**Szegedi Szakképzési Centrum Vasvári Pál Gazdasági és Informatikai Szakgimnáziuma**

**Az 54 213 05 számú Szoftverfejlesztő szakképesítés záródolgozata**

# Fotós segéd

Készítette:

Marosi Márk Dániel

**Szeged**

**2019**

# Tartalom

# BEVEZETÉS

Egyik hobbim a fotózás, így mikor a témaválasztásra került sor, elég hamar arra a következtetésre jutottam, hogy egy, a fotózással kapcsolatos rendszert szeretnék fejleszteni. Mivel a témában jártas vagyok, és valamelyest érint is, így nem volt nehéz feltárni milyen szükségletek és követelmények merülnek fel egy ilyen alkalmazás fejlesztése során.

# 1. FEJLESZTŐI DOKUMENTÁCIÓ

## 1.1 ADATBÁZIS TERVEZÉS

A fejlesztést az adatbázis tervezéssel kezdtem. Minden azonosításra használt értékema MySQL auto increment funkcióját használja, ezen kívül a dátumoknál használtam még a CURRENT_DATE illetvre a CURRENT_TIMESTAMP beépített függvényket. Legelőször feltérképeztem, melyek azok a legfontosabb adatok, amelyeket érdemes nyilvántartani egy fényképezőgépről, és hogy az adott tulajdonságokat milyen típusú változóban kellene tárolni.

1.sz.táblázat

*Fényképezőgépek tábla*

| Adat | id | gyárto | sorozat | típus | pixel | szenzor | objektív | ár |
|---|---|---|---|---|---|---|---|---|
| Változó | int | varchar | varchar | varchar | float | varchar | varchar | int |

Miután a fényképezőgépek adatait felvittem az adatbázisba, azután következtek az objektívek. Vannak olyan fényképezőgépek, melyekhez nem tartozik objektív foglalat, de a többséghez igen, így az objektív foglalat típusával kötöttem össze a két táblát. Ezentúl itt is fel kellett mérnem, hogy az objektívek mely tulajdonságait szeretném eltárolni, ugyanis rengeteg tulajdonsággal rendelkezhetnek, de szerettem volna csak a legfontosabb tulajdonságokat eltárolni, így spórolva az adatbázis kapacitással. Itt is figyelnem kellett a megfelelő változók használatára, mert nem mindegy, hogy egy objektív legnagyobb blende értéke 2.1 vagy 2.8, valamint egy logikai változót is használtam “stabil” név alatt, amely igaz értéke jelzi, ha az adott objektív rendelkezik beépített stabilizátorral, és hamis értéke azt, ha nem.

2.sz.táblázat

*Objektívek tábla*

| id | típus | gyártó | név | gyújtótáv | stabil | Min blende | Max blende | hossz | súly | ár |
|---|---|---|---|---|---|---|---|---|---|---|
| int | varchar | varchar | varchar | int | bool | float | float | int | int | int |

Miután a fényképezőgépek és az objektívek táblát is feltöltöttem megfelelő mennyiségű adattal, a tervezés következő része a felhasználók nyilvántartására létrehozott tábla lett. Ezzel egyidejűleg fejlesztettem a weboldal regisztrációs és bejelentkező felületét, így egyből ellenőrizni tudtam, hogy a tábla minden szükséged adatot megfelelően tárol el, vagy ki kell még egészíteni. Kezdetben a regisztráció után közvetlenül, titkosítás nélkül tároltam el a jelszavakat a táblában, de a korszerű követelmények miatt később titkosítást alkalmaztam a jelszavakra, ennek következményeként az adatbázisban található, jelszó tárolására használatos oszlop tulajdonságát is ehhez kellett igazítanom. Későbbi változtatás során került az adatbázisba még a regisztrációs dátum, valamint egy logikai változó, melynek igaz értéke az adott felhasználó admin jogokkal való rendelkezését hivatott eltárolni. Ezt természetesen webes felületről történő regisztráció során nem lehet beállítani, csakis a már admin jogokkal rendelkező felhasználók által kezelhető asztali alkalmazás segítségével lehet egy felhasználónak admin jogokat biztosítani.

3.sz.táblázat

*Felhasználók tábla*

| Id | Username | Password | Neme | Email | Bemutatkozás | Regisztrációs dátum | admin |
|---|---|---|---|---|---|---|---|
| Int | Varchar | Varchar | Varchar | Varchar | Text | Date | Bool |

Még mielőtt a weboldal legkomplexebb részéhez kezdtem volna el megtervezni az adatbázis-hátteret, előtte egy élő beszélgetés, azaz chat funckiót szerettem volna megvalósítani. A tábla megtervezésével nem volt gondom, tudtam, melyek azok az értékek, melyeket el szeretnék tárolni, de a legfontosabbak a chat üzenet tartalma és az üzenetet küldő felhasználó egyéni azonosítója. Ezek mellé még került egy posztolási dátum is. Annak érdekében, hogy pontosan visszakövethető legyen az összes üzenet, a megszokott Date változó helyett a Timestamp változót használtam, mely nem csak év-hónap-nap formátumban tárolja el a dátumot, de ezek mellé még eltárolja az óra-perc-másodperc adatokat is, mely hasznos lehet egy olyan rendszerben, mint az élő beszélgetés.

4.sz.táblázat

*Chat tábla*

| Üzenet_Id | Üzenet | User_id | Dátum |
|---|---|---|---|
| Int | Varchar | Int | timestamp |

Az adatbázis tervezés utolsó fázisában a weboldal legkomplexebb funkciója – a képfeltöltés - megvalósításához szükséges táblákat terveztem meg. Elsősorban kellett egy tábla, mely eltárolja a feltöltött képek adatait. Mivel külön galériarendszert nem hoztam létre, így elérési útvonalat nem, de fájlnevet el kellett tárolnom. Ezen túl a képfeltöltés során megadható adatok közül néhány még ebbe a táblába kerül, ilyen például a kép címe, leírása, készítésének helye, vagy éppen a kép készítésének dátuma, melyet szintén a feltöltő ad meg. Emellett természetesen tárolásra kerül a kép feltöltésének időpontja is, mely a MySQL beépített függvényét használja. Viszont a feltöltő profilja illetve a kép készítéséhez használt fényképezőgép egy másik táblában kerül eltárolásra, amelyre lentebb térek ki.

5.sz.táblázat

*Képek tábla*

| Id | Fájlnév | Cím | Leírás | Hely | Feltöltés | Készítés |
|---|---|---|---|---|---|---|
| Int | Varchar | Varchar | Varchar | Varchar | Date | Date |

A felhasználónak képfeltöltés során lehetősége van kiválasztani a fénykép elkészítéséhez használt fényképezőgépet. Ezen érték, illetve a felhasználó azonosítója tárolására hoztam létre a fotókészítés táblát. Ebben a táblában a userid érték azonos a felhasználók tábla id értékével, így azonosítva a felhasználót. A kep_id érték a képek tábla id értékével kapcsolódva azonosítja a feltöltött képet, valamint a gep_id és a gépek tábla id értékének azonossága azonosítja a kép készítéséhez használt fényképezőgépet.

6.sz.táblázat

*Fotókészítés tábla*

| Kép azonosító (kep_id) | Gép azonosító (gep_id) | Felhasználó azonosító (userid) |
|---|---|---|
| Int | Int | Int |

## 1.2 WEBOLDAL FEJLESZTÉS

A weboldal megvalósítása során a legfontosabb tulajdonságok az átláthatóság, a könnyű navigáció és a reszponzivitás voltak, melyeket figyelembe vettem a fejlesztés során. Legelőször a kezdőoldalt terveztem meg statikus formában, Bootstrap 4 segítségével, melyet természetesen az egész oldalon használok. A kezdőoldal tartalmaz egy képnézegető ablakot, melyben a weboldalra feltöltött utolsó 5 kép között lehet váltogatni, alatta pedig egy felsorolás, hogy mely funkciók válnak elérhetővé, ha valaki regisztál az oldalon.

### 1.2.1 REGISZTRÁCIÓ

A regisztráció során a felhasználó által kitöltött mezők értékei rögzítésre kerülnek az adatbázis megfelelő táblájában a PHP nyelvben megírt reg.php-ban található MySQL parancs segítségével. A regisztráció viszont csak akkor lehet sikeres, ha a felhasználó által felvitt adatok minden feltételnek megfeleltek. Ilyen feltétel vizsgálatnak vetem alá JavaScript és RegExp segítségével például a beviteli mezőket. A RegExp vagy RegEx rendkívül hasznos, mert egyetlen sor kóddal vizsgálható, hogy az adott szöveg megfelel-e a megkívánt kritériumoknak, míg egyszerű feltételvizsgálattal ez hosszas kódsorokat igényelhetne. Ilyen kritérium nálam például az, hogy az email cím tartalmaz-e ‘@, jelet, illetve található-e a végén pont, és utána még legalább egy betű. Ugyan így a jelszóra is írtam egy vizsgálatot. A jelszónak tartalmaznia kell legalább egy nagy betűt, egy számot, és egy speciális karaktert. Végül pedig vizsgálom azt is, hogy a jelszó, és a jelszó megerősítése mezőbe beírt adatok egyeznek-e. Amennyiben ezen feltételek közül valamelyik nem teljesül, akkor azt a felhasználó tudomására juttatom a hibás beviteli mező alatt, valamint a regisztráció gombot is letiltom. Amennyiben a JavaScript vizsgálat minden feltétele teljesül, a regisztráció gomb aktívvá válik, és lefut a regisztrációért felelős PHP állomány. Ugyanakkor ebben az állományban is vannak vizsgálatok, ilyen például az, hogy a felhasználónév nem egyezhet a jelszóval, de még fontosabb, hogy lefuttatok egy lekérdezést, amely vizsgálja, hogy létezik-e már a regisztrálni kívánó felhasználó által beírt felhasználónév vagy email cím az adatbázisban. Amennyiben létezik, akkor a regisztrációs folyamat megszakad, és jelzést ad a felhasználónak a hiba okáról. Ha a felhasználó által megadott adatok minden feltételnek megfelelnek, akkor végbemegy a regisztráció, melynek sikerességéről szintén tájékoztatást adok, és lehetőséget adok neki egyenesen a bejelentkezési fülhöz eljutni.

### 1.2.2 Bejelentkezés, profiladatok kezelése, módosítása

Sikeres bejelentkezés után a felhasználó számára feloldódik minden funkció. A menüsávban a kijelentkezés gomb mellett található ‚Beállítások' gombra kattintva lenyílik egy menü, melyből el lehet érni a saját profilunk oldalál, de mégfontosabb, hogy innen érhető el a profil módosítása is. A profil módosítása oldalon találkozik a felhasználó a profilkép feltölés lehetőségével, ezen kívül még a felhasználónév, az email cím, a bemutatkozás vagy a jelszó is megváltoztatható. Mivel a profilkép eltárolására egyedi megoldást használok – melyre hamarosan kitérek – és nem adatbázisban tárolom, így külön formot kapott, és egy külön PHP állományt hoztam létre a feltöltéshez. Mivel ez a rész független a többi beviteli mezőtől, így anélkül lehet profilképet módosítani, hogy adatot vinnénk fel az adatbázisba, vagy bármilyen más adatbázis műveletet végeznénk. Amennyiben a felhasználó valamely adatmezőben módosítást végez, és rányom az adatok frissítése gombra, akkor az update.php feldolgozó állomány segítségével az adatbázisban is módosulnak az adatok. Email cím változtatásnál, valamint jelszó változtatásnál is érvényesek a RegExp feltételek, ugyan úgy, mint a regisztrációnál, illetve a jelszót itt is meg kell erősíteni ahhoz, hogy végbemehessen a módosítás, melynek sikerességéről vagy éppen sikertelenségéről is tájékoztatom a felhasználót.

A profilkép feltöltésre egyéni megoldást dolgoztam ki. Úgy gondoltam, felesleges lenne még egy táblát létrehozni az adatbázisban a profilképek tárolására, ezért úgy döntöttem, hogy a feltöltött profilképet eltárolom egy almappában, és a fájl neve minden esetben a feltöltő felhasználó egyéni azonosítója lesz. Így mikor a kép tallózása után a felhasználó a kép feltöltése gombra kattint, akkor egy PHP állomány a feltöltött képet átnevezi és elhelyezi a megfelelő almappában, ahonnan később be lehet majd tölteni. Kicsit több munkát igényelt a már meglévő profilkép kitölrése, és az újra való lecserélése. Elsősorban meg kellett vizsgálnom azt, hogy létezik-e már az adott user id-jával rendelkező fájlnevű .jpg kiterjesztésű fájl a profilképek almappában(ezt a file_exists() PHP függvénnyel vizsgáltam), és ha igen, akkor azt az unset() függvény segítségével töröltem, majd áthelyeztem az újonnan feltöltött képet az almappába a move_uploaded_file() segítségével, és végül átneveztem a megfelelő id-ra a rename() használatával. És ugyan ezt megtettem .png kiterjesztésre is, lásd 1.sz.kép. Összességében nem volt nehéz megoldani, de sokat kellett kísérletezni és tesztelni vele, hogy hibamentesen működjön.

1.sz.kép

```
if (in_array($imgType, $imgFormat) && $imgSize < 16000000){

    if (file_exists($profkepFolder.$id.".jpg")) { //vizsgálja, hogy van-e már adott id névvel .jpg fájl
        unlink($profkepFolder.$id.".jpg"); //kitörli ha van
        move_uploaded_file($imgTmpName, $profkepFolder.$imgName); //áthelyezi az újat
        rename($profkepFolder.$imgName, $profkepFolder.$id.".".$ext); //átnevezi a user id alapján
        header("Location: profil_modositasa.php");
    } else { //vizsgálja, hogy van-e már adott id névvel .png fájl
      unlink($profkepFolder.$id.".png");  //kitörli ha van
      move_uploaded_file($imgTmpName, $profkepFolder.$imgName); //áthelyezi az újat
      rename($profkepFolder.$imgName, $profkepFolder.$id.".".$ext); //átnevezi a user id alapján
      header("Location: profil_modositasa.php");
    }
```

A következő lépés a profilkép megjelenítése volt. Összesen 3 oldalon jelenítem meg, ezek a profil módosítása, a profil áttekintése, és a chat szoba. Mind a 3 oldalon ugyan azt a megoldást alkalmaztam, aminek a lényege az, hogy egy feltételvizsgálat megkeresi az adott felhasználó azonosítójával megegyező .jpg vagy .png fájlt, és azt elhelyezi egy változóban. Amennyiben pedig nem talál az adott id-val megyező képet – azaz a felhasználó még nem töltött fel saját képet – akkor a placeholder.jpg nevű, alapértelmezett profilképet jeleníti meg.

2.sz.kép

```
$profImg = "";
if(file_exists("../kepek/profkep/".$id.".jpg"))
{
  $profImg = '../kepek/profkep/'.$id.'.jpg';
} else if(file_exists("../kepek/profkep/".$id.".png"))
{
  $profImg = '../kepek/profkep/'.$id.'.png';
}
else {
  $profImg = '../kepek/profkep/placeholder.jpg';
}
```

### 1.2.3 Táblázatok

Mivel az adatbázis rendelkezik egy adadtokkal feltöltött gépek, és objektívek táblával, akkor érdemes lenne felhasználni a weboldalon. Elsősorban a képfeltöltés során használhatóak fel igazán hasznosan ezen adatok, de mindenképpen szerettem volna valamely más módon is felhasználni a weboldalon. Mivel tanórán tanultunk táblázat feltöltést adatbáziból, illetve lapozható, dinamikus táblázatot létrehozni, így az órán tanultakat felhasználtam a projektemben, és létrehoztam egy-egy külön táblázatot a fényképezőgépeknek és az objektíveknek is, külön oldalakra. Figyelmet fordítottam arra,

hogy az adatbázisból beolvasott adatok után megjelenítsem a mértékegységeket a megfelelő formában minden adatnál, illetve a logikai érték is szövegesen jelenjen meg, lásd 3.sz.kép.

3.sz.kép

| Gyújtótáv | Stabilizátor | Legszűkebb blende | Legnagyobb blende | Hossz | Súly |
|---|---|---|---|---|---|
| 60mm | Van | f/18 | f/0.8 | 80mm | 800g |
| 50mm | Nincs | f/16 | f/0.95 | 75mm | 700g |
| 85mm | Nincs | f/22 | f/4 | 95mm | 310g |

A lapozhatóság szintén PHP alapú. Az adatbázis lekérdezés eredményét sorokban kapja vissza a result változó, amelyből a ‚num_rows' függvény segítségével megkapjuk, hogy hány sorból állna a kiiratásunk. Ezt az értéket elosztjuk annyival, ahány sort szeretnénk látni oldalanként. Ezentúl definiálunk egy változót, amely értéke megegyezik majd az aktuális oldallal, ahova lapoztunk a táblázatban, és ezen értékek segítségével kiegészítjük a lekérdezésünket, lásd 4.sz.kép. Ezek után egy ciklus segtségével kiiratjuk a lapozást lehetővé tevő számokat, valamint alá magát a táblázatot.

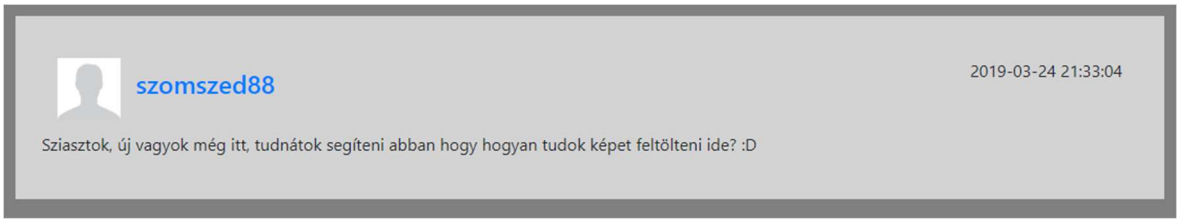
4.sz.kép

```php
$sql = "SELECT * FROM gepek";
      $res = $conn -> query($sql);
      $numRows = $res -> num_rows;

        if (isset($_GET['page'])){
            $page = $_GET['page'];
        } else {
            $page = 1;
        }

        $productNum = 10;
        $record = ($page-1) * 10;
        $sql .= " LIMIT $record, $productNum";
        $res = $conn -> query($sql);
```

### 1.2.4 CHAT SZOBA

Midenképpen szerettem volna valamilyen kommunikációs lehetőséget biztosítani a felhasználók számára. Tervben volt valamilyen féle forum, esetleg privát üzenetküldési lehetőség, vagy éppen a chat. Azért a chat-re esett a választásom, mert manapság egyre elterjedtebbek a valós idejű beszélgetések, jobban le tudja kötni az embert, ha társaságra, esetleg gyors segítségre, válaszra van szüksége. Továbbá a forum szerű oldalak kezdenek eltűnni, egyre kevésbé népszerűek és elterjedtek a felhasználók körében. A chat megtervezését statikusan, HTML oldalon kezdtem el. Fő szempontok az átláthatóság és az egyszerű kezelés volt, és hogy a fókusz magán a chat üzeneten legyen, körülötte csak a legfontosabb információkkal ellátva, ilyen a profilkép és felhasználónév párosa – hogy könnyen beazonosíthatóak legyenek később is a társalgó partnereink – valamint az üzenet elküldésének másodpercre pontos dátuma, lásd 5.sz.kép.

5.sz.kép




Fontosnak tartottam, hogy az üzenetek ne ömlesztve jelenjenek meg az egész oldalon, hanem körbefogtam egy div elemmel, mely nagyjából kihasználja az oldal éppen megjeleníthető részét, és azon belül egy vertikális görgetősávval lehet a chatben visszaolvasni, egész pontosan az utolsó 25 elküldött üzenet fog megjelenni. Ez alá egy beviteli mezőt pozícionáltam, melynek maximális hosszát 350 karakterben határoztam meg. Ide lehet írni az elküldeni kívánt üzenetet, majd a mellette lévő gombbal azt továbbítani a chat.php állománynak feldolgozásra, amely a felhasználó azonosítóját is megkapja ‚$_GET' superglobal használatával. Ezen értékeket feltölti a chat táblába, és onnantól meg fog jelenni a chat oldalon az éppen elküldött üzenet is. Ezen kívül a feldolgozó állományban még vizsgálom azt is, hogy nem üres-e a beviteli mező. Amennyiben üres, akkor nem történik meg az adatbázisba való feltöltés. Használom még a trim() nevű függvényt is, amely minden felesleges szóközt levesz a chat üzenet elejéről és végéről, így megakadályozva a felesleges helyfoglalást az adatbázisban. Miután elégedett voltam a chat HTML-alapú kinézetével, és megtörtént az első adatfelvitel, átalakítottam PHP-vel, és egy while() ciklus segítségével megjelenítettem az utolsó 25 elküldött üzenetet. Ezentúl megtörtént az adott felhasználó

profilképének és nevének a megjelenítése, amelyet kattintható hivatkozássá tettem. A hivatkozás segítségével megtekinthetjük a kiválasztott felhasználó minden adatát, feltöltött képeinek számát, valamint innen elérjük a felhasználó saját galériáját is.

### 1.2.5 KÉPFELTÖLTÉS

A weboldalam legfontosabb funkciója a fényképfeltöltés. Szerettem volna egy viszonylag komplex rendszert létrehozni, melynek alapja a fényképezőgépek és a feltöltött fotók közti kapcsolat lett. Először a képfeltöltés lehetőségét biztosító oldallal kezdtem. Formázáshoz az eddig már több helyen is felhasznált Bootstrap kártyát vettem igénybe, és a már megtervezett adatbázis alapján felraktam a beviteli mezőket, így az alap szöveges beviteli mezőkön túl még lett egy fájl tallózás, egy dátum mező, és két legördülő lista, a fényképezőgép kiválasztásához, lásd 6.sz.kép.

6.sz.kép


Ennek a háttere egy SQL lekérdezésen alapul, mely lekérdezi az összes fényképezőgép gyártót, és ezekkel töltöm fel az első legörülő listát. Amennyiben a felhasználó a “Válassz„ értéktől eltérő adatot választ ki, abban az esetben lefut egy jQuery kód, és átadja a kiválasztott gyártó nevét egy PHP állománynak, amely a megkapott érték segítségével lefuttatott lekérdezés eredményeképpen megkapja az adott gyártó összes fényképezőgépét. A lekérdezésből kapott értékeket még a PHP állományban a legördülő listához adom, majd ezt az új listát kiiratom az eredeti oldalon jQuery segítségével, valamint a PHP állományból kapott fényképezőgép sorozatszámot is eltárolom egy változóban, mivel még szükség lesz rá. Amennyiben a felhasználó kitöltötte a szükséges mezőket, onnantól elindul a feltöltés folyamata az upload.php állomány segítségével. Ennek az állománynak viszont át kell adni a legördülő menüből kiválasztott fényképezőgép azonosítóját is. A fényképezőgép márkája

azért nem elég, mert egy márkához több gép is kapcsolódik a táblában, így egy konkrét fényképezőgépet tartalmazó sor azonosítóját kell a feltöltő állomány rendelkezésére bocsátani, melyet egy külső, a márkához tartozó sorozatszámokat lekérdező PHP fájlban definiáltuk, de éppen azért tároltuk el jQuery-ben is ezt az értéket amikor a legördülő lista már feltöltve tért vissza a PHP állományból, hogy ezt az értéket át tudjuk majd adni a kép feltöltését végző állománynak. Ennek segítségére egy HTML attribútum kicserélésére vagy felülírására szolgáló kóddal a form feldolgozó állományának az elérési útvonalát manipuláltam, lásd 7.sz.kép, hozzáfűzve a kiválasztott fényképezőgép sorozatát.

7.sz.kép

```
$(document).on("change","#gyarto", function(){
let gyarto = $(this).val();
console.log(gyarto);

    if(gyarto != "alap"){
      $.get("select_sorozat.php?gyarto="+gyarto,function(kiir){
          $("#select").html(kiir);
      });
    }
  });

  $(document).on("change","#sorozat",function(){
    let sorozat = $(this).val();
    $("#form_action").attr("action", "upload.php?sorozat="+sorozat);

  });
```

A sorozatszámot – ami természetesen nem feltétlenül egy szám – átadva már le tudjuk kérdezni az azonosítóját a felhasználó által kiválasztott gépnek. Ezek után a form-ból bekérem a többi kitöltött beviteli mező értékét, és eltárolom ezeket egy változóban, csakúgy, mint a feltöltött fájl nevét. Mielőtt az értékeket feltöltöm az adatbázisba, ellenőrzöm, hogy a feltöltött fájl típusa vagy jpg/jpeg vagy png. Amennyiben ezek közül valamelyik, és a maximális méret korlátozásnak is megfelel, akkor a fájlnév alapján felviszem új sorként a képek táblába. Ezen túl a feltöltő Id-ját szintén eltárolva feltöltésre kerül további két érték a fotókészítés nevű táblába, mely feladata az, hogy beazonosítható legyen az adott fénykép feltöltője. Miután minden érték sikeresen eltárolásra került, onnantól a képek elérhetővé válnak lekérdezésre.

### 1.2.6 Galéria

Miután a fényképfeltöltés funkció hibátlanul működött, el kellett kezdenem megvalósítani a feltöltött fényképek megjelenítésére szolgáló Galéria nevű oldalt. Az elsődleges galéria, mely a legörülő menü első pontjára kattintva érhető el, funkciója szerint

az összes feltöltött képet megjelenítette volna, felhasználótól függetlenül. Ez viszont gondot okozhatott volna a későbbiekben, mikor már nagyon nagy mennyiségű képet kellene megjeleníteni. Ezért erre az oldalra írt SQL lekérdezésen belül lekorlátoztam a megjelenítendő képek számát 30-ra, és a sorrendiség a fénykép feltöltésének dátuma alapján történik, így az első fénykép mindig a legfrissebben feltötött kép lesz. A galéria megjelenítésre szintén a Bootstrap 4 lehetőségeit használtam ki, mely kellemesen kiemeli a fényképeket, felhasználóbarát és reszponzív. Az oldal letisztultságának megtartása érdekében a képeken kívül semmilyen információt nem jelenítek meg a Galéria oldalon. Helyette lehetővé tettem a felhasználó számára, hogy a képre kattintva eljusson a kép adatait tartalmazó oldalra. Mivel rengeteg adatot kell megjeleníteni, így sok értéket át kellett adnom az adatok lekérdezéséért felelős PHP állománynak, lásd 8.sz.kép.

8.sz.kép

```
echo '<div class="row mb-3">';
while($row = $res -> fetch_assoc()){
    echo '<div class="col-md-4">
    <div class="card photo">
    <a class="lightbox" href="kep_adatok.php?kepid='.$row["id"].'
        &gepid='.$row["gepid"].'&userid='.$row["userid"].'">

    <img src="../kepek/galeria/'.$row["fajlnev"].'"
        alt="'.$row["cim"].'" class="card-img-top">
    </a>
    </div>
    </div>';
}
echo '</div>';
```

Ezen átadott értékek segítségével fogjuk tudni meghatározni, hogy melyik képre kattintott a felhasználó, hogy azt a képet ki töltötte fel, valamint a kiválasztott képhez tartozó fényképezőgépet is. Miután sikeres az átirányítás a feldolgozást végző PHP fájlra, az átadott értékek segítségével lekérdezést tudunk végezni a fényképezőgép pontos meghatározására, a felhasználó pontos kilétére, valamint a kép minden adatára. Ilyen adatok a kép címe, leírása, készítésének a helye és a dátuma. Erről az oldalról a feltöltő felhasználónevére kattintva eljuthatunk a felhasználó profiljának áttekintésére, ahonnan elérhetkjük az összes feltöltött fényképét.

A kiválasztott felhasználó képei megjelenítésére is ugyan azt a megjelenítési módszert használtam, mint amit a fő galériánál. A különbség annyi, hogy az SQL lekérdezésben csak a kiválasztott felhasználó által feltöltött képeket kérem le, és jelenítem meg a weboldalon. A menüsávon található legördülő Galéria menü 2. pontja a saját képek, mely szintén az eddig felvázolt galériákon alapul, megintcsak annyi különbséggel, hogy az adatbázis lekérdezésben csak a bejelentkezett felhasználó által feltölött képek jelennek meg, de ezek nincsenek lekorlátozva egy maximális számban úgy, mint a fő, közös galériában.

Miután az utolsó funkció is megfelelően működött, a következő lépés az oldal finomítás volt. Ilyen szempontból a főoldallal foglalkoztam a legtöbbet, hiszen az teljesen statikus volt, szimpla HTML. A főoldal közepén lévő, képeket lapozató elem is teljesen HTML alapon olvasott be a kódban meghatározott fényképeket. Ezt szerettem volna úgy átalakítani, hogy a megjelenített képek szintén az adatbázisból legyenek lekérve, természetesen korlátozva egy bizonyos mennyiségre. De hogy ne legyen annyira galéria-szerű, így egy kicsit többet is meg szerettem volna jeleníteni, mint csak magát a fényképet. Ennek segítségéül a Bootstrap 4 „Carousel" lapozható galériaelemet használtam fel, mely a képmegjelenítésen kívül lehetőséget ad arra is, hogy pár sornyi adatot is megjelenítsek. Ezen adatsorok kihasználására a fényképhez tartozó címet, valamint a készítő felhasználónevét is megjelenítettem. Ahhoz, hogy ez működjön, megint PHP-t hívtam segítségül. A PHP kódban lekérdeztem az 5db utoljára feltöltött kép nevét, és az ahhoz tartozó adatok egy részét. A lekérdezésben (lásd 9.sz.kép) a képek táblát összekapcsoltam a fotókészítés nevű táblával, melynek segítségével beazonosítható a feltöltő ember felhasználóneve is.

9.sz.kép

```
$sql="SELECT user.username, kepek.fajlnev, kepek.cim
FROM kepek
INNER JOIN fotokeszites ON kepek.id = fotokeszites.kepid
INNER JOIN user ON fotokeszites.userid = user.id
ORDER BY feltolt_datum DESC
LIMIT 5;";
$res = $conn -> query($sql);
```

A lekérdezés tehát visszaadott 5 sort, soronként pedig 3 értéket, az egyik a fájl neve, a másik a feltöltő felhasználóneve, valamint a képhez tartozó cím.

A megjelenítés kicsit több tervezést igényelt. A „Carousel” képnézegető elem kódja megköveteli, hogy az első megjelenített kép class tulajdonságában szerepeljen az “active„ osztály. Mivel az adatbázisból bekért képeket egy while ciklussal jelenítem meg, így a megjelenítendő képek számától függetlenül csak egy darab „Carousel” elemet kell a kódnak tartalmaznia. Viszont ha ebbe az egyszeri kódba felveszem az “active„ osztályt, akkor minden képhez megkapja azt, és a „Carousel” elem nem fog megfelelően működni. Amennyiben nem veszem fel ezt az osztályt, abban az esetben egyetlen kép sem kapja meg, és megint nem fog megfelelően funkcionálni a képlapozó elem. Erre megoldásképpen felvettem egy counter nevű változót, mely minden ciklus végén növekszik eggyel, így meg lehet határozni, hogy éppen hanyadik kép megjelenítésénél tart a ciklus. A számlálót felhasználva egy feltételvizsgálat segítségével meg lehet oldani, hogy az első megjelenített kép osztályához az “active„ osztályt is hozzárendeljük, lásd 10.sz.kép. A while ciklus többi részében a képek és a szöveg megjelenítéséért felelős kódot találhatjuk, melyben az SQL lekérdezés eredményeképp kapott adatokat kiíratva elkészül a végleges, már adatbázison alapuló lapozható képnézegető.

10.sz.kép

```
$counter = 1;
while($row = $res -> fetch_assoc()){
    echo'<div class="carousel-item ';
    if($counter <= 1){
      echo "active"; //hozzáadjuk a class-hez az active értéket
      }
      echo '"><img src="../kepek/galeria/'.$row['fajlnev'].'" alt="'.$row["cim"].'">
    <div class="carousel-caption d-none d-md-block">
    <h4>'.$row["cim"].'</h4>
    <p>'.$row["username"].' fényképe</p>
    </div>
    </div>';
    $counter++;
  }
```

Ezzel a weboldalam teljessé vált, de további fejlesztési terveim természetesen még vannak.

## 1.3 ASZTALI ALKALMAZÁS

Az asztali alkalmazás megtervezésénél figyelembe vettem, hogy milyen funkciókra lesz használva, milyen feladatokat hivatott ellátni. Mivel egy fotós számára az általam tervezett funkciók teljes mértékben elérhetőek a weboldalról, így az asztali alkalmás feladata az adminok feladatának kisegítése lett.

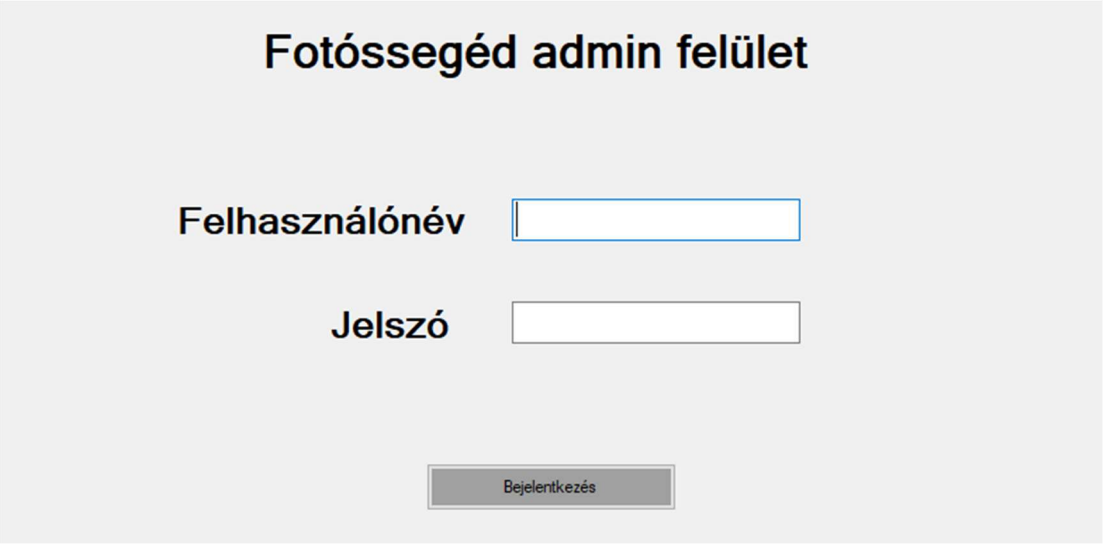
### 1.3.1 TERVEZÉS, BIZTONSÁG

A program az iskolában használt Visual Studio környezetben készült, C# nyelven. A programot úgy terveztem, hogy az egyszerűvé tegye az adminok számára az adatbázisban található adatok kezelését, legyen az adatfelvitel, adatmódosítás, vagy éppen törlés. Ezen túl maga az adatmegjelenítés is fontos, melyre az órák során is használt DataGirdView nevű táblás megjelenítési módszer használtam. Mivel több DataGridView-t is meg kellett jelenítenem, emiatt az adatkezelési formot is több oldalra kellett osztanom. Erre a TabControl nevű, több oldal kezelésére lehetőséget adó, beépített funkciót használtam. Mivel ezzel az alkalmazással tulajdonképpen bármilyen adatot módosítani lehet, emiatt figyeltem a kellő biztonságra is. Regisztrálni csak a weboldalon keresztül lehet, ahonnan a jelszó a mai biztonsági követelményeknek megfelelően titkosítva kerül eltárolásra az adatbázisban. De mivel az asztali alkalmazás lehetőséget ad adatok módosítására, vagy akár kényes adatok eltulajdonítására, így csak olyan felhasználók léphetnek be, akiknek admin jogosultságot adott a fő adminisztrátor, jelen esetben ez én vagyok. Admin jogosultág pedig csak ebből az alkalmazásból adható bármely regisztrált felhasználónak.

### 1.3.2 BEJELENTKEZÉS

A bejelentkező felületet teljesen letisztultra terveztem, két beviteli mezővel, és egy belépés gombbal, lásd 11.sz.kép. Amennyiben a felhasználó valamelyik mezőt rosszul töltötte ki, abban az esetben egy felugró ablak jelzi felé, hogy a felhasználó neve, vagy éppen a jelszava helytelenül lett megadva.

11.sz.kép




Mivel az adatbázisban a jelszavakat titkosítva tárolom el, ezért a bejelentkezés sikeres létrejöttéhez szükséges lenne összehasonlítani a beviteli mezőbe beírt jelszót az adatbázisban eltárolttal. Ennek a segítségére használtam a „**CryptSharpOfficial**” nevű Visual Studio bővítményt, melyet a **tools** lenyíló menüben a **NuGet Package Manager** pontban, azon belül a **Manage NuGet Packages for Solution pontban** lehet a projekthez rendelni. Ez a bővítmény képes jelszó titkosításra, valamint adatbázisból bekért jelszó és egy változóban eltárolt érték összehasonítására, lásd 12.sz.kép.

12.sz.kép

```
string username = textBoxUsername.Text;
string password = textBoxPassword.Text;

userDT = mdi.getToDataTable("SELECT username,password FROM user WHERE username = '" + username + "'");
if (userDT.Rows.Count != 0)
{
    foreach (DataRow row in userDT.Rows)
    {
        if (Crypter.CheckPassword(password, row["password"].ToString()))
        {
            //sikeres belépés
            this.Hide();
            FormGepek fg = new FormGepek();//átvezetjük a felhasználót a FormGepek-re
            fg.ShowDialog();
            this.Close();
```

### 1.3.2 Funkciók tervezése

A programot az iskolában tanultak alapján Objektum Orientáltan készítettem, MVC alapokra helyezve. Ennek megfelelően külön mappákba rendeztem minden osztályt, feladatától függően. Az osztálydiagram (lásd 13.sz.kép a mellékletekben) segítségével

könnyen átláttam, melyik osztályban milyen változók, illetve milyen metódusok találhatóak. Csakúgy, mint a bejelentkezésnél, itt is a letisztultságra és az áttekinthetőségre törekedtem. A színek megválasztásánál arra is figyeltem, hogy azok hasonlóak legyenek, mint a weboldalamon. Az alkalmazásba 2 fő funkciót terveztem. Az egyik az adatbázisban található adatok kezelése, a másik pedig statisztikák mutatása. Az adatbázis-kezelést 3 táblámra tettem lehetővé, az első a fényképezőgépek, a második az objektívek, a harmadik pedig a felhasználók tábla. Az adatbázisból való bekérés csak akkor sikeres, ha van kapcsolat az magával az adatbázissal. A kapcsolat megteremtésére létrehoztam egy külön osztályt, hogy az bármikor könnyen konfigurálható legyen. Ezen túl még felhasználtam az órai munkák során szintén felhasznált MySQLDatabaseInterface nevű fájlt, mely az adatbázis műveletek elvégzésétben segít, ilyenek például a módosítások egy adott sorban, vagy éppen egy komplett sor törlése.

### 1.3.3 GÉPEK KEZELÉSE

A gépek adatainak eltárolása a gepek nevű osztályban történik. Az osztály rendelkezik egy konstruktorral (lásd 14.sz.kép), mely tartalmaz minden adatot. Ezen kívül minden változóra írtam Get és Set metódust is, valamint egy override-ot is, mely visszatér a gépek összes adatával.

14.sz.kép

```
public Gepek(int id, string gyarto, string sorozat, string tipus, double pixel, string szenzor, string objektiv, int ar)
{
    this.id = id;
    this.gyarto = gyarto;
    this.sorozat = sorozat;
    this.tipus = tipus;
    this.pixel = pixel;
    this.szenzor = szenzor;
    this.objektiv = objektiv;
    this.ar = ar;
}
```

A gépek megjelenítéséért egy DataGridView felelős. A TabControl ezen oldalán megtalálható minden szükséges gomb, ami az adatok betöltéséhez, azok módosításához vagy éppen törléséhez kellhetnek. A formon megtalálható gombok, feliratok és beviteli mezők természetesen nem jelennek meg egyszerre, csakis akkor, amikor a felhasználó meghívja azt a funkciót, ami szükségessé teszi megjelenítésüket. Például amint a felhasználó az új gép felvitele gombra kattint, akkor megjelennek az adatok felviteléhez szükséges üres bemeneti mezők, egy gomb, amellyel az adatok rögzítésre kerülhetnek a táblában, valamint egy gomb, amely lehetőséget ad a felhasználónak kilépni az új adat felviteli módból. Ezeket a vezérlő beállításokat szemlélteti a 15.sz.kép.

15.sz.kép

```
/// <summary> Program indulása után csak a betöltés gomb legyen látható
private void beallitVezerloketIndulaskor()...

/// <summary> Adatok betöltése után csak a módosít gomb legyen látható, a DGV le ...
private void beallitVezerloketBetolteskor()...

/// <summary> Módosításra megjeleníti a gombokat
private void beallitVezerloketModositaskor()...

/// <summary> Megjeleníti a beviteli mezőket
private void beallitVezerloketUjAdatFelvitelhez()...
```

Mint látható, több ilyen eseményre is létrehoztam ezeket a metódusokat, és mindegyik felel valamilyen állapotért, legyen az adatbetöltés előtti, adatbetöltés utáni, módosításra felkészített, vagy új adat felvitelére szolgáló állapot. Amennyiben a felhasználó módosítás módban van, és duplán kattint bármelyik cellára a DataGridView-n, akkor az abba történt módosítások mentésre kerülnek a programban definiált DataTable-ben, de az adatbázisban csak akkor módosulnak, ha a felhasználó a mentés gombot is megnyomta, így kiszűrve annak a lehetőségét, hogy nem kívánt módosítás történik az adatbázisban. Ilyenkor a mégsem gombra kattintva visszaáll az eredeti állapot. Egy sor törlésénél pedig egy felugró ablak fogja figyelmeztetni a felhasználót arra, hogy éppen véglegesen törölni akar egy sort. Miután megtörtént a törlés a DataTable-ből, még mindig lehetősége van a felhasználónak visszavonnia a műveletet a mégsem gomb segítségével, ugyanis itt is a végleges adatbázis művelet csak a mentés gombra kattintva fog végbemenni. Ahhoz, hogy meghatározzam, történt-e változtatás a DataGridView-ban, a DataGridView-hoz tartozó események közül legeneráltam a CellValueChanged eventet (lásd 16.sz.kép), melybe egy korábban már deklarált logikai változó értékét igazra állítom, így meg tudom állapítani, hogy történt-e változtatás.

16.sz.kép

```
private void dataGridViewGepek_CellValueChanged(object sender, DataGridViewCellEventArgs e)
{
    modositva_gep = true;
}
```

Új adat felvitelénél a beviteli mezők értékét kiovassa a program, és eltárolja őket egy-egy változóban. Ezek után ezeket a változókat felhasználva feltöltöm az új gép adatait az adatbáztisba INSERT INTO utasítás segítségével, lásd 17.sz.kép.

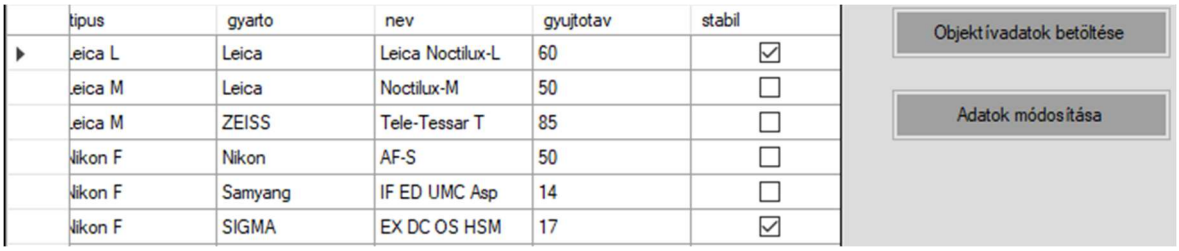
17.sz.kép

```
Database update = new Database();
mdi = update.kapcsolodas();
mdi.open(); string query = "";
query += "INSERT INTO gepek (id,gyarto,sorozat,tipus,pixel,szenzor,objektiv,ar) VALUES ";
query += "(" + ujId + ", ";
query += "\"" + gyarto + "\", ";
query += "\"" + sorozat + "\", ";
query += "\"" + tipus + "\", ";
query += pixel + ", ";
query += "\"" + szenzor + "\", ";
query += "\"" + objektiv + "\", ";
query += ar + ")";
mdi.executeDMQuery(query);
mdi.close();
```

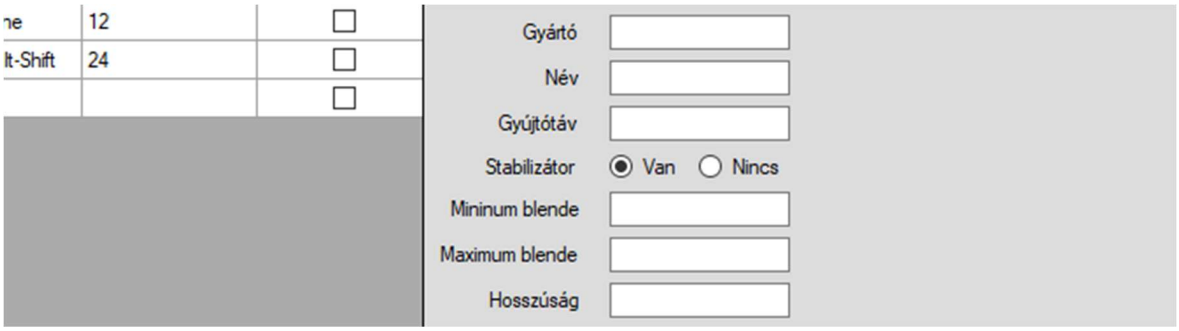
### 1.3.4 OBJEKTÍVEK KEZELÉSE

Az objektívek kezelése ugyan úgy működik, mint a fényképezőgépeké, egyedül az adatok, és azok típusai térnek el. Itt például megtalálható egy boolean, azaz logikai típusú változó, melynek értéke 0, azaz hamis, vagy 1, azaz igaz lehet. Ezt a DataGridView CheckBox formájában jeleníti meg, lásd 18.sz.kép, az adatfelvitelnél pedig RadioButton használatával határozhatja meg a felhasználó, hogy az adott objektív rendelkezik-e beépített stabilizátorral, lásd 19.sz.kép.

18.sz.kép




19.sz.kép



### 1.3.5 Felhasználók kezelése

A felhasználók kezelése annyiban különbözik a gépek vagy az objektívek kezelésétől, hogy itt nem lehet új sort hozzáadni az adatbázishoz, azaz nem lehet új felhasználót regisztrálni. Viszont adatot módosítani lehet, ami esedékes lehet akkor, ha valakinek admin jogot szeretnénk adni, valamint lehet felhasználót törölni az adatbázisból.

### 1.3.6 Statisztikák

A statisztikák olyan adatok, melyekkel nem árt tisztában lenni, főleg az alkalmazás kezelőinek. Csak a legfontosabb statisztikákat szeretném megjeleníteni az adminok számára, ilyen adatok például a regisztrált felhasználók száma. Minden adat SQL lekérdezések segítségével történik a Repository mappában tárolt StatisztikakOperations nevű osztályban. Ebben az osztályban minden metódus egy-egy lekérdezést foglal magában, lásd 20.sz.kép, és visszatérési értéke adja át a formnak a lekérdezés eredményét, mely kiíratásra kerül. Minden visszatérési értékem szöveg típusú, elkerülve ezzel a változók típusának konvertálását, mely további kódsorokat igényelt volna. Amennyiben a felhasználók száma gyorsan növekszik, akkor az oldal nagy népszerűségnek örvend. Ennek ellentétére utalhat az, ha régen történt már az utolsó képfeltöltés, ezt az utoljára feltöltött kép statisztikával lehet ellenőrizni. Ezeken kívül megjelenik még a feltöltött képek száma és az utolsó chat aktivitás dátuma is.

20.sz.kép

```
public string getUserNumber(string userNumber)
{
    Database md = new Database();
    mdi = md.kapcsolodas();
    mdi.open();
    string query = "SELECT COUNT(id) FROM user;";
    userNumber = mdi.executeScalarQuery(query);

    return userNumber;
}
```

### 1.3.7 Tesztelés

Az egységtesztek írása rendkívül fontos, hiszen ezek nélkül nem mehetünk biztosra abban, hogy az értékek, melyeket visszakapunk valóban helyesek-e. Ennek a megállapítására írtam egy egységtesztet például a statiszikákra, azon belül is a regisztrált felhasználók számára, lásd 21.sz.kép.

21.sz.kép

```
[TestMethod()]
public void getUserNumberTest()
{
    StatisztikakOperations so = new StatisztikakOperations();
    string actual = so.getUserNumber("0"); //Az adatbázisból lekért felhasználók száma
    string expected = "5"; //A valós eredmény, amit elvárunk
    Assert.AreEqual(expected, actual, "Felhasználók számára nem jó eredményt ad!");
}
```

Ehhez hasonló módon szintén írtam egy tesztet az adatbáziban eltárolt képek mennyiségére (lásd 22.sz.kép). A képeket manuálisan megszámoltam, és ezt a számot adtam meg az elvárt értéket tartalmazó változónak, csakúgy, mint minden tesztben az megszokott.

22.sz.kép

```
[TestMethod()]
public void getKepekNumber()
{
    StatisztikakOperations so = new StatisztikakOperations();
    string actual = so.getKepekNumber("0"); //Az adatbázisból lekért fényképek száma
    string expected = "9"; //A valós eredmény, amit elvárunk
    Assert.AreEqual(expected, actual, "Fényképek számára nem jó eredményt ad!");
}
```

Amennyiben a tesztek hiba nélkül lefutnak, azt a program zöld jelzéssel tudatja velünk, lásd 23.sz.kép.

23.sz.kép

# 2. FELHASZNÁLÓI DOKUMENTÁCIÓ

## 2.1 TELEPÍTÉSI ÚTMUTATÓ

### 2.1.1 ASZTALI ALKALMAZÁS TELEPÍTÉSE

Az asztali alkalmazás elindításához futtassa le a FotossegedAdminApp.exe nevű fájlt, mely alapértelmezetten a projekt mappáján belüli bin, és az azon belüli Debug mappában található. Telepítés nem szükséges. Az alkalmazásba való belépésre alapértelmezetten csak admin jogokkal rendelkező felhasználó léphetne be, de tesztelési célzattal egyelőre bármely felhasználóval lehetésges a belépés. Példa adatok az alkalmazás kipróbálásához:

- Felhasználónév: admin
- Jelszó: Abc123@

Az alkalmazás lefuttatásával egy időben már működnie kell az Apache és a MySQL szervereknek. Az adatbázis kapcsolódásáért felelős fájl konfigurálása a Database nevű mappán belül található, Database.cs néven. Ez bármikor átírható arra az adatbázisra, amelyikre éppen kapcsolódni szeretnénk.

### 2.1.2 WEBES ALKALMAZÁS HASZNÁLATA

Első lépésként indítsa el az Apache kiszolgálót és a MySQL szervert. Egy tetszőleges böngészőt kiválasztva vagy közvetlenül, vagy a localhost/ cím segítségével nyissuk meg a weboldal gyökérmappáján belül elhelyezkedő php mappát, és azon belül található egy index.php. Ez a főoldal, ahonnan a navigációs sáv segítségével eljuthatunk a regisztrációs fülre. Miután regisztráltunk, az összes funkció elérhetővé válik. Természetesen tesztelési célokkal ide is be lehet lépni a fentebb megadott felhasználói adatokkal. Az adatbázis konfigurálása a config mappán belül elhelyezkedő connect.php fájl tetszőleges átírásával történhet. Fontos, hogy az Apache kiszolgáló konfigurációjánál található PHP.ini fájlban átírjuk a feltöltésre vonatkozó maximális megengedett fájlméretet 16 megabyte-ra, lásd 24.sz.kép.

24.sz.kép

```
; Maximum allowed size for uploaded files.
; http://php.net/upload-max-filesize
upload_max_filesize=16M
```

### 2.1.3 Adatbázis telepítése

Az adatbázis egyszerűen telepíthető a localhost/phpmyadmin címen elérhető adatbáziskezelő rendszer segítségével, amennyiben előzetesen elindítottuk a MySQL szervert az XAMPP segítségével. Amennyiben az oldal megnyílt, navigáljunk a felső sávon található importálás menüpontra, ahol a Fájl kiválasztása gombra kattintva betallózhatjuk a fotosseged.sql nevű fájlt, amely automatikusan létrehozza az adatbázist, benne a táblákkal és minden adattal.

## 2.2 Asztali alkalmás példa funkció

### 2.2.1 Új objektívadatok felvitele

Miután belépett az alkalmazásba, navigáljon a TabControl segítségével az objektívek menüpontra. Itt kattintson az objektívadatok betöltése gombra, hogy megjelenjen az adatbázisból feltöltött DataGridView. Ezután kattintson az adatok módosítása gombra, mely engedélyezni fogja az adatmódosítási módot. Miután megjelentek az új vezérlőelemek, kattintson az Új objektívek felvitele feliratú gombra, és a megjelent beviteli mezőket töltse ki az adott tulajdonsághoz hű értékekkel, például a névhez szöveg, a gyújtótávhoz egyszerű szám, a maximum vagy minimum blendéhez pedig lebegőpontos szám kerüljön. Amennyiben minden mezőt kitöltött, kattintson a Listához adás gombra. Ekkor a beviteli mezőkbe beírt értékek kitörlődnek, ezzel lehetőséget adva a gyors adatfelvitelre. Amennyiben viszont nem szeretne több adatot felvinni, akkor a vissza gombra kattintva kiléphet ebből a módból.

## 2.3 Weboldali funkciók

### 2.3.1 Fényképfeltöltés

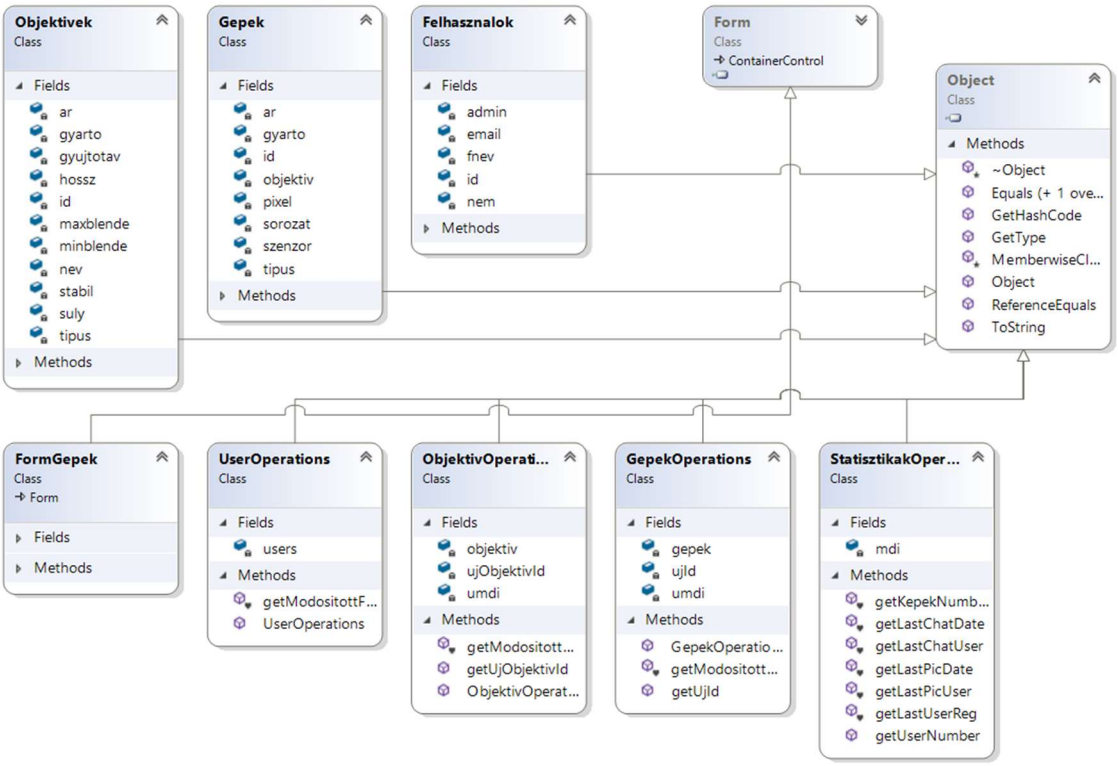
Belépés után a navigációs sávon nyissa le a fénykép menüpontot, és válassza ki a fénykép feltöltése opciót. Először a Browse gombra, vagy a tallózás feliratra kattintva válassza ki a feltöltendő képet. Ez lehet jpeg vagy png formátumú, maximum 16 megabyte méretű kép. Adjon a képnek egy rövid, frappáns címet. A legördülő listából válassza ki az ön által használt fényképezőgép gyártóját, majd a második legördülő listából a fényképezőgép pontos típusát. Töltse ki a maradék néhány mezőt, és ha mindennel kész van, kattintson a feltöltés gombra. Feltöltött képének meg kell jelennie mind a galéria, mind a saját képek oldalon.

# 3. Összegzés

Az eredetileg tervezett funkciók túlnyomó részét sikeresen megvalósítottam, voltak problémák melyek segítséget igényeltek, de örömmel tölt el, hogy rengeteg funckiót egyedül, az iskolában elsajátított tudás alapján meg tudtam oldani. Úgy gondolom, hogy az asztali és a webes alaklmazásom az adatbázissal együtt összeállt egy nagy komplex alkalmazássá, melyek együttes használata zökkenőmentesen kivitelezhető, egymás működéséhez nélkülözhetetlenek. Néha felmerült egy-két probléma, de azokat kitartó munkával sikerült megoldani, ilyen volt például az adatbázis háttérrel nem rendelkező profilkép feltöltési rendszer. A jövőben szeretném a szoftverből kihozni a lehető legtöbbet, így további funkciók beépítését is tervezem. Az egyik legfontosabb funkció, amely kifejlesztésére már nem jutott idő, a feltöltött képek törlése. Ez egy egyszerűen megoldható feladat, mégis sokat hozzáadna a szoftver minőségéhez. Ezentúl az asztali alkalmazás is rengeteg lehetőséget tartalmaz, de ahhoz további tapasztalatszerzésre lesz szükségem. Ettől függetlenül rendkívül sokat tanultam a dolgozatom fejlesztése során, sok új ismeretet szereztem és új problémamegoldási opciókra leltem.

# 4. MELLÉKLETEK

13.sz.kép



# 5. FORRÁSMEGJELÖLÉSEK

- https://stackoverflow.com/
- https://getbootstrap.com/
- https://jquery.com/
- https://www.php.net/
- https://www.w3schools.com/
- Órákon, tanárok segítségével írt kódsorok
- Órai munkák anyagai
- MySqlDatabaseInterface

Profilképként felhasznált képek:

- https://pixabay.com/photos/man-face-wet-male-head-person-945482/
- https://pixabay.com/photos/cat-animals-cats-portrait-of-cat-778315/
- https://pixabay.com/vectors/no-symbol-prohibition-sign-39767/

Az alkalmazásban minden más felhasznált kép saját magam által készített fénykép, minden szerzői jog a tulajdonomban áll.

# 6. PLÁGIUMNYILATKOZAT

TANULÓI NYILATKOZAT

Alulírott Marosi Márk Dániel, Szegedi Szakképzési Centrum Vasvári Pál Gazdasági és Informatikai Szakgimnáziuma tanulója kijelentem, hogy a „Fotós segéd” című záródolgozat a saját munkám.

Kelt:

_____

aláírás