

## DEBRECENI SZC BEREKSZÁSZI PÁL TECHNIKUM

4032 Debrecen, Jerikó u. 17.  
OM azonosító: 203033



Tel.: 52/ 503-150

Fax: 52/314-204

<http://www.dszcberegszaszi.hu>

E-mail:

[titkarsag@dszcberegszaszi.hu](mailto:titkarsag@dszcberegszaszi.hu)

---

Szakképesítés megnevezése: Szoftverfejlesztő  
OKJ száma: 54213 05

# ZÁRÓDOLGOZAT RAKTÁRKEZELŐ PROGRAM

**Készítette:**  
VÁCZI ZSOLT

**Konzulens:**  
KOVÁCS ATTILA RÓBERT

**Debrecen, 2021**

# Tartalomjegyzék

A. Felhasználói dokumentáció.....	3
1. Specifikáció.....	3
2. Telepítés.....	4
3. Rendszerkövetelmények.....	6
4. Bejelentkezés.....	7
5. Raktáros kezelőfelület.....	10
6. Műszerész kezelőfelület.....	14
7. Gazdálkodó kezelőfelület.....	18
B. Fejlesztői dokumentáció.....	21
1. Bevezetés.....	21
2. Specifikáció.....	22
3. Használt szoftverek.....	24
4. MySql adatbázis.....	25
5. A program felépítése.....	31
5.1 Közös osztályok.....	31
5.2 Belépés:.....	37
6. Tesztelés:.....	41
7. Továbbfejlesztési ötletek.....	45
8. Irodalomjegyzék.....	45

# A. Felhasználói dokumentáció

## 1. Specifikáció

Ez egy több felhasználós raktárkezelő program, amely három különböző munkakör együttműködését támogatja. A három munkakör:

1. gazdálkodó, aki a rendeléseket, beszállításokat intézi
2. műszerész, aki a karbantartást végrehajtja, és a gépek működéséért felelős
3. raktáros, a raktárban dolgozó személy, aki az ottani alkatrészekért felelős

A bejelentkezés felhasználónévvel és jelszóval történik, ezáltal illetéktelenek nem tudnak hozzáférni az adatokhoz. Lehetőség van új felhasználó létrehozására, vagy már meglévő jelszó megváltoztatására. A felhasználó meg tudja változtatni az IP címet programon belül, ahova csatlakozni szeretne az adatbázishoz. Munkakörnek megfelelően különböző dolgokat tudnak csinálni a felhasználók programon belül.

Bármelyik munkakörben dolgozó személy tud bizonyos adatokat lekérdezni. Tud törzsadatokat, időintervallum, vagy hely alapján is szűrni.

A raktáros tud új alkatrészt felvenni, már meglévőhöz bevételezni, kiírni, selejtezni, módosítani és törölni.

A műszerész tud kiírni, ütemezett karbantartást rögzíteni, gépet felvenni, módosítani, vagy törölni.

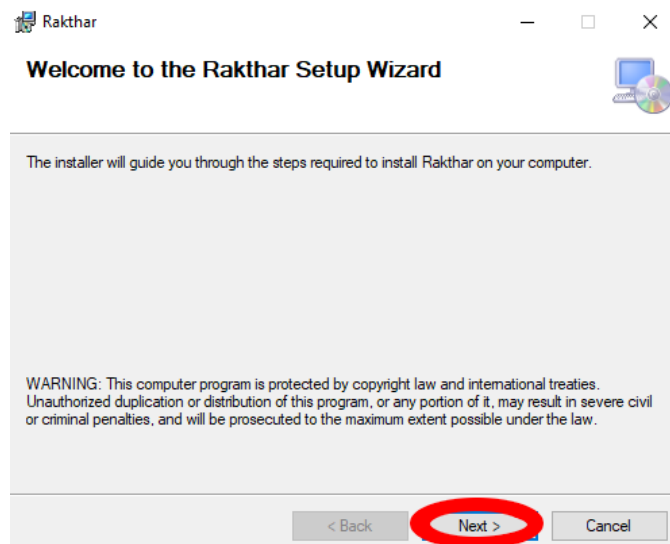
A gazdálkodó tud rendeléseket felvenni, új beszállítót felvenni, módosítani, vagy törölni. A program az alábbi dolgokat naplózza dátummal(nem mi állítjuk be a dátumot, hanem az adott nappal kerül rögzítésre a cselekmény, amikor elvégeztük):

- bejelentkezés , hogy melyik felhasználó, mikor jelentkezett be
- rendelés, hogy milyen cikket rendeltünk, melyik beszállítótól
- kiírás, melyik alkatrész, melyik gépre lett kiírva → karbantartás
- selejtezés, milyen cikk, honnan lett lesejtezve, és milyen indokból

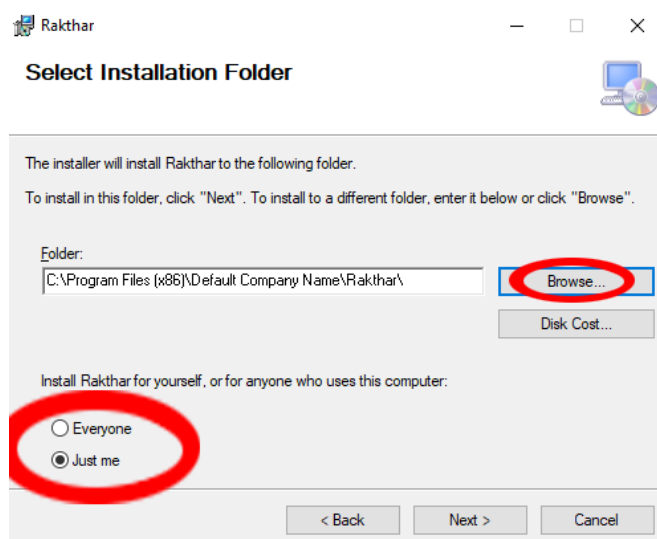
## 2. Telepítés

A Rakthar ikonra dupla bal egérgomb kattintással tudjuk elindítani a telepítést.

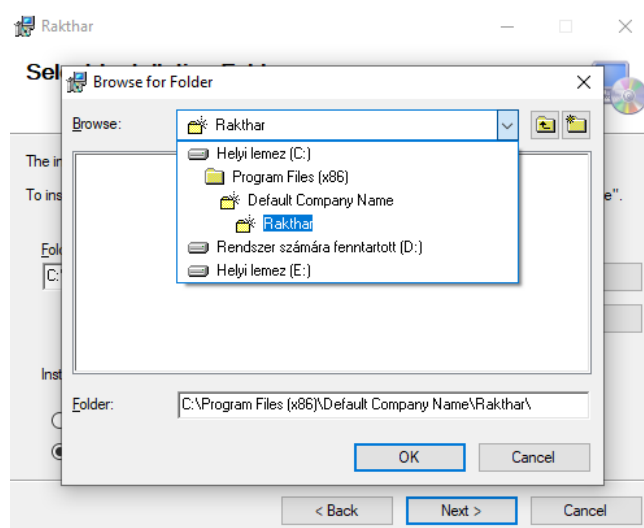
1. A telepítés során megjelenő első oldalon a Next gombra kell kattintanunk, ha szeretnénk a programot feltelepíteni.



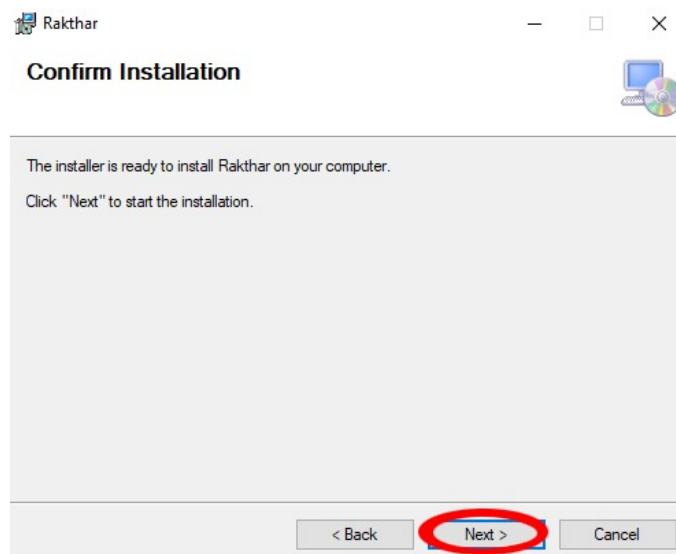
2. A következő oldalon azt kell eldöntenünk, hogy hova szeretnénk feltelepíteni a programot, illetve, hogy mindenki számára elérhető legyen, vagy csak nekünk (Windows felhasználó).



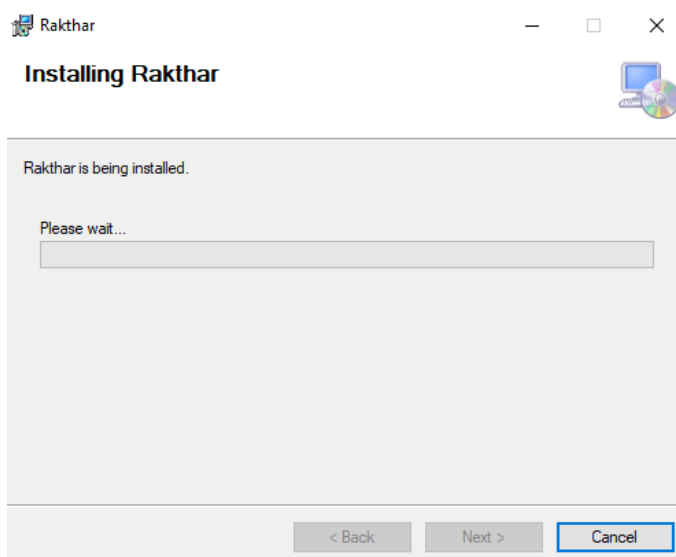
3. A lenyíló mezőben tudjuk megadni, hogy hova, pontosan melyik mappába szeretnénk feltelepíteni.



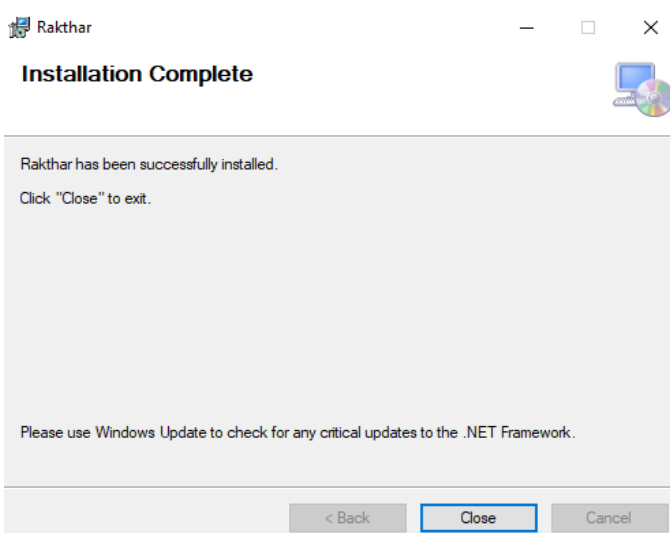
4. A következő oldalon nincs más dolgunk, mint megerősíteni a telepítést a Next gombra rákattintva.



5. Ezt követően már csak meg kell várnunk, hogy a telepítés végigmenjen. Ezt a "Please wait..." szöveg alatti csíkon tudjuk nyomonkövetni.



6. Ha a telepítés befejeződött, az alábbi oldal jelenik meg az "Installation Complete" szöveggel. Itt már csak a Close gombra kell kattintanunk, és végeztünk is a telepítéssel.



### 3. Rendszerkövetelmények

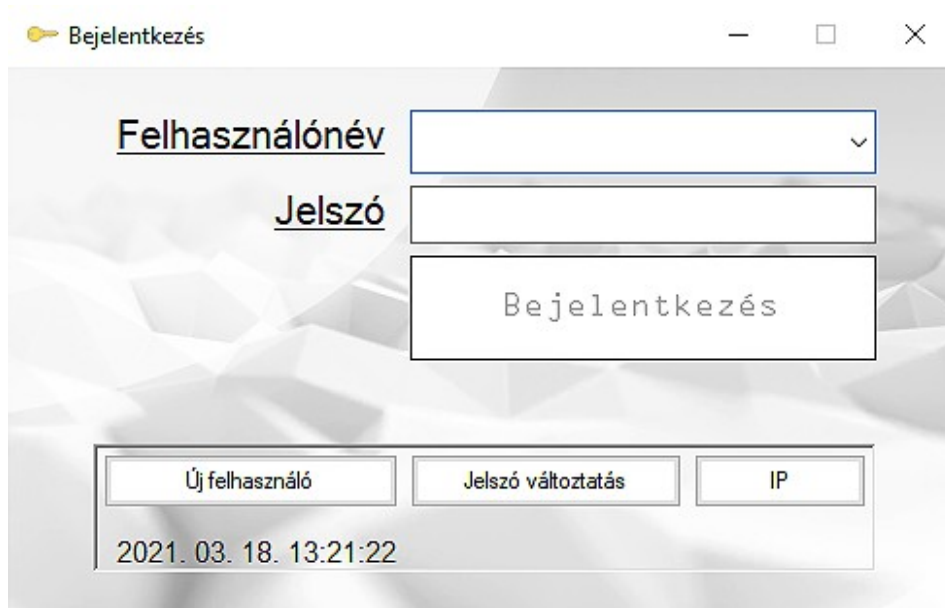
#### **Hardver követelmény:**

- 1 gigahertzes (GHz) vagy gyorsabb 32 bites (x86) vagy 64 bites (x64) processzor\*
- 1 gigabájt (GB) RAM (32 bites rendszerhez) vagy 2 GB RAM (64 bites rendszerhez)
- 16 GB (32 bites rendszerhez) vagy 20 GB (64 bites rendszerhez) szabad lemezterület
- DirectX 9 grafikus eszköz WDDM 1.0 vagy újabb illesztőprogrammal

#### **Szoftver követelmény:**

- Operációs rendszer: minimum Windows 7
- .NET 4.7.2 keretrendszer
- Xampp

## 4. Bejelentkezés



A program elindításakor a Bejelentkezés ablak jelenik meg, ahol a felhasználó az alábbi lehetőségek közül választhat:

1. -megadva a felhasználónevét és jelszavát belépni arra a formra, aminek a használatához jogosultsága van
  2. -új felhasználóként regisztrálni
  3. -megváltoztatni a jelszavát
  4. -IP címet változtatni
- 
1. A felhasználó a megfelelő mezőbe belekattintva(bal egérgombbal), be tudja írni az adatokat, majd a **Bejelentkezés** gombra rákattintva, megkísérelheti a bejelentkezést. Amennyiben jó felhasználónevet és jelszót adott meg, abban az esetben betöltődik számára a munkaköréhez megadott ablak. Ha viszont nem megfelelő valamelyik, akkor a program nem engedi továbblépni, és ezt jelezni is fogja számára, számolva a hibás próbálkozásokat. A harmadik próbálkozás után automatikusan kilép.

2. Az **Új felhasználó** gombra kattintva betöltődik egy másik ablak, ahol meg tudja adni a hozzá szükséges adatokat.

A regisztrációhoz szükséges dolgok:

- felhasználónév

- munkakör, az alábbi három közül lehet választani: 1. gazdálkodó 2. műszerész 3. raktáros

- email cím

- jelszó, minimum 8 karakternek kell lennie, kis és nagybetűt, illetve számot kell tartalmaznia

- a jelszót be kell írni még egyszer ellenőrzés céljából, aminek meg kell egyeznie a felette lévő mezőbe írttal

Ha a felhasználó kitöltötte a mezőket, akkor a Regisztráció gomb megnyomásával létre tudja hozni az új felhasználót. A Reset gomb megnyomásával pedig ki tudja törölni a mezőkbe beírt adatokat.

3. A **Jelszó változtatás** gombra kattintva betöltődik egy másik ablak, amin a felhasználó meg tudja változtatni a jelszavát.

Ehhez szükséges megadni a következőket:

- email cím (a regisztrációkor megadott email cím)

- régi jelszó, meg kell adnunk a korábban használt jelszavunkat, amit meg akarunk változtatni

- új jelszó, 8 karakternél hosszabbnak kell lenni, kis és nagybetűt, illetve számot kell tartalmaznia

- új jelszó még egyszer, meg kell egyeznie az Új jelszó mezőbe írt jelszóval

Ha az összes mezőt kitöltötte megfelelően, akkor a Megváltoztatás gombra kattintva a program végrehajtja azt. Amennyiben valamit rosszul írtunk be, vagy nem szerepel az adatbázisban, a program hibát fog jelezni.



4. Az **IP cím** gombra kattintva betöltődik egy másik ablak, ahol be tudjuk állítani, hogy melyik IP címhez szeretnénk csatlakozni(megadva további paramétereket is), ahol az adatok találhatóak, amivel dolgozni szeretnénk.

Ha nem értünk hozzá, érdemes olyan ember segítségét kérni, aki ért hozzá.

## 5. Raktáros kezelőfelület

A raktáros az alábbi dolgokat tudja tenni programon belül:

1. Keresni
2. Új cikket létrehozni
3. Adott cikkből betárolni
4. Cikket kiírni karbantartáshoz
5. Selejtezni
6. Cikk adatokat módosítani
7. Cikket törölni



## 1. Keresés:

Keresni tud cikkekre, leadott rendelésekre, gépekre, karbantartási adatokra, illetve ütemezett szervizekre. Tudja mettől-meddig időintervallum alapján szűrni az adatokat, és tud hely alapján is keresni.

A raktáros ki tudja jelölni, hogy pontosan mire akar keresni, azáltal, hogy a megfelelő rádiógombokra kattint, majd a Keresés felirat alatt lévő mezőbe be tudja gépelni. Ha bevittük a keresni kívánt adatot, nem szükséges Enter-t nyomnunk, vagy bármilyen más billentyűt, ahogy belekattintottunk a keresőmezőbe, és elkezdtünk gépelni, a keresési folyamat egyből megkezdődik magától, amennyiben ki lett választva két rádiógomb. Egy kereséshez ugyanis két rádiógombnak kell bejelölve lenni. Az első, főrádiógombnak (például Cikk), amely bekapcsolására megjelenik mellette a többi alrádiógomb (például Típus) is. Ezekből szintén választanunk kell egyet.

Ha a Cikk rádiógomb van kiválasztva, lehetősége van a megjelenő Hely gombra kattintva megadni, hogy hol szeretnénk keresni, és ez alapján kapjuk meg a szűrt adatokat. Például, ha szeretné megnézni, hogy raktáron belül egy adott oszlopban milyen termékek vannak tárolva, akkor fentről lefelé haladva ki kell töltenie, hogy hol keres pontosan. Amely mezőkbe bevitt adatokat, az azok melletti négyzetet be kell pipálnia. Ha például oszlopra keresünk, akkor a hiearchiában felette lévő helyeket is meg kell adnunk, vagyis jelen esetben az épületet és raktárt is, és mellettük az üres négyzetet ki kell pipálni. A keresési hely megadásánál nem lehet üres hézagokat hagyni. Tehát úgy például nem tudunk keresni, hogy megadjunk egy épületet, és egy boxot. Ha boxra akarunk keresni, akkor az előbb említett módon ki kell tölteni a felette elhelyezkedő mezőket is.

Ha időintervallum alapján szeretnénk szűrni:

- Karbantartás rádiógombot kiválasztjuk, azon belül a Karbantartva rádiógombot

- Ütemezett rádiógombot kiválasztjuk, azon belül a Dátum rádiógombot

- Berendelve rádiógombot kiválasztjuk, azon belül pedig a Rendelve, vagy a Várható rádiógombot

Ha ezek valamelyikét kiválasztottuk, akkor felugrik egy új ablak, ahol a mettől-meddig dátumot tudjuk megadni. Ha megadtuk a két dátumot, akkor a Szűrés gombra kattintva frissülni fog az adatmegjelenítő mezőnk.

## 2. Új cikk:

A kezelő felületünk jobb oldalán található Új cikk nevű gombra rákattintva megjelenik az ablak.

A mezők kitöltése után a Mentés gombra kattintva tudjuk rögzíteni az újonnan létrehozott cikket, vagy. Fontos, hogy az összes mezőben legyen adat, ha akár egy is üresen lett hagyva, akkor a program jelezni fogja, hogy “Hiányos adatok!”, és nem tárolja le a bevitt adatokat. Amire még oda kell figyelnünk, hogy a minimum-maximumként megadott értékek is stimmeljenek. A minimum érték nem lehet nagyobb a maximumnál, illetve egyiksem lehet nulla. A megadott mennyiség szintűgy nem lehet nulla. A Reset gomb megnyomásával ki tudjuk törölni az összes mezőből a beírt adatokat.

## 3. Betárolás:

A Betárolás gombra kattintva tudunk meglévő cikket bevételezni.

Már meglévő cikkhez tudunk hozzáadni, hogy hova, és mennyit szeretnénk belőle betárolni. Kizárólag az adatbázisban már szereplő cikkel tudjuk ezt végrehajtani, amit a cikknévvel és gyártói számmal tudunk beazonosítani. Ha olyan helyre szeretnénk betárolni, ahol már van ilyen cikk, akkor a Korábbi helyre kell kattintani. Segítségül be fognak tölteni a lenyíló mezőkbe a tárolási helyek, hogy hova lettek letárolva. Ha viszont olyan helyre akarjuk tárolni, ahol még nem volt, akkor az Új helyre kattintsunk.

## 4. Kiírás:

A Kiírás gombra kattintva tudunk cikket kiírni karbantartáshoz.

A mezők kitöltése után tudjuk kiírni az adott cikket, az adott gépre, a Kiírás gombra kattintva. Ha a program nem jelez hibát, akkor az adatbázisunkban a karbantartáshoz lesz mindez rögzítve, az adott dátummal, amikor a kiírás megtörtént. Ha esetlegesen többet szeretnénk kiírni, mint ami ténylegesen megtalálható abból a cikkből a raktárban, akkor a program ezt egy hibaüzenettel fogja számunkra jelezni. A Reset gomb megnyomásával törölni tudjuk a mezőkbe beírtakat.

## 5. **Selejtezés:**

A Selejtezés gombra kattintva tudunk meglévő cikket selejtezni.

A meglévő cikkek közül tudunk selejtezni, ha esetleg az nem megfelel meg a szakmai követelményeknek, hibásan vételeztünk be belőle párat, nincs meg a raktárban valamilyen okból kifolyólag, illetve, ha már nincsen szükség rá, mert újabbat használnak helyette.

## 6. **Módosítás:**

A Módosítás gombra kattintva tudjuk karbantartani a cikk törzsadatait.

Ha esetleg elgéptünk valamit a cikkalap létrehozásakor, vagy időközben különböző változások voltak az adott cikkel kapcsolatban, akkor lehetőségünk van arra is, hogy azt módosítsuk. A cikkhez rögzített összes adat módosítására lehetőségünk van. A Reset gomb megnyomásával ki tudjuk törölni az adatokat a beviteli mezőkből.

## 7. **Törlés:**

A Törlés gombra kattintva tudunk cikket törölni.

Ha egy alkatrészt már nem használnak, mert például egy olyan gépbe való, amit szintén nem használnak már, vagy egy újabb, jobb van helyette, akkor lehetőség van rá, hogy az adott cikket töröljük a rendszerből. Törölni csak akkor lehet, ha már nincs belőle raktáron egysem, ezért a törlést megelőzően előbb le kell selejteznünk a megmaradt darabokat, és ki kell választanunk azt az indokot, hogy "Mást használnak helyette". A lenyíló mezőbe csak azok a cikkek töltődnek be, amiből valahol már nullára fogyott a készlet, de fontos, hogy ez nem azt jelenti, hogy más raktárban sincs belőle egysem.

## 6. Műszerész kezelőfelület

A műszerész az alábbi dolgokat tudja tenni programon belül:

1. Keresni gépekre, cikkekre, leadott rendelésekre, karbantartási adatokra, és előre ütemezett karbantartásokra
2. Kiírni alkatrészt karbantartáshoz
3. Ütemezni karbantartást különböző gépekhez, időponttal
4. Új gépet rögzíteni
5. Gép adatokat módosítani
6. Gépet törölni

Műszerész

Menu Kiírás Ütemezés Új gép Módosítás Törlés MENÜSOR

ADATMEGJELENÍTŐ FELÜLET

Keresés 1.

☐ Cikk  
☐ Karbantartás  
☐ Gépek  
☐ Ütemezett  
☐ Berendelve

KERESŐ FELÜLET

2. Kiírás 3. Ütemezés  
4. Új gép 5. Módosítás  
6. Törlés

ABLAKBETÖLTŐ GOMBOK

Bejelentkezve: b  
2021. 03. 21. 13:05:30

## 1. Keresés:

Keresni tud cikkekre, karbantartási adatokra, gépekre, ütemezett szervizekre, illetve berendelt cikkekre. Tudja mettől-meddig időintervallum alapján szűrni az adatokat, és tud hely alapján is keresni.

A műszerész ki tudja jelölni, hogy pontosan mire akar keresni, azáltal, hogy a megfelelő rádiógombokra kattint, majd a Keresés felirat alatt lévő mezőbe be tudja gépelni. Ha bevittük a keresni kívánt adatot, nem szükséges Enter-t nyomnunk, vagy bármilyen más billentyűt, ahogy belekattintottunk a keresőmezőbe, és elkezdünk gépelni, a keresési folyamat egyből megkezdődik magától, amennyiben ki lett választva két rádiógomb. Egy kereséshez ugyanis két rádiógombnak kell bejelölve lenni. Az első, főrádiógombnak (például Cikk), amely bekapcsolására megjelenik mellette a többi alrádiógomb (például Típus) is. Ezekből szintén választanunk kell egyet.

Ha a Cikk rádiógomb van kiválasztva, lehetősége van a megjelenő Hely gombra kattintva megadni, hogy hol szeretnénk keresni, és ez alapján kapjuk meg a szűrt adatokat. Például, ha szeretné megnézni, hogy raktáron belül egy adott oszlopban milyen termékek vannak tárolva, akkor fentről lefelé haladva ki kell töltenie, hogy hol keres pontosan. Amely mezőkbe bevitt adatokat, az azok melletti négyzetet be kell pipálnia. Ha például oszlopra keresünk, akkor a hierarchiában felette lévő helyeket is meg kell adnunk, vagyis jelen esetben az épületet és raktárt is, és mellettük az üres négyzetet ki kell pipálni. A keresési hely megadásánál nem lehet üres hézagokat hagyni. Tehát úgy például nem tudunk keresni, hogy megadunk egy épületet, és egy boxot. Ha boxra akarunk keresni, akkor az előbb említett módon ki kell tölteni a felette elhelyezkedő mezőket is.

Ha időintervallum alapján szeretnénk szűrni:

- Karbantartás rádiógombot kiválasztjuk, azon belül a Karbantartva rádiógombot

- Ütemezett rádiógombot kiválasztjuk, azon belül a Dátum rádiógombot

- Berendelve rádiógombot kiválasztjuk, azon belül pedig a Rendelve, vagy a Várható rádiógombot

Ha ezek valamelyikét kiválasztottuk, akkor felugrik egy új ablak, ahol a mettől-meddig dátumot tudjuk megadni. Ha megadtuk a két dátumot, akkor a Szűrés gombra kattintva frissülni fog az adatmegjelenítő mezőnk.

## 2. Kiírás:

A Kiírás gombra kattintva betöltődik a hozzátartozó ablak.

A mezők kitöltése után tudjuk kiírni az adott cikket, az adott gépre, a Kiírás gombra kattintva. Ha a program nem jelez hibát, akkor az adatbázisunkban a karbantartáshoz lesz mindez rögzítve, az adott dátummal, amikor a kiírás megtörtént. Ha esetlegesen többet szeretnénk kiírni, mint ami ténylegesen megtalálható abból a cikkből a raktárban, akkor a program ezt egy hibaüzenettel fogja számunkra jelezni. A Reset gomb megnyomásával törölni tudjuk a mezőkbe beírtakat.

## 3. Ütemezés:

Az Ütemezés gombra kattintva betöltődik az ütemező ablak.

A mezők kitöltése után tudunk az adott géphez, az adott dátumra, órára pontosan, időponttal felvenni ütemezett karbantartást, a Mentés gombra rákattintva. Ehhez szükséges megadnunk a gép nevét, azonosítószámát, majd a dátumválasztó panelt lenyitva megadnunk egy dátumot, alatta pedig ki kell választanunk még azt is, hogy hány órára tervezzük.

Ha mindent jól csináltunk, az itt bevitt adatok az ütemezett karbantartásokhoz lesznek rögzítve. A Reset gomb megnyomásával törölni tudjuk a mezőkbe beírtakat.

## 4. Új gép:

Az Új gép gombra kattintva tudjuk behozni az ablakot.

Új gépet tudunk felvenni, megadva annak nevét, azonosítószámát, majd a Mentés gomb megnyomásával tudjuk rögzíteni az adatbázisba. A Reset gomb megnyomásával pedig törölni tudjuk a beírt adatokat a mezőkből.

## 5. Módosítás:

A Módosítás gombra kattintva tudjuk behozni ezt az ablakot.

Lehetőség van a gépek adatainak a módosítására, amennyiben elgéveltük, vagy valamiféle változás állt be. Tudjuk a gépnek a nevét és/vagy azonosítószámát is módosítani.



## 6. Törlés:

A Törlés gombra kattintva lehetőségünk van gépet törölni.

Lehetőség van gépek törlésére a rendszerből, ha már az adott gépet nem használják különböző okokból kifolyólag, például újabb, jobb van helyette, vagy leselejteztek. Ehhez meg kell adnunk a gép nevét és azonosítóját.

## 7. Gazdálkodó kezelőfelület

A gazdálkodó az alábbi dolgokat tudja végrehajtani a programon belül:

1. Keresni tud cikkekre, leadott rendelésre, beszállítóra, selejtezésre, karbantartásra, illetve arra, hogy a raktáron belül melyek azok a cikkek, amelyek a minimum értékkel egyenlőek, vagy kevesebbek
2. Új beszállítót rögzíteni
3. Leadott rendeléseket rögzíteni
4. Beszállító törzsadatokat módosítani
5. Beszállítót törölni



## 1. Keresés:

Keresni tud cikkekre, leadott rendelésekre, beszállítókra, selejtezett cikkekre, karbantartásokra, illetve minimum értéken lévő cikkekre. Tudja mettől-meddig időintervallum alapján szűrni az adatokat, és tud hely alapján is keresni.

A gazdálkodó ki tudja jelölni, hogy pontosan mire akar keresni, azáltal, hogy a megfelelő rádiógombokra kattint, majd a Keresés felirat alatt lévő mezőbe be tudja gépelni. Ha bevittük a keresni kívánt adatot, nem szükséges Enter-t nyomnunk, vagy bármilyen más billentyűt, ahogy belekattintottunk a keresőmezőbe, és elkezdtünk gépelni, a keresési folyamat egyből megkezdődik magától, amennyiben ki lett választva két rádiógomb. Egy kereséshez ugyanis két rádiógombnak kell bejelölve lenni. Az első, főrádiógombnak (például Cikk), amely bekapcsolására megjelenik mellette a többi alrádiógomb (például Típus) is. Ezekből szintén választanunk kell egyet.

Ha a Cikk rádiógomb van kiválasztva, lehetősége van a megjelenő Hely gombra kattintva megadni, hogy hol szeretnénk keresni, és ez alapján kapjuk meg a szűrt adatokat. Például, ha szeretné megnézni, hogy raktáron belül egy adott oszlopban milyen termékek vannak tárolva, akkor fentről lefelé haladva ki kell töltenie, hogy hol keres pontosan. Amely mezőkbe bevitt adatokat, az azok melletti négyzetet be kell pipálnia. Ha például oszlopra keresünk, akkor a hierarchiában felette lévő helyeket is meg kell adnunk, vagyis jelen esetben az épületet és raktárt is, és mellettük az üres négyzetet ki kell pipálni. A keresési hely megadásánál nem lehet üres hézagokat hagyni. Tehát úgy például nem tudunk keresni, hogy megadunk egy épületet, és egy boxot. Ha boxra akarunk keresni, akkor az előbb említett módon ki kell tölteni a felette elhelyezkedő mezőket is.

Ha idő intervallum alapján szeretnénk szűrni:

-Karbantartás rádiógombot kiválasztjuk, azon belül a Karbantartva rádiógombot

-Selejtezett rádiógombot kiválasztjuk, azon belül a Dátum rádiógombot

-Berendelve rádiógombot kiválasztjuk, azon belül pedig a Rendelve, vagy a Várható rádiógombot

Ha ezek valamelyikét kiválasztottuk, akkor felugrik egy új ablak, ahol a mettől-meddig dátumot tudjuk megadni. Ha megadtuk a két dátumot, akkor a Szűrés gombra kattintva frissülni fog az adatmegjelenítő mezőnk.

## 2. Új beszállító:

Az Új beszállító gombra kattintva tudunk új beszállítót rögzíteni az adatbázisba.

Meg kell adni a beszállító nevét, mennyi idő szállítja be a tőlük rendelt alkatrészeket, ez rögzítve van a szerződésben, hogy hány napon belül kell teljesítenie a szállítást, városát, címét, telefonszámát, email

címét. Ha megfelelően ki van töltve az összes mező, akkor a Mentés gomb megnyomásával tudjuk menteni az adatokat. A Reset gomb megnyomásával törölni tudjuk a mezőkbe beírtakat.

## 3. Rendelés:

A Rendelés gombra kattintva tudunk rendelést rögzíteni.

Meg kell adnunk a kitöltendő mezőkbe, hogy melyik cikkből akarunk rendelni, mennyit, melyik beszállítótól, majd végül a dátumválasztó részt lenyitva azt is megadhatjuk, hogy mikorra várható a beérkezése. Ha jól lettek kitöltve a mezők, akkor a Mentés gombra kattintva rögzítjük mindezt az adatbázisba.

## 4. Módosítás:

A Módosítás gombra kattintva tudjuk módosítani a beszállítói törzsadatokat.

Ebben az ablakban meg kell adni, hogy melyik beszállítót akarjuk módosítani, és mire. Ehhez először is ki kell választanunk legfelül, a lenyíló mezőben egy beszállítót, majd az Adatok betöltése gombra kattintva be tudjuk tölteni az adott beszállítóhoz tartozó adatokat. Lehetőségünk van a módosítani: a nevét, szállítási idejét, városát, címét, telefonszámát és email címét is. Majd, ha megfelelően kitöltöttük a módosítani kívánt mezőket, akkor a Módosítás gomb megnyomásával tudjuk menteni azt.

## 5. Törlés:

A Törlés gombra kattintva betöltődik az adott ablak.

Amennyiben megszűnt a szerződés egy adott beszállítóval bizonyos okokból kifolyólag, és a cég már nem rendel tőle semmit, akkor lehetőségünk van törölni azt a rendszerből.

Itt a lenyíló mezőből kell kiválasztanunk, hogy melyik beszállítót szeretnénk törölni a rendszerből, majd az ebben az ablakunkban megtalálható Törlés gombra kattintva törölhetjük azt.

## B. Fejlesztői dokumentáció

### 1. Bevezetés

A záródolgozatom célja egy gyógyszergyártó cég műszaki raktárában található eszközök, alkatrészek számontartására, illetve a hozzájuk köthető egyéb tevékenységek lebonyolítására szolgáló (elsősorban) raktárkezelő szoftver készítése, amely ezenfelül egyfajta belső összekötő kapocsként is funkcionál a gyár területén belül dolgozók számára, összehangolva az ottani munkájukat, hogy minél jobban leegyszerűsítse számukra a különböző munkafolyamatokat. Mint elég sok mindent, ezt is jelentős mértékben meg tudja könnyíteni egy megfelelően elkészített szoftver, amely ezenfelül az adott raktár további igényeinek is megfelel, amelyre később visszatérek a specifikációnál.

A gyógyszergyártásban elengedhetetlen a gyógyszereket gyártó gépek folyamatos megfelelő működése, ezért egy esetlegesen felmerülő, nem várt hiba esetén, lehetőleg minél gyorsabban meg kell javítani, amely szükségessé teszi egy jól összehangolt műszerész brigád 3 műszakban történő váltott, folyamatos készenléti állapotát. Továbbá, hogy a gépek szervizeléséhez helyben megtalálhatók legyenek a különböző alkatrészek, amelyek meghibásodhatnak az adott gépben, és cserélni kell. Maga a gyártási folyamat több részből tevődik össze, ebből adódóan különböző gépekre van szükség. Minden egyes gépben megannyi alkatrész található, amelyek összeadva körülbelül 2000 különböző cikknek felelnek meg, beleértve az elektronikus, illetve mechanikus alkatrészeket is. Hogy a műszerészek minél gyorsabban tudják végezni a munkájukat, arra van szükségük, hogy egy jól átlátható, könnyen kezelhető rendszer segítségével tudják minden egyes alkatrésztől, beleértve ebbe akár egy kisebb méretű csavart is, hogy pontosan hol található a raktáron belül, hogy melyik oszlop, melyik polcán van az. Az anyaggazdálkodónak tudnia kell, hogy mely alkatrészek vannak a minimum szinten, vagy alatta, az melyik beszállítótól van, mennyi a várható szállítási idő, stb.

Az általam elkészített raktárkezelő program ebben nyújt számukra segítséget.

## 2. Specifikáció

Ez a program egy grafikus kezelő felülettel ellátott raktárkezelő program. Három, jelszóval védett, illetve munkaköri jogosultsághoz kötött form van: Gazdálkodó, Műszerész, illetve Raktáros. Hogy melyik jogosultsághoz kötött form töltődik be a felhasználónak bejelentkezéskor, az a regisztrációkor megadott munkakörtől függ. Bejelentkezéskor a program lekéri az adatbázisban tárolt felhasználónévhez tartozó munkakört, és az alapján tölti be a megfelelő formot a felhasználónak, amennyiben a felhasználóneve és jelszava megfelelő.

Háromféle munkakör van: műszerész, raktáros, gazdálkodó. Regisztrációkor az email cím, és felhasználónév csak egyszer szerepelhet az adatbázisban, ezért a program nem engedi, ha már szereplő email címmel, vagy felhasználónévvel szeretnénk regisztrálni.

Bejelentkezésnél, három hibás próbálkozás után a program kilép. Lehetőség van a jelszó megváltoztatására, amihez nem kell külön webes megoldás, elég, ha a felhasználó megadja az email címét, és az email címhez tartozó jelszót meg lehet változtatni. A jelszó titkosítva kerül fel az adatbázisra, ha valaki hozzá is fér az adatbázishoz, akkor se tudja a felhasználó tényleges jelszavát megszerezni. Lehetőség van új felhasználó létrehozására, ami szükséges hozzá: felhasználónév, jelszó, email cím és munkakör. Egy formból egyszerre csak egy töltődhet be, nem lehet ugyanabból többször megnyitni. Ha kilépünk valamelyik jogosultsághoz kötött, jelszóval védett formból, akkor azokat a formokat is bezárja egyidejűleg, amelyeket onnan nyitottunk meg.

A raktáros tud betárolni már meglévő cikket, új cikket létrehozni, kiírni, selejtezni, meglévő cikket módosítani, illetve cikket törölni. Ezenfelül le tud kérdezni bizonyos adatokat: cikkekről, rendelésről, gépekről, karbantartásról, illetve ütemezett szervízeiről, és szűkíteni tudja a keresést az adott táblán belül.

A műszerész is tud kiírni cikket, mint a raktáros, ütemezni tudja egy adott gépre a karbantartást, új gépeket tud regisztrálni, módosítani, illetve tud régi, már selejt, vagy nem használt gépet törölni. Ő is tud lekérni adatokat cikkekről, karbantartásokról, gépekről, ütemezett szervízeiről, és berendelt cikkekről.

Az anyaggazdálkodó fel tud venni új beszállítót, illetve tudja azt módosítani, vagy törölni, és rögzíteni tudja az adatbázisba a különböző leadott rendeléseket. Emellett, mivel ő felel a kiadásokért, illetve rendelésekért, le tudja kérdezni a cikkeket, beszállítókat, a selejtezett cikkeket, a karbantartásokat, illetve azokat a cikkeket, amelyek az adott cikk minimum értékét elérték, vagy annál kevesebb van már belőlük, illetve le tudja kérdezni a leadott rendeléseket.

Aznapi dátummal naplózza a bejelentkezéseket, a cikk kiírásokat(karbantartás), rendeléseket, illetve a selejtezéseket, mindezt rögzítve az adatbázisba, a megfelelő adattáblába.

A különböző raktáron tárolt cikkekről információk tárolása. Milyen típusú az adott cikk, elektromos, vagy mechanikus. Mennyi a minimum-maximum érték, aminek lennie kell raktáron, gyártói szám tárolása. A gyár területén belül melyik épületben van, azon belül melyik raktár helyiségben, ott melyik oszlopban, melyik polcon, illetve, ha van, akkor melyik boxban található. Az adott cikk melyik beszállítótól van, milyen árban rendelték.

A karbantartás nyomonkövetése, mikor, melyik gép lett szervízelve, illetve ahhoz milyen eszközök lettek a raktárból felhasználva(kiírás). Ütemezett szervizelések rögzítése.

Milyen nevű gépek találhatók a gyáron belül, névvel és azonosítószámmal megkülönböztetve. Beszállítók tárolása, szállítási idővel, címmel, telefonszámmal, email címmel.

### 3. Használt szoftverek

A programot C# nyelven írtam, Visual Studio 2019 fejlesztőkörnyezetben, a .NET keretrendszer 4.7.2 számú verziójának segítségével készítettem el. Adatbázis kezeléshez XAMPP-ot használtam MySQL-el. A .NET framework, ami a korábbi COM platformot váltotta fel, számos előnyt nyújt, mint például a platformfüggetlenség, gyors alkalmazásfejlesztés, amit angolul csak RAD-nak rövidítenek (Rapid Application Development). A képek szerkesztését Adobe Photoshop CS6-ban végeztem.

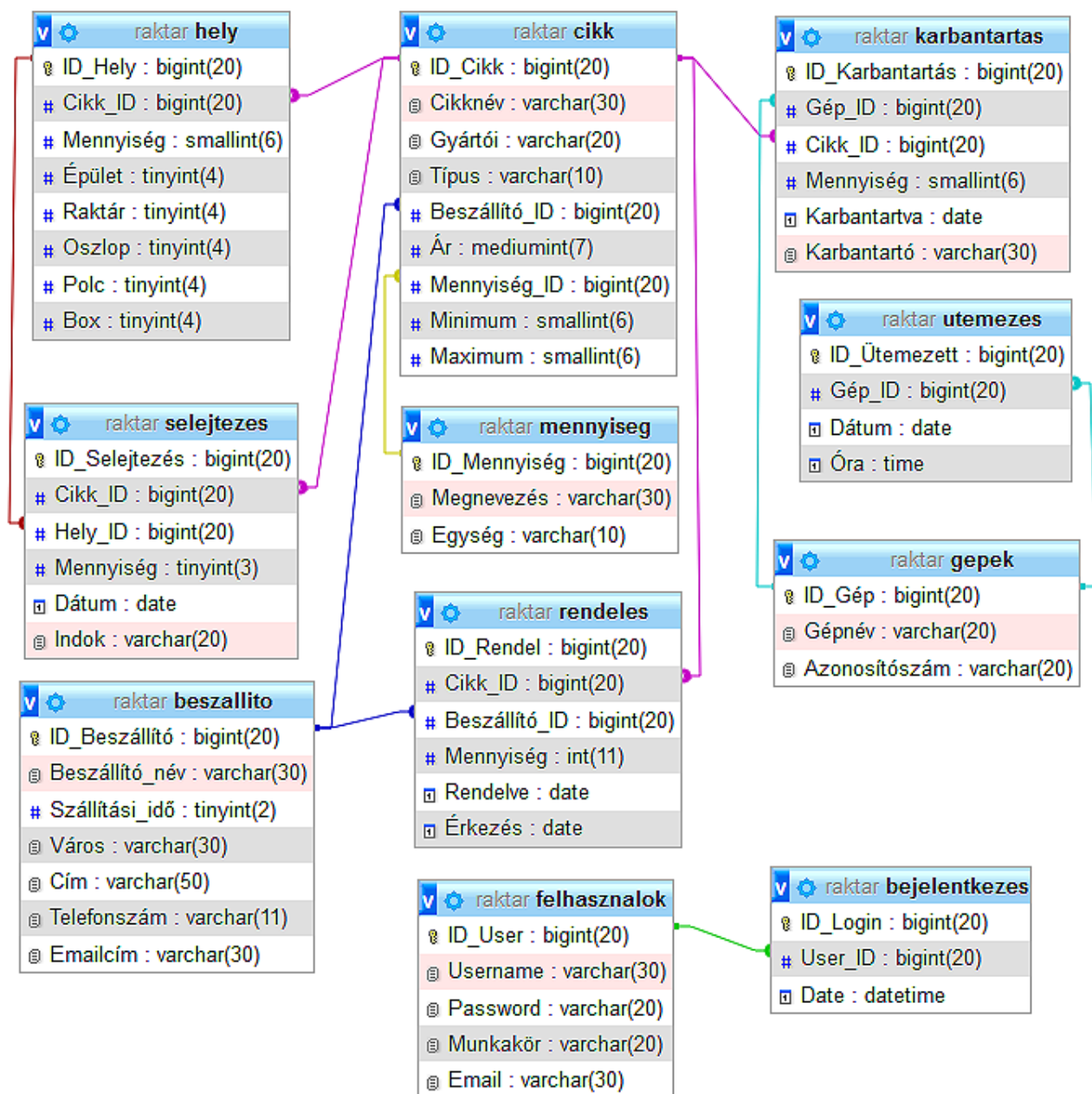
Magáról a keretrendszer megalkotásáról, és történetéről röviden annyit, hogy a kilencvenes évek közepén az új Java platform megalkotásával, az elég nagy népszerűsége miatt, amely más cégeket háttérbe szorított, többek közt a Microsoftot is. Ez egy könnyen tanulható, platformfüggetlen, nagy és könnyen kezelhető osztálykönyvtárral rendelkező környezet volt, ami a többi cégnek kihívást jelentett, hogy ők is felzárkózzanak, ezért fejlesztésekbe kezdtek.

A kilencvenes évek végén Bill Gates felügyelete alatt kezdődtek meg a munkálatok, majd 2001-ben a C# szabványosítását elfogadta az ECMA.

A .NET Framework 1.0-át pedig 2002. január. 5.-én mutatták be, a népszerűségét rá 3 évre a 3.0 megjelenésével szerezte meg, 2005.-ben, és a fejlesztése folyamatosan tart napjainkban is. Széles elterjedését elősegítette, hogy különböző platformokra lehet vele fejleszteni, asztali, webes, mobil alkalmazásokat. Az alapja a CLI, azaz a Common Language Infrastructure, ami olyan szabályok összessége, ami egy nyelvfüggetlen fejlesztői környezet ír le, amelynek az implementációja .NET-ben a CLR, azaz a Common Language Runtime.



## 4. MySql adatbázis



Adatbázis neve: **raktar**

## 11 adattábla:

**1. bejelentkezés** -- rögzíti a bejelentkezéseket felhasználó ID-val és időponttal

- ID\_Login – bigint(20) - Elsődleges kulcs
- User\_ID – bigint(20) - Külső kulcs → felhasználok(ID\_User)
- Date – datetime

### Idegen kulcs megszorítások:

UserID\_reláció = User\_ID → felhasználok(ID\_User)

ON DELETE = CASCADE

ON UPDATE = CASCADE

**2. beszallito** -- a beszállítókról különböző adatok

- ID\_Beszállító – bigint(20) - Elsődleges kulcs
- Beszállító\_név – varchar(30)
- Szállítási\_idő – tinyint(2)
- Város – varchar(30)
- Cím – varchar(50)
- Telefonszám – varchar(11)
- Emailcím – varchar(30)

**3. cikk** -- a cikkekről különböző adatok

- ID\_Cikk – bigint(20) - Elsődleges kulcs
- Cikknév – varchar(30)
- Gyártói – varchar(20)
- Típus – varchar(10)
- Beszállító\_ID – bigint(20) - Külső kulcs → beszallito
- Ár – mediumint(7)
- Mennyiség\_ID – bigint(20) - Külső kulcs → mennyiseg
- Minimum – smallint(6)
- Maximum – smallint(6)

**Idegen kulcs megszorítások:**

cikk\_beszallito = Beszállító\_ID → beszallito(ID\_Beszállító)

ON DELETE = CASCADE

ON UPDATE = CASCADE

cikk\_mennyiseg = Mennyiség\_ID → mennyiseg(ID\_Mennyiség)

ON DELETE = CASCADE

ON UPDATE = CASCADE

**4. felhasznalok** -- felhasználónév + jelszóval történő bejelentkezéshez felhasználói adatok

-ID\_User – bigint(20) - Elsődleges kulcs

-Username – varchar(30)

-Password – varchar(20)

-Munkakör – varchar(20)

-Email – varchar(30)

**5. gepek** -- a gépekről különböző adatok

-ID\_Gép – bigint(20) -Elsődleges kulcs

-Gépnév – varchar(20)

-Azonosítószám – varchar(20)

**6. hely** -- adott cikk hol található, és mennyi van belőle

-ID\_Hely – bigint(20) - Elsődleges kulcs

-Cikk\_ID – bigint(20) - Külső kulcs → cikk

-Mennyiség – smallint(6)

-Épület – tinyint(4)

-Raktár – tinyint(4)

-Oszlop – tinyint(4)

-Polc – tinyint(4)

-Box – tinyint(4)

**Idegen kulcs megszorítások:**

hely\_cikk = Cikk\_ID → cikk(ID\_Cikk)

ON DELETE = CASCADE

ON UPDATE = CASCADE

**7. karbantartas** -- gépek karbantartása, milyen és mennyi cikket használtak fel hozzá, és mikor

-ID\_Karbantartás – bigint(20) - Elsődleges kulcs

-Gép\_ID – bigint(20) - Külső kulcs → gepek

-Cikk\_ID – bigint(20) - Külső kulcs → cikk

-Mennyiség – smallint(6)

-Karbantartva – date

-Karbantartó – varchar(30)

**Idegen kulcs megszorítások:**

karb\_cikk\_relacio = Cikk\_ID → cikk(ID\_Cikk)

ON DELETE = CASCADE

ON UPDATE = CASCADE

karb\_gep\_relacio = Gép\_ID → gepek(ID\_Gép)

ON DELETE = CASCADE

ON UPDATE = CASCADE

**8. mennyiseg** -- használatban lévő mennyiségi egységek, és rövidítésük

-ID\_Mennyiség – bigint(20) - Elsődleges kulcs

-Megnevezés – varchar(30)

-Egység – varchar(10)

**9. rendeles** -- adott cikkekre leadott rendelések

-ID\_Rendel – bigint(20) - Elsődleges kulcs  
-Cikk\_ID – bigint(20) - Külső kulcs → cikk  
-Beszállító\_ID – bigint(20) - Külső kulcs → beszállito  
-Mennyiség – int(11)  
-Rendelve – date  
-Érkezés – date

**Idegen kulcs megszorítások:**

rendeles\_beszallito = Beszállító\_ID → beszállito(ID\_Beszállító)

ON DELETE = CASCADE

ON UPDATE = CASCADE

rendeles\_cikk = Cikk\_ID → cikk(ID\_Cikk)

ON DELETE = CASCADE

ON UPDATE = CASCADE

**10. selejtezes** -- milyen cikk, és miért lett leselejtezve, dátummal naplózva

-ID\_Selejtezés – bigint(20) - Elsődleges kulcs  
-Cikk\_ID – bigint(20) - Külső kulcs → cikk  
-Hely\_ID – bigint(20) - Külső kulcs → hely  
-Mennyiség – tinyint(3)  
-Dátum – date  
-Indok – varchar(20)

**Idegen kulcs megszorítások:**

selejt\_cikk = Cikk\_ID → cikk(ID\_Cikk)

ON DELETE = CASCADE

ON UPDATE = CASCADE

selejt\_hely = Hely\_ID → hely(ID\_Hely)

ON DELETE = CASCADE

ON UPDATE = CASCADE

**11. utemezes** -- előre betervezett gép átnézés(szervízelés), dátummal, órára pontosan rögzítve

-ID\_Ütemezett – bigint(20) - Elsődleges kulcs

-Gép\_ID – bigint(20) - Külső kulcs → gepek

-Dátum – date

-Óra – varchar(20)

**Idegen kulcs megszorítások:**

utem\_gep\_relacio = Gép\_ID → gepek(ID\_Gép)

ON DELETE = CASCADE

ON UPDATE = CASCADE

## 5. A program felépítése

### 5.1 Közös osztályok

A programban jópár form található, így hogy logikusan felépített és könnyebben áttekinthető legyen, illetve az objektum orientált szemléletnek is megfeleljen, (ami egy C#-al fejlesztett szoftvernél elvárható, és kihasználható szempont) több közös osztály van. Ezekben a közös osztályokban olyan változók, és metódusok találhatók, amelyek több helyen felhasználhatók. Ezek a kódsorok egy **Közös\_osztályok** néven lévő mappában találhatók, amelyet, ha használni akarunk, be kell emelni a névtérbe:

```
using Raktár.Közös_osztályok;
```

Ha meg van hívva, akkor elérhetővé válnak a benne tárolt különböző osztályok. Azért statikus osztályok, mert nincs külön szükségünk rá, hogy példányosítva legyen az adott osztály minden formában, mivel csak egy-egy feladat elvégzésére hívjuk be az adott metódusát.

Hogy használni tudjuk az ezen osztályokban használt adatbáziskezelő metódusokat, és benne példányosított osztályokat, az alábbi névteret kell beemelnünk:

```
using MySql.Data.MySqlClient;  
using System.Data;
```

### Adatbazis.cs:

Az első és egyben talán legfontosabb osztályunk **Adatbazis.cs** néven található meg. Ebben vannak a különböző MySQL lekérdező stringek, és ezeket fogjuk meghívni szituációtól függően a különböző formokon belül. Példa:

```
public static string cikk_query =
    "SELECT cikk.Cikknev,cikk.Gyártói,cikk.Típus,cikk.Ár AS
      Ár_Forint" +
    ",hely.Mennyiség,menyiseg.Egység,beszallito.Beszállító_név AS
      Beszállító" +
    ",cikk.Minimum,cikk.Maximum," +
    "hely.Épület,hely.Raktár,hely.Oszlop,hely.Polc,hely.Box " +
    "FROM cikk " +
    "JOIN menyiseg ON cikk.Mennyiség_ID=menyiseg.ID_Mennyiség " +
    "JOIN beszallito ON cikk.Beszállító_ID=beszallito.ID_Beszállító"
    +
    "JOIN hely ON cikk.ID_Cikk=hely.Cikk_ID ";
```

Fontos megjegyezni, hogy az itt tárolt lekérdezésekben, egyben sincs felhasználó által bevitt adat, ezek csak összekapcsolt táblás általános lekérdezések. Ezeket fogja a felhasználó kiegészíteni azokkal az adatokkal, amire pontosan keresni szeretne. A felhasználó által beírt adatokat majd paraméteresen fogjuk átadni, hogy biztonságosabb legyen a lekérdezés, és ne lehessen különböző MySQL injenction-öket bejuttatni a felhasználói felületről. Egy példa a paraméteres érték átadásra:

```
CMD_SELECT_cikk.Parameters.Add("@cikknev", MySqlDbType.String).Value =
cikknev_txtbox.Text;
```

A felhasználó beír egy cikknevet, az adott Textbox értékét pedig átadjuk paraméterként egy MySqlCommand-ba, megadva a lekérdezésben megadott paraméter nevét, és a típust, hogy hogyan kezelje azt az adatbázis. Itt fontos figyelembe venni azt is, hogy milyen típusként lett rögzítve az adatbázisban az adott oszlop. Eltérő típus esetén hibára fut, ezért erre is oda kell figyelni.



## Adatkezeles.cs:

Következő ilyen közös osztályunk az **Adatkezeles.cs** néven tárolt osztály. Ebben található két olyan metódus, amelyekre a paraméteres lekérdezéseknél szükségünk van. Az egyik string paramétert vár, a másik int-et. Ez valószínű megoldható lenne egy metódussal is, de mivel egy kisebb metódusról van szó, amely csak egy paramétert kezel, és annak megfelelően lekérdez, ezért ez a program gyorsaságát tekintve észrevehetően a felhasználó számára.

```
public static void datagrid_STRINGparam(string query, DataGridView dgv, string
searchvalue)
{
    var result = new DataTable();
    string sql = query;

    using (var con = new MySqlConnection(Ipcim.constr))
    using (var cmd = new MySqlCommand(sql, con))
    using (var sda = new MySqlDataAdapter(cmd))
    {
        cmd.Parameters.Add("@search", MySqlDbType.String).Value = searchvalue;
        sda.Fill(result);
    }
    dgv.DataSource = result;
}
```

A felhasználó az általa használt MainForm-on beír a keresőmezőbe egy értéket, és a radio button-ök segítségével kiválasztott táblán rákeres az adott értékre. Ezt a beírt értéket veszi át paraméterként a fentebb lévő metódus. Tehát radio button-től függ, hogy mely lekérdező stringet hívjuk meg a közös Adatbazis.cs osztályunkból. Majd ehhez a stringhez fűzzük hozzá a pontosított, paraméterrel elátott stringünket, kiegészítve azt a pontos kereséshez. A metódus vár egy string query paramétert, amely az egyik általános string az Adatbazis.cs osztályból, vár egy DataGridView dgv-t, ahova betölti a lekérdezést, és várja a string searchvalue-t, ahova a felhasználó által beírt adatot kapja.

```
if (cikk_RB.Checked)
{
    if (cikknev_cRB.Checked)
    {
        Adatkezeles.datagrid_STRINGparam(Adatbazis.cikk_query + "WHERE Cikknev =
@search      ", this.dataGridView1, valueToFind); //" + valueToFind + ""
    }
}
```

## ComboBoxKezelo.cs:

Következő ilyen közös osztályunk a **ComboBoxKezelo.cs**. Mivel több olyan form van, amin combobox található, és ugyanazokat a műveleteket kell rajtuk elvégezni, ezért érdemes ezeket is külön közös osztályban tárolni. Hogy használni tudjuk, a névtérbe be kell emelnünk a System.Windows.Forms-ot.

Első és talán legfontosabb metódusa a **public static void ComboBoxFeltoltes(Form f)**, ami egy publikus, statikus, void metódus, ami paraméterként egy Formot vár, ahol név alapján feltölti a comboboxokat az adatbázisból. Végigmegy a paraméterként kapott form összes control komponensén, és név alapján beazonosítva a comboboxot, feltölti azt a megfelelő értékekkel, amit vagy adatbázisból kap, vagy konkrét értéket megadva kódból.

```
func = (controls) => {};
```

Ez a kódsor annyit tesz, hogy létrehoz egy anonimusz, tehát név nélküli funkciót, és formailag előrehozza a kapott paramétert, majd azt követi a funkció "teste". Ezt lambda kifejezésnek hívják. Ennek a név nélküli funkciónak fogjuk átadni paraméterként, a formunk control komponenseit, az alábbi módon.

```
func(f.Controls);
```

A funkción belül található kódsorból egy részlet, nem végigmenve az összes ágon:

```
foreach (Control control in controls){
    if (control is ComboBox)
    {
        if (control.Name == "cikknév_combo")
        {
            using (MySqlConnection mysqlcon = new MySqlConnection(Ipcim.constr))
            {
                mysqlcon.Open();
                MySqlCommand cikk_comm = new MySqlCommand("Select Cikknév From cikk", mysqlcon);
                MySqlDataReader cikk_reader = cikk_comm.ExecuteReader();
                while (cikknév_reader.Read())
                {
                    (control as ComboBox).Items.Add(cikk_reader.GetString("Cikknév"));
                }
                cikk_reader.Close();
                mysqlcon.Close();
            }
        }
    }
}
```

Egy foreach-el végigmegy az összes control komponensen, leellenőrzi, hogy ComboBox, majd név alapján kiválasztva feltölti az adatbázisban megtalálható összes cikknévvel a comboboxot.

Következő említésre méltó metódusa a:

**public static void ComboBoxSpecialisFeltoltes(ComboBox epulet, ComboBox raktar, ComboBox oszlop, ComboBox polc, ComboBox box, ComboBox cikk, ComboBox gyartoi)**

Amely azért felelős, hogy a helyet fentről-lefelé szűrjük, és csak az annak megfelelő értékeket töltsük a hely hierarchiában lentebb lévő comboboxokba. Tehát cikknév, vagy gyártói alapján meghatározzuk, hogy az a cikk hol található meg a gyár területén. A comboboxokban kiválasztott értékekkel pedig fokozatosan szűkítjük a keresési találatot. Ha egy cikk például megtalálható az egyes és kettes épületben is, mi ezt kiválaszthatjuk, hogy hol szeretnénk tovább szűrni. Ha a kettes épületet választjuk, akkor a raktár, oszlop, polc, box comboboxba már csak azok az értékek fognak betöltődni, amelyek a kettes épülethez köthetők. És ezen a módon szűkíthetjük tovább a szűrést, ha lentebb haladunk a comboboxoknál.

Paraméterként bekéri az adott form épület,raktár,oszlop,polc,box combobox-ának a text mezőjét, továbbá a beírt cikknevet és gyártóit.

Ebben az osztályban találhatók továbbá azok a metódusok, amelyek megoldják azt, hogy egy bevitt adatból megjelenítsük a hozzátartozó másik adatot textboxban, megkönnyítve így a felhasználó dolgát, mivel nem kell külön begépelnie. Ilyen metódus pl. Cikknev\_GyartoiSzambol, vagy Azonosito\_Gepnevbol, és a többi.

### **FormEllenorzo.cs:**

Itt olyan metódusok találhatók, amik a form megnyitását, bezárását, az azon található text mezők kitörlését teszi lehetővé. Emellett található még egy GroupBox kezelő, ami csak az éppen használatban lévő radio button-öket tartalmazó groupboxokat jeleníti meg, illetve egy bevétel ellenőrző metódus is, ami leellenőrzi, hogy ki van-e töltve az összes bevételi mező.

Itt jópár metódus használja az Application.OpenForms property-t, ami lényegében az applikációban éppen megnyitott formok gyűjteménye.

### Metódusok(nem az összes):

**public static void form\_ellenorzes**(string formnev)

Paraméterként egy stringet vár, végigmegy a megnyitott formokon, és név alapján beazonosítva azokat(amit paraméterként kap), megakadályozza, hogy ugyanabból a formból egyszerre több legyen megnyitva.

**public static void vissza\_Loginhoz**<T>(T f) where T : Form

Ha bezárjuk a main formunkat, akkor az összes megnyitott alformot is bezárja, és visszatér a bejelentkezéshez. Ez egy általános típusú paramétert váró metódus, ahol nincs pontosan megadva a kapott paraméter típusa. Jelen esetben a Form összes gyermek elemét elfogadja.

**public static bool Bevitel\_ellenorzo**(Form f)

Paraméterként egy Form-ot vár, és egy számláló segítségével összeszámolja az összes üres beviteli mezőt a kapott formon, ha nem nulla a számláló értéke, akkor false-al tér vissza a metódus, tehát vannak még kitöltetlen egységek.

### **Reset.cs:**

A Reset osztály **public static void ClearAll**(Form f) metódusát meghívhatjuk bárhol, ahol a beviteli mezőket ki akarjuk törölni. Paraméterként egy Form-ot vár, amelynek az összes ComboBox, TextBox, NumericUpDown, RadioButton, PictureBox elemén végigmegy, és alaphelyzetbe állítja azokat.

---

Ezeket a közös osztályokat, és azok metódusait alkalmazva a megfelelő helyen, lehetővé teszi, hogy a kód tisztább maradjon, és ne kelljen felesleges kód ismétlésbe bocsátkoznunk.

## **5.2 Belépés:**

Belépési engedélyhez kötött formok:

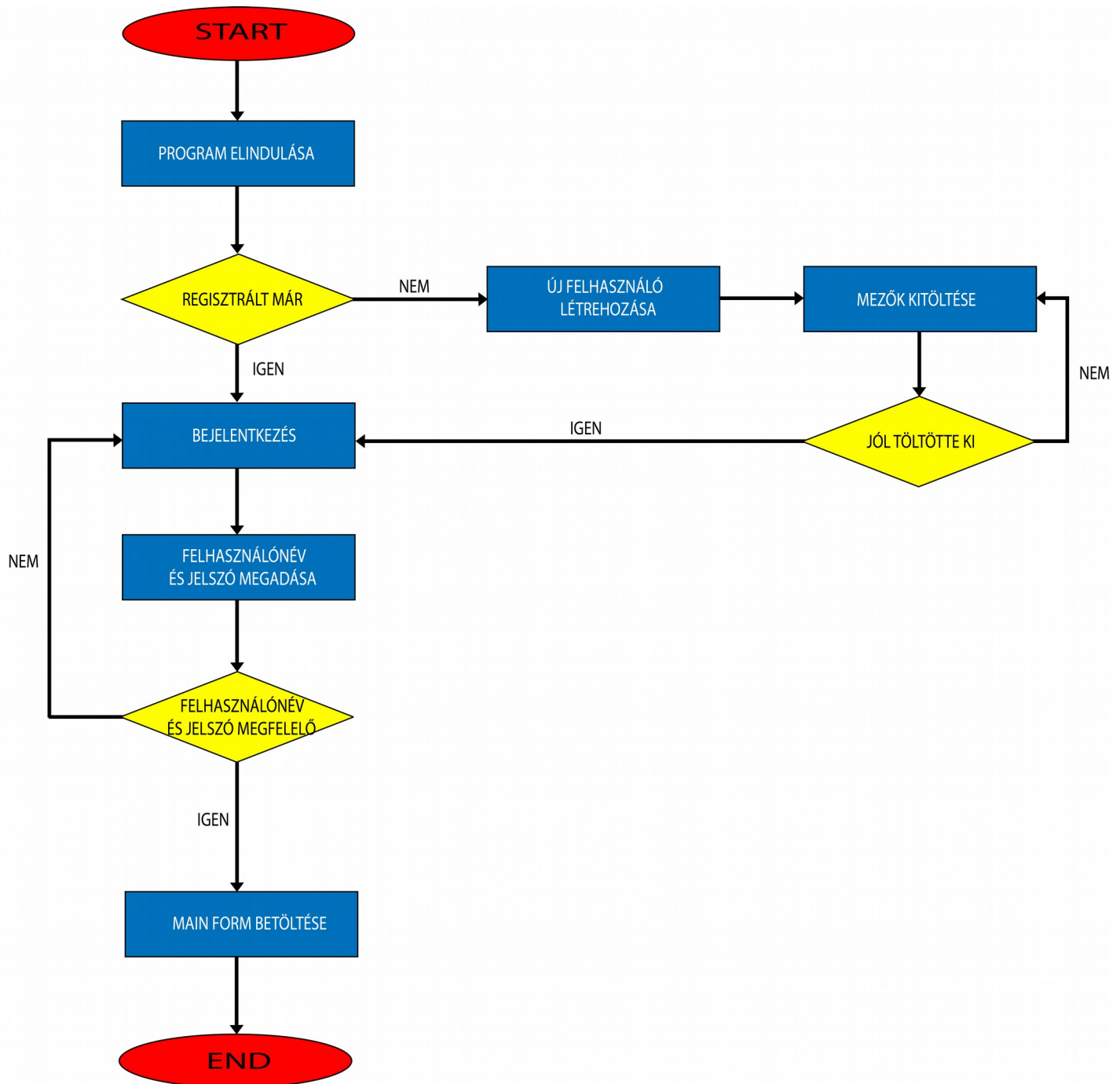
-raktáros form - Raktaros

-műszerész form -Muszeresz

-anyaggazdálkodó form -Gazdalkodo

A többi alform ezek valamelyikéből érhető el.

Bejelentkezés folyamata:



## Bejelentkező form fontosabb kódrészletei:

### Ip cím alapbeállítások:

A form betöltésekor az alábbi kódsorral tesszük lehetővé azt, hogy a felhasználó által bevitt adatok alapértelmezettek legyenek a programunk számára:

```
Ipcim.megadott_ip = Properties.Settings.Default.Ipcim;  
Ipcim.constr = Properties.Settings.Default.Constring;  
Ipcim.megadott_port=Properties.Settings.Default.Port;  
Ipcim.megadott_username = Properties.Settings.Default.Username;  
Ipcim.megadott_password = Properties.Settings.Default.Password;
```

Ha ezeket nem tölténénk be a form betöltésekor, akkor a program nem tudna arra az IP címre csatlakozni, amit korábban a felhasználó megadott futási időben az IP cím változtató formon(Ipcim.cs). Tehát csak kódból lehetne a fent felsorolt paramétereket megváltoztatni, ami telepítés után használhatatlanná tenné a programot, ha nem megfelelő a kódban rögzített IP cím.

### Jelszó:

Három dolgot kell megvizsgálunk a jelszónál, az egyik, hogy megvan-e minimum 8 karakter, a másik, hogy tartalmaz-e kis és nagy betűt, illetve számot, a harmadik pedig, hogy a jelszó ismétlésnél bevitt adatok megfelelnek-e az először megadott jelszóval.

Kis és nagybetű vizsgálata, tartalmaz-e számot, rövidebb-e 8 karakternél, illetve, hogy a két megadott jelszó azonos-e.

A felhasználó által bevitt jelszót és email címet a Regex osztállyal tudjuk megvizsgálni, hogy megfelelő formátumú-e. Létrehozunk egy objektumot az előbb említett osztályból, amiben megadjuk a várt formátumot, majd az IsMatch metódusa segítségével, amelynek átadjuk paraméterként a beviteli mező szövegét, letároljuk egy bool változóba a vizsgálat eredményét, amelyet később fel tudunk használni.

```
bool jelszocheck= regex_password.IsMatch(jelszo_Txt.Text);
```

### jelszó regex:

```
Regex regex_password = new Regex("^(?=.*[A-Z])(?=.*[0-9])[a-zA-Z0-9]+$");
```

### email regex:

```
Regex regex_email = new Regex(@"^[^\w!#$%&'*\+\/=?\^_`{|}~]+(\.[^\w!#$%&'*\+\/=?\^_`{|}~]+)*" +  
    "@" + @"((([^\-\\w]+\.)+[a-zA-Z]{2,4})|(([0-9]{1,3}\.){3}[0-9]{1,3}))$");
```

## A jelszó titkosítása az alábbi kódsor segítségével történik:

A titkosításhoz be kell emelni a névtérbe a System.Security.Cryptography-t, hogy el tudjuk érni a hozzá szükséges osztályokat.

```
var data = Encoding.ASCII.GetBytes(jelszo_Txt.Text); var  
md5 = new MD5CryptoServiceProvider();  
var md5data = md5.ComputeHash(data);  
string pwhash = "";  
  
foreach (var item in md5data)  
{  
    pwhash += item;  
}
```

Először is deklarál egy byte változót data néven, amibe lekódolja byte-ba a jelszo\_Txt textboxból bevitt stringet, majd deklarál egy titkosító osztályt md5 névvel. Ezután létrehoz egy md5data tömböt, amibe letárolja a Hash kódot úgy, hogy meghívja a ComputeHash() metódust. Ezután már csak annyit kell tenni, hogy deklarálunk egy stringet, amibe letároljuk a hasht összeállítva. Amihez előbb végigmegyünk egy foreach-el a tömbünkön.



## 6. Tesztelés:

A tesztelési folyamat igen összetett volt, mivel több formos a program, ezért egyenként végig kellett ellenőrizni mindet. Mivel elég sok textbox található összességében, ezért különös figyelmet kellett fordítani a karakter bevitel tesztelésre, hogy a program megfelelően működjön.

Egy programot lehet feketedobozos(black-box) technikával tesztelni, mikor a specifikáció alapján történik a tesztelés, tehát, hogy a specifikációban leírtakat végre tudja-e hajtani a program. Illetve lehet fehérdobozos(white-box) technikával, másnéven strukturális teszttel, mikor a forráskód alapján tesztelünk. Tehát, ha a forráskód a rendelkezésünkre áll, akkor tudunk fehérdobozos tesztelést végrehajtani, ha nem, akkor pedig specifikáció alapján. Forráskódnál tudunk különböző algoritmusokat, elágazásokat, metódusokat vizsgálni, specifikációnál viszont csak a bemenet-kimenetet tudjuk vizsgálni, és összevetni azt a specifikációban rögzítettel, hogy adott bemenetnél milyen kimenetnek kell megjeleníteni.

### 1. teszteset:

A tesztelés során az első hiba amibe ütköztem az a form betöltése volt egy másik formból, ami egy komponens teszt, mivel a programnak csak egy komponensét teszteli önmagában. Ez a hiba `ObjectDisposedException`-t dobott, amit akkor dob, ha egy object implementálja az `IDisposable` interface-t és meg lehet hívva rajta a `Disposable()` metódusa, majd azután ezt az objectet próbáljuk elérni. Tehát, ha van egy objektumunk, amin meghívjuk ezt a metódust, akkor azon nem tudunk már bizonyos műveleteket elvégezni. Ez a probléma merült fel itt is, ugyanis, ha bezárunk egy formot, az automatikusan meghívja a `Close()` metódusát, ami implementálja a `IDisposable.Dispose` metódusát. Ez a metódus annyit tesz, hogy felszabadítja a nem használt erőforrásokat, amit az objektum használ, amely implementálta ezt az interface-t és a metódusát. Vagyis, ha egy formot bezárunk a "hagyományos" módszerrel, akkor azt már nem tudjuk elérni. Ami ebben az esetben lényeges, az az, hogy nem mindegy hol deklaráljuk azt a formot, amelyiket meg szeretnénk nyitni a másik formból, ugyanis az előbb említettek miatt, ha a formot nem a `button_click` eventhandlerjén belül deklaráljuk, akkor az történik, hogy egyszer létrehozuk, majd a bezárást követően az a példány megsemmisül, éppen ezért nem is tudjuk elérni. Viszont, ha a `button` eventhandlerjében történik a deklaráció, akkor ez nem törthénhet meg, mivel minden egyes kattintásnál létrehozunk egyet, amit bezáráskor meg is semmisítünk.

### 2. teszteset:

A második tesztelésem az előzőhöz köthető. A form betöltése többször végrehajtható, viszont nem a specifikációban rögzítetteknek megfelelő az eredmény, ugyanis ugyanazt a formot többször is meg lehet nyitni. Erre az a megoldás, hogy meg kell vizsgálni az éppen megnyitott formokat, végigmenni rajtuk egy foreach-el, és ha az adott form Name tulajdonsága megegyezik azzal, amit meg akarunk nyitni, akkor a BringToFront() metódust hívjuk meg rajta, ezáltal az előtérbe hozva azt. Ha nincs megnyitva, akkor pedig példányosítjuk és a Show() metódus segítségével megjelenítjük.

### 3. teszteset:

A 3. teszteset is a form bezárásához köthető. A probléma az, hogy mikor kilépünk a Raktáros formból, ami úgymond egy main form, akkor nem zárja be azokat a formokat, amik erről a formról lettek megnyitva. Tehát kilépésnél csak maga a Raktáros form záródik be, visszalép a Login formra, de továbbra is megnyitva hagyja azokat a formokat, amiknek a használatához csak a raktárosnak van jogosultsága. Ha ezek a formok le vannak rakva a tálcára, és a raktáros elfelejti, hogy valamelyiket nem zárta be, akkor ez probléma lehet a munkahelyen, ha többen használják az adott gépet, hogy olyan fér hozzá, aki nem jogosult a form használatához.

Ezt a problémát az alábbi kódsorral oldottam meg:

```
public void Raktaros_FormClosing(object sender, FormClosingEventArgs e)
{
    Login log = new Login();
    log.Show(); this.Dispose();

    for (int i = Application.OpenForms.Count - 1; i >= 0; i--)
    {
        if (Application.OpenForms[i].Name != "Login") Application.OpenForms[i].Close();
    }
}
```

Mivel a Login formon mindig meghívjuk a Hide() metódust a bejelentkezésnél, ezért itt nem kell külön arra figyelni, hogy BringToFront(), vagy Show() metódust használjunk a megjelenítéshez. Példányosítjuk, és a Show() metódussal megjelenítjük. Az adott formunkon pedig meghívjuk a Dispose() metódust. Fontos, hogy ha az adott formot az adott form FormClosing event-jével zárjuk

be, vagyis, ha a form jobb felső sarkában lévő X-re kattintunk, akkor az eventhandler metódusába mindenképp a Dispose() metódust hívjuk meg, és ne a Close()-t. Tesztelés során az előbb említett esetben a Close() metódus meghívásánál az történt, hogy a Login form megnyitása bekerült egy végtelen loop-ba, és egymásután nyitotta meg sorban azt.

A for ciklusban végigmegy az összes éppen futó formon, majd egy if-el megvizsgálja, hogy az éppen megnyitott form Name tulajdonsága a Login. Ha nem az, akkor bezárja azt.

#### 4. teszteset:

Ha úgy hajtunk végre egy SQL lekérdezést, hogy a hozzá szükséges paraméterek egyikét textboxból adjuk meg, akkor figyelniünk kell arra, hogy a lekérdezés csak akkor hajtsódjon végre, ha az adott textbox nem üres. Ellenkező esetben hibaüzenetet dob a program. Ezt úgy tudjuk elkerülni, hogy egy if segítségével megvizsgáljuk, hogy az adott textbox nem üres-e. Ezt az alábbi módon tudjuk megtenni: `!String.IsNullOrEmpty(cikkNev_txtbox.Text)` A String osztálynak van egy olyan metódusa, hogy `IsNullOrEmpty`, ami segítségével meg tudjuk vizsgálni, hogy az adott string üres-e, null értékű, vagy whitespace-t tartalmaz. Egy string akkor üres, ha deklarálva van üresként, ami két módon lehetséges; vagy úgy deklaráljuk, hogy `String.Empty`, vagy pedig átadunk neki értékül egy üres stringet, így: `""`. A white space pedig a textboxba beírt szóközöket vizsgálja. Ha ezt a három dolgot megvizsgáljuk, és hamis értéket dob vissza, akkor a stringbe mindenképp van valami írva, így a lekérdezéshez átadott paraméterünk biztos, hogy nem üres.

#### 5. teszteset:

Ezt a tesztesetet, illetve problémát csak általánosságban veszem át. A felmerülő probléma az volt, hogy a megnyitott jó pár `MySqlConnection` nem lett lezárva, ezáltal nem lettek felszabadítva a lefoglalt memóriaterületek, ez pedig ahhoz vezetett, hogy `StackOverflowException`-t dobott a program, amit akkor tesz, ha a végrehajtási stackje megtelik a meghívott metódusokkal. Ezt úgy oldottam meg, hogy ezeket a `MySqlConnection`-öket egy using blokkon belül deklaráltam, így miután elvégezte a program a feladatát az adott connection példányával, megsemmisítette azt.

### 6. teszteset:

A hibaüzenetek megjelenítése. Felhasználói tesztelésnél zavaró tényezőként jelentkezett, hogy a hibaüzenetek külön-külön egymásután ugrottak fel, ha egyszerre több hiba merült fel például új felhasználó létrehozásakor.

Ezt az alábbi módon javítottam ki: deklaráltam egy üres stringet error néven, majd minden egyes hibánál ehhez a stringhez hozzáadtam a hibaüzenetet egy új sorban. Majd megvizsgáltam, hogy a string üres-e, ha nem, akkor megjelenítettem a hibákat, mindet külön sorban.

```
error += "Hiányos adatok!" + Environment.NewLine;

if (!String.IsNullOrEmpty(error))
{
    MessageBox.Show(error, "Hiba", MessageBoxButtons.OK,
        MessageBoxIcon.Exclamation);
}
```

Ezután már a hibák egy sorban jelentek meg a felugró figyelmeztetésben.

## 7. Továbbfejlesztési ötletek

Még további funkciókat lehetne beépíteni a programba. Az alformok betöltésére szolgáló gombok panelje igény szerint bővíthető. Esetleg a programon belüli levelezést is meg lehetne valósítani, ha van rá külön igény, bár ez nem létszükségletű, mivel más levelező oldalak már vannak, amelyek ingyenesek, és majdnem teljesen bugmentesek, ezért megítélésem szerint erre nincs külön szükség, de, ha az ügyfél úgy dönt, akkor egy ilyen modult is lehetne bele implementálni. Meg lehetne valósítani benne, hogy az email cím változtatásnál a konkrét megadott email címre küldené el a változtatási kérelmet. Lehetne még alakítani esetleg a kezelő felületen, vagy a design-on, bár számomra ez a mostani is kényelmes volt. A menu stripbe lehet több menüpontot is beletenni.

## 8. Irodalomjegyzék

Használt könyvek:

- C# programozás lépésről lépésre - Reiter István
- Head first C# - Andrew Stellman (2008)

Használt oldalak:

- <https://docs.microsoft.com>
- <https://stackoverflow.com>

Felhasznált képek (a projekten belül az összes kép ingyenes, szabadon felhasználható):

- <https://cloudcommercepro.com/home2019/warehouse-icon-2>
- [https://www.pngfind.com/mpng/ihxoxh\\_free-png-download-dollar-sign-in-gold-png/](https://www.pngfind.com/mpng/ihxoxh_free-png-download-dollar-sign-in-gold-png/)
- [https://www.teahub.io/viewwp/ihxoRJ\\_stunning-white-polygon-wallpaper-images-for-free-download/](https://www.teahub.io/viewwp/ihxoRJ_stunning-white-polygon-wallpaper-images-for-free-download/)
- <https://www.pngaaa.com/detail/41731>