**SZEGEDI SZAKKÉPZÉSI CENTRUM**

**VASVÁRI PÁL GAZDASÁGI ÉS INFORMATIKAI**

**SZAKGIMNÁZIUMA**

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

**„TraDem”**

**Használt áru kereskedelmi**

**alkalmazása**

**Készítette:**

**Kovács Péter**

**Szeged**

**2020**

Tartalom

[1.Bevezetés 3](#_Toc36996597)

[2. Áttekintés 4](#_Toc36996598)

[2.1. Célszoftver meghatározása 4](#_Toc36996599)

[2.2. Asztali alkalmazás 4](#_Toc36996600)

[2.3 Web Alkalmazás 4](#_Toc36996601)

[2.4 Adatbázis 5](#_Toc36996602)

[2.5 Megjelenítési tervek 5](#_Toc36996603)

[2.5.1 WEB 5](#_Toc36996604)

[2.5.2 Asztali alkalmazás. 6](#_Toc36996605)

[3. Adatbázis tervezése és megvalósítása 7](#_Toc36996606)

[3.1. Adatbázis tervezés 7](#_Toc36996607)

[3.2. „TraDem” Adatbázis Táblái 8](#_Toc36996608)

[3.3 E-K DIAGRAM 10](#_Toc36996609)

[4. Webalkalmazás megvalósítása 12](#_Toc36996610)

[4.2 Webes alkalmazás fájl állománya 13](#_Toc36996611)

[4.3. Webalkalmazás funkciói, és adatok 17](#_Toc36996612)

[4.3.1 Szerver oldalon 17](#_Toc36996613)

[5. Asztali alkalmazás megvalósítása 18](#_Toc36996614)

[5.1 Az asztali alkalmazás fájl állományai 19](#_Toc36996615)

[5.2 Models 19](#_Toc36996616)

[5.3 Repo(Repository réteg) 20](#_Toc36996617)

[5.4. Presenter (presenter réteg) 21](#_Toc36996618)

[5.5 Interface 22](#_Toc36996619)

[5.6 View 22](#_Toc36996620)

[5.7 Asztali alkalmazás tesztesetek 23](#_Toc36996621)

[6. Felhasználói dokumentáció 23](#_Toc36996622)

[6.1 Bevezetés 23](#_Toc36996623)

[6.2 Asztali alkalmazás 24](#_Toc36996624)

[6.2.1 Asztali alkalmazás minimum / ajánlott hardwer követelményei 24](#_Toc36996625)

[6.3 Asztali alkalmazás használati utasítás 24](#_Toc36996626)

[6.4. Webes alkalmazás 25](#_Toc36996627)

[7. Használt programozási nyelvek 28](#_Toc36996628)

[7.1. Asztali alkalmazás 28](#_Toc36996629)

[7.2 Webalkalmazáshoz használt nyelvek 29](#_Toc36996630)

[8. Fejlesztői környezet 32](#_Toc36996631)

[11. Melléklet 35](#_Toc36996632)

# 1.Bevezetés

Záródolgozatom központi témája egy kereskedelmi szoftver létrehozása, amely elsősorban már a hétköznapokban nem használatos, de nem szemét használt termékek kereskedelmét segíti elő.

A téma választása mögött több indok is van, elsősorban egy üresen álló pozíció kitöltése az interneten. Mindenki találkozott már olyan weboldalakkal amelyek használt termékek hirdetésével foglalkoznak, legyen szó járművekről, vagy háztartási cikkekről, vagy bármilyen kategóriáról, de egyik weboldal se nyújt konkrét lehetőséget ezeknek a tárgyaknak nem fizetőezköz alapú cseréjéhez.

A „TraDem” pont ezt célt szolgálja, a szoftvercsomag tartalmaz egy webes alkalmazást, a célközönségnek használatra, egy asztali alkalmazást, amellyel adminisztrátori feladatokat láthatunk el, kezelhetjük a felhasználók által megtett hirdetéseket, és magukat a felhasználói fiókokat is egyben, és végül egy összetett, kapcsolatokkal ellátott adatbázist, amely a felhasználók, és hirdetések információit tartalmazza.

Másodszor úgy vélem nem kell belemélyednem modern korunk egy kiemelkedő problémájába, a környezetszennyezésbe ,így is mindenki tudja hogy mekkora gonddal nézünk szembe. A folyamatosan növekvő népességszám magával járó következménye ez, amely ellen csak kis lépésekben tudunk fellépni. Részben egy ilyen fellépésnek tekintem szoftveremet, amely lényegében mint egy újra hasznosító közeg működne azáltal hogy a már nem használatos tárgyainkat nem azonnal a szemétbe, hanem valaki máshoz juttassuk, úgy hogy egyben mi is „megmentsünk” egy hasonló kaliberű tárgyat a kidobástól.

Célom ugyan az ilyen típusú használt áru kereskedelem támogatása, de úgy vélem teljes mértékben ez nem kivitelezhető, így a szoftver egyben támogatni fogja a hagyományos, fizető eszköz alapú használt termék árusítást is.

# 2. Áttekintés

## 2.1. Célszoftver meghatározása

Ahogy azt az imént már megírtam, a célszoftver („Tradem”) egy használtáru csere alapú kereskedelmével foglalkozó szoftver. Ezt egy asztali alkalmazás, egy webes alkalmazás, és egy az előbbiek által használt adatbázis létrehozásával hoztam létre. Ezek mind saját funkcióval rendelkeznek, saját feladattárral, amit az alábbiakban részletesebben is kifejtek.

## 2.2. Asztali alkalmazás

A szoftver asztali része a regisztrált felhasználók, és hirdetéseik megtekintésére, és kezelésére készült. Ezt kizárólag adminisztrátori használatra készült, átlagos felhasználók számára nem nyújt belépési lehetőséget. Pont ezen biztonsági okokból az adminisztrátori adatok nem is adatbázisban kerültek tárolásra, a szoftver maga tartalmazza őket.

A szoftver alapvető funkciói röviden:

* Az adatbázis adatainak szerkesztése, törlése grafikus felületen

-Az adatbázisban szereplő felhasználók adatainak módosítása

-Az adatbázisban szereplő felhasználók törlése

-Igényre új felhasználó hozzáadása az adatbázishoz

-Az adatbázis szereplő hirdetések módosítása

-Az adatbázisban szereplő hirdetések törlése

* Az adatbázisban szereplő adatok megtekintése DataGridView megjelenítő eszköz segítségével.

## 2.3 Web Alkalmazás

A szoftver számottevő részét ez alkotja. Lényege az átlag felhasználók számára egy olyan felület garantálása, ahol használtáru hirdetés lehetséges. Regisztráció nélkül is használható, de a termék teljes használatához szükség van rá.

A szoftver alapvető funkciói röviden:

* Hirdetés feladása

-A hirdetés adatbázisban való tárolása

-Képek csatolása a hirdetéshez

-Regionális adatok automatikus felhasználása a felhasználói információkból a hirdetés besorolásához

-

* Más felhasználók által hirdetett termékek megtekintése
* Más felhasználók által hirdetett termékek szűrése
* Saját hirdetett termék megtekintése/szerkesztése
* Saját profil megtekintése/szerkesztése
* Regisztráció a szoftver teljes funkcióinak elérése érdekében

## 2.4 Adatbázis

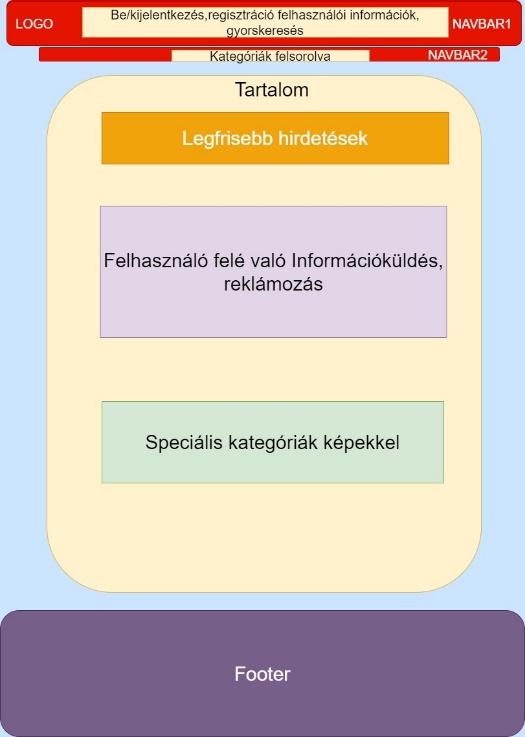
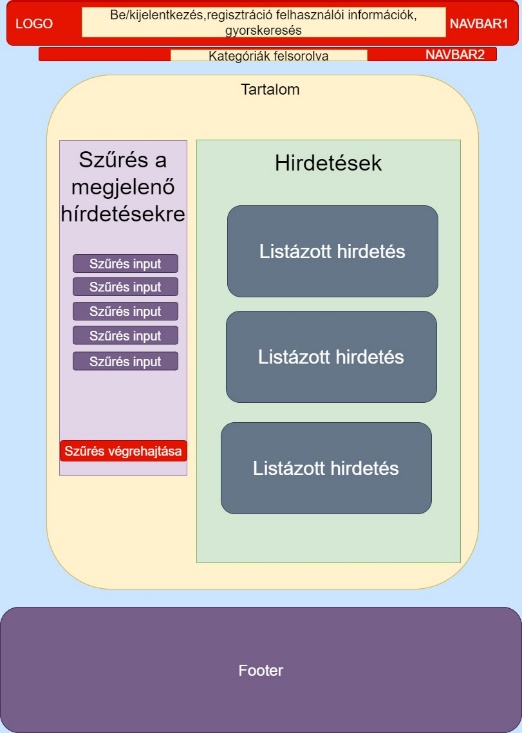
A szoftver által tárolandó felhasználói és termék információk raktározására lett létrehozva. Az adatbázis táblái mind kulcsokkal vannak ellátva, logikus relációkkal kiépítve. Tartalmazza még Magyarország településeinek, és megyéinek listáját, mely a hirdetések szűrésének egyszerűsítéséért lett hozzáadva. Közvetlen hozzáférés csak adminisztrátori jogosultsággal van hozzá.

## 2.5 Megjelenítési tervek

### 2.5.1 WEB

A weboldalak megjelenítésénél igyekeztem egy már megszokok egyszerű stílusban visszaadni a használt termékeket árusító oldalak stílusát. Az elemek helyezkedése a nézőszög középpontjában van, nincsenek túlzsúfolva. A design tervet indokolta az egyszerű és gyors weboldal használat. Főbb megvalósítandó weboldalrészek:

1. Kezdőlap
2. Egy oldal a hirdetések megjelenítésére, és szűrésére
3. Egy oldal a saját profil megtekintésére, és szerkesztésére
4. Felhasználó kezelő oldalak(login,registration)
5. Kapcsolat, webalkalmazás információk

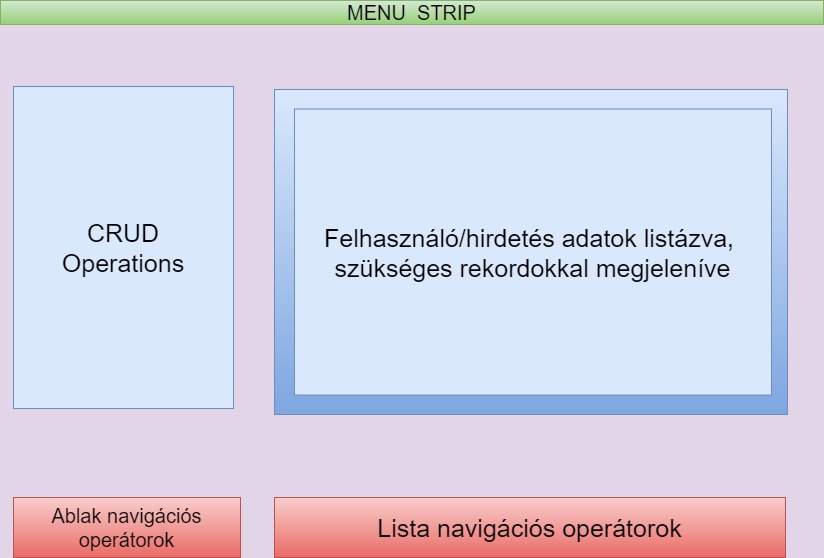


Kezdőlap oldalterv Hirdetés feladás oldalterv

### 2.5.2 Asztali alkalmazás.

Mivel az asztali alkalmazásomnak nem sok feladatot kell ellátnia, egy szintén egyszerű megjelenítést tervezek hozzá. A cél hogy egy gombokkal, és más eszközökkel nem túlzsúfolt felület legyen elérhető rajta, ahol az adminisztrátor rugalmasan tudja kezelni az adatokat. Felépítésének főbb megvalósítandó részei:

1. Bejelentkező ablak
2. Main ablak, útelválasztó a felhasználók, és a cserék között
3. Összes felhasználót megjelenítő ablak, szűréssel, kereséssel
4. Összes hirdetést megjelenítő ablak, szűréssel, és kereséssel
5. Külön-külön ablak ahol a kijelölt felhasználót, vagy a kijelölt hirdetést lehet részletesen megtekinteni/szerkeszteni.



Felhasználó/hirdetés lista Felhasználó/hirdetés

Megjelenítés és kezelés Egyed megjelenítés és kezelés

# 3. Adatbázis tervezése és megvalósítása

## 3.1. Adatbázis tervezés

**„Adatbázis-tervezés** alatt jellemzően logikai adatbázis-tervezést értünk, hiszen az esetek túlnyomó többségében az adatbázis kezelő rendszerek nem teszik lehetővé a fizikai struktúra lényeges befolyásolását. Adatbázis-tervezés kapcsán tipikusan relációs adatbázis-tervezést szokás érteni két okból is. Egyrészt ami a relációs adatbázisokra helyes tervezési módszer és nézőpont, az jellemzően a többi adatmodell alapján felépített adatbázisokra is alkalmazható, másrészt mert ez leggyakoribb adatbázisfajta.

A helyes tervezés általában nem könnyű feladat, számos, helyenként ellentétes szempontot érdemes mérlegelni döntéseink előtt. Amire célszerű tekintettel lenni:

* A létrehozott adatszerkezetben kódolt jellemzők és adatok a leírandó világot az adott célnak, nézetnek megfelelő teljességében reprezentálják.
* Ne lehessen olyan lekérdezést megadni, amely a leírandó világgal nincs összhangban.
* Adat (információ) ne vesszen el.
* Az adatot és annak összetartozó részeit a lehető legkevesebb helyre kelljen beszúrni, illetve helyen kelljen módosítani vagy törölni.
* Az adatbázis lekérdezésének hatékonyságát az adatszerkezet kialakítása ne rontsa jelentős mértékben.”

A tervezésem első lépésében meghatároztam hogy pontosan milyen adatokkal fogok dolgozni, hogy mit kell az adatbázisomon tárolnom. Másodszor az adatok közötti kapcsolatokat dolgoztam ki, mielőtt áttértem volna az adatbázistáblák, és a mezők felépítésére. A tervezés folyamán nagy figyelmet fektettem a redundancia (adatismétlés) elkerülésére, és a normalizálás betartására. A normalizálás az adatbázis adatainak rendszerezését jelenti, hogy az adatbázis táblái, és adatai a lehető leginkább megfeleljenek a normalizálás szabályainak, és ha lehetséges mellőzzék a redundanciát.

A normalizálás szabályai különböző fokozatokon léteznek, ezeket másnéven normálforma szabályainak nevezzük.

**Első normálforma szabályai:**

-A táblák között nem lehetnek ismétlődő csoportok.

-Az kapcsolatokkal ellátott táblák adatait mindenképp elsődleges kulccsal el kell látni

-A különböző kategóriába sorolható adatokat különböző, a saját kategóriájának megfelelő táblában kell elhelyezni

**Második normálforma szabályai:**

-A rekordok csak a táblák elsődleges kulcsaitól függhetnek.

**Harmadik normálforma szabályai**

-A kulcstól nem függő mezők kiiktatása az adatbázisból

## 3.2. „TraDem” Adatbázis Táblái

Teljes adatbázis relációs séma

Az adatbázis alapjainak megtervezése után a táblák létrehozása volt fő célom.A táblák közötti kapcsolatot a táblákban elhelyezett elsődleges, és idegen kulcsok segítségével alkalmaztam. Ezek a kapcsolatok különböző viszonyban vannak, ezeket az imént említett kulcsok határozzák meg. A kapcsolatok formái lehetnek:

**Egy az egyhez kapcsolat (1:1)**

-A tábla akkor van B táblával egy az egyhez kapcsolatban, ha A tábla minden egyes rekordjához legfeljebb egy rekord tartozhat a B táblában, és a B tábla minden egyes rekordjához is csak legfeljebb egy rekord tartozhat az A táblában

**Több a többhöz kapcsolat(N:M)**

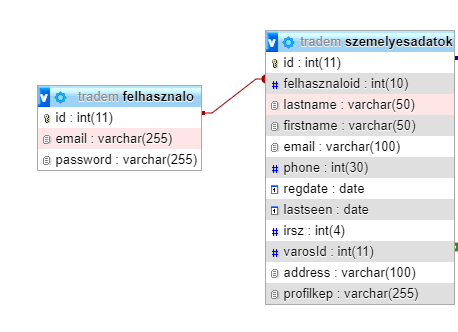
-Az A tábla valamely rekordjához több rekord is tartozhat a B táblában, és a B tábla valamely rekordjához is több rekord tartozhat az A táblában. Ehhez definiálnunk kell egy harmadik táblát, az ún. illesztőtáblát, melynek elsődleges kulcsa két mezőt tartalmaz: az A tábla és a B tábla külső kulcsát. A több-a-többhöz kapcsolat tehát válójában két egy-a-többhöz kapcsolat egy harmadik tábla részvételével.

**Egy a többhöz kapcsolat (1:M)**

A tábla valamely rekordjához több rekord tartozhat a B táblában, de a B tábla valamennyi rekordjához csak egy-egy rekord tartozhat az A táblában. Ez a leggyakrabban használatos kapcsolati típus. Az általam készített adatbázis táblái is nagyrészt ebbe a kapcsolati kategóriába esnek bele.

Mivel egy termékekkel foglalkozó kereskedelmi szoftver létrehozása volt a célom így már eleinte is egyértelmű számomra hogy 2 fő táblával kell majd dolgoznom, amelyek a felhasználók, és a termékek adatait tartalmazzák. Először a felhasználók táblára koncentráltam, hiszen Nélkülük nincsenek hirdetett termékek.

A táblát biztonsági okok miatt két részre osztottam, egy táblára, amely csak az autentikáció segíti elő, és egy másikra, amely már részletesebben tartalmazza a felhasználó adatait



Bejelentkezési adatok-Személyes adatok

A felhasználó bejelentkezési adatait tartalmazó tábla a „felhasznalo” nevet kapta, a felhasználó összes többi adatát tartalmazó pedig a „szemelyesadatok”-at.

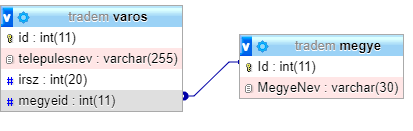
A „szemelyes” tábla tartalmaz egy profilkép mezőt, itt tároltam a felhasználó saját kedvére változtatható profilképének nevét, és kiterjesztését.

**Fontos megemlítenem, hogy a szoftver több képfájl-al is dolgozik, ezeknek a tárolása már a kezdeti fázistól szerver oldalon történt. Az adatbázison való képtárolás egyszerűbb, és akár gyorsabb is lehet, de könnyebben túlterhelhető, és az esetek nagy részében nem nyújt akkora teljesítményt, mint egy fájl server.**

Az adatbázis külön táblában tartalmazza a felhasználó megadott városát, és a városhoz tartozó megyét.

Ennek az oka, hogy egy használt termékekkel foglalkozó szoftvernél véleményem szerint elengedhetetlen hogy a felhasználók lássák a más felhasználók által meghirdetett termékek városra, és megyére pontos beosztását, ezáltal egy fontos szűrési szempontot építve ki.

Mivel a szoftvert egyelőre csak Magyarországra terveztem, így csak a táblák csak magyar városokat és megyéket tartalmaznak. Ezeket hivatalos forrásból töltöttem le, és importáltam az adatbázisomba.



város-megye adatbázis kapcsolat

A termékek adatainak tárolására „cserek” néven hoztam létre táblát. A tábla tartalmaz minden fontosabb mezőt amivel egy használt terméknek rendelkeznie kell. Mivel a szoftver tárgyi cserék hirdetésére fókuszál, külön mezőket hoztam létre a felhasználó oldaláról elcserélendő tárgynak {„megnevezes”=> A tárgy rövid, címszerű leírása

”kategoriaid”=> Külön táblában tárolt, a tárgy kategóriáját mutatja meg

”alkategoriaid”=> Külön táblában tárolt, a tárgy egy kategórián belüli alkategóriáját mutatja meg.

”allapotid”=> Külön táblában tárolt, a tárgy fizikai állapotát mutatja meg

”ertek”=> A tárgy értéke forintban megmutatva

”feltoltesdatuma”=> A hirdetés létrehozásának ideje

,”reszletek”=> A tárgy hosszabb, részletes leírása.},

 és cserébe várt termék(ek)nek {„cserekat”=>A ”kategoria” táblában tárolt, a cserére lehetséges tárgyak kategóriáját mutatja meg.

„cserealkat”=> Az ”alkategoria” táblában tárolt, a cserére lehetséges tárgyak alkategóriáját mutatja meg.

„csereleiras”=>Rövid leírás a cserére várt tárgyak listájának szűkítésére. }.

A kategóriák, alkategóriák, és a tárgy állapotának külön táblában tárolásának oka egy úgy kialakítható felület volt, ahol a felhasználó könnyedén tud ezek a feltételek alapján szűrni, böngészni, és hirdetést feladni.

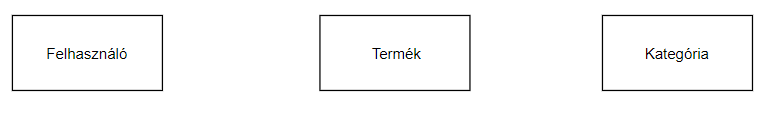
## 3.3 E-K DIAGRAM

„Az egyed-kapcsolat diagram a tárolandó adatok és kapcsolataik grafikus ábrázolására szolgál. Segítségükkel könnyen áttekinthető, a függőségi diagrammokkal ellentétben sok adatot tartalmazó, rendszervázlatot készíthetünk. Az egyed-kapcsolat diagram háromféle összetevőt tartalmaz:

* egyedek
* attribútumok
* kapcsolatok

3.3.1 Egyedek

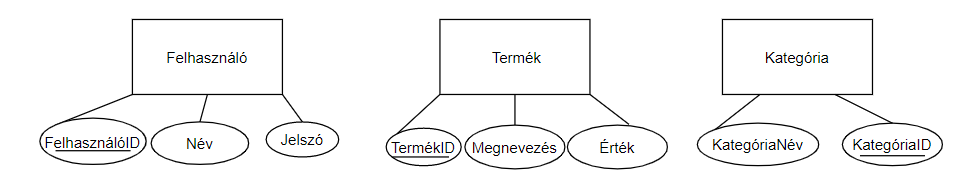
Azok a természetben megtalálható elemek vagy elvont fogalmak, amelyek tulajdonságait (attribútumait) szeretnénk tárolni. Egyed lehetnek például személyek, berendezések, stb. Az egyedeket egy téglalappal ábrázoljuk, melyben felül megjelenik az egyed neve.



Az egyedek az azonos tulajdonságokkal (attribútumokkal) bíró csoportot jelképeznek, az adatbázisban egyes konkrét példányait tároljuk az egyednek.

3.3.2 Attribútumok

Az egyedeket az attribútumaikkal írjuk le. Az attribútum a legkisebb egység, amely az egyed egy tulajdonságát tárolja. Az attribútumokat az egyedek neve alatt soroljuk fel.



A kulcsokat kiemeljük aláhúzással vagy félkövér betűkkel. A több kulccsal, vagy összetett kulccsal rendelkező entitások nem egyértelműek, ezért célszerű csak az elsődleges kulcsot feltüntetni. Például a Tárgyalás egyed összetett kulccsal rendelkezik, míg a Jármű egyednek a Rendszám és az Alvázszám is kulcsa. Helyesebb a Jármű egyed egyik kulcsát jelölni csak.

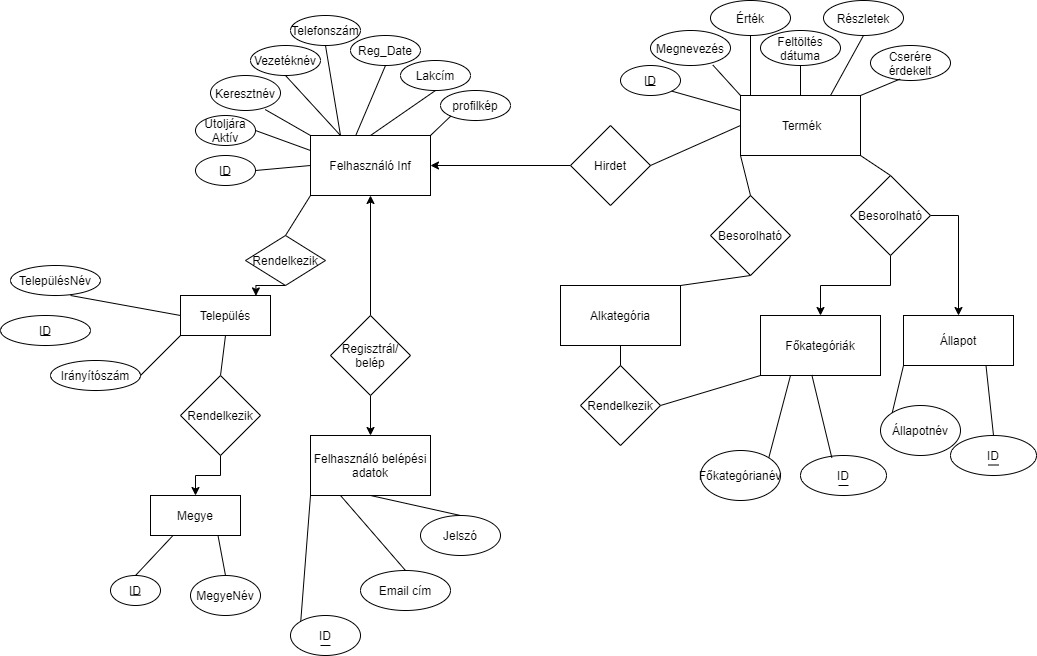
3.3.3 Kapcsolat

Kapcsolat két különböző típusú egyed között lehetséges. Például:

* Személy járművet ***vezet***
* Személy ***részt vesz***  a tárgyaláson

Amíg az egyedek megnevezésében általában főneveket használunk, addig a kapcsolat leírására általában igéket használunk. A kapcsolatok három típusát különböztethetjük meg:

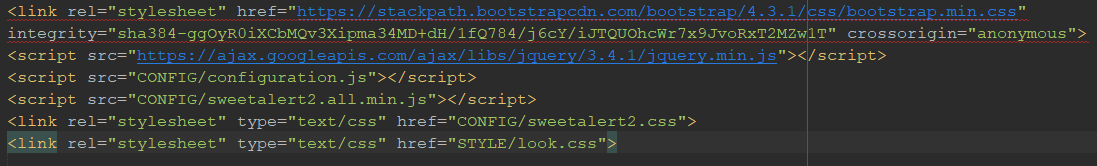
* Egy az egyhez kapcsolat (1:1)
* Egy a többhöz kapcsolat (1:n)
* Több a többhöz kapcsolat (n:m)”



„TRADEM” teljes EGYED-KAPCSOLAT diagram

# 4. Webalkalmazás megvalósítása

A webalkalmazás a „TraDem” elsőszámú felülete, itt van lehetőség felhasználói használatra. A weboldal minden oldala teljesen reszponzívan készült, minden modern eszközön az elvárt vizuális megjelenést nyújtja a felhasználónak. A megfelelő reszponzivitás elérése érdekében „Bootstrap 4” használata mellett döntöttem, amelyet „Bootstrap” CDN segítségével csatoltam a weboldalhoz. Ezáltal a weboldalnak folyamatos internet elérésre van szüksége a megfelelő megjelenéshez.



TraDem Web HTML fejléc

A „Bootstrap” mellett még használtam „Sweetalert2”-t, amely a felhasználó felé érkező értesítések megjelenéséről gondoskodik.

A webalkalmazás összes oldala alatt egy statikus header, és footer működik. Ezek külön fájlokban kerültek tárolásra, és az éppen használatban levő php fájl hívja meg őket. Az oldalakon két navigációs sávval találkozhatunk, az első tartalmazza az alapvető navigációt a weboldal különböző oldalai között (ez a felhasználó bejelentkezésétől függően változik), és a második egy gyors ugráslehetőséget nyújt a termékekhez, azoknak a főkategóriáit linkként felsorolva.

A „TraDem” Webfejlesztése NetBeans IDE 8.0.2 fejlesztőkörnyezetben készült (részletesebben lejjebb). Négy programozási nyelv is használatba került a készítés során.

A weboldal elemi részei HTML nyelven íródtak, ezeknek a kinézetét „Bootstrap” mellett CSS nyelven írtam meg. A backend műveletek PHP –ban, míg a frontend műveltek Jquery segítségével lettek megírva. A futását a XAMPP alatt futó apache szerver segíti elő.

## 4.2 Webes alkalmazás fájl állománya

Fájl térkép

CONFIG => Tartalmazza a weboldal alapvető funkcióit különböző fájlokban.

Itt találhatóak a javascript műveletek, ajax műveletek, és alapvető php funkciók.

HTML => A weboldal alapvető elemeit helyeztem el itt, főleg .html fileok. Itt található például a webalkalmazás header, és footer része külön html fájl okban tárolva, de a felhasználóhoz illeszkedő navigációs sáv is.

IMG => A felhasználók, és hirdetéseik képeinek szerver oldali tárolóhelye.

IMG-SITE=> A weboldal esszenciális képeit tartalmazó mappa.

STYLE => A weboldal kinézetéért felelő fileok tárolóhelye.

Minden más backend feladat ellátásáért létező .php file a projektet root könyvtárában található. Ezek a fájlok nyújtják a weboldal alapját, kezelik a HTML és JAVASCRIPT fileokat, a „munka” nagy része itt folyik.

Fontosabb fileok, és feladatuk:

4.2.1. index.php

A weblap kezdőlapja, ez az első oldal ami fogadja a felhasználót. Belépett felhasználótól független ugyanazt az adatot mutatja mindenkinek, ennek az oka a látogatóközönség effektív bevonása/érdeklődés felkeltés. A fájl maga nem sok kódot tartalmaz, főleg más fájlokat használ az elemek megmutatására, ez nagyrészt igaz a projekt összes többi php elemére is. Funkcionális feladatai megjeleníteni a fejlesztő által közlendő információkat, kiemelt, legfrissebb hirdetéseket, és átlátható navigációt nyújtani a hirdetések. A kezdőoldal legelején egy „bootstrap” által készített carousel található. Carousel segítségével könnyedén, képek segítségével közvetíthetünk információt a látogatók számára. A carousel képei a webalkalmazás „IMGsite/” mappájában kerülnek tárolásra, hiszen ezek a weboldal elemét képzik, és nem a felhasználókhoz kötődnek.

4.2.2 connect.php

Ahhoz hogy a céljának megfelelő működést érjünk el a weboldaltól folyamatos kommunikációra van szükségünk a weboldalunk szervere, és az adatbázis szerver között.

Ezt az esetek nagyrészében a „connect.php” fájl meghívásával értem el. A fájl tartalmazza az adatbázis kapcsolat létesítéséhez szükséges kódot, sikeres kapcsolat esetén (utf8) karakter beállítást végez, sikertelen csatlakozás esetén pedig megállítja a munkamenetet.

4.2.3. functions.php

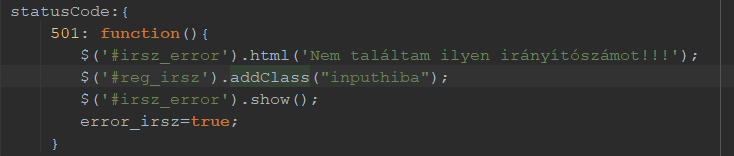
Az átlátható kódolás elérése érdekében a webalkalmazás által használt szerver oldali funkciókat itt tároltam, melyek majdnem az összes oldalon meghívásra kerülnek.

4.2.4 start.php

Majdnem minden php fájl használja, meghívásával elindíthatjuk a munkamenetet „session\_start()”, meghívjuk a connect.php-t adatbázis kapcsolatért, és a functions.php, a webalkalmazás funkcióinak használatáért.

4.2.5 reg.php

A felhasználók adatbázisba való regisztrációját végzi. A mezők helyes kitöltésének ellenőrzését a reg\_check.js végzi. Mivel a weboldal fontos részét képezi a régiókra való keresés így a várost nem engedélyeztem saját bevitelre. Az irányítószám bevitele után egy

****„hover” jqery esemény lefutásával az oldal elindít egy AJAX kérést. Ez a kérés ellenőrzi hogy létezik e Magyarország irányítószámainak listájában a bevitt irányítószám. Amennyiben pozitív eredménnyel fut le a kérés, az visszaadja az irányítószámhoz tartozó város nevét, és automatikusan beilleszti a város „input text” mezőjébe. Negatív válasz esetén figyelmezteti a felhasználót a nem megfelelő adatról, hiszen továbblépni csak hiteles irányítószám megadása után engedélyezett. Az Ajax fájt (keresIrsz.php) nem létező irányítószám esetén az 501 es http hibakóddal tér vissza. A hibásan kitöltött input elemek kinézete előre megírt CSS osztály hozzáadásával változik, „small” típusú HTML elemben írt hibaüzenet megjelenik, helyes kitöltés az osztály, és hibaüzenet eltávolításra kerül.

501 http\_respose\_code visszatérés

4.2.6. login.php

A felhasználó beléptetéséért felelős weboldal alapja. A start.php n keresztül elindítja a munkamenetet, és csatlakozik az adatbázishoz, ahol a felhasználók bejelentkezéshez szükséges adati vannak tárolva. A kódfolyamat egy „if” feltételen keresztül csak akkor indul el ha a felhasználó kitöltötte a belépéshez szükséges text mezőket. A lekérdezést itt, és más érzékeny helyen „prepared statement”-eken keresztül hajtottam végre, a biztonságos adatkezelés érdekében. A prepared statement lényege hogy a gyors adatelérés, és az sql injekció elkerülése érdekében nem tisztán SQL kóddal hajtjuk végre a lekérdezés,

hanem azt előkészítjük, paramétereket fűzünk hozzá, és végül végrehajtjuk.

Miután a lekérdezés megtörtént, és a felhasználó sikeresen bejelentkezett egy $\_SESSION

szuperglobálban tárolásra kerülnek az azonosításhoz szükséges adatai, és frissül az adatbázisban a hozzá tartozó „last seen” mező. A sikertelen bejelentkezési kísérletek if feltételekkel, és „alert()”/”alertThanMove()” funkciók segítségével vannak kezelve.

4.2.7 logout.php

Rövid kódsorozattal lezárja a jelenlegi munkamenetet, a session\_destroy() metódussal.

Ennek következtében a bejelentkezett felhasználó adatai törlődnek a weboldal munkamenetéből, és egy header() funkció segítségével automatikus átirányításra kerül a kezdőoldalra.

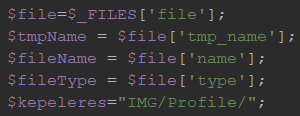
4.2.8. tradesAll.php

A webalkalmazás egyik legkomplexebb része. Feladata egy összetett szűrésen keresztül listázni az adatbázisra feltöltött hirdetéseket, képekkel, és személyes információkkal.

Több linken keresztül is elérhető, a második navigációs részleg kategórianév linkjei ezt az fájlt nyitják meg egy switch case állításon keresztül, a linkre kattintott kategória szerint szűrve. A hirdetések listázása „bootstrap card” elemeken keresztül történik, melyek tartalmazzák a hirdetés címét „card-title” mezőben, a hirdetés fizikai értékét forintban mérve, a feltöltés dátumát, és a feladóhoz tartozó regionális adatokat „card-text” mezőben.

Ezek a „bootstrap card” elemek automatikusan vannak generálva, az adatbázis lekérdezés alapján, egy „while” cikluson keresztül, ami jelenít meg újabb elemeket, amíg talál őket az adatbázisban. A weboldal bal oldalán elhelyezkedő szűrés egy HTML „form” elemben van összesűrítve. A keresés gombra való kattintás után a „tradesAll.php” végigmegy az összes input elemen, ellenőrzi hogy a felhasználó milyen adatokat vitt be, és az alapján egyesével módosítja/kiegészíti a lekérdezést. Miután az összes elem ellenőrzésre került, és a SQL lekérdezés ez alapján kiegészült, és a „$csere” változóban megkezdődik a kártyák kialakításért felelős HTML kód összeépítése , melyet az oldal végén egy „echo()” beépített PHP funkció segítségével kerül megjelenésre. A lekérdezést a „query()” ,PHP funkció futtatja le, a „$res” változóban kerül tárolásra. Ezután indul a „while” ciklus, ami addig kerül végrehajtásra amíg a „$res” tartalmaz elmentett hirdetést. A hirdetések képeinek beillesztéséért a „keresKepek()” funkció felelős. Ez kikeresi az adatbázisban a hirdetéshez tartózó képek adatait, az alapján megkeresi őket a képek tárolásáért kialakított mappában, egy tömbben eltárolja az elérési adatokat, és fájlneveket, majd egy index alapján az első képet megjeleníti a kártyán. A fájlban szintén használatra kerül a „keresUserById()”, és a „keresVaros()”, amelyek használatát a szétbontott adatbázis indokolja.

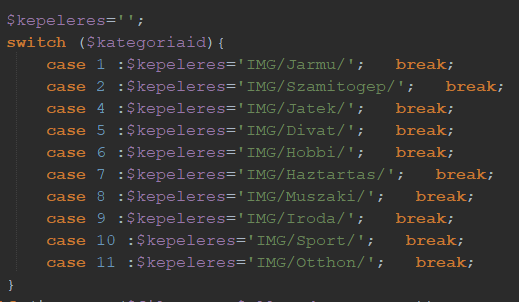
4.2.9. personal.php

A bejelentkezett felhasználó személyes adatainak megtekintésére készült. A felhasználó megtekintheti regisztrált adatait, feladott hirdetéseit, és módosításokat hajthat végre rajtuk. Csak bejelentkezést követően elérhető, másképp egy hibaüzenettel átirányítja a felhasználót az oldalról. A jelenlegi profilkép a „$keresprofilkep” segítségével jelenik meg, amely amennyiben ha a felhasználó még nem használ saját képet a profiljához egy univerzális képet tölt be megtekintésre. Az oldal csak kép fájltípusú fájlokat fogad el profilképnek(például jpg, png, jpeg).Miután kiválasztásra, és elküldésre került a feltöltendő kép, az szétbontásra kerül, fájltípusra, fájlnévre, és ideiglenes tárolási helyre.

A fájlismétlődés elkerülése érdekében a fájl nevéhez hozzáfűződik a mai dátum time(). funkcióval, majd az ideiglenes tárolási helyről áthelyezésre kerül a kiválasztott fájl.

Ezután egy „prepare statement” SQL update lekérdezéssel ezeket a szétbontásra került adatok feltöltésre kerülnek az adatbázisba. A saját hirdetések megtekintése itt is „bootstrap carousel” elemekkel történik. A saját adatok változtatása AJAX kéréssel történik. Az email esetében a szerkesztés gombra kattintva elindul egy jquery esemény a háttérben, ez a megadott email adattal elindítja az ajax kérést, ami a szerkEmail.php ben kerül végrehajtásra. A szerkemail.php ellenőrzi az adatbázisban hogy a megadott adatok szerepelnek e már valahol, és csak negatív eredmény után folytatja a folyamatot. Pozitív lefutás esetén „http\_response\_code()” elküldi az 503 az http választ a jquerynek, ami a felhasználó számára is látható hibaüzenetet csatol hozzá. Végül Egy ”prepare statement” lekérdezés megváltoztatja az adaatbázisban szereplő email rekordokat, és é javascript oldalról értesítjük a felhasználót a sikeres email cseréről. Jelszó változtatás esetében az ajax kéréses folyamat ugyanaz, viszont itt már egy külön oldalon(passwordchange.html) történik a változtatás, és csak akkor kerül átadásra az Ajax oldalra(passwordChange.php) a jelszó, ha az előlegesen megírt feltételeknek megfelel.

4.2.10. makeTrade.php

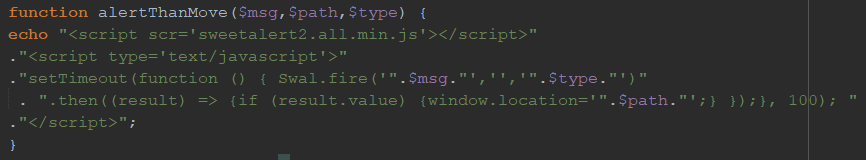
A „makeTrade.php” felelős a hirdetések feltöltéséért. Felhasználóhoz kötött, csak bejelentkezést követően válik elérhetővé. Az input mezők kitöltése után a php fájl maga, és a javascript konfigurációs fájl ellenőrzéseket hajt végre rajtuk. Amennyiben a hirdetés megfelelő adatokkal lett kitöltve a php fájl eltárolja ezeket lokális változókban, és feltölti őket a „cserek” adatbázismezőbe. A fájl több kép csatolására is alkalmas. Amennyiben csatolásra került legalább 1 fájl egy for ciklus elindul minden egyes képre, eltárolja változókba a képek nevét, átmeneti helyét, és típusát, és egy „switch case” ciklus a hirdetés kategóriája alapján megkeresi a leendő tárolási helyet a képenek(eknek). A létező képek elleni védelem itt is meg van írva, amennyiben a felhasználó már létező nevű képfájlt tölt fel, az automatikus átnevezésre kerül. Sikeres feltöltés esetén alert() függvény értesíti a felhasználót.

## 4.3. Webalkalmazás funkciói, és adatok

A weboldalon használt funkciók szerves részét képzik a működésnek. Elősegítik a gyors és tisztán átlátható kódolást, amelyre az egész projekt alatt törekedtem. Backend oldalról ezek megtalálhatóak a functions.php file-ban. Néhány a weboldal fontosabb funkciói közül:

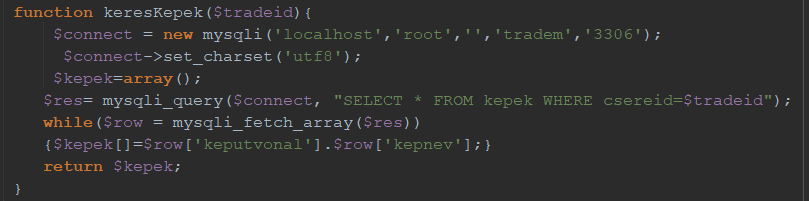
### 4.3.1 Szerver oldalon

* isLogged() => bool visszatérési értékkel megadja hogy van e bejelentkezve felhasználó. Segítségével könnyedén megvalósíthatóak elem mint a menü megjelenítése, és bizonyos lehetőségek letiltása/engedélyezése.
* alertThanMove() => Egy egyszerű függvény a php beépített header() függvénye helyett. Értesíti a felhasználót egy a fejlesztő által megadott üzenetre, majd átlépteti őt egy megadott oldalra. A felhasználóbarátabb megjelenés érdekében sweetalert2-t használ



alertThanMove()

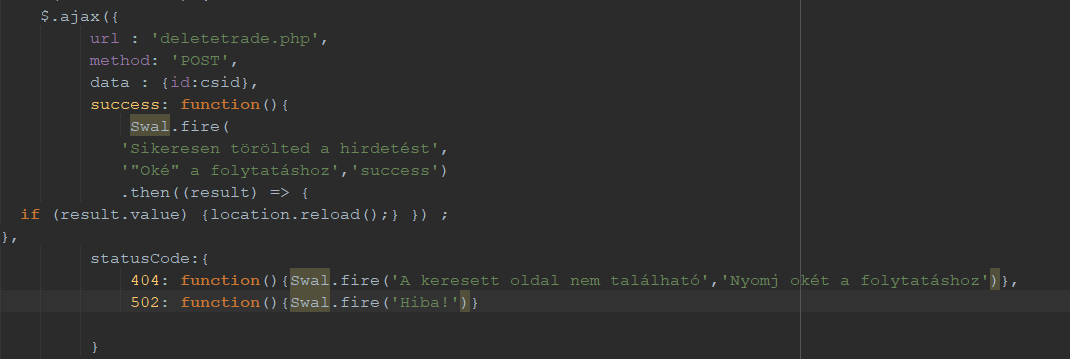
* keresXY() => megadott érték alapján megkeres egy bizonyos adatot(adatokat) az adatbázisból. Használatát főleg a gyors, átlátható kódolás indokolta.



keresKepek()

* lastSeen() => A felhasználó bejelentkezésével frissíti az adatbázis lastseen mezőjét.

4.3.2. AJAX funkciók

* Text kiegészítés(Regisztráció) => Ajax kérés segítségével a weboldal automatikusak kiegészít beviteli mezőket az adatbázisban szereplő adatok alapján. Például a regisztráció során megadott irányítószám alapján automatikusan kitölti a felhasználóhoz tartozó települést, és a hozzárendelt megyét.
* Törlés, és módosítás => AJAX kérés segítségével felhasználói utasításra a weboldal elérhetővé teszi a felhasználó adatainak módosítását (jelszó, email), amennyiben az új adatok megfelelnek a weboldal feltételeinek (Előzővel nem megegyező adatot adott meg a felhasználó, megadott email nem létezik még az adatbázisban).

Ajax kérés jqueryben hirdetés törlésére

4.3.3. Nem szerver oldalon

* Teljes adatbevitel ellenőrzés=>A weboldal teljes felülete védve van nem megfelelő adatok bevitele ellen. Ezt Jquery segítségével megírt funkciók teszik lehetővé, letiltják az adatbevitelt, és figyelmeztetik a felhasználót a problémás mezőkről, amennyiben az adatok nem felelnek meg a beviteli mező elvárásaival.
* Felületi elemek grafikus megjelenítésének kontrollálása=>A weboldal több része is események alapján dolgozik, jelenít meg HTML elemeket, adatokat.

A web alkalmazás folyamatosan igazodik a felhasználóhoz. Bejelentkezés során egy

$\_SESSION szuperglobálon keresztül tárolásra kerülnek a felhasználó beazonosításához szükséges adatok. Ez az alkalmazás során több helyen is használatra kerül, így csak a jelenlegi felhasználó.

# 5. Asztali alkalmazás megvalósítása

A „TraDem” asztali alkalmazása adminisztrátori használatra készül. Segítségével az adatbázisban szereplő adatok egyszerűen kezelhetővé válnak megfelelő jogosultság mellett. Az alkalmazást MVP (Model-View-Presenter) szoftvertervezési minta alapján készítettem. Az MVP az MVC mintából alakult ki, és hasonlóan hozzá komplex alkalmazások műveletvégzéseiben nyújt segítséget.

„Az MVP modell három rétegre osztja az alkalmazást. Mindhárom rétegnek jól körülírható feladatai vannak, és csak a szomszéd réteggel kommunikálhatnak. Ez nagyfokú rugalmasságot ad: a két szélső komponens egymás tudta nélkül bármikor lecserélhető. Például egy új adatbázismotor támogatásához nem kell módosítani a felhasználói felületet, illetve egy új "publikus interfész" (pl. web-szerviz) bevezetése sem igényli a modell módosítását.

* A **modell** az alkalmazás által megjelenített és feldolgozott adatok reprezentációja.
* A **nézet** megjeleníti a modellben tárolt adatokat a felhasználó számára, illetve a felhasználói interakció során bekövetkező eseményeket továbbítja a prezenter felé.
* A **prezenter** összegyűjti az adatokat, illetve formázza a nézet számára feldolgozható módon.”

## 5.1 Az asztali alkalmazás fájl állományai

File térkép az asztali alkalmazáshoz

## 5.2 Models

A benne lévő fájlok az adatbázissal való kommunikációért felelnek.Az adatbázis alapján a program egy saját modell készült, ami ennek a felépítésére alapszik. Minden egyes kommunikáció során az adatbázissal a program ezen a modellen keresztül azonosítja be hogy hogyan kérje le az adatokat. A program készítése során többször is frissítésre szorult, minden egyes komolyabb adatbázis szerkesztés megigazolja ezt.

Az itt található TrademModels.edmx fájl felelős a program számára az adatbázis táblák felépítésére. Tartalmazza a táblák adatainak megfelelő osztályokat, a kontextus fájlt, és grafikus tábla modellt. A szükséges adatelérés érdekében egy „Partials” mappában tároltam részosztályokat, mivel a TraDemModel.edmx osztályai minden adatbázisfrissítés után újra íródnak, bennük információ szerkesztés/hozzáadás nem effektív.

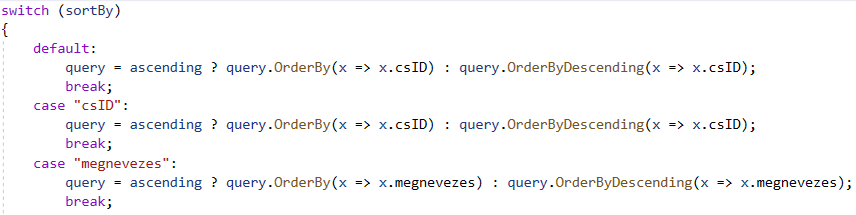
## 5.3 Repo(Repository réteg)

A teljes adatbázissal való kommunikáció itt történik. Fájljai metódusokkal vannak ellátva, melyek a presenter rétegből való meghívásra elvégeznek egy adott lekérdezést az adatbázisból.

5.3.1 UsersRepo.cs és CsereRepo.cs

Mindkét osztályt egyszerre mutatom be, mert funkcionalitásukat, és kódtartalmukat tekintve rendkívül hasonlóak. Mindkettő egy publikus „IDisposable” osztály, az adatbázis hirdetésekkel foglalkozó tábláját dolgozzák fel a felsőbb rétegek számára.

Metódusok:

-GetAllCsere()=> Felelős a teljes hirdetés lista lekérdezéséért. Még mielőtt kapcsolatba lépne az adatbázissal, LinQ segítségével a „query” változóban rakja össze a lekérdezést, jövőbeli eseményektől függően. A „query” értéke keresés esetén a keresett értékre kap szűrést, rendezés esetén a tábla rekordjait a megadott feltételek szerint rendezi.

Az összes lekérdezett adat megszámlálását az integer totalItems változóban tároltam. 

Szintén itt történik még a mezők mennyisége szerinti oldalbontás elkészítése.

Miután a query elkészült egy új „BindingList” elemben tárolásra kerül annak eredménye.

-**Insert()**,**Update(),Delete()**,**Save()** => Az alapvető adatbázis műveletek mind létre vannak hozva itt metódusok formájában.

**-DeleteF()**,**UpdateF()=>**Mivel az adatbázisban egy felhasználóhoz két tábla tartozik, így mindkettő külön metódus által kerül módosításra, a Delete()/Update() a felhasználó személyes adatainak tábláját töröli, míg a DeleteF()/UpdateF() a bejelentkezéshez szükséges adatokkal ellátott tábláját.

-**Exists()**=> Bool típusú visszatérési értékkel megadja hogy létezik e az adatbázisban az kérdéses rekord.

5.3.2. TRepo.cs és UsersFormRepo.cs

Szintén rendkívül hasonló a két repository osztály. Feladatuk a hirdetés/felhasználó tábla bizonyos kiválasztott rekordjának szerkesztése, törlése, vagy indokolt esetben hozzáadása. Nem tartalmaznak sok metódust, nincs is rá szükség.

5.3.3LoginCheck.cs

Egyszerű repository osztály, az asztali program bejelentkezése saját döntésre biztonsági okokból nem adatbázisból történik, hanem beépített jelszóval, fióknévvel. Azért hogy a program csak akkor engedje be a felhasználót ha van adatbázis kapcsolat ez az osztály felelős.

## 5.4. Presenter (presenter réteg)

A presenter mappa az MVP modell presenter ágának feladatait végzi.

Kizárólagosan ez kommunikál a repository réteggel, adatokat kezel, funkciókat futtat. Közvetlen összeköttetésben áll az szükséges interfész-el, és segítségével alakítja ki a megjelenítendő adatok formáját.

5.4.1 UserPresenter.cs és TradePresenter.cs

Mindkét osztály rendkívül hasonló, a megjelenítendő felhasználó(UserPresenter) és hirdetés(TradePresenter) listát rakják össze, és módosítják azt.

A felhasználókért listájáért felelős osztály az „IDataGridList” interfész, és a CsereRepo repository, míg a hirdetések listájáért szintén az „IDataGridList” interfész , és a „UsersRepo”alapján dolgozik.

Fontosabb metódusok

LoadData() => A repositoryt meghívva összegyűjti az esetleges szűrész alapján a felhasználókat, feltölt egy „BindingList et”, rendezi, és összeszámolja.

Add() =>”szemelyesadatok” vagy „cserek” objektum alapján hozzáad egy elemet a listához, és az adatbázishoz

Remove()=> Index alapján töröl egy adatok a listából, és az adatbázisból

Modify()=> „szemelyesadatok” vagy „cserek” objektum alapján frissíti az adatbázist.

Save()=> Elmenti a végzett változtatásokat az adatbázisba.

5.4.2 Upresenter.cs és TPresenter.cs

Egy bizonyos, a felhasználó által kiválasztott rekord módosítását végzik külön formokon. A Upresenter a felhasználókért felel, a TPresenter pedig a hirdetésekért.

**Metódusok:**

**Save**()=> Új adat felvétele, vagy már meglévő adat szerkesztése esetén meghívandó.

Feladata a felhasználó, vagy hirdetés beviteli adatainak ellenőrzése, és hiba esetén egy saját hiba változóba annak specifikus leírása. Ezek a hibák az IUser.cs interfész en keresztül elérhetőek.

**keresVaros**()=> string típusú településnév bevitelével ezt elküldi a repositorynak, és az visszaadja az objektumot, ha talált a névvel megegyező objektumot az adatbázisban.

5.4.3 LoginPresenter.cs

A programba való bejelentkezést ellenőrzi , az erre megírt „Authenticate()” függvénnyel.

## 5.5 Interface

Interfész fájlokat tartalmaz, ezek hasonlítanak az osztályokhoz, azt mutatják be, milyen deklarációk lesznek az adott osztályunkban. Tudnunk kell, hogy az interface-k ,( más néven felületek), nem valósítanak meg semmit, nincs is bennük konkrét kód utasítás formájában.

## 5.6 View

A megjelenésért felelős windows form fileok itt találhatóak. Közvetlen kapcsolódik a presenter réteghez, innen kapja a megjelenéshez szükséges adatokat.

5.6.1 LoginForm.cs

Bejelentkezéshez elkészített form. Jogosultság kizárólag az adminisztrátornak jár, biztonsági okokból a programba való bejelentkezés adatai nem adatbázison lettek tárolva.

Hibás beviteli mező esetén errorProviderek gondoskodnak a felhasználó figyelmeztetéséről.

5.6.2. MainForm.cs

Úgynevezett főmenü, elválasztó. Egyetlen célja az adminisztrátor megfelelő helyre való navigálása kattintás után.

5.6.3. TradeForm.cs és Usersform.cs(Listák megjelenítése)

Az adatbázis rekordjainak listázott, grafikus megjelenítése itt történik. A formok kommunikációban vannak a hozzájuk tartozó presenter, és interfész fájlokkal.

Init()=>A listázásra vonatkozó adatokat alapértékkel látja el (pl objektum oldalanként).

Az interfész tulajdonságai megtalálhatóak benne, a megfelelő „set/get” beállításokkal.

Események:

-Datagridview headerre való kattintás hatására egy switch case állítás alapján a srtBy változó értéket kap, majd ezt átadja a presenter LoadData() függvényének.

5.6.4. Regform.cs(felhasználó felvétel)

Új felhasználó manuális beviteléért felelős. A presenter által egy teljes input mező ellenőrzésen megy keresztül.

-Város mező ellenőrzése: az adatbázisban megkeresi hogy található e azonos nevű város rekord, és csak pozitív visszatérés esetén engedi a további ellenőrzést.

-Üres mezők ellenőrzése: a program minden megjelent mező kitöltését ellenőrzi, ki nem töltött elemek esetén nem enged továbbhaladást a kódban.

-Nem megfelelő formátumú mezők: Néhány input mező speciális adatokat igényel (például email), a program vagy azonnal letiltja a nem megfelelő adatok bevitelét, vagy ellenőrzésnél nem engedi tovább haladni a kódot.

A hibák kezeléséért, és a felhasználó figyelmeztetéséért errorProviderek felelősek.

5.6.5. UForm.cs és TForm.cs (adatmegjelenítés/adatmódosítás)

Mindkét windows form hasonló elven működik. Fő feladatuk a kijelölt objektum részletes megjelenítése az adminisztrátor számára. A felhasználók esetében ez kibővül a felhasználók adatainak szerkesztésére is.

A felhasználók, és hirdetések képei „picture box” elemekben kerülnek megjelenítésre.

A Tform 4 ilyen elemmel rendelkezik, ezek adatait az adatbázis alapján egy „foreach” ciklus feltölti egy listába, majd a lista elemeinek száma alapján „switch case” állítással feltölti őket a picture boxokba.

## 5.7 Asztali alkalmazás tesztesetek

Az asztali alkalmazás metódusainak tesztelésére Unit test eket alkalmaztam. „A unit-teszt, vagy más néven egységteszt, a metódusokat teszteli. Adott paraméterekre ismerjük a metódus visszatérési értékét (vagy mellékhatását). A unit-teszt megvizsgálja, hogy a tényleges visszatérési érték megegyezik-e az elvárttal. Ha igen, sikeres a teszt, egyébként sikertelen. Elvárás, hogy magának a unit-tesztnek ne legyen mellékhatása.”

5.7.1 Tesztelt metódusok

**ConnExists- Teszt** => A bejelentkezéshez szükséges „ConnectExists” metódust teszteli.

A teszt sikeresen lefut, adatbáziskapcsolat esetén nem dob hibát, amennyiben nincs adatbáziskapcsolat hibát dob, ahogy az elvárható.

**LenghTest teszt =>** A metódus beviteli mezők hosszát ellenőrzi. A teszt sikeresen lefutott, a metódus hibát dobott ha a limitet átlépte a hossz.

# 6. Felhasználói dokumentáció

## 6.1 Bevezetés

A „TraDem” szoftver csomag 3 különböző részből áll. Tartalmaz egy asztali alkalmazást, adminisztrátori tevékenységek gördülékeny elvégzésére, egy adatbázist, ami a használathoz szükséges adatok tárolásának környezetét biztosítja, és végül a szoftver lényegi részét tartalmazza.

-Az adatbázis használatához folyamatos élő szerverkapcsolatra van szükség, melyet jelenleg a „XAMPP” program biztosít.

-A webes alkalmazás használatához valamilyen böngésző szükséges, a program használata Google Chrome, és Firefox programok alatt ajánlott.

-Az asztali alkalmazás Windows alapú operációs rendszereken működésképes, Microsoft.NET framework keretrendszer elengedhetetlen!

## 6.2 Asztali alkalmazás

Az adatbázisban szereplő adatok(hirdetések /felhasználók) manipulációját teszi lehetővé.

Felhasználó adatok szerkesztését indokolttá teheti a felhasználó adatainak való életben történő megváltozása. Amennyiben az adatok változtatását nem indokolja semmi, kérem ne hajtson végre műveleteket a rekordokon.

### 6.2.1 Asztali alkalmazás minimum / ajánlott hardwer követelményei

Tárhely => 20 MB szabad hely - 50 MB Szabad hely

RAM => 512 MB - 2024 MB

CPU => 1 Ghz - 1-2Ghz (Intel dual core)

Egér, billentyűzet

## 6.3 Asztali alkalmazás használati utasítás

A „TraDem” asztali alkalmazás használatához adminisztrátori jogosultság szükséges. Amennyiben nem a megfelelő jogosultsággal nem rendelkezik, kérem ne használja az alkalmazást. Az asztali alkalmazás a weboldal felhasználóinak, és hirdetéseinek adminisztrátori kezelésére készült.

6.3.1 Bejelentkezés

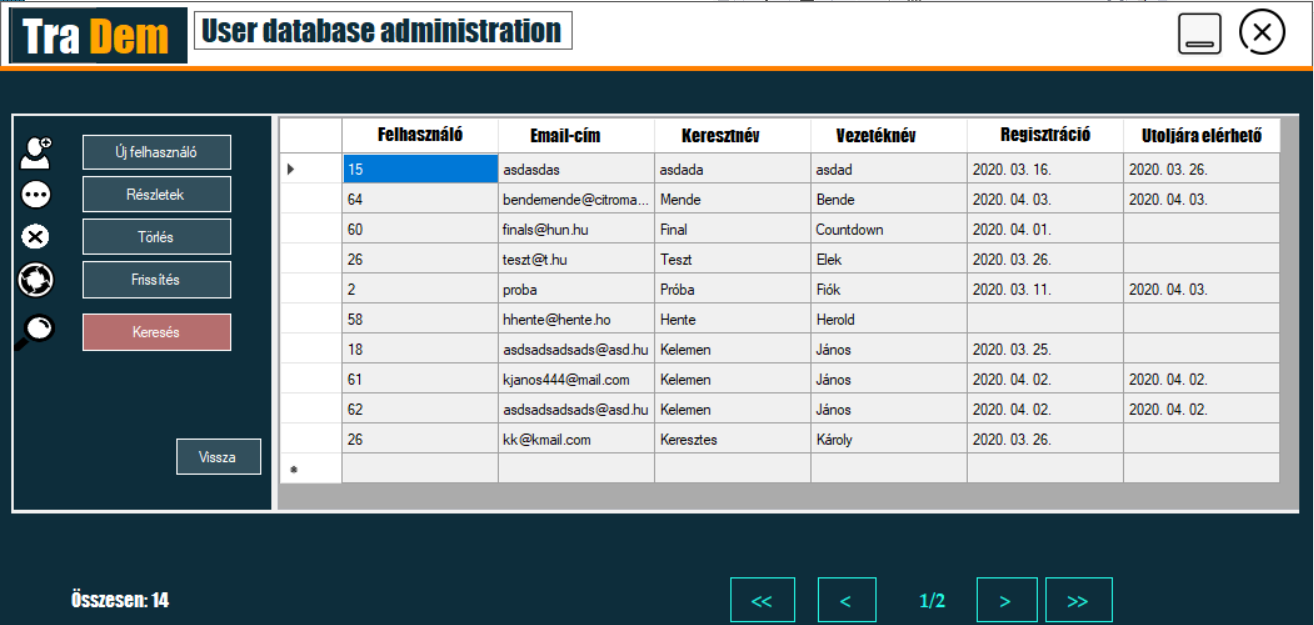
A program indítása után az kapásból bejelentkezéssel indít. A program használata csak bejelentkezés után lehetséges. A bejelentkezéshez szükséges adatokat kérje a készítőtől. Amennyiben az adatok helyes beütése után sem sikerült bejelentkezni kérem ellenőrizze hogy van e megfelelő adatbázis kapcsolat.

Login ablak

6.3.2 Fő-menü

Tartalmazza a felhasználó listához vezető, és a hirdetések táblához vezető gombot. Funkciója a navigáció ezek az ablakok között.

6.3.3 Felhasználó kezelés

A teljes felhasználó megjelenítés, és kezelés itt történik. Az ablak bal oldalán található a kezelési menü, bal oldalon az adatbázisban szereplő felhasználók, és adataik. A megjelent listából megnézhetjük a felhasználóhoz tartozó azonosító kódot, emailt, vezetéknevet, keresztnevet, regisztrációjának dátumát, és az utolsó elérhetőségének dátumát. A felhasználó többi adatát a részletek gombra kattintva tekinthetjük meg, ez új ablakot nyit meg. A táblák frissítését a frissítés gomb kezeli, amennyiben adatváltozás történt ezzel a gombbal frissíteni tudjuk a listánkat hogy megjelenítsük az új adatokat. Új felhasználó felvételét az „Új felhasználó” gombbal tehetjük meg, ez egy új ablakon történik. Felhasználó törléséhez jelöljünk ki egy felhasználóhoz tartozó teljes sort, majd kattintsunk a „Törlés” gombra.

Felhasználó Kezelés

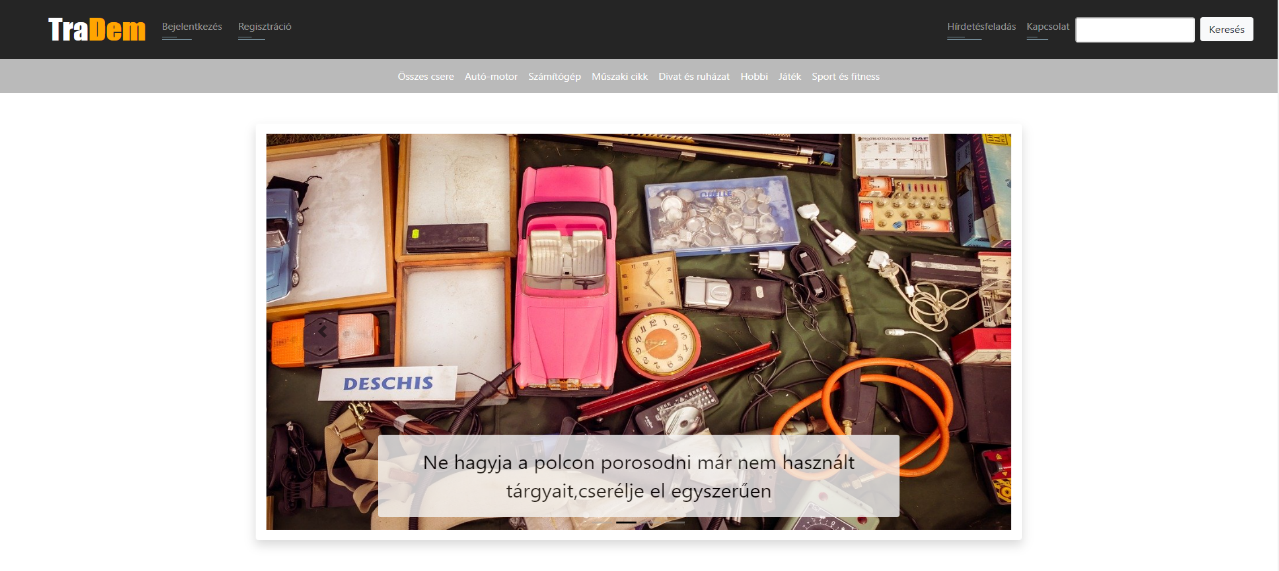
A megjelenő listában való kereséshez nyomjuk meg a keresés gombot, ez megjeleníti a keresett szöveg beviteléhez szükséges textbox mezőt. Miután kitöltöttük az elemet a keresésre szánt szöveggel, a „Keresés” gomb ismételten való lenyomásával frissíthetjük a listát a szűrés szerint

6.3.4 Hirdetések kezelése

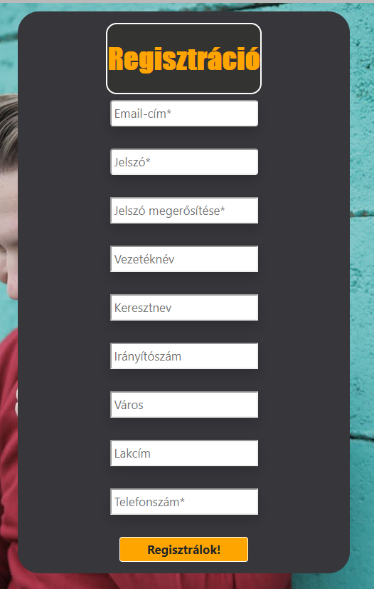
Hasonló menüponttal rendelkezik mint a felhasználókezelés. Hirdetések módosítására, és felvételére nincs lehetőség, csak megtekintésükre, és törlésükre. A részletek menüponttal új ablakban megjelennek a hirdetés pontos adatai, és képei. Hirdetés törlésére itt is lehetőség van.

## 6.4. Webes alkalmazás

A webalkalmazást bármilyen modern böngésző képes futtatni. Egy egységes felhasználói felületet teremt, ahol a „TraDem” felhasználói használt termék hirdetéseket tehetnek közzé más felhasználók általi megtekintésre.

6.4.1. Főoldal

A weboldal minden lapján felül találhatóak a navigációs sávok. A felső navigációs sáv az oldalon való navigációt szolgálja. Itt található a bejelentkezés, regisztráció, hirdetés feladás, kapcsolat, és egy kereső sáv is. A hirdetésfeladás csak bejelentkezett felhasználók számára elérhető, de megtekinteni a hirdetéseket bejelentkezés nélkül is lehetséges. A második navigációs sáv a hirdetések kategóriát sorolja fel, a linkekre kattintva a hirdetések jelennek meg, a kiválasztott kategória szerint szűrve.

6.4.2. Regisztráció

Az oldalra való regisztráció egyszerűen elérhető a navigációs sávból. Regisztrálni akárkinek lehet jogosultsága, amennyiben valós adatokat használ.

Miért fontos? A teljes weboldal használata csak ez után válik elérhetővé, ez kitér a hirdetések feladására, és a felhasználói profil kezelésére.

Regisztráció feltételei:

-A megadott email cím még nem szerepelhet az adatbázisban.

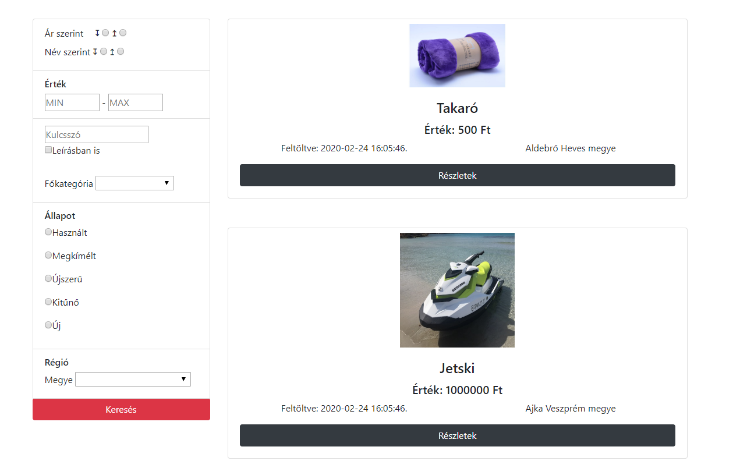
-A megadott jelszó megerősítésnek meg kell egyeznie a megadott jelszóval.

-A megadott városnak egy létező Magyar városnak kell lennie, ez a szűrés szempontjából elengedhetetlen.

6.4.3. Böngészés a hirdetések között

A megjelent hirdetések alapvetően feltöltési idő szerint, csökkenő sorrendben jelennek meg. Az oldal lehetőséget ad az oldalanként megjelent hirdetések számának megváltoztatására, „Hirdetés feladás” oldal felső részén, gombokkal elválasztva. Alapvetően 10 hirdetés jelenik meg oldalanként, ezt módosíthatjuk 5-re, és 20-ra is.

A hirdetések jobb oldalt, kártyákon jelennek meg. A kártya felső része a hirdetéshez tartozó képet tartalmazza, majd utána megjelenik a tárgy megnevezése, a fizikai értéke, feltöltés dátuma, és az érdeklődési kör is. A hirdetések szűrését a kártyák mellett jobb oldali mezőn találhatjuk.

A hirdetések rendezésére lehetőségünk van:

-Ár szerint

-Név szerint

A hirdetések szűrésére lehetőségünk van:

-Ár szerint, minimum, és maximum megadása szerint

-Kulcsszó megadásával, amely a hirdetések megnevezésében keres

-Állapot alapján

-Régió szerint

Ezek a szűrések külön-külön, de akár egyszerre több kijelölésével is egyaránt működnek.

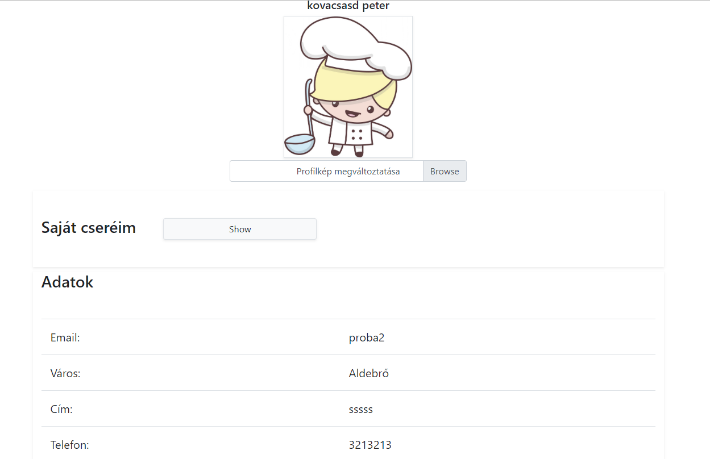
6.4.4. Hirdetés feladás

A hirdetés feladása oldal a navigációs menü „Hirdetés-feladása” gombjára kattintva válik elérhetővé. Első lépésnek választanunk kell egy fő kategóriát a tárgyunknak, e-nélkül nem léphetünk tovább. Kép kiválasztása a hirdetéshez nem kötelező, de erősen ajánlott. A képpel ellátott hirdetések sokkal megnyerőbbek, kedveltebbek az emberek számára. Maximum 4 képet tölthetünk fel hirdetésünkhez.

Következő gomb => Megjeleníti a hirdetés adatainak további beviteli mezőit.

A megjelent ablak minden mezője kötelező adat. Kitöltésük után a feltöltés mezőre kattintva már fel is töltöttük hirdetésüket.

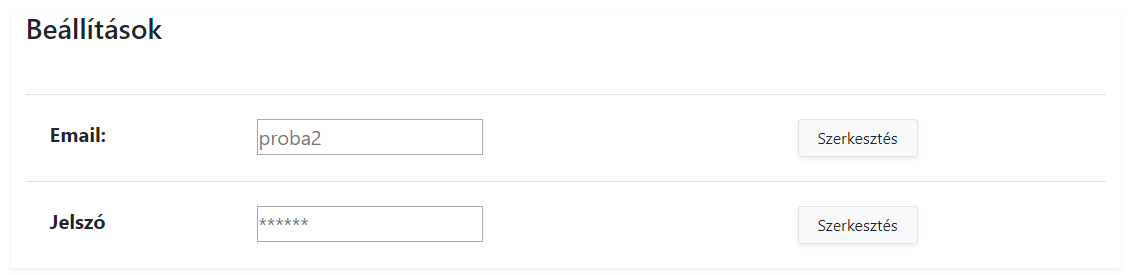
6.4.5. Profil kezelés

****Profilunk kezelésére a navigációs sávon megtalálható saját fiókom/ profilképre kattintva van lehetőségünk. Kattintás után megjelenik a profil oldal, ahol saját láthatjuk, és kezelhetjük saját adatainkat.

**Profilkép** => A fiókhoz tartozó profilképet a „Browse” gomb-ra kattintva választhatunk ki a számítógépünkről. Amennyiben sikeresen kiválasztottunk egy képet, a „Mentés” gomb kattintásával véglegesíthetjük a beállítást

**Saját hirdetések =>** A „Show” gomb megjeleníti a feladott hirdetéseinket. A hirdetések megtekintéséhez a hirdetés kártyáján található részletek gomb kattintásával válik lehetővé, a törlése pedig a törlés gomb megnyomásával.

**Email, és jelszó változtatás** => Az oldal alján a beállítások menüben van rá lehetőségünk.

Az emailt helyben is meg tudjuk változtatni, a jelszó egy új oldalra navigál minket. A regisztráció során már megadott beviteli feltételek itt is érvényesek!

6.5.6 Kapcsolat

A navigációs sávból elérhető, itt találhatunk információkat a weboldalról, gyakori kérdéseket, és válaszokat.

# 7. Használt programozási nyelvek

## 7.1. Asztali alkalmazás

**7.1.1. C#**

A C# az a programozási nyelv, ami a legközvetlenebb módon tükrözi az alatta működő, minden .NET programot futtató .NET keretrendszert, valamint erősen függ is attól: nincsen nem menedzselt, natív módban futó C# program. A primitív adattípusai objektumok, a .NET típusok megfelelői. Szemétgyűjtést használ, valamint az absztrakcióinak többsége (osztályok, interfészek, delegáltak, kivételek…) a .NET futtatórendszert használja közvetlen módon.

A C vagy C++ nyelvhez hasonlítva a C# több korlátozást és továbbfejlesztést is tartalmaz. A lehetőségei közül néhány:

• A mutatók és a nem ellenőrzött aritmetika csak egy speciális, nem biztonságos módban (unsafe mode) használható. A legtöbb objektum-hozzáférés csak biztonságos hivatkozásokon keresztül tehető meg, és az aritmetikai műveletek debug módban túlcsordulás szempontjából ellenőrzöttek.

• Az objektumok nem szabadíthatók fel közvetlen módon, ehelyett a szemétgyűjtő szabadítja fel őket, mikor már nincs rájuk hivatkozás. Ez a módszer kizárja a nem létező objektumokra való hivatkozás lehetőségét.

• A destruktorok (~) elérhetőek. A megfelelően megírt IDisposable interfész (Disposable programozási minta), aminek a lefutását garantálja using blokk, együtt kikényszerítheti az azonnali felszabadítást az osztályon belüli natív erőforrások esetében. A nem natív erőforrások felszabadítását ebben az esetben is a szemétgyűjtő (Garbage Collector) végzi. A finalizerek szintén rendelkezésre állnak, de nem váltanak ki azonnali felszabadítást. Finalizer a Dispose eljárás javasolt implementációs módozatában nem fut le, mivel az ajánlás szerint a szemétgyűjtőt utasítani kell a finalizer hívás kihagyására.

• A nyelv csak egyszeres öröklődést támogat, de egy osztály több interfészt is megvalósíthat.

• A C# sokkal típusbiztosabb, mint a C++. Az egyetlen implicit konverzió a biztonságos konverzió, úgy mint az egészek tágabb intervallumba konvertálása vagy a leszármazott osztályok alaposztályba konvertálása. Nincs implicit konverzió az egészek és a logikai típus (boolean) között, a felsorolás tagok és az egészek között. Nincsenek void mutatók (bár az Object osztályra mutató mutatók hasonlóak), valamint bármely, a felhasználó által definiált implicit konverziót explicit módon meg kell jelölni.

• A felsorolás adattagjai a saját névterükben helyezkednek el.

• A 2.0-s verziótól felfelé már rendelkezik a generikus programozás néhány eszközével.

• Tulajdonságok (Properties) használhatók, amelyek úgy tesznek lehetővé kódfuttatást mezők beállításakor és olvasásakor, hogy közben az adattagok szintaxisát használja.

## 7.2 Webalkalmazáshoz használt nyelvek

**7.2.1 HTML**

„A HTML5 a HTML (Hypertext Markup Language, a web fő jelölőnyelve) korábbi verzióinak az átdolgozott változata. A kifejlesztésének egyik fő célja, hogy a webes alkalmazásokhoz ne legyen szükség pluginek (pl. Adobe Flash, Microsoft Silverlight, Oracle JavaFX) telepítésére.”

**7.2.2 CSS**

„A A CSS (Cascading Style Sheets, magyarul: lépcsőzetes stíluslapok) a számítástechnikában egy stílusleíró nyelv, mely a HTML vagy XHTML típusú strukturált dokumentumok megjelenését írja le. Ezenkívül használható bármilyen XML alapú dokumentum stílusának leírására is, mint például az SVG, XUL stb. A CSS-t a weblapok szerkesztői és olvasói egyaránt használhatják, hogy átállítsák vele a lapok színét, betűtípusait, elrendezését, és más megjelenéshez kapcsolódó elemeit. A tervezése során a legfontosabb szempont az volt, hogy elkülönítsék a dokumentumok struktúráját (melyet HTML vagy egy hasonló leíró nyelvben lehet megadni) a dokumentum megjelenésétől (melyet CSS-sel lehet megadni). „

**7.2.3 PHP**

„A PHP egy általános szerveroldali szkriptnyelv dinamikus weblapok készítésére. Az első szkriptnyelvek egyike, amely külső fájl használata helyett HTML oldalba ágyazható. A kódot a webszerver PHP feldolgozómodulja értelmezi, ezzel dinamikus weboldalakat hozva létre. Rasmus Lerdorf 1995-ben indította útjára. Ma a The PHP Group tartja fenn és fejleszti. A PHP szabad szoftver, de licence nem csereszabatos a GNU licenccel, mivel megkötéseket tartalmaz a PHP név használatára.

A PHP születésekor csupán egy makrókészlet volt személyes honlapok karbantartására. Innen jön az eredeti név is: Personal Home Page Tools.A rövidítés jelentése később PHP: Hypertext Preprocessor lett, így rekurzívvá vált. Később a PHP képességei bővültek, így egy önállóan használható programozási nyelv alakult ki, amely képes nagyméretű webes adatbázisalapú alkalmazások működtetésére is.

A PHP nyelv népszerűsége képességeinek bővülésével folyamatosan nőtt. A NetCraft elemző cég felmérései szerint a PHP-t 1999 novemberében több mint 1 millió kiszolgálón használták. Ez a szám 2001 szeptemberére 6 millióra ugrott, 2003 októberében pedig már állítólag 14 millió gépre telepítették a PHP-t. A SecuritySpace.com szerint a PHP az Apache legnépszerűbb modulja, a mod\_ssl-t, a perl modult és a FrontPage-et is maga mögé utasítva. 2007 áprilisában több, mint 20 millió honlap és 1 millió szerver használta.

A hagyományos HTML lapokkal ellentétben a kiszolgáló a PHP-kódot nem küldi el az ügyfélnek, hanem a kiszolgáló oldalán a PHP-értelmező motor dolgozza fel azt. A programokban lévő HTML elemek érintetlenül maradnak, de a PHP kódok lefutnak. A kódok végezhetnek adatbázis-lekérdezéseket, létrehozhatnak képeket, fájlokat olvashatnak és írhatnak, kapcsolatot létesíthetnek távoli kiszolgálókkal. A PHP-kódok kimenete a megadott HTML elemekkel együtt kerül az ügyfélhez.

A PHP-t parancssori alkalmazásként is telepítik, így alkalmas parancsfájlok készítésére. Számos rendszergazda automatizálási célokra is a PHP-t használja, pedig ezt a feladatot hagyományosan Perl- vagy shell scriptekkel oldották meg.”

**7.2.5 JQUERY**

„A jQuery népszerű JavaScript könyvtár, mely a HTML kód és a kliensoldali JavaScript közötti kapcsolatot hangsúlyozza. 2006 januárjában jelentette meg a Mozilla Alapítvány népszerű JavaScript evangelistája, John Resig. A függvénykönyvtár MIT és GNU kettős licenc alatt jelent meg. A jQuery ingyenes, nyílt forrású szoftver.

Számos ismert IT cég is alkalmazza a jQuery-t saját projektjeiben, például a Microsoft erre építette a Visual Studióban is elérhető ASP.NET AJAX platformját.

A jQuery célja, hogy segítsen minél inkább leválasztani a JavaScript kódot a HTML-ről, és kényelmes kommunikációt biztosítson a weblap elemeivel – eseményvezérlők és azonosítók (ún. CSS szelektorok) használatával.”

**7.2.6 SQL**

„Az SQL, azaz Structured Query Language (strukturált lekérdezőnyelv) relációsadatbázis-kezelők lekérdezési nyelve.

Angol nyelvterületen 'eszkjuel' a kiejtése. A hagyományokhoz való hűség jegyében sokan 'szíkvel'-nek ejtik, ugyanis korábban Structured English Query Language (SEQUEL) volt az elnevezés, és ezt rövidítették le.

A relációsadatbázis-kezelők általában az SQL nyelven programozhatók. Az SQL alapvető utasításait közel egyformán valósítják meg, de a később beépült nyelvi elemek körében nagyon nagy az eltérés, az inkompatibilitás, emiatt számos SQL nyelvjárásról beszélhetünk.

Jellegét tekintve ez a szakterület-specifikus nyelv részben procedurális, részben deklaratív.”

# 8. Fejlesztői környezet

**A webalkalmazás teljes része** NetBeans IDE 8.2 alatt készült. „A NetBeans egy integrált fejlesztői környezet, ami a Java nyelven alapul. A program grafikus fejlesztőfelületet kínál a különböző alkalmazások, Appletek vagy akár JavaBeanek elkészítéséhez, amelynek segítségével könnyebben, gyorsabban tudjuk fejleszteni saját programjainkat.”

**Az alkalmazás asztali része** pedig Microsoft Visual Studio 2019 alatt készült. „A Visual Studio a Microsoft több programozási nyelvet tartalmazó fejlesztőkörnyezete, amely az évek során egyre több új programnyelvvel bővült. Jelenleg a F#, C++, C# (ejtsd: Szí-sárp) és Visual Basic programozási nyelveket, valamint az XML-t támogatja. A csomag része még a MASM (Microsoft Macro Assembler) is, ami részleges assembly támogatást biztosít.”

Az alkalmazás elkészüléséhez segítséget nyújtott

-Entity framework 6.4

-Bootstrap

-Sweetalert2

# 9. Összegzés

A záródolgozatom célkitűzése egy olyan szoftver csomag elkészítése volt, amely webalkalmazás részen használt áru hirdetési lehetőséget nyújt a felhasználó számára, asztali felületen pedig a felhasználói, és hirdetési adatok kezelését végzi. Az alkalmazások funkcionalitásukat tekintve elérték a nekik szánt célt, viszont még van hely a tovább fejlesztésre. A fejlesztés során folyamatosan újabb, és újabb ötletekkel bővült a tervezésem, ezeknek még nem sikerült minden részét megvalósítani. A későbbiekben mindenképpen szeretnék még hozzáadni kommunikációs lehetőséget a felhasználó-felhasználó között, ami egyszerűsítheti a cserék/eladások lebonyolítását, külön oldalt, ahol a saját hirdetéseket kezelni lehet, és lehetőséget a már lebonyolított hirdetések megtekintésére. A weboldalt megjelenés terén is tervezem még bővíteni, ezt animációk hozzáadásával, és képek, figyelemfelkeltő tartalmak beillesztésével szeretném elérni. Az asztali alkalmazásba hozzáadnék lehetőséget az adminisztrátor, és a felhasználók közötti kommunikációra. Ez megegyszerűsítené az adminisztrátor feladatainak elvégzését, hiszen így a felhasználói kéréseket sokkal egyszerűbben le lehetne bonyolítani.

Amikor elkezdtem a szakdolgozatom készítését még egy nehezen elkészíthető projektet láttam magam előtt. Ötleteimmel zsúfoltan ugrottam neki a kódsorok megírásához, de a program megvalósításhoz szükséges információk csak töredékével rendelkeztem, mind a PHP, és C# nyelvekben. Ezeknek hiánya folyamatosan csökkent a projektel eltöltött minden egyes nappal, és az óráimon tanultakkal, ahol rengeteg segítséget, kaptunk a megvalósítással, tervezéssel kapcsolatban. Ha újból kezdhetném a szakdolgozatom készítését jóval több időt fordítanák annak megtervezésére. Akkor még nem tudtam, de egy jó terv elkészítése elengedhetetlen, hosszú órákat menthet meg a fejlesztő számára. Én magam többször is beleestem a tervezéshiányosságainak hibáiba, amely kódsorok, és adatok többszöri újra írásával járt. Ez főleg az adatbázisomat érintette, ahol bizonyos mérföldkövek elérésével többször is be kellett látnom, hogy a jelenlegi adatbázisom kevés, vagy relációit tekintve nem működik. Mégis folyamatosan motivált voltam, az általam elkészített alkalmazás fejlődésének folyamatos látványa hatalmas ösztönző erővel irányított az új ismeretek, és tapasztalatok eléréséhez.

# 10. Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondai tanáraimnak és témavezetőimnek, Bálint Róbert, Gyuris Csaba, Boros Bence, és Rédai Dávid szaktanáraimnak. Oktatásukkal, és segítőkészségükkel folyamatosan hozzájárultak ismereteim megszerzésének, szakmai képességeim fejlesztésének.

# 11. Melléklet

Források:

Adatbázis kapcsolatok

<http://www.viszki.sulinet.hu/tananyagtar/informatika/Kapin/12_evf/adatbaziskezeles.pdf///>

Programozási nyelvek

<https://hu.wikipedia.org/wiki/C_Sharp>

<https://hu.wikipedia.org/wiki/Cascading_Style_Sheets>

<https://hu.wikipedia.org/wiki/HTML5>

<https://hu.wikipedia.org/wiki/PHP>

<https://hu.wikipedia.org/wiki/JQuery>

A projekt készítéséhez használt

bootstrap 4

sweetalert 2

animate.css