkapcsolat
* 12 MB tárhely
* Minimum Net framework verzió: 4.6.1

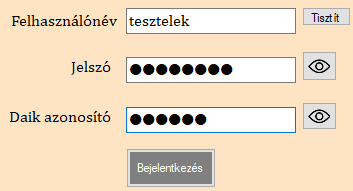
### Ki használja ezt az alkalmazást és milyen célból

A programot az adminisztrátorok és a fő adminisztrátorok használják, akik a várhoz tartozó programokat kezelik. Az alkalmazással az ügyintézők kezelhetik saját fiókjuk adatait és kezelhetik a programokhoz tartozó tulajdonságokat és képeket.

### Bejelentkezés az alkalmazásba

Ha megnyitotta a programot, akkor megjelenik maga előtt egy bejelentkezési felület, ahol ahhoz, hogy használatba tudja venni az alkalmazás további funkcióit meg kell adnia az adminisztrátori fiókjához tartozó felhasználónevet, jelszót és a daik azonosítót. Ha megadta, akkor lehetősége van továbblépni a menübe. A bejelentkezést a „Bejelentkezés” szövegű gombra való kattintással tudja megtenni.

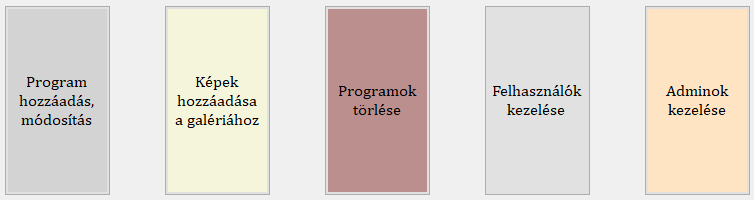
Ha bizonytalan a jelszóval kapcsolatosan, amit a jelszó és daik mezőbe írt, akkor a bementi mező mellett található szem ikonnal rendelkező gomb lenyomásával meg tudja jeleníteni az eddig beírt értéket. A „tisztít” nevű gombbal pedig ki tudja törölni a felhasználónevet a mezőből.



### Menü használata

#### Menügombok

Ha sikeresen be tudott lépni a menübe, akkor a kívánt funkció megnyitásához az ahhoz tartozó gombra kell rákattintania. A gombokon szerepel az azzal megnyitható funkció felületének neve.



#### Fiókváltás

A menü bal felső sarkában el van helyezve egy vezérlő, amire a „Fiókváltás” szöveg van írva. Ha rákattint erre a gombra, akkor a program visszadobja önt a bejelentkezés felületére, ahol beléphet egy másik fiókkal.



#### Felhasználónév

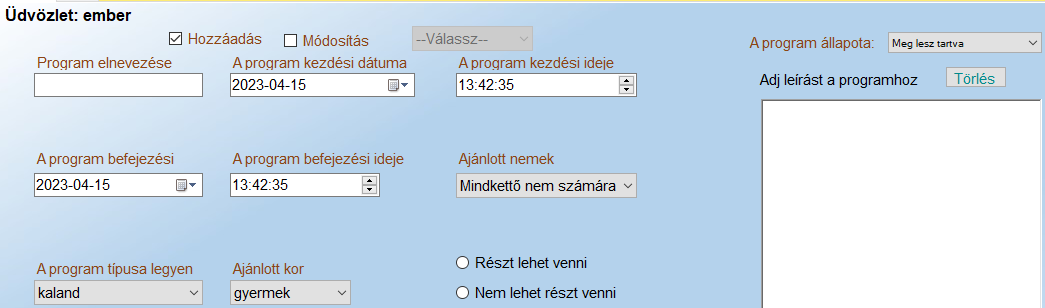
A fiókjának a felhasználóneve, amivel belépett, az minden oldalon a bal felső sarokban jelenik meg.

### Programok hozzáadásának, módosításának felülete

#### Mit nyújt a felület a felhasználó számára?

A programkezelő ablakban képes új programokat hozzáadni a várban szerveződő programok weboldalon való megjelenítésére. Pontosabban képes az új program nevét, idejét és tulajdonságait és a hozzá társítani kívánt képeket megadni a felületen található vezérlők segítségével. Továbbá módosítást is nyújt az alkalmazás, amivel az események tulajdonságait szekeszthetjük a bementi mezők értékeivel.

#### Egy program hozzáadása a felületen



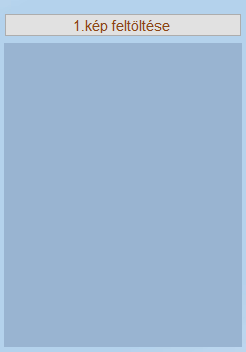
Ha a felhasználó új programot akar hozzáadni, akkor ki kell töltenie a következő mezőket:

* **Program elnevezése:** az új program neve lesz
* **Program kezdeti dátuma:** A kezdetének évét, hónapját, napját állíthatja be, az önnek megfelelő idő kiválasztásával
* **Kezdési idő:** Az esemény kezdésének idejét kalibrálhatja, az óra, perc, másodperc átírásával. Alapesetben a betöltéskor számolt idő lesz rajta.
* **Program befejezési dátuma**
* **Program befejezési ideje**
* **Ajánlott nemek:** Kiválaszthatja a lenyúló listából, hogy milyen nem számára ajánlatos a hozzáadandó program.
* **Program típusa:** Megadhatja lenyúló listával, hogy milyen típusú programról van szó.
* **Ajánlott kor:** A programhoz ajánlott kor szakaszának kiválasztása.
* **Részt lehet e venni opció:** Kijelölheti 2 opcionális gomb közül, hogy a megrendezésre került programon részt lehet e venni, vagy csak szemlélni lehet.
* **Program leírása:** Hosszabb programleírást képes megcsinálni 2000 karakterig terjedően, amit a felület jobb oldalán található nagy fehér szövegdobozba lehet begépelni, vagy bemásolni.

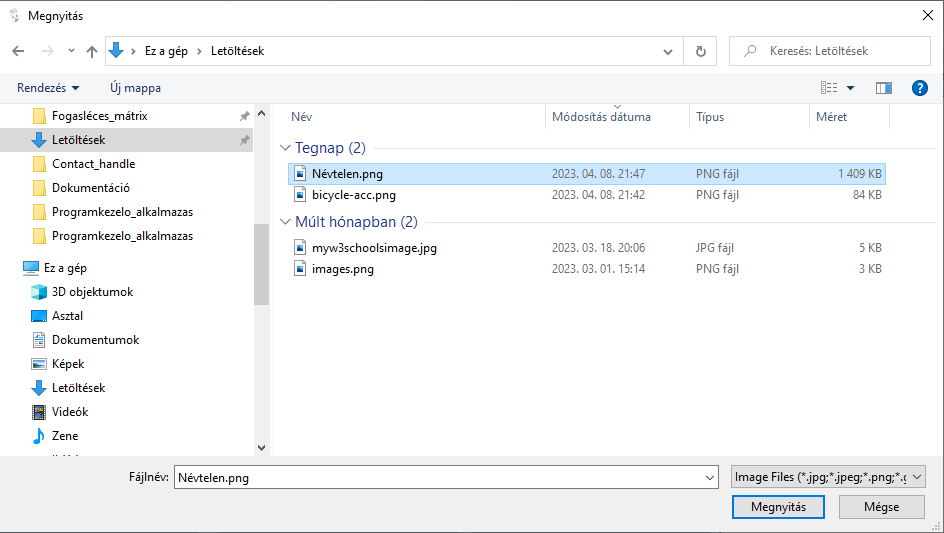
A fent említett összes mezőt ki kell tölteni, különben a program nem végzi el a mentést és hibaüzenetet jelenít meg az ablak jobb alsó sarkában piros színnel. Ha sikeres a mentés, akkor értesül a felhasználó a program által, egy üzenetdoboz formájában.

##### Kép kiválasztása a programhoz

A felhasználónak 3 képet is ki kell választania a hozzáadás elvégzéséhez. Ezt úgy teheti meg, hogy először megnyomja a „kép felöltése” feliratú gombok valamelyikét.



Ha megnyomta a „kép feltöltése”gombot, akkor megjelenik egy kép kiválasztását biztosító felület, ahol el kell navigálnia a kiválasztani kívánt kép könyvtárába és rákattintani a kép nevére, ezután pedig a „Megnyitás” szövegű lehetőségre. A felülethez csak jpg, jpeg és png formátumú képeket képes kiválasztani.



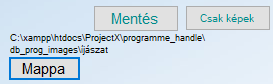
Ha kiválasztotta a képet, akkor megjelenik a kijelölt fotó elérési útvonala a megjelenített kép fölött.

##### Célmappa kiválasztása

A felület jobb alsó sarkában található a „Mappa” szövegű gomb, aminek a segítségével kijelölheti, a mappát ahová szeretné másolni a képeket. Így a bemásolt képeket közvetlenül tudja egy szerver mappába másolni, vagy előkészíteni egy szerverre való feltöltésre.

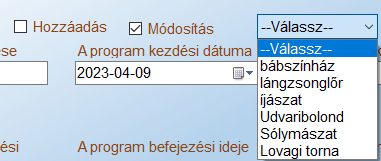
#### Csak képek másolása

Ha csak képeket kíván átmásolni minden adatbázis művelet és mező megadása nélkül, akkor egyszerűen a „Csak képek” gomb megnyomásával átmásolhatja a kijelölt képeket, ha megadta a „Mappa” feliratú vezérlő rányomása után, hogy hova szeretné másolni a fotókat. A másolásra kijelölt kép célmappáját a gombok között lehet megtekinteni.



#### Kijelölt program módosítása

Ha a felhasználó a felület felső részében kijelölte a módosítás lehetőséget, akkor lehetősége adódik a kiválasztani a szerkeszteni kívánt programot. A kiválasztásnál a bemeneti mezők automatikusan kitöltődnek a program paraméterei alapján, így könnyen észrevehető, hogy milyen értéket is szándékozik megváltoztatni.



A hozzáadásnál a listát nem lehet és nem is szükséges használnia és nem kötelező képeket kiválasztani és feltölteni. Gyakorlatilag ezután megadja a bemeneti mezőket, és ha minden mezőt kitöltött és ha, rákattintott a „Mentés” gombra, akkor módosulnak a megváltoztatott értékek a kiválasztott program érintett mezőiben.

### A galéria képek feltöltésének funkciója

#### Mit tehet a funkció segítségével a felhasználó?

A felülettel feltöltheti a befejezett programokról készített képeket, amelyeket a Váci várweb oldal jelenít meg a „Galéria” lapon.

#### Képek feltöltésének menete

A feleület bal felső részében láthat egy lenyúló listát, amely tartalmazza a már befejeződött események neveit. A megnevezések közül először ki kell jelölnie, hogy mely befejezett programhoz tartozzanak a kiválasztandó képek. Ezzel gyakorlatilag kijelölheti a weboldalon található galéria képek oldalon az új kép alatt mely program neve és ideje legyen feltüntetve.

Ezután ki kell választania legalább egy darab képet, amit szándékozik átmásolni a szerver mappájába. A képet úgy tudja kiválasztani, hogy a még a még üres és szürke képdoboz alatt rákattint a „Kép” feliratú gombra, ami megjelenít a számára egy kép kijelöléséért felelős ablakot, ahol meg tudja azt tenni.

Ha van legalább egy darab kép kiválasztva, akkor utána az ablak jobb alsó területén egy „Célmappa” szövegű menüikont, amivel hasonló módon megnyílik egy mappa kiválasztó Windows felület, amellyel azt a mappát jelölheti ki, ahová kívánja a képet vagy képeket másolni, ahonnan egyszerűen előkészítheti a szerver számára a fotók megjelenítését.

Legvégül, ha megvalósította a fent leírt műveleteket, és ha rákattint a „Feltöltés” gombra, akkor végrehajtódik a másolás és a feltöltés. A sikeres, vagy sikertelen feltöltés bekövetkeztéről egy üzenetdoboz értesíti a felhasználót.

### Programok törlése

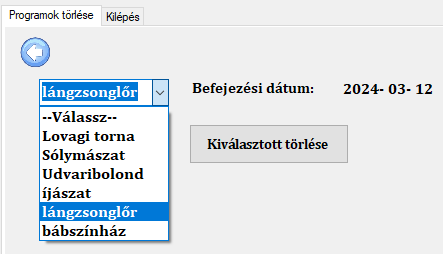
#### Mit kezdhet ezzel a felülettel?

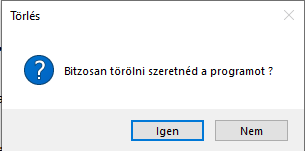
A programok törlése funkcióval eltűntethet eseményeket a rendszerből és azzal együtt eltávolíthatja a törlendő programhoz tartozó megjegyzéseket, kedvenc megjelöléseket és kép adatokat.

#### Hogyan tud törölni?

A törléshez, ki kell a felhasználónak választania először, hogy melyik programot kívánja eltávolítani. Érdemes megnéznie, hogy mikor fejeződik, vagy fejeződött már be a kiválasztott program a jobb oldalon található dátum alapján, nehogy olyan programot töröljön véletlenül, amely még nem ért véget a várban.

Ha kiválasztotta a programot, akkor meg kell, hogy nyomja a törlés gombot. Ezután, ha megerősítette a törlést az előugró ablak felületén, akkor megtörténik a program törlése a rendszerből, így már a weboldal sem lesz képes megjeleníteni az eltávolított eseményhez tartozó adatokat.

******



### Adminisztrátorok kezelése

#### Mire képes ez a felület?

Az ügyintézők kezelése funkcióval megváltoztathatja a saját fiókjának a jelszavát, email címét és daik kulcsát.

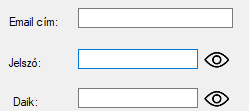
#### Név kiválasztása

Ha a menüben az adminok kezelése lehetőségre kattintott rá, akkor megjelenik számára egy névlista, ahol az összes adminisztrátor szerepel. Nem kell sokat keresnie a nevét a felsorolásban, mivel az ön által éppen használt felhasználónevet mindig megtalálhatja a lista tetején bézs színűvel kiszínezve.



Ha rákattint a nevére, akkor a program átdobja önt a kezelő felületre, ahol beállíthatja a megváltoztatni kívánt paramétereket.

Be kell gépelnie legalább egy megváltoztatni kívánt értéket, a kritériumoknak megfelelően.

****Az alábbiakat adhatja meg és módosíthatja:

* **Email cím**
* **Jelszó**
* **Daik kulcs**

Ha bizonytalan azzal kapcsolatban, amit beírt a mezőbe, akkor használja a jelszó megjelenítése funkciót a szem ikont tartalmazó gombra való rákattintással. Ha már nem akarja, hogy látszódjon a jelszó, akkor engedje el a kurzort a gombról.

Végül a felület alján található „Módosít” gombra való kattintásra és megerősítésre elvégzi a program a módosítást az ön számára.

### Visszalépés a menübe

Ha vissza szeretne lépni a menübe, akkor használja a kék hátterű fehér színű nyilat ábrázoló ikont, úgy, hogy rákattint. Ezzel vissza is navigált a menübe. Ez az ikon szinte az összes felületen megtalálható.

# **Munkabeosztás a csapatban**

## Less Máté munkássága

1. A programokhoz szükséges adatbázis táblák és mezők tervezése.
2. Az adatbázis és tábláinak létrehozása.
3. A táblák kapcsolatainak megvalósítása megszorításokkal.
4. A weboldal php osztályainak és metódusainak kidolgozása és működtetése. (Backend)
5. Weboldal adatbázis kapcsolatának megvalósítása.
6. Regisztráció oldalon található bementi értékek helyességének kimutatása JavaScript állományokkal.
7. Regisztráció hitelesítése és végrehajtása php-val.
8. Bejelentkezési rendszer a szűréssel együtt php programnyelvben.
9. Recapcha validáció.
10. Google APi térkép beimplementálása
11. Vár és a felhasználó elhelyezkedésének a megjelenítése a térképen JavaScripttel.
12. Email küldő rendszer, php osztályok és phpmailer segítségével.
13. Composer futtatása és az autoloader megvalósítása.
14. Adatbázis adatok lekérése a weboldal számára.
15. A bemeneti mezők értékeinek feltöltése az adatbázisba php osztályokkal.
16. A program és kép adatok lekérése, valamint listázása a weboldalon való megjelenítésre.
17. Szűrések a weboldalon PHP-val és JavaScripttel.
18. JavaScript és PHP állományok létrehozása a program hozzáadása a kedvencekhez funkció működéséhez.
19. Megjegyzés írása funkció a csillagok legenerálásával PHP és JavaScript programnyelvek felhasználásával.
20. Sessionkezelés és felhasználónév használatának kezelése, szűrése egyes funkcióknál.
21. Kijelentkezés megvalósítása.
22. A várkezelő alkalmazás összes funkciója a C# nyelvben megoldva.
23. Telepítő létrehozása a Várkezelő alkalmazáshoz.
24. Tesztadatok legenerálása az adatbázisba.

## Nemecz Krisztián munkássága

1. Stílus megtervezése a weboldalhoz.
2. Fajléc (header) megszerkesztése.
3. Főoldal, bejelentkezés, regisztráció, galéria, programok, megjegyzések, vártörténet, elérhetőség, lábfej (footer) megformázása CSS, bootstrap és JavaScript segítségével (Frontend).
4. A mozgó ágyú animációja a fejlécben.
5. Az égő fáklya animációja a főoldalon.
6. A galambbal levelet küldő emberke animációja az elérhetőség oldalon.
7. Reszponzivitás megvalósítása az összes oldalhoz.
8. A fejléc alatt található menüpontok stílusának és dinamikus megjelenítésének biztosítása.
9. A CSS média lekérdezések felhasználása a nagyobb reszponzivitás eléréséért.
10. A képlapozó csúszka funkció létrehozása a főoldalon.
11. Képek kiválasztása az oldalakhoz.
12. Képek megszerkesztése.
13. CSS áttünések létrehozása a bemeneti mezők fókuszára.
14. A vártörténet oldal megalkotása a szövegcsúsztató animációval.
15. A 404. hibakódból betöltött felület megformázása.
16. A kijelentkezés dialógus formázása és reszponzivitásának biztosítása.
17. A megfelelő színek és formák megválasztása az oldalhoz.
18. A fejlesztéshez szükséges kutatómunka támogatása.

# **Irodalomjegyzék**

(https://www.w3schools.com/php/php\_oop\_classes\_objects.asp, w3schools, OOP Tutorial, 2023)

(https://www.php.net/manual/en/function.session-status.php, PHP.net, Session\_státusz szűrése, 2023)

(https://www.php.net/manual/en/pdo.construct.php, PHP.net, PDO\_adatbázis\_csatlakozás, 2023)

(https://stackoverflow.com/questions/68833018/save-multiple-data-to-database-with-php-pdo, stackoverflow.com, PHP adatok feltöltése adz adatbázisba, 2023)

(https://www.php.net/manual/en/function.file-get-contents.php, php.net, Szöveg beolvasása phpval, 2023)

(https://www.phptutorial.net/php-pdo/php-fetchall/, phptutorial, adatbáizs adatok kinyerése és összegyűjtése, 2023)

(https://www.php.net/manual/en/function.preg-match.php, php.net, Bemenetek szűrése, 2023)

(https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage, mozilla.org, JavaScript helyi tárolók, 2023)

(https://www.w3schools.com/howto/howto\_js\_animate.asp, w3schools, JavaScript transitions, 2023)

(https://developers.google.com/maps/documentation/javascript/adding-a-google-map, Google Cloud Platform, Elhelyezkedés mutatása a térképen, 2023)

(https://fonts.google.com/specimen/MedievalSharp, Google Fonts, Weboldal hozzáadott stílusos szövegformátum , 2023)

(https://www.w3schools.com/Css/css3\_mediaqueries\_ex.asp, w3schools.com, CSS média lekérdezések, 2023)

(https://linuxhint.com/css-not-hover/, linuxhint, Not hover, 2023)

(https://getcomposer.org/doc/00-intro.md, Composer futtatása, 2023)

(https://www.google.com/recaptcha/about/, Google, Get recapcha, 2023)

(https://cloud.google.com/, Google, Google cloud szolgáltatás, 2023)

(https://mapsplatform.google.com/, Google, Google térkép platformja, 2023)

(https://stackoverflow.com/questions/11993211/how-to-fill-datatable-with-sql-table, stackoverflow , C# adattábla feltöltése adatbázisból , 2023)

(https://learn.microsoft.com/en-us/dotnet/api/system.windows.forms.form.showdialog?view=windowsdesktop-8.0, microsoft.com, Form Showdialog , 2023)

(https://stackoverflow.com/questions/1142828/add-timer-to-a-windows-forms-application,stackoverflow.com ,Timer for Winform app, 2023)

(https://www.c-sharpcorner.com/UploadFile/mirfan00/uploaddisplay-image-in-picture-box-using-C-Sharp/,c-sharpcorner.com ,Select picure in your project, 2023)

(https://stackoverflow.com/questions/11477083/hashing-text-with-sha-256-at-windows-forms, stackoverflow.com, Sha256 encryption in Winfrom , 2023)

(https://dev.mysql.com/doc/connector-net/en/connector-net-connections-string.html, dev.mysql.com ,C# MySQL kapcsolódás, 2023)

(https://cooltext.com/, Stílusos szövegek létrehozása, 2023)

(Szerző: Blum Richard, Cím: PHP, MySQL, & JavaScript All-in-One For Dummies, Évszám: 2018, kiadó : For Dummies)

(Szerző: Reiter István, Cím: C# programozás lépésről lépésre , Évszám: 2018, kiadó : JEDLIK OKTATÁSI STÚDIÓ KFT.)

# **Mellékletek**

#### A Váci várweb működéséhez szükséges porgramok:

* XAMPP (PHP 8)
* Composer
* Adatbázis lefuttatása és MySQL szerver, futó állapota
* MySQL Connector
* MySQL Connector NET 8.0.32 a telepített alkalmazáshoz

#### A Várkezelő alkalmazás működéséhez szükséges programok

* XAMPP
* Adatbázis lefuttatása és MYSQL szerver, futó állapota

Ezeket a programokat megtalálják a csatolt adathordozóban a szakdolgozat/ alkalmazások telepítése mappában.

#### Adatbázis dump állomány futtatása

Az SQL\_dump\_szakdolgozat.sql fájl tartalmazza az adatbázis adatokat és a kezdeti tesztadatokat a porgramokhoz. A megfelelő működéshez futtatni kell a localhost/phpmyadmin felületen az importálás lehetőséggel, Ha kijelöli a dump fájl egész tartalmát és lefuttatja az SQL oldalon, akkor ugyan- úgy működni fog a futtatás. Fontos, ha importálja, akkor NE módosítson a beállításokon.

#### Belépéshez és teszteléshez szükséges jelszavak és nevek:

Adminisztrátor a várkezelőhöz:

**Felhasználónév:** Admin1

**Jelszó:** Admin06

**Daik:** Admin06

Fő adminisztrátor:

**Felhasznalónév:** Foadmin1

**Jelszo:** Foadmin06

**Daik:** FAdmin06

Felhasználó a weboldalhoz:

**Felhasznalónév:** Felhasznalo1

**Jelszo:** Felhasz06

Adminisztátor a weboldalhoz:

**Felhasznalónév:** Admin1

**Jelszo:** Admin06

#### A mellékelt adatok elhelyezkedése és ismertetése

* Alkalmazás mappa: Itt találhatóak az elékszített programok.

1. ProjectX.rar: A weboldal állományait tartalmazza tömörített formátumban.
2. Várkezelo\_install.msi: A várkezelő program telepítője, ami egy futtatható állomány.
3. Varkezelo\_alkalmazas\_alfa.rar: A várkezelő program a fejlesztési adataival és kódsoraival együtt, amely a visual studio fejlesztési környezettel módosítható és megtekinthető.

* Dokumentáció mappa: Tartalmazza a szakdolgozat dokumentációt word és pdf formátumban egyaránt.

1. Less Máté, Nemecz Krisztián szakdolgozat dokumentáció.docx
2. Less Máté, Nemecz Krisztián szakdolgozat dokumentáció.pdf

* Szükséges programok telepítése mappa: Itt találhatók a működéshez alapvető fontosságú programok telepítési állományai.
* Felhasználók\_és\_jelszavaik.txt: Tartalmazza az alkalmazásokhoz létrehozott teszt fiókok belépési adatait.
* SQL\_dump\_szakdolgozat.sql: Az adatbázis telepítéséhez szükséges SQL fájl, ami tartalmazza az adatbázis és tábláinak létrehozását és a hozzájuk tartozó teszt adatokat.
* A github feltöltések eléréséhez tartozó link:

1. https://github.com/krisz45/ProjectX.git

1. 1(https://www.letscode.hu/2015/03/12/composer-a-php-fejlesztok-kedvenc-zeneszerzoje/, 2023)

   2(https://en.wikipedia.org/wiki/Visual\_Studio\_Code, 2023)

   3 (https://webiskola.hu/php-ismeretek/php-apache-xampp-letoltese-telepitese/, 2023) [↑](#footnote-ref-1)
2. 5(https://en.wikipedia.org/wiki/Google\_APIs, 2023)

   7(https://docs.fileformat.com/hu/web/json/, 2023)

   2(https://en.wikipedia.org/wiki/MySQL\_Connector/ODBC, 2023)

   6(https://hu.wikipedia.org/wiki/Google\_T%C3%A9rk%C3%A9p, 2023)

   1(https://gremmedia.hu/mi-az-a-bootstrap-4-hogyan-hasznaljuk, 2023)

   6(https://support.google.com/google-ads/answer/9004655?hl=hu, 2023)

   8(https://www.awh.hu/kb/webtarhely/mi-az-a-htaccess-fajl-es-mi-celja, 2023)

   9(https://hu.wikipedia.org/wiki/Kriptogr%C3%A1fiai\_hash\_f%C3%BCggv%C3%A9ny, 2023) [↑](#footnote-ref-2)
3. https://fonts.google.com/specimen/MedievalSharp [↑](#footnote-ref-3)
4. https://fontawesome.com/icons [↑](#footnote-ref-4)