SZEGEDI SZC VASVÁRI PÁL   
GAZDASÁGI ÉS INFORMATIKAI TECHNIKUM

Az 5 0613 12 03 Szoftverfejlesztő és - tesztelő szakképesítés vizsgaremeke

**IT Bolt leltározási nyilvántartó programja**

Minden nyilvántartható

Készítette:

1. Benkő Ferenc

2. Pilisi István

Szeged

2023

**HALLGATÓI NYILATKOZAT**

Alulírott Benkő Ferenc és Pilisi István a Szegedi

Gazdasági Szakképző Iskola Vasvári Pál Tagintézménye hallgatója kijelentem, hogy a

IT Bolt leltározási nyilvántartó programja című záródolgozat a saját munkám.

Dátum 2023

aláírás

Tartalomjegyzék

[Bevezetés 4](#_Toc134002671)

[1. Fejlesztői dokumentáció 5](#_Toc134002672)

[A komplex alkalmazás rendszere: 5](#_Toc134002673)

[2. Az adatbázis tervezése, megvalósítása 5](#_Toc134002674)

[2.1 Használt program 5](#_Toc134002675)

[2.2 Az adatbázis tervezése és szerkezete 6](#_Toc134002676)

[2.2.1 BELÉPÉS TÁBLA: 7](#_Toc134002677)

[2.2.2 BOLT TÁBLA: 8](#_Toc134002678)

[2.2.3 VASARLAS TÁBLA: 8](#_Toc134002679)

[2.2.4 VASARLO TÁBLA: 8](#_Toc134002680)

[2.2.5 RAKTAR TÁBLA: 8](#_Toc134002681)

[2.2.6 LELTARI\_ESZKOZOK TÁBLA: 8](#_Toc134002682)

[2.2.7 ESZKOZOK TÁBLA: 9](#_Toc134002683)

[2.2.8 GYARTO TÁBLA: 9](#_Toc134002684)

[2.2.9 KATEGORIA TÁBLA: 9](#_Toc134002685)

[3. Asztali alkalmazás megtervezése és megvalósítása 10](#_Toc134002686)

[3.1 Asztali alkalmazás megvalósítása 10](#_Toc134002687)

[3.2 Néhány osztály jellemzése 12](#_Toc134002688)

[4. A Webes alkalmazás megtervezése és megvalósítása 15](#_Toc134002689)

[4.1 Webes alkalmazás megvalósítása 15](#_Toc134002690)

[4.2 A Webes alkalmazás néhány főbb osztálya 17](#_Toc134002691)

[5 Megállapítások 18](#_Toc134002692)

[6 Az elemzések értékelése 19](#_Toc134002693)

[7 Következtetések, javaslatok 23](#_Toc134002694)

[Összegzés 24](#_Toc134002695)

[Ábrajegyzék 25](#_Toc134002696)

[Irodalomjegyzék 26](#_Toc134002697)

# Bevezetés

1. Az általunk készített projektmunka egy leltározást segítő logisztikai projekt a boltok, áruk, vásárlások és egyéb adatok nyomon követésére és esetleges lekérdezése céljából. Az asztali alkalmazásban egy adatbázisban tárolt adatokat lehet megjeleníteni az a boltban lévő eszközökről, boltokról, raktárakról. Ezeket az adatokat módosítani, törölni és új adatokat felvinni lehetséges felhasználótól függően melyet az adat-bázisban el is tárol a program. A webes alkalmazásban ugyanúgy a boltok, áruk és raktárak információi találhatók, illetve a látogatók számára néhány fontosabb információval. A webes alkalmazásban van lehetőség a munkatársaknak, raktári dolgozóknak nyomon követni a raktári készleteket. A webes alkalmazást elsősorban PDA raktári rendszerekhez terveztük. Azért választottuk ezt a témát, mert raktározással kapcsolatosan eléggé sok tapasztalatunk van és ezzel a projekttel könnyen segíteni tudjuk a logisztikai cégek munkáját.

# 1. Fejlesztői dokumentáció

Mivel már előre elhatároztuk, hogy egy komplex alkalmazást szeretnénk elkészíteni, viszont maradt néhány tisztázandó kérdés. A legfontosabb kérdés az volt, hogy milyen programozási nyelveket használjunk fel a projekt megvalósítása során. Választásunk egyértelműen azokra a programokra és programnyelvekre irányult, amiket a szakképzés során tanultunk. A programok készítése során C# nyelv és Vue.js keretrendszert amelyen belül Javascript, HTML CSS volt használva. Az adatbázishoz klasszikus SQL nyelvet használtunk amit phpMyadminon valósítottunk meg XAMP program segítségével.

Mind a Webes és mind az asztali alkalmazás esetén a Visual Studio Community Edition fejlesztői környezetet használtunk ami nagyban megkönnyítette a munkánkat és a fejlesztési folyamatok nyomon követését Githubon. Ennek a környezetnek a segít segítségével gyorsan és precízen tudtunk dolgozni, a beépített funkciói miatt.

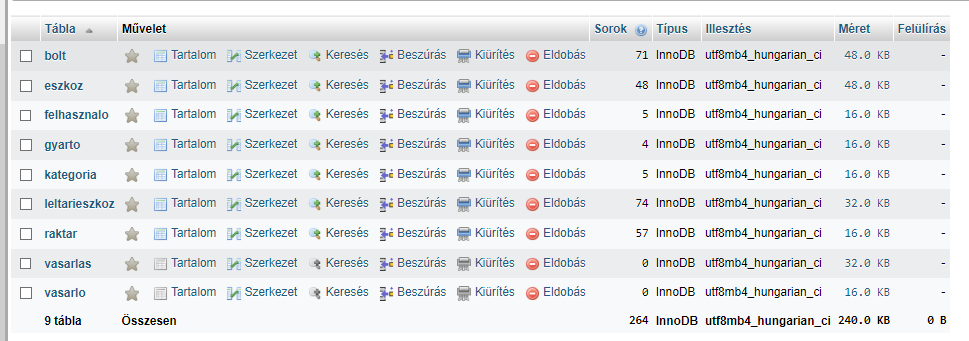
### A komplex alkalmazás rendszere:

Az alkalmazás három fő részből tevődik össze. Ezek egy webes felületből, egy asztali alkalmazásból és egy adatbázisból állnak. Ez a három főbb rész szorosan együttműködve alkotnak egy rendszert. Az Adatbázis egyik előnyös tulajdonsága, hogy önmagában is alkalmazható, exportálható és importálható. Az asztali alkalmazásban felvitt adatok ugyan úgy megjelennek a webes alkalmazásban és fordítva is. A már létező eszközöket, boltokat, raktárakat lehet módosítani és törölni.

# 2. Az adatbázis tervezése, megvalósítása

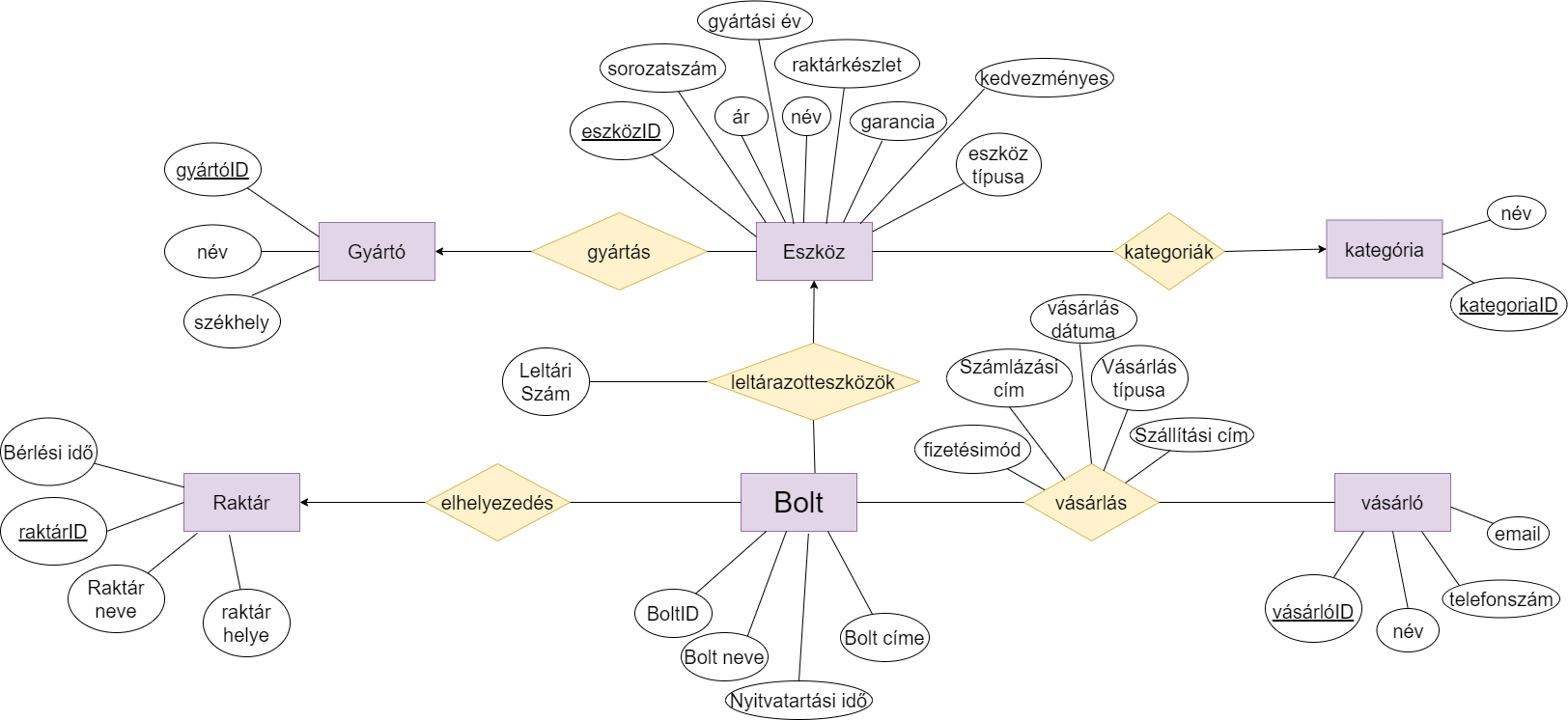
### 2.1 Használt program

Az adatbázisunkat a XAMPP nevű programban készítettük el, mert nyílt forráskódú és ingyenes program amely könnyen kezelhető. Ezzel a programmal voltunk képesek arra, hogy elérjük az adatbázist, valamint arra is, hogy a tervek alapján elkészítsük az adatbázisunkat. PhpMyAdmin-t, az Apache és MySQL bekapcsolásával, az admin gombra kattintva jelent meg a PhpMyAdmin, ahol az adatbázis és a táblák szerkezetét tudtam létrehozni és módosítani. A már kész adatbázisnak itt tudtuk különböző nézeteit megtekinteni és később a prezentációban ábrázolni a táblák közötti kapcsolatot ezeknek a nézetek segítségével.

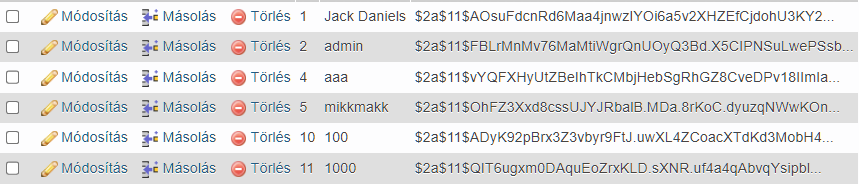


1. ábra: A teljes adatbázis

### 2.2 Az adatbázis tervezése és szerkezete

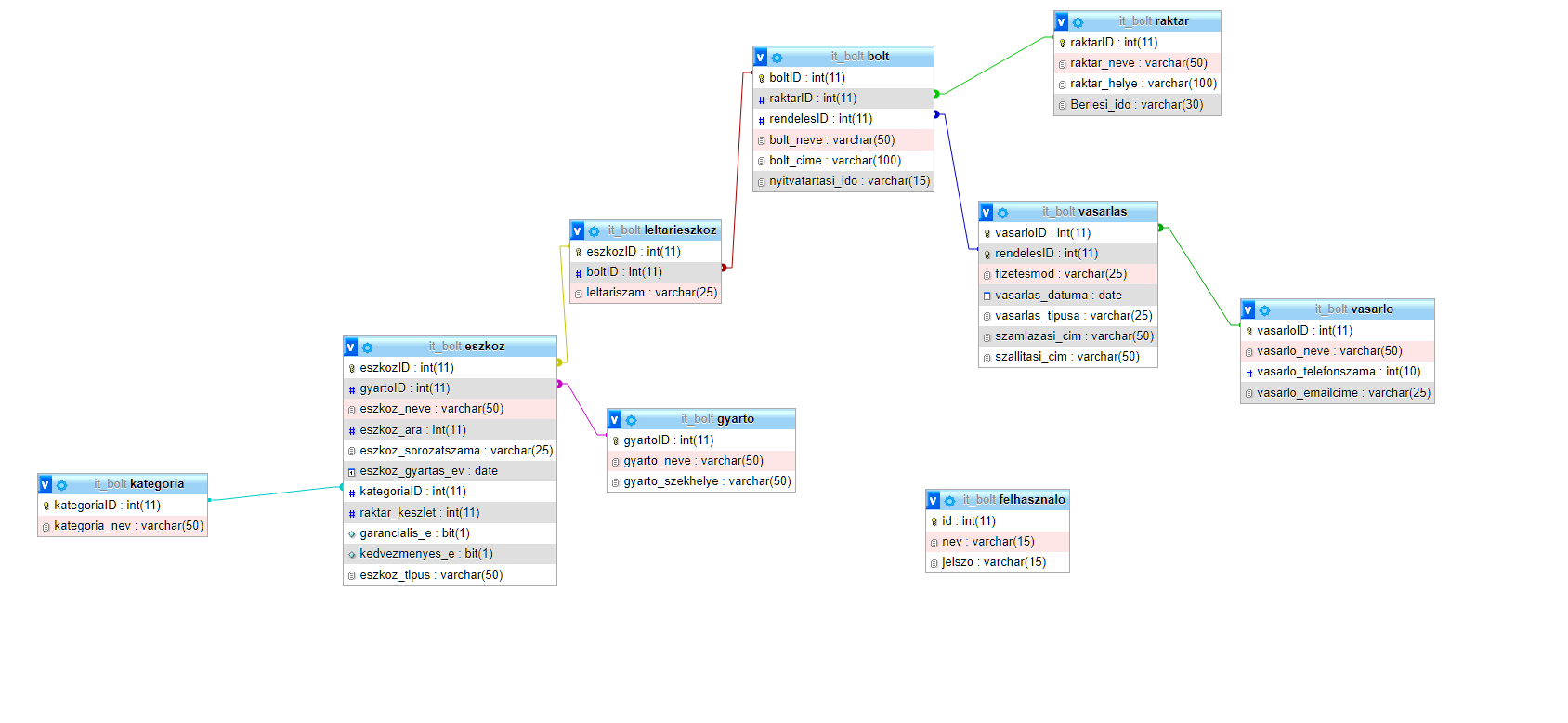
1. Az adatbázis tervezését előszőr „E-K” diagramm segítségével oldottuk meg. „Az E-k diagramm Egyedosztályok, kapcsolatok, típusok, egyéb feltételezések ábrázolását jelenti”[1].
2. 

2. ábra: Az adatbázis E-K diagramja

1. A közösen létrehozott adatbázis a „PhpMyAdmin” oldalon található meg, melynek neve „it\_bolt”. Az adatbázis külön táblában tartalmazza a felhasználókat és a jelszavakat, amik nincsenek összeköttetésben más táblákkal. Ennek első sorban adatbiztonsági okai vannak, hogy egy esetleges összetett lekérdezésnél ne jelenjenek meg a felhasználók adatai illetéktelenek számára.
2. 

3. ábra: A felhasználói adatok táblája az adatbázisban

Az adatbázisunk rengeteg táblából áll, sajnos arra már nem volt további kapacitásunk és időnk , hogy mindet a komplex alkalmazásba beleépítsük (ezt a későbbi fejlesztések során gondoljuk megvalósítani, ezzel kibővítve és tovább fejlesztve a már meglévőt). Rengeteg adatott kellett felvennünk minden általunk használt táblához legalább 50 darabot, hogy megfelelően tudjuk tesztelni a programunkat és a dokumentációt elkészíteni.

1. 

4. ábra: Az adatbázis Bachmann-ábrája

Az Adatbázisunk főbb tábláinak tartalma:

### 2.2.1 BELÉPÉS TÁBLA:

id: szám (INT) a felhasználó id-ja

nev: szöveg (VARCHAR(15) – a felhasználó belépési neve

jelszo: szöveg (VARCHAR(15) – a felhasználó belépési jelszava

### 2.2.2 BOLT TÁBLA:

* 1. **boltID**: szám (INT) –a bolt azonosítója - elsődleges kulcs

1. nev : szöveg (VARCHAR(50) – a bolt megnevezése
2. cim :szöveg (VARCHAR(100) – a bolt címe
3. nyitvatartasiIdo: dátum (DATE) – a bolt nyitvatartási ideje
4. raktarID: szám (INT) –a raktár azonosítója - idegen kulcs
   * 1. rendelesID: szám (INT) –a rendelés azonosítója idegen kulcs

### 2.2.3 VASARLAS TÁBLA:

1. ***vasarloID****: szám (INT)–a vásárló azonosítója - elsődleges kulcs*
2. *rendelesID: szám (INT)–a rendelés azonosítója - elsődleges kulcs*
3. fizetesmod: szöveg (VARCHAR(25) – fizetésmód (készpénz, bankkártya, stb.)
4. vasarlasDatuma: dátum (DATE) – a vásárlás dátuma
5. vasarlastipus: szöveg (VARCHAR(25) – vásárlás típusa (helyben, online, stb.)
6. szamlazasi\_cim: szöveg (VARCHAR(50) – számlázási cím
7. szallitasi\_cim: szöveg (VARCHAR(50) – szállítási cím

### 2.2.4 VASARLO TÁBLA:

1. **vasarloID**: szám (INT) –a vásárló azonosítója - elsődleges kulcs
2. nev : szöveg (VARCHAR(50) – a vásárló neve
3. telefonszam: szám (INT) – a vásárló telefonszáma
4. email: szöveg (VARCHAR(50) – a vásárló email címe

### 2.2.5 RAKTAR TÁBLA:

1. **raktarID**: szám (INT)–azonosító - elsődleges kulcs
2. nev: szöveg (VARCHAR(50) – a raktár megnevezése
3. helye: szöveg (VARCHAR(100) – a raktár címe
4. berlesiIdo: dátum (DATE) – a raktár bérlésének ideje

### 2.2.6 LELTARI\_ESZKOZOK TÁBLA:

1. eszkozID: szám (INT) –a eszköz azonosítója - elsődleges kulcs
2. boltID: szám (INT) –a bolt azonosítója - idegen kulcs
3. leltariszam: szöveg (VARCHAR(25) –a eszköz azonosítója

### 2.2.7 ESZKOZOK TÁBLA:

1. **eszkozID**: szám (INT) –a eszköz azonosítója - elsődleges kulcs
2. nev: szöveg (VARCHAR(50) – a eszköz megnevezése
3. ar: szám (INT) – az eszköz ára
4. sorozatszam: szöveg (VARCHAR(25) –a eszköz sorozatszáma
5. gyartasiev: dátum (DATE) – az eszköz gyártási éve
6. eszkoztipusa: szöveg (VARCHAR(50)) – eszköz típusának megnevezése
7. kedvezmenyes-e: logikai (BINARY) – 1 ha kedvezményes az eszköz, 0 ha nem
8. garancialis-e: logikai (BINARY) – 1 ha van garancia az eszközön, 0 ha nem
9. raktarkeszleten-e: logikai (BINARY) – 1 ha van raktáron az eszköz, 0 ha nem
10. gyartoID: szám (INT) –a gyártó azonosítója - idegen kulcs
11. kategoriaID: szám (INT)–a kategória azonosítója idegen kulcs

### 2.2.8 GYARTO TÁBLA:

1. **gyartoID**: szám (INT)–a gyártó azonosítója - elsődleges kulcs
2. nev: szöveg (VARCHAR(50) – a gyártó megnevezése
3. szekhely: szöveg (VARCHAR(50) – a gyártó címe

### 2.2.9 KATEGORIA TÁBLA:

1. **kategoriaID**: szám (INT)–a kategória azonosítója - elsődleges kulcs
2. kaegorianev: szöveg (VARCHAR(50) – a kategória megnevezése

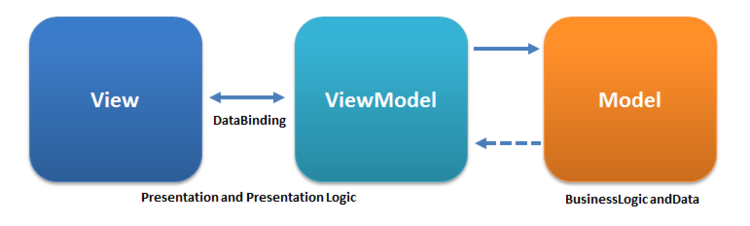
## 3. Asztali alkalmazás megtervezése és megvalósítása

Az asztali alkalmazás interface kialakítása során figyelembe kellett vennünk olyan szempontokat, mint a tiszta programkód, a könnyű kezelhetőség az átláthatóság és az egyszerű alkalmazhatóság.  Ezek alapján egy viszonylag egyszerű felépítésű és kinézetű alkalmazásnak az ötlete született meg.

### 3.1 Asztali alkalmazás megvalósítása

Az asztali alkalmazást C# nyelven írtuk meg. Az alkalmazást alapjáraton mindig is ebben a programozási nyelvben képzeltük el, mivel órákon ezt a nyelvet tanultuk és alkalmaztuk legtöbbször, illetve ezt a nyelvet találtuk a lehető legmegfelelőbbnek egy „desktop” alkalmazás elkészítéséhez. Fejlesztői környezetnek a „Microsoft Visual Studio”-t választottuk, mivel ebben a környezetben programoztunk a legtöbbet és az otthoni és iskolai gépeken is ez volt feltelepítve, vagy bármikor eltudtuk érni, ha szükségét éreztük. A legnagyobb gondot és odafigyelést minden esetben a verziószámok követése adta, mivel ha ezek nem egyeztek a programok nem tudtak helyesen lefutni. Ezért közös megállapodás alapján mindig a legfrisseb csomagokat és frissítéseket használtuk (például a NET.7 verzióját).

„Az architektúra azon döntések összessége, amik a kódban realizálódnak. Lényegében mondhatni kódolási szabályok, amelyeket a szoftver elején lefektetünk. Ezeket a szabályokat nagy valószínűséggel nem a nulláról kell kitalálnunk, hiszen architektúrára is vannak tervezési minták.”[2]



5. ábra: MVVM struktúra ábrája [3]

Az alkalmazás készítésénél MVVM struktúrát használtuk, ami alapján eltudtuk különíteni az alkalmazás logikai programrészét a felhasználói felülettől.. MVVM programozás előnye, hogy a program logikai része az változatlan marad és így bármilyen változtatás történik a felhasználói felület programkódján belül nincs kihatással a programunk logikai részére. Az ITBolt.api amiben a Controllerek találhatók az adatbázis alapján modelleztük le ez alkotja a Backend alapját.

A controllerek: feladata, kapcsolatot teremteni adatbázis és a View réteg között. A lekérdezett adatokat úgy alakítják át a Controllerek, hogy azok könnyen megtudjanak jelleni a View oldalakon. Az itt lévő metódusok és függvények biztosítják továbbá az adatbázisba történő adatok feltöltését és módosítását is.

A Views mappájában találhatóak azok az XML fileok, amelyek az adatok bekérésén, és a szükséges controller függvények meghívásán kívül kezelnek még elágazásokat is.A további feladata még a felhasználói számára történő alkalmazhatóság. Bindingok segítségével hatására meghívja vagy lekéri a megfelelő osztályok funkcióit, mind ezek alapján módosul a felületet. Az input és az mely paraméterek kezelésére szolgál a controllereknek majd a modellek számára.

A ViewModels mappa olyan tulajdonságokat és filokat tartalmaz, amelyek adatkötést végeznek a View-hoz és változások esetén módosítások történnek a View-ban. A ViewModels felelős a Views és a modellosztályok összehangolásáért, úgy hogy az adatokat összekapcsolja.

A Models mappában találhatóak azok az osztályok, melyeket a folyamatok során a controllerek osztályai felhasználhatnak, ezek az adatbázis csatlakozásával kapcsolatosak. Alapvető feladata az adatok tárolása, és az azokon végezhető elemi műveletek támogatása. Ezt a részt már meglévő MYSQL adatbázis alapján generáltuk le. Ez a réteg biztosít kapcsolatot az adatbázis szerverrel, valamint tartalmaz minden olyan függvényt, ami képes az adatok lehívására az adatbázisból és azok eltárolását teszik lehetővé adattáblákban.

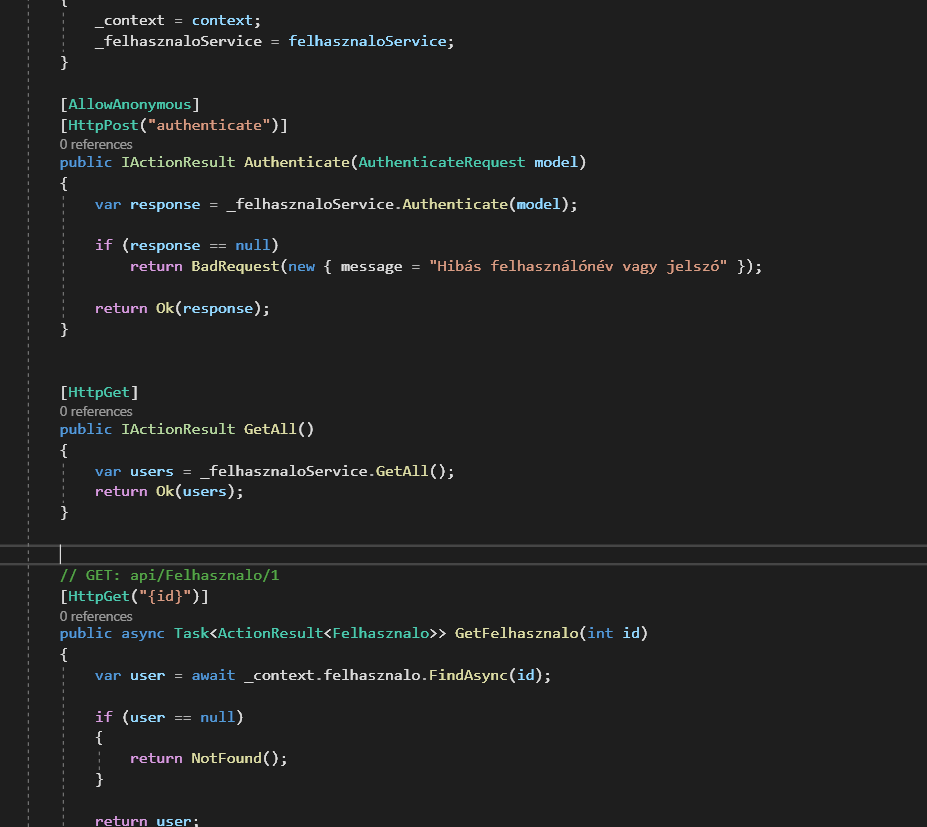
Tervezett funkciók az Asztali alkalmazáshoz:

* Belépés
* Boltok klistázása
* Boltok keresése
* Boltok adatainak módosítása
* Új bolt felvitele
* Bolt törlése
* Eszközök kilistázása
* Eszközök keresése
* Új eszköz felvitele
* Eszköz törlése
* Eszközök adatainak módosítása
* Raktárak kilistázása
* Raktárak keresése
* Raktárak adatainak módosítása
* Új raktár felvitele
* Raktár törlése

### 3.2 Néhány osztály jellemzése

„Az OOP világában egy osztály olyan adatok és műveletek összessége, amellyel leírhatjuk egy modell (vagy entitás) tulajdonságait és működését. Egy osztálynak vannak tulajdonságai. Amikor programot írunk, akkor az adott osztályból (osztályokból) létre kell hoznunk egy (vagy több) példányt, ezt példányosításnak nevezzük.”[4]

A FelhasznaloController osztály végzi az Entities mappában található Felhasznalo.cs segítségével a felhasználók beléptetését úgy, hogy a „BelepesView” ablakból lekért felhasználónevet és jelszót összeveti az adatbázisban lévőkkel, és amennyiben talál ilyet, megkeresi, hogy a bejelentkezni kívánó felhasználó jogosult –e a belépéshez. További funkciója a belépésnek a validálás, hogy megfelelően adta meg –e a felhasználó a jelszavát és az azonosítóját. A sikeres bejelentkezés esetén aktívvá válik a többi Views amely a megfelelő ViewModell segítségével a Controller osztályokat használják.

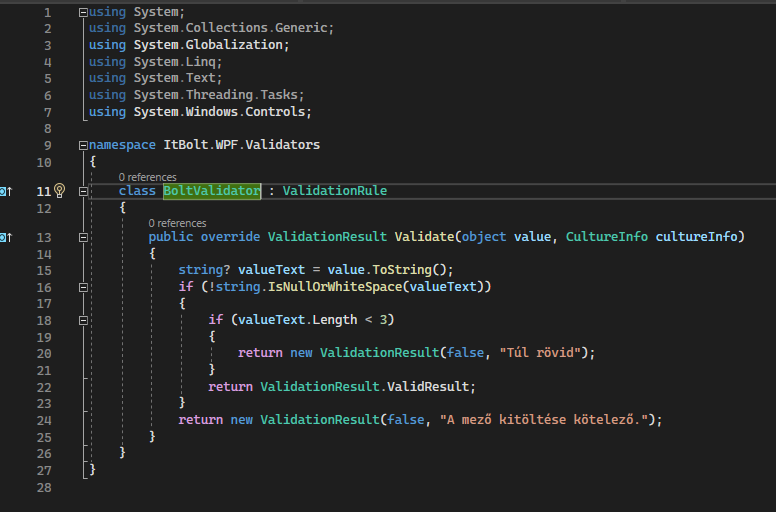
******

6. ábra: Ábra a FelhasznaloController.cs osztályról

A többi View XML file hasonlóan működik ott is az Entities és a Controllers mappában lévő osztályok segítségével történik az adatok lekérdezése, felvitele, módosítása.

A „validátor” osztályok szerepe, ellenőrzik hogy a program futásakor a felhasználó megfelelő adatokat adott –e meg.

Az egyik ilyen validator osztály a BoltValidator.cs amely ellenőrzi a bolt nevének a hosszát és azt, hogy minden adatot kitöltött –e a felhasználó.



7. ábra: BoltValidator.cs osztály ábrája

„A ViewModels olyan tulajdonságokat és parancsokat implementál, amelyekhez a nézet adatkötést végezhet, és a változásértesítési eseményeken keresztül értesíti az állapotváltozások nézetét. A nézetmodell által biztosított tulajdonságok és parancsok határozzák meg a felhasználói felület által kínált funkciókat, de a nézet határozza meg a funkció megjelenítésének módját.”3 Afő nézetek megjelenítéséért a mainviewmodel a felelős amely tartalmazza a szolgáltatások regisztrálását és az összekötését, ha ez nincs nem jöhetne létre a View változtatása és így a többi VieWmodelből való adatok lekérését.

Ilyen osztály például a BoltKimutatasViewModel.cs, amely a boltok táblának a kimutatásaira van hatással.

## 4. A Webes alkalmazás megtervezése és megvalósítása

Az asztali alkalmazást C# nyelven írtuk meg, a cél az volt, hogy a webes alkalmazás hasonló legyen mint az asztali alkalmazás, ezért Vue keretrendszert használtunk az alkalmazásunk felépítéséhez.

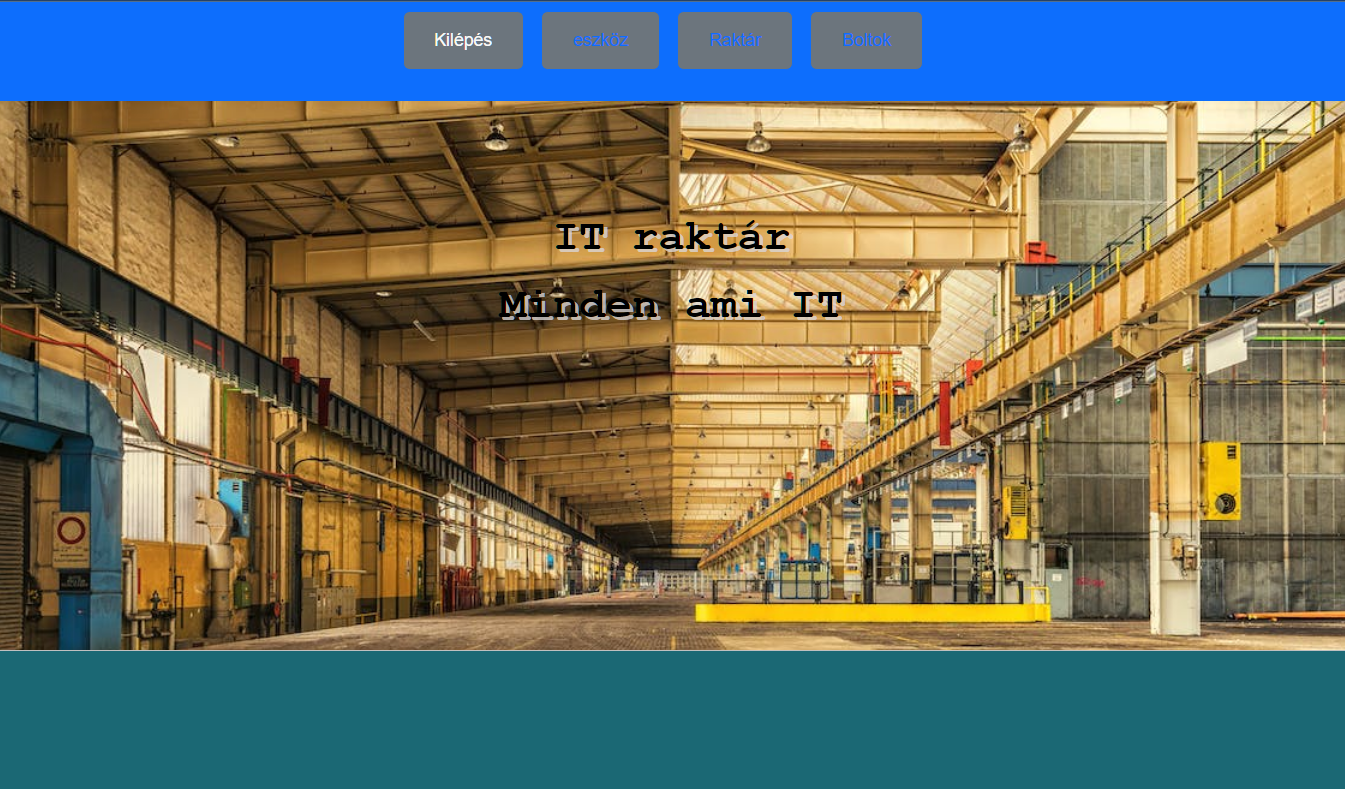
### 4.1 Webes alkalmazás megvalósítása

A webes alkalmazás megvalósításához három statikus weboldalt készítettünk az egyszerű látogatók részére és egy beléptető rendszert, amin keresztül elérhetővé válik a dolgozók számára az adatbázis adatainak lekérése.A weboldalaknak alkalmazkodnia kell a megjelenítendő eszközök (főleg PDA) felbontási képességeikhez. Ezért az oldalakat Bosstrep megfelelő osztályait használva a alkalmaztuk az elemeken, amelyeket az alkalmazás fejlesztés kezdetén telepítettünk.A bootstrap egy olyan reszponzív rácsnézetet használ mely előre definiált töréspontokat határoz meg. Ezek paramétereinek megváltoztatásával változtatható az oldal megjelenése. A reszponzív designon kívül a Bootstrap segítségével számos más előre definiált class-okat tudtunk használni, amik segítségével könnyen meglehetett formázni a weboldalunkat. Ezek segítségével értük el, hogy milyen töréspontok szerepeljenek a különböző eszközök esetén, valamint a menüsáv kialakítása is Bosstrep segítségével történt.

Továbbá a style részben alkalmazni lehet különféle CSS módosításokat, amelyek esztétikusabb külsőt kölcsönöznek a weboldalainknak.

A weboldal menüszerkezete a felső navigációs sávban lett kialakítva. A kezdőlap menüpontjai olyan hivatkozásokra irányítanak melyek a kezdőoldalon megfogalmazott információkat.A navigációs menüsávból két verzió van az első amit az átlag látogató láthat a másik amit belépés után a felhasználó tud kezelni. A belső navigációs részről bármikor átléphetünk a külső részre, viszont a belsőre csak bejelentkezés után.

A kijelentkezés gomb a felső menüsáv bal szélén helyezkedik el.



8. ábra: Webes alkalmazás kezdő felület

A webalkalmazás Backend részéhez a már meglévő adatbázist és ITBolt.API-t használtuk fel.

A lekérdezések minden oldalnál a script részben történnek és ezeket a lekérdezéseket használtuk fel később az oldalakon az adatok megjelenítéséhez.

Tervezett funkciók a webes alkalmazáshoz:

* Belépés
* Boltok kilistázása
* Boltok keresése
* Boltok adatainak módosítása
* Eszközök kilistázása
* Eszközök keresése
* Eszközök adatainak módosítása
* Raktárak kilistázása
* Raktárak keresése
* Raktárak adatainak módosítása

A tervezés során további funkciók is voltak tervezve, de ezeket idő és kapacitás hiány miatt ki kellett hagynunk a projektünkből. Ezek voltak a vásárlók adatainak lekérdezése, vásárlások dokumentálása. Továbbá tervbe volt véve, hogy az eszközöket részletesseb információkat biztosítunk az eszközökről, mint gyártók és az eszköz kategóriáinak bemutatása lekérdezés során.

### 4.2 A Webes alkalmazás néhány főbb osztálya

src könyvtár elemei:

views: itt találhatók azok a vue filok amik a kinézetét adják a honlapnak.

components: Ebben találhatók a menüsávok vue fileai.

assets: itt találhatók a felhasznált képek és design elemek amelyek az oldal kinézetét szebbé teszik.

router: az ebben lévő index.js file biztosítja az útvonalakat az oldalak között



9. ábra: Webes alkalmazás eszközök felületének kódja

index.html: Az alkalmazásunk csontváza, amely egy egyszerű html file amin belül fog élni az alkalmazásunk. A másik fő funkciója, pedig a javascript gyökerét is elérhetővé teszi src="/src/main.js” segítségével.

package.json: függőségek és fejlesztési függőségeket tartalmaz

az src könyvtár és a benne található alkönyvtárak közül a legfontosabbak:

## 5 Megállapítások

A tervezés során a programokkal kapcsolatos főbb megállapítások:

Feltétlenül szükséges volt egy új tábla felvitele az adatbázisba, amely tartalmazza a felhasználó adatait.

Az iskolairendszer és az otthoni számítógépeken nem ugyan az a verziójú Microsoft Visual Studioműködik ezért csak otthon tudtuk elvégezni a szükséges módosításokat.

## 6 Az elemzések értékelése

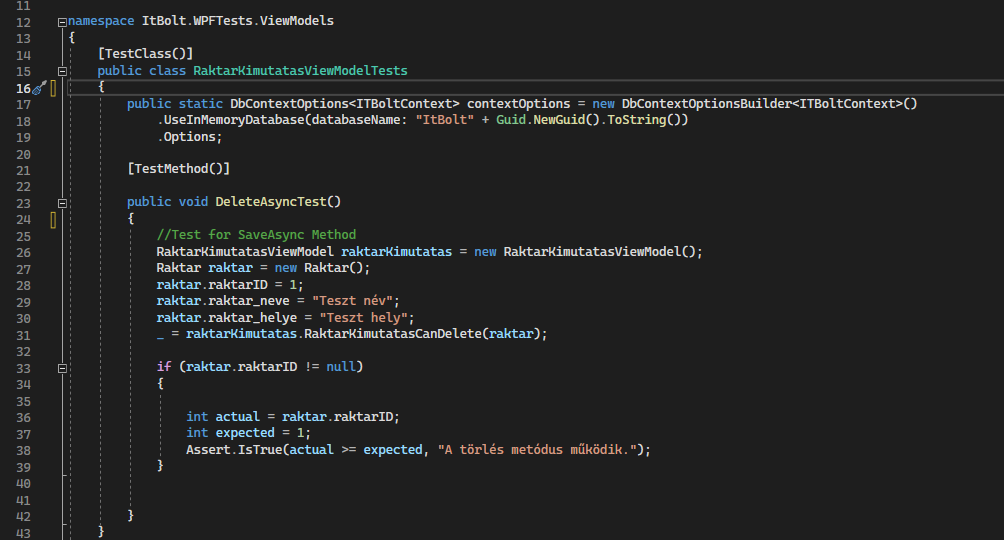
A különböző tesztek és eredményeik kiértékelése:

Egységtesztek:

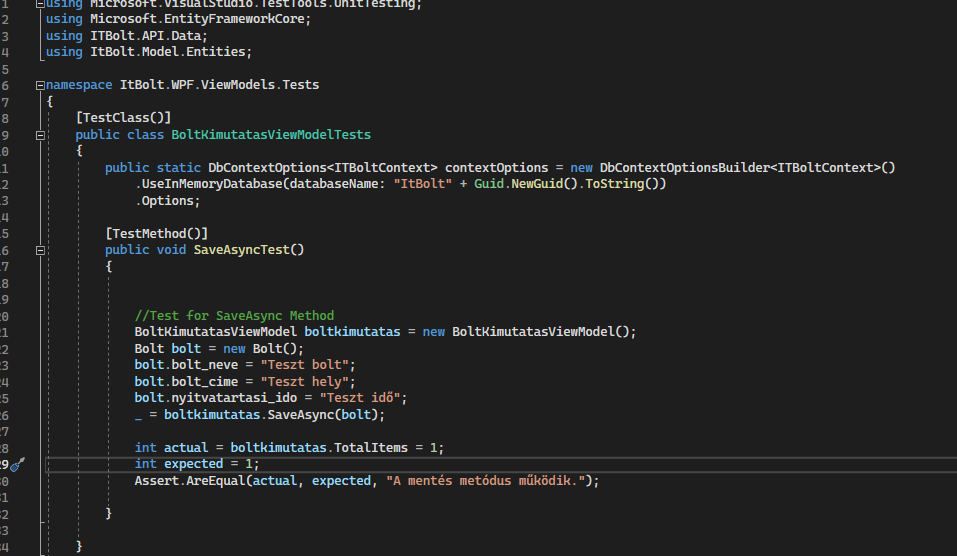
Az egységtesztelések során az alábbi teszteket hajtottuk végre.

ViewModel osztályokat vizsgáltuk meg, hogy az általunk megadott adatokkal. Az adatfelvétel és a törlés megvalósítható –e.

Mind a két esetben úgy tapasztaltuk, hogy a törlés és az adat felvétel egyaránt működött. A két teszt sikeres volt.



10. ábra: törlési test példa



11. ábra: mentési test példa

Rendszertesztek:

Rendszer tesztelés során az alábbi teszt-eseteket néztük meg és értékeltük ki:

Lekérdezések Teszt esete:

A táblák lekérdezése, hogy minden adat megjelenik –e és az adatok száma és értékeik megegyeznek –e az adatbázisban lévő adatok értékeivel.

Tapasztalat: Az oldalon mindig adott számú és értékű adat jelent meg.

Eszt a tesztelést mind a 3 táblára elvégeztük, a megfelelő adatokkal.

|  |  |
| --- | --- |
| **Tesztelési jegyzőkönyv** | |
| **A teszteset leírása és célja:** | Bolt adatok lekérdezése az adatbázisból. |
| **A tesztelt folyamat / funkció leírása:** | A szerverről lekérdezni az adatokat, hogy megjelenhessenek a megfelelő felületen. |
| **A tesztelés előfeltételei:** | Számítógép, Frontend-Backend kapcsolat, Frontend felület megléte. |
| **A tesztelés dátuma és ideje:** | 2023-03-20 |
| **A tesztadatok típusa:** | String |
| **A tesztet végző személy(ek):** | Benkő Ferenc |
| **A tesztelt rendszer beállításai:** | Windows 10 alatt futott program.  Xamp futtatása közben.  Microsoft visual studio segítségével  Amelyben ai ITBolt.API  ITBolt.Wpf egyzerre fut |
| **A teszteset elvárt eredménye:** | A táblában lévő boltok megjelenése  névvel címmel és nyitvatartással. |
| **A tesztelés eredménye:** | 🞏 megfelelt  🞏 nem felelt meg  🞏 megfelelt megjegyzésekkel |
| **Megjegyzések:** |  |
| **A tesztelési jegyzőkönyvet készítette (név, aláírás):** | Benkő Ferenc |

12. ábra: jegyzőkönyv példa

Adatbevitel teszt esete:

Mindegyik táblára a beviteli menüpontokon több adatot is felvittünk. Az elvárt eredmény az volt, hogy jelezzen, a rendszer, ha rossz adatot akartunk felvinni, illetve a mentés után az adatok kerüljenek be az adatbázisba.

Tapasztalat: Az oldalon minden esetben megjelentek az adatok, ha jól adtuk meg őket. Ellenkező esetben nem jelentek meg és hibás üzenetet kaptunk vissza.

Eszt a tesztelést mind a 3 menüpontban elvégeztük, a megfelelő adatokkal.

Belépés teszt esete:

Vizsgáltuk a beléptető rendszert, hogy csak a megfelelő adatokkal tudunk belépni. Megfelelő adat az adatbázisban eltárolt felhasználói adatok voltak.

Tapasztalat: A fő menübe csak abban az esetben tudtunk belépni, ha megfelelő felhasználói adatokat adtunk meg ellenkező esetben nem tudtunk belépni és hibaüzenetet kaptunk.

Ezt a tesztelést három felhasználó adataival végeztük el amelyből 2 létező felhasználó volt az adatbázisban egy pedig fiktív.

## 7 Következtetések, javaslatok

Az általános következtetésünk, hogy programunk egy kis és közép vállalkozás általános adatrögzítési és elemzési feladatait eltudja végezni.

A későbbiek során alkalmas lehet a projektünk, hogy további problémák megoldását is képes legyen megoldani. Akár logisztikai, vagy egyéb vállalat irányítási feladatok elvégzésére. Ezért a későbbiekben tervezzük további funkciók üzembehelyezését a projektünkben. Illetve az időkorlát miatti elmaradt funkciók megvalósítását.

# Összegzés

1. A projekt fejlesztése során sokat fejlődtünk mind programozás mind csapatmunka terén. Több olyan megoldást is találtunk, amelyet nem vagy csak részlegesen vettünk tanulmányaink során, de ezeket és a már meglévő tudásunkat jól tudtuk kamatoztatni a projekt megvalósítása közben. A vizsga remekünk jó alapot ad a tovább fejleszthetőségre és a tovább fejlődéshez. A későbbiekben szeretnénk a projektet tovább fejleszteni Androidra illetve Mauis alkalmazásként.

# Ábrajegyzék

[1. ábra: A teljes adatbázis 6](#_Toc134002575)

[2. ábra: Az adatbázis E-K diagramja 6](#_Toc134002576)

[3. ábra: A felhasználói adatok táblája az adatbázisban 7](#_Toc134002577)

[4. ábra: Az adatbázis Bachmann-ábrája 7](#_Toc134002578)

[5. ábra: MVVM struktúra ábrája [3] 10](#_Toc134002579)

[6. ábra: Ábra a FelhasznaloController.cs osztályról 13](file:///C:\Users\36304\Desktop\Benkő_Pilisi_prezentácio_2_14b\Benkő%20Pilisi%20Szakdolgozat%20Fejlesztői.docx#_Toc134002580)

[7. ábra: BoltValidator.cs osztály ábrája 14](#_Toc134002581)

[8. ábra: Webes alkalmazás kezdő felület 16](#_Toc134002582)

[9. ábra: Webes alkalmazás eszközök felületének kódja 17](#_Toc134002583)

[10. ábra: törlési test példa 19](#_Toc134002584)

[11. ábra: mentési test példa 20](#_Toc134002585)

[12. ábra: jegyzőkönyv példa 21](#_Toc134002586)

# Irodalomjegyzék

1. dr. Attila Kiss Adatbázis tervezés

2 <https://csharptutorial.hu>

3. <https://en.wikipedia.org/wiki/Model–view–viewmodel>

4. Reiter István C# programozás lépésről lépésre