

# ZÁRÓDOLGOZAT

Nagy Dávid

2019



# **Készletnyilvántartó**

**Nagy Dávid**

**SZÁMALK-Szalézi Szakgimnázium**

**Szoftverfejlesztő**

**Konzulens: Kaczur Sándor**

### Nyilatkozat

a záródolgozat eredetiségéről

Alulírott Nagy Dávid {Dobos Edit (anyja neve) 080807PA (szem. ig. szám)} büntetőjogi és fegyelmi felelősségem tudatában kijelentem és aláírással igazolom, hogy a záródolgozat saját munkám eredménye. A felhasznált irodalmi és egyéb információs forrásokat az előírásoknak megfelelően kezeltem, a záródolgozat készítésre vonatkozó szabályokat betartottam.

Kijelentem, hogy ahol mások eredményeit, szavait vagy gondolatait idéztem, azt a záródolgozatomban minden esetben, beazonosítható módon feltüntettem, a dolgozatban található fotók és ábrák közzétételével pedig mások szerzői jogait nem sértem.

Kijelentem, hogy a záródolgozatom elektronikus változata teljes egészében megegyezik a nyomtatott formával.

Hozzájárulok ahhoz, hogy az érvényben lévő jogszabályok és a Számalk-Szalézi Szakgimnázium belső szabályzata alapján az iskola saját könyvtárában megtekinthető (olvasható) legyen a záródolgozatom.

A záródolgozat titkos/nem titkos.

Budapest, 2019. április 1.

.....  
Tanuló aláírása

## Tartalomjegyzék

1	Bevezetés .....	4
1.1	Eszközök: .....	4
2	Feladatspecifikáció .....	5
3	Tervezés .....	6
3.1	Az adatbázis megtervezése .....	6
3.2	Adatelérési Objektum Data Access Object (DAO) minta.....	12
3.3	MVC Modell .....	13
3.4	A program osztályai .....	13
3.5	A felhasználói felület megtervezése.....	15
4	Megvalósítás .....	21
4.1	Bejelentkezés.....	21
4.2	Termék hozzáadása .....	22
4.3	Fájl beolvasás .....	24
4.4	Termék készlet megtekintése .....	26
4.5	Keresés eredmények között.....	28
4.6	Szállítólevél.....	30
4.7	Jelentések .....	33
5	Tesztelés.....	35
6	Felhasználói dokumentáció.....	39
6.1	Bejelentkezés: .....	39
6.2	Termék hozzáadása: .....	40
6.3	Raktár készlet megtekintése: .....	41
6.4	Szállítólevél készítés: .....	41
6.5	Vevők kezelése: .....	42
6.6	Beszállítók Kezelése: .....	43

6.7	Felhasználók kezelése: .....	43
6.8	Jelentések: .....	43
6.9	Kijelentkezés: .....	44
6.10	Programot készítette: .....	44
7	Továbbfejlesztési lehetőségek .....	45
8	Összefoglalás .....	47
	Irodalomjegyzék .....	48
	A melléklet tartalma.....	49
	Köszönetnyilvánítás.....	50

## Ábrajegyzék

1. E-K diagram.....	7
2. Kapcsolati-ábra .....	11
3. MVC-modell.....	13
4. Dao-minta .....	12
5. Bejelentkezési felület terv.....	15
6. Hozzáadási felület terv.....	16
7. Kezelési felület terv .....	17
8. Kezelési felület terv .....	18
9. UML osztály diagram .....	14
10. Bejelentkezési felület képernyőkép .....	39
11. Kliens felület képernyőkép .....	40
12. Kezelési felület képernyőkép .....	41
13. Vevők feltöltése felület képernyőkép .....	42
14. Jelentések felület képernyőkép .....	43

# **1 Bevezetés**

Egy cég számára alapvető a megfelelő szervezettség és átláthatóság az eladásra kínált termékei, illetve ezek kezelését végző alkalmazottaik, felett a hatékony és minél profitálóbbr működéshez. Egészen a termékek beszerzésétől az eladás pillanatáig végig kell tudnunk követni, illetve befolyásolni a folyamatokat és képesnek kell lennünk ezek megkönnyítésére szolgáló megoldásokat kifejleszteni. Gördülékenyebbé tenni, a tapasztalatokat leszűrve fejleszteni a hatékonyságunkat. Az általam választott szakdolgozati téma ezen szervezési ütemezési problémákat kívánja megoldani. Ezen kívül nem csupán egy általam az üzleti életben hasznosnak tartott szoftver megvalósítása volt ami miatt ezen téma mellett döntöttem, de szakmai kihívást, minél több új technológia tervezési minta és módszer megismerése, valamint elsajátítása is motivált.

## **1.1 Eszközök:**

programozási nyelvek: Java, SQL

Adatbáziskezelő: Microsoft Sql Server Manager 17.9

Fejlesztőkörnyezet: Netbeans IDE 8.2

## 2 Feladatspecifikáció

A készletnyilvántartó szoftver célja többfelhasználó számára lehetőséget biztosítani egy adatbázishoz való hozzáféréshez valós időben. Bővebben kifejtve funkciókat biztosítani nekik a raktáron lévő termékek, partnerek és dolgozók adatainak tárolására, kezelésére, illetve egyéb raktári folyamatok például: szállítólevél generálás kimutatások készítése, illetve ezek adataik kezelése és áttekinthetőbbé tétele. A bejelentkezésnél két jogosultsági szint érhető el egy felhasználónak, aki csak egy ilyen jogosultsági szinthez tartozhat, amit az első felvitt legmagasabb jogosultságú felhasználó(Menedzser)-től kezdve Menedzserek feladata meghatározni mindenki másnak ,illetve ezen kívül ad hozzá minden adatot az adatbázishoz ezzel kreálva nekik felhasználói fiókot ami teljesen kiváltja a regisztrációt. Minden jogosultsági körhöz egy sémára épülő de funkciók számában eltérő, ezeket elérhetővé tévő menü felület tartozik



## **3 Tervezés**

### **3.1 Az adatbázis megtervezése**

#### **Az ügyviteli funkciók:**

##### **Felhasználók kezelése**

- Felhasználók törzsadatainak rögzítése, karbantartása Menedzser által
- egy Menedzser oszt jogot
- jelszó módosítása
- senki sem regisztrál

##### **Termékek kezelése**

- új termékek adatainak rögzítése
- beszerzések rögzítése, a pillanatnyi készlet frissítésével
- termékek logikai törlésekor frissíteni törölt-e mező értékét alapértelmezett 0-ról 1-re

##### **Partnerek kezelése**

- A megrendelők és beszállítók törzsadatainak rögzítése
- partnerek törzsadatainak módosítása, illetve törlése

##### **Szállítólevél készítés**

- a szállítólevelek összeállítása valamely partnerhez tartozó, teljesítésre váró rendelési tételekből, a termékek készletének figyelembevételével

##### **Rendelések**

- a megrendelések felvitele
- feltétel szerinti törlés
- a rendelések teljesíthetőségével kapcsolatos ellenőrzések
  - csak olyan termékekből állítható össze amelyek logikai törlésen nem estek át, azaz termék törölt-e értéke az alapértelmezett hamis értékű
- árukészlet csökkentése megrendelt árucikkek mennyiségével
- bruttó végösszeg számítása konstans 27% áfakulccsal

## Táblák:

(jelölés: TÁBLA (Elsődlegeskulcs, leíró, **Külsőkulcs**)

Termék:(termékID, terméknév, kategóriaID, egységár, mennyiség, leírás, törölt-e)

Beszállító:(beszállítóID, beszállítónév, elérhetőségID)

Beszerzés:(beszerzésID, **beszállítóID**, **termékID**, mennyiség, dátum)

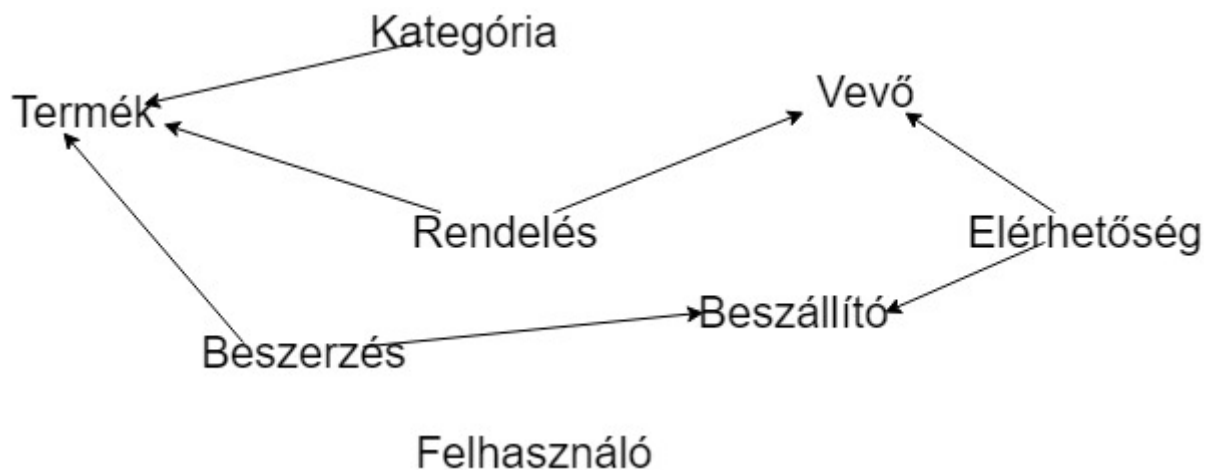
Vevő:(vevőID, vevőnév, elérhetőségID)

Rendelés:(rendelésID, dátum, **vevőID**, **termékID**, mennyiség, végösszeg)

Felhasználó:(felhnév, jelszó, jogkör, nyomtatási név, **elérhetőségID**)

Kategória szótár tábla(kategóriaID, kategórianév)

Elérhetőség(elérhetőségID, email, telefon, irsz, város, utca, házszám, weblap)



## 1. E-K diagram

### Lekérdezések:

- Havi bruttó és nettó bevételek
- Termékek aktuális raktárkészlete
- Negyedéves bevételek
- Készletbecslés
- top 5 legnépszerűbb termék
- top 5 legtöbbet vásárló
- top 5 legtöbbet beszállító

## Táblák szerkezete a táblaszintű megszorításokkal

### Termék

mező	adattípus	szerep	korlátozás
termékID	int	Kulcs, generált	
terméknév	varchar(50)	kötelező	
kategóriaID	int	Külső Kulcs	
mennyiség	int	kötelező	mennyiség $\geq$ 0
egységár	money	kötelező	egységár $>$ 0
leírás	varchar(100)	nem kötelező	
törölt-e	bit	kötelező	alapértelmezett 0-hamis

### további megszorítások:

Rendelés összeállításához bármilyen nyilvántartás csak törölt-e alapértelmezett hamis értéken álló termékeket tartalmazhat.

### Beszállító

mező	adattípus	szerep	korlátozás
beszállítóID	int	Kulcs	
beszállítónév	varchar(50)	kötelező	
elérhetőségID	int	Külső Kulcs	

### Beszerezés

mező	adattípus	szerep	korlátozás
beszerzésID	int	Kulcs	
beszállítóID	int	Kulcs, Külső Kulcs	
termékID	int	Kulcs, Külső Kulcs	
mennyiség	int	kötelező	mennyiség $>$ 0
dátum	date	kötelező	$\geq$ getdate()

### Vevő

mező	adattípus	szerep	korlátozás
vevőID	int	Kulcs	

vevőnév	varchar(50)	kötelező	
elérhetőségID	int	Külső Kulcs	

### Rendelés

mező	adattípus	szerep	korlátozás
rendelésID	int	Kulcs	
dátum	date	Kulcs	>=getdate()
vevőID	int	Kulcs, Külső Kulcs	
termékID	int	Kulcs, Külső Kulcs	
mennyiség	int	kötelező	mennyiség>0
végösszeg	money	kötelező	bruttó, Forint

### Felhasználó

mező	adattípus	szerep	korlátozás
felhnév	varchar(15)	Kulcs	
jelszó	varchar(15)	kötelező	egyedi
jogkör	bit	kötelező	0 vagy 1
nyomtatottnév	varchar(25)	kötelező	
elérhetőségID	int	Külső Kulcs	

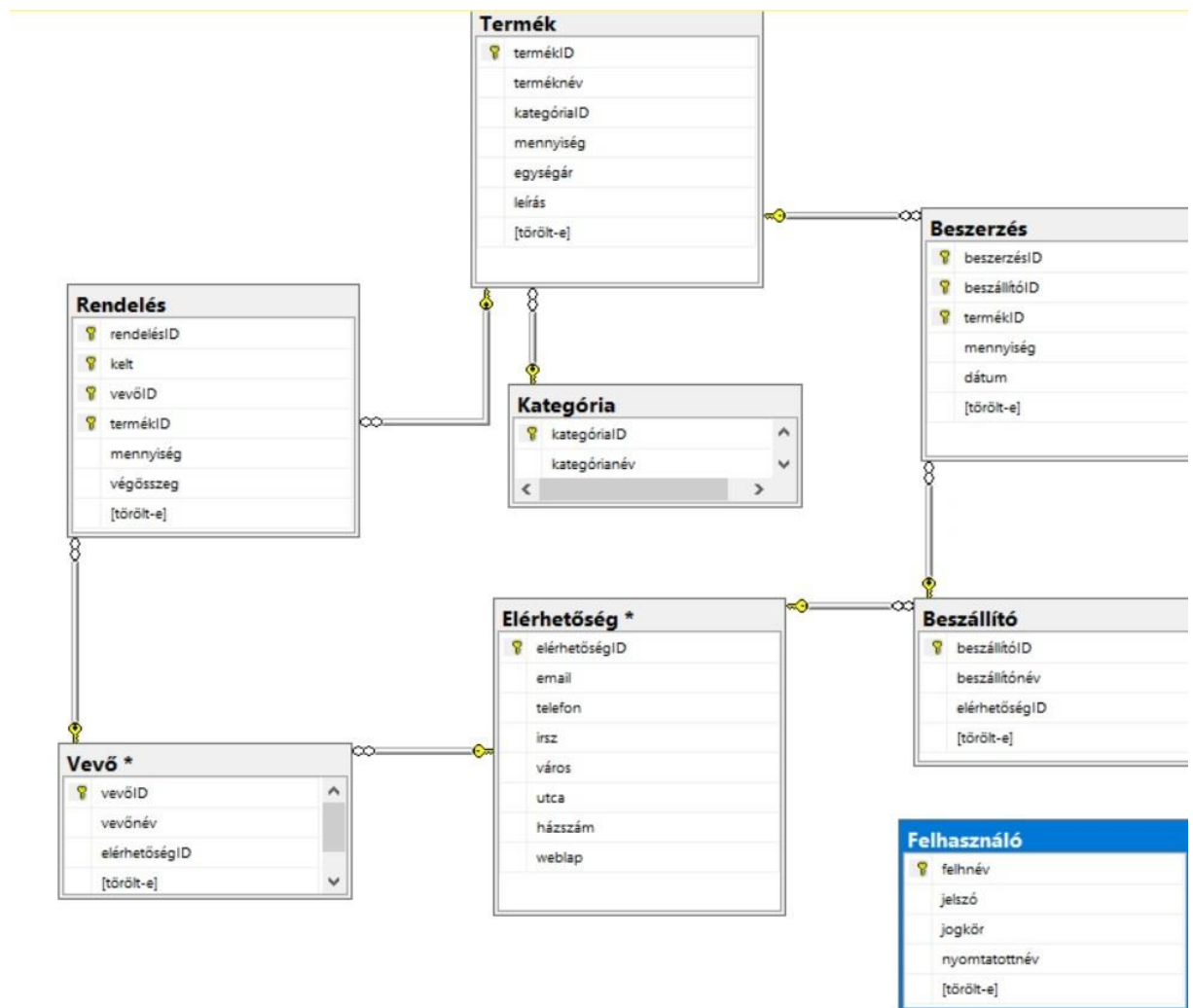
jogkör csak két állapot lehet ha 0 dolgozó, ha 1 eladó

### Kategória

mező	adattípus	szerep	korlátozás
kategóriaID	int	Kulcs	
kategórianév	varchar(50)	kötelező	

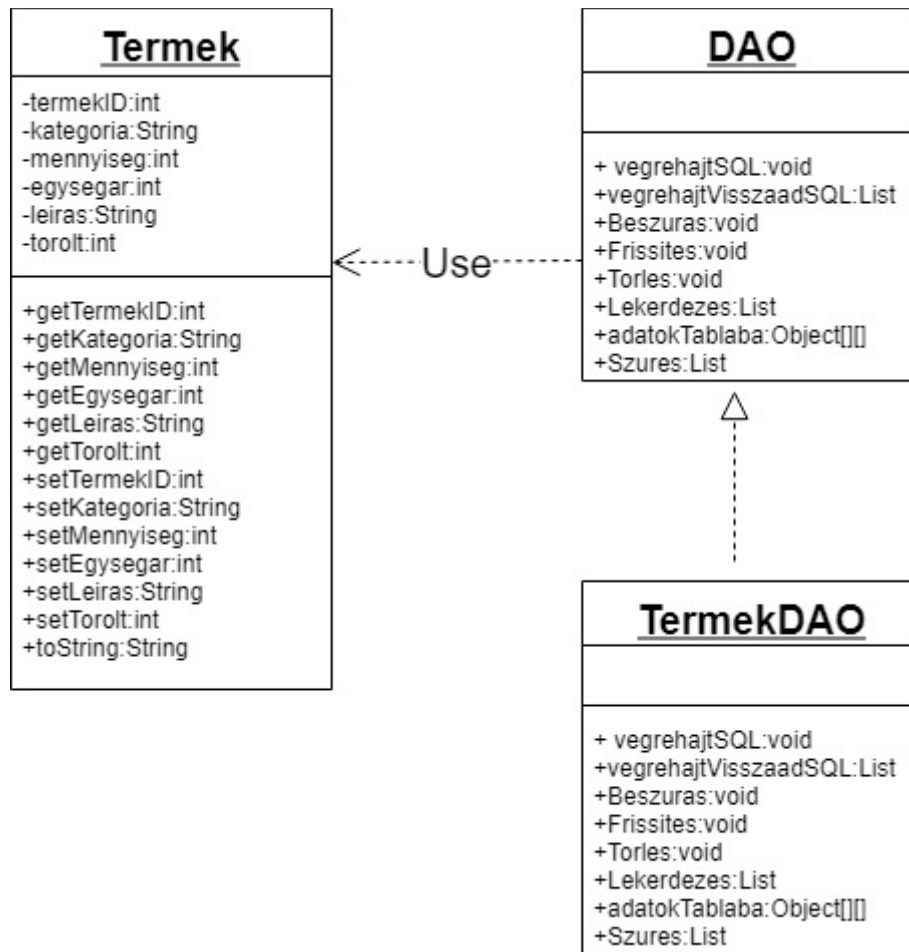
### Elérhetőség

mező	adattípus	szerep	korlátozás
elérhetőségID	int	Kulcs	
email	varchar(30)		
telefon	varchar(20)	kötelező	
írsz	varchar(4)	kötelező	
város	varchar(50)	kötelező	
utca	varchar(25)	kötelező	
házszám	int	kötelező	
weblap	varchar(25)		



2. Kapcsolati-ábra

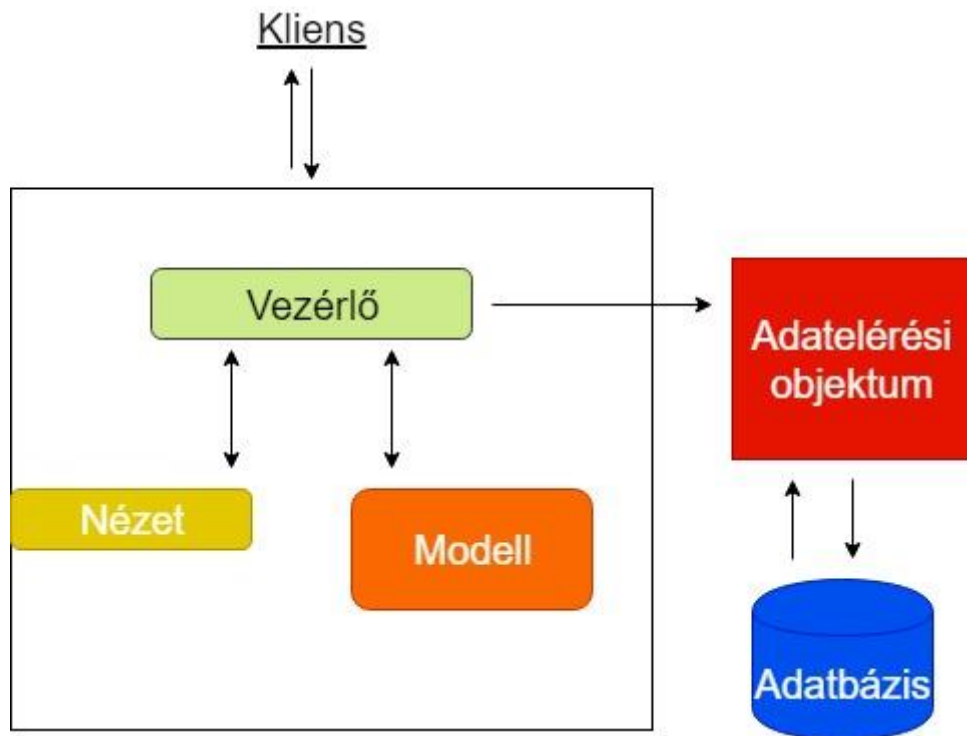
### 3.2 Adatelérési Objektum Data Access Object (DAO) minta



#### 3. Dao-minta

Az adatbázis elérési objektum lényegében egy séma amely adatbázis alapl műveleteket beszúrást, törlést, frissítést és lekérdezést foglal magába. Ezen felül pedig kapcsolatot teremt az adatbázis és az objektumok között hogy azok adattagjainak és adatbázis műveleteivel képesek legyünk befolyásolni az adatbázis tartalmát. Az eredeti dao egy interface, amit minden egyedi osztály esetében implementálunk és személyre szabjuk metódusainak tartalmát, az osztály és az adatbázis táblastruktúráinak fényében.

### 3.3 MVC Modell



4. MVC-modell

Az MVC modell célja elválasztani egymástól a felhasználói nézetet és az üzleti logikát. Ezáltal jól átlátható és könnyebben karbantarthatóvá válik az alkalmazás. Ezen MVC paradigma elemei és azok kapcsolatai jól megfigyelhető a fentebb látható ábrán. Nézet és modell sosem lehet egymással közvetlen kapcsolatban ezt Vezérlő közbeágyazása biztosítja. Vezérlő hívja Adatelérési objektumot, amely kapcsolatot teremt az adatbázissal.

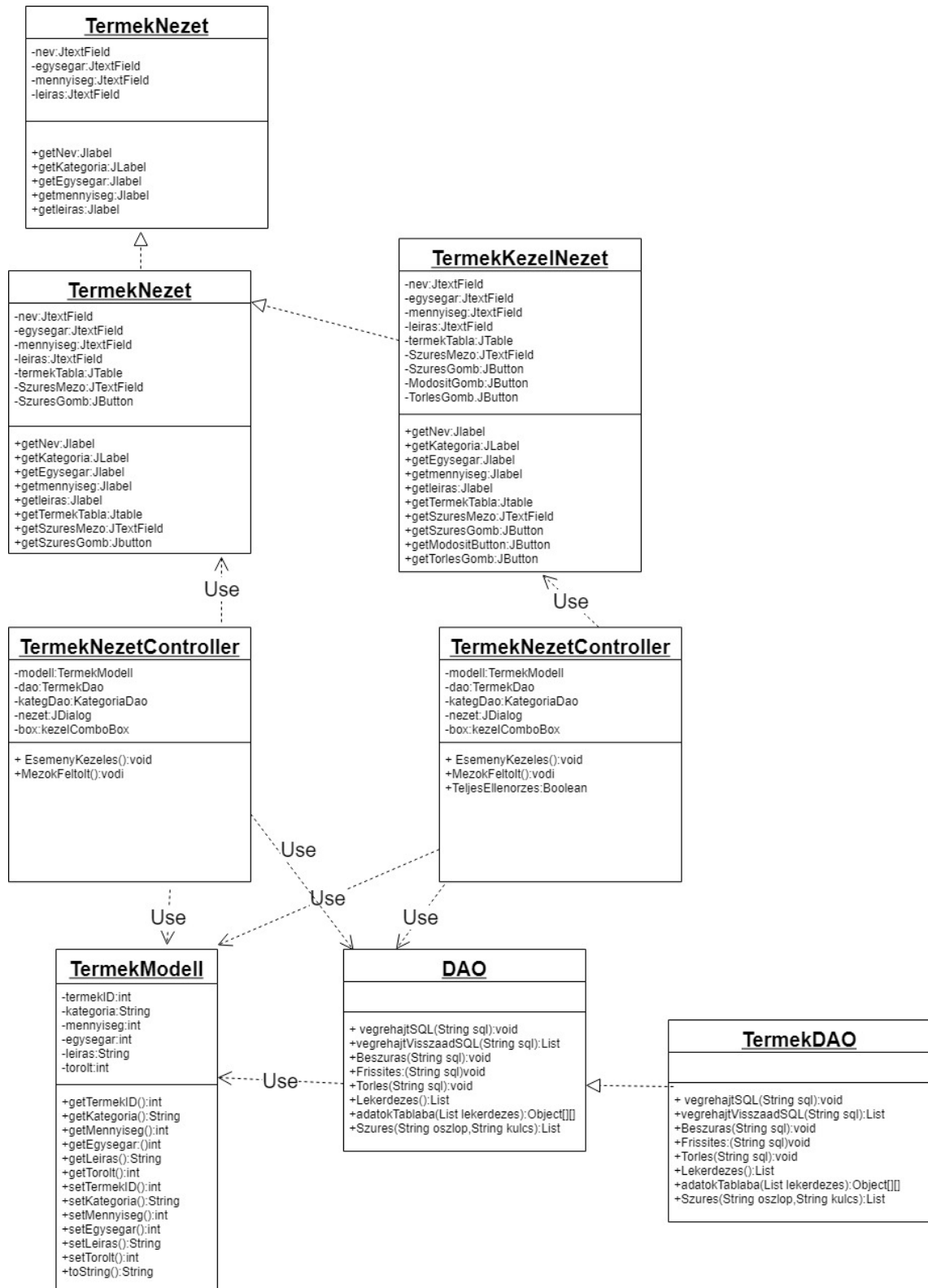
### 3.4 A program osztályai

Az alábbi ábra bemutatja program Termékekhez tartozó osztályainak szerkezetét és azok kapcsolatait egymással az MVC tervezési minta szerint. A program többi területét érintő osztályok is ezen tervezési architektúrára épülnek, természetesen adattagjaik és néhány metódust leszámítva, mindazonáltal ezen ábra képes áttekinthető rövid bemutatást biztosítani program működéséhez.

Nézetek a legprimitívebbtől kezdve öröklik az előző nézet felület adattagjai és beállításait annak legösszetettebb tagjáig. Minden nézethez amely már öröklő tartozik



egy vezérlő, annak függvényében, hogy adott funkció jogosultságok között hány féle variációban oszlik meg. Modell és DAO működését ezek nem érintik.



5. UML osztály diagram

### 3.5 A felhasználói felület megtervezése

Összesen 1 Bejelentkezési, 2 felhasználói kliens felület név szerint Menedzser és Dolgozó, valamint ezekhez tartozóan szétosztva különböző karbantartási és egyéb ügyviteli felületek

Cím	
Felhasználónév	<input type="text" value="Szövegmező"/>
Jelszó	<input type="text" value="Jelszómező"/>
<input type="button" value="Bejelentkezés"/>	

6. Bejelentkezési felület terv

A program indulásakor elsőként betölt a bejelentkezési felület. Regisztrációra nincs lehetőség minden alkalmazottnak jogkörtől függetlenül egy már létező Menedzser hozhat létre fiókot felhasználónévvel, jelszóval amit később megváltoztathat a felhasználó saját maga, illetve meghatározhat neki egy jogkört a 2 fent említettből ami fix. Ha rendelkezik ezekkel az adatokkal akkor juthat tovább a jogkörének megfelelő felülethez és ahhoz tartozó funkciókhoz.

Általános séma szerint a vevők, beszállítók, felhasználók, illetve termékekhez tartozóan adott egy csak felvitelre szolgáló felület, illetve egy megtekintésre, szűrésre alkalmas, bizonyos esetekben a magasabb jogkörnek kiterjesztve ez utóbbi felülethez tartozóan módosítás és törlés is elérhető nevezzük ezt karbantartási felületnek. Természetesen az adott felvitel és karbantartás felületek más adatokkal operálnak az adott egyed függvényében de felépítési struktúrájukban szinte teljesen megegyeznek ezért minden eset bemutatása helyett egyes típusokra hozok példákat.

### Termék felvitel felület:

Cím	
Terméknév szövegmező	
Kategóriák legördülőlista	
Mennyiség szövegmező	
Egységár szövegmező	
Leírás szövegdoz	
Importálás gomb	Hozzáadás gomb

#### 7. Hozzáadási felület terv

Minden ilyen felület tartalmaz textfieldeket az adatok bevitelére, egy importálás gombot, ami adott excel fájl tartalmával kiváltja a kézi kitöltést és egy hozzáadás gombot a véglegesítéshez és adatbázisba való feltöltéshez. Ez a felület pluszként tartalmaz egy kategóriák lenyíló listát ahonnan kiválasztható előre felvitt kategóriák, hogy ne kelljen minden esetben beírni a gyakran előforduló eseteket. Természetesen ehhez pluszban tartozik egy kategória felvitel felület az új esetekhez, ezen kívül minden felvitel azonos struktúrán alapul.

### Termék nézet, kezelés felület:

Cím	
Termék azonosító	Szűrési feltétel legördülőlista
Terméknév szövegmező	Szűrés szövegmező
Kategóriák legördülőlista	Szűrés gomb
Mennyiség szövegmező	Termék táblázat
Egységár szövegmező	
Leírás szövegdoz	
Törés gomb	
Módosítás gomb	

#### 8. Kezelési felület terv

A megnyitáskor automatikusan megjelennek minden eddig felvitt termék adatai. Szűrhetünk közöttük név, vagy kategória esetleg egységár alapján. Kurzorral rákattinthatunk egy adott sorra és akkor adatai betöltődnek az adott mezők be ahol átírhatóak és módosítás gombbal ha megfelelnek a módosított adatok a feltételeknek és nem kapunk hibaüzenetet, akkor végrehajtódik a módosítás az adott termékhez. Természetesen törölhetünk is egy ilyen kijelölt terméknel ahol kapunk egy ablakot ahol jóvá kell hagyni, hogy biztosak vagyunk-e benne. Illetve Menedzser kezelheti a legenerált szállítólevelek eltárolt adatait.

vevők szűrése szöveg mező			Szűrési feltétel legördülő lista			Szűrés gomb			Termékek szűrése Szöveg mező			Szűrési feltétel legördülő lista			Szűrés gomb		
Vevők adatai táblázat									Termékek adatai táblázat								
									mennyiség beviteli mező								
Hozzáadás gomb									PDF készítés gomb								

## 9. Kezelési felület terv

A két táblázatot egy az egyében átvesszük a vevők és termékek kezelése nézettől adataikkal együtt és ugyanúgy szűkítjük és így választjuk ki a szállítólevélhez szükséges szinte minden adatot. PDF-et generál belőle véglegesen illetve egy előnézet PDF is elérhető amit a végleges kreálásakor eldob.

## 4 A program működési elve

### Nyilvántartás/karbantartás:

termékek, vevők és beszállítók, illetve felhasználók adatainak felvitele egy grafikus felhasználói felületen és ellenőrzésen átment adatok feltöltése egy relációs adatbázisba. Adatbázisban tárolt adatokat tudjuk módosítani, illetve törölni, valamint különféle lekérdezésekkel kinyerni a felületek számára ezeket az adatokat.

### PDF generálás:

szállítólevél vagy egy más néven hivatkozott bizonylathoz, ami alapján nyilvántartható a be és kimenő termékek és pénzösszegek. A vevő cég, vagy magánszemély adatai a termékek azonosítója, megnevezése, mennyisége végösszege. az aktuális ár alapján. Vevők és termékek adatait adatbázisból lekérdezzük és hozzáadjuk PDF tartalmához, illetve saját céges adatokat fájlból importáljuk.

Illetőleg A jelentések felületen megadható két dátum közötti intervallumhoz generál kimutatás PDF-et például: mennyi bevétel volt ebben a hónapban, illetve kiadás, valamint termékek azonosítónként rendezve a bejött és eladott mennyiség, a bent lévő mennyiség összértéke stb. Rögtön generálható egy végleges PDF, de egy előnézet is amit eldob amint elkészült a végleges példány.

### Importálás:

Partnerek, illetve termékek adatainak importálása excel fájlból a beviteli felületek számára kézi kitöltés kiváltására. Egy fájlválasztó ablak segítségével kiválasztható a kívánt fájl, ha a fájl kiterjesztése, szerkezeti integritása és minden adata átmegy az ellenőrzésen, illetve UTF-8 kódolásúak, akkor azok feltöltésre kerülnek az adatbázisba. Egyébként megadott hibaüzenettel felugróablak jelenik meg.

### Jogok, Hozzáférések kezelése

#### Dolgozó hozzáférése:

- termékek nézete felülethez ahol szűrhet a termékek között,
- vevők hozzáadása, illetve felvitt vevők adatainak megtekintése, módosítása
- Szállítólevél generálása

### Menedzser hozzáférése:

a kisebb jogkör által is használható felületekhez, illetve kiterjeszti azokat:

- termékek nézet felületen törölhet, illetve módosíthat is.
- Beszállítók adatainak felvitele felület, illetve adataik karbantartása felület
- új felhasználók felvitele és már létezők karbantartása felület
- Jelentések/kimutatások generálása
- Szállítólevelek kezelése

## 5 Megvalósítás

### 5.1 Bejelentkezés

```
public void egyeztetes() {
    bejelentAblak.getBtnBejelentkezes().addActionListener(e->{
        felhNev=bejelentAblak.getTfFelhNev().getText();
        System.out.println(felhNev);
        jelszo=new String(bejelentAblak.getTfJelszo().getPassword());
        System.out.println(jelszo);
        int n=0;
        String üzenet="";
        System.out.println(titkos.titkositas(jelszo));
        try {
            felhLista=dao.Lekerdezes();
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(ablakElado, ex);
        }
        for (felhasznaloModell felhasznalo : felhLista) {
            if(felhasznalo.getFelhNev().equals(felhNev)&&
                felhasznalo.getJelszo().equals(titkos.titkositas(jelszo))) {
                n++;
                nyomtatottNev=felhasznalo.getNyomtatottNev();
                jog=felhasznalo.getJogkor();
                break;
            }
        }
        if(n==0) {
            üzenet="Hibás felhasználó név vagy jelszó";
        }
        else{üzenet="Sikeres bejelentkezés";}
        if(n!=0 && jog==1) {
            ablakMenedzser= new MenedzserKliens();
            ablakMenedzser.setVisible(true);
            bejelentAblak.dispose();
        }
        else if(n!=0 && jog==0) {ablakElado=new eladoKliens();
            ablakElado.setVisible(true);
            bejelentAblak.dispose();
        }
        JOptionPane.showMessageDialog(bejelentAblak, üzenet,"Üzenet" , 1);
    });
}
```

BejelentkezesController [nézettől](#) átvett gombhoz eseménykezelést rendel hozzá amely lényege, hogy lenyomást követően vizsgálja, hogy felületen felvitt mezőben lévő felhasználónév megtalálható-e az adatbázis felhasználó táblájában lévő valamely rekorddal, illetve jelszó Sha-256-os kódolással megegyezik az ehhez a felhasználóhoz tartozó szintén ilyen kódolással tárolt jelszóval. Amennyiben van egyezés megvizsgálja, hogy milyen jogosultsági körbe tartozik az adott felhasználó és ennek megfelelően betölti a számára elérhető kliens felületet. Ez lehet például Menedzser kliens



## 5.2 Termék hozzáadása

```
public termékHozzaadController() {  
  
    dao = new TermékDAO();  
    vizsgálat = new Validálás();  
  
    nezet = new termékHozzaadNezet(new adminKliens(), true);  
    nezet.setVisible(true);  
  
    box = new kezelComboBox();  
    box.feltoltComboBox(nezet.getScrKategória(), kategDAO, 0, 0);  
    esemenyKezeles();  
    nezet.setVisible(true);  
}
```

TermekHozzaAdasController kezeli a a felhasználói felületet , ami JDialog típusú osztály ami adatok grafikus megjelenítésért felelős felhasználók számára, illetve adatokat vár.

```
public termékHozzaadNezet(java.swing.JFrame parent, boolean modal) {  
    super(parent, modal);  
  
    initComponents();  
    setTitle("Termék hozzáadása");  
    setLocationRelativeTo(parent);  
}
```

Konstruktorában megadjuk úgy nevezett parent komponens amin megjelenik a Dialog ablak ez az én projektben mindig az az adott kliens felület lesz. Illetve minden Dialognál megadjuk a címét setTitle() metódussal és pozícionálását setLocationRelativeTo(parent) metódusával paramétere pedig konstruktor paramétere lesz. Hibakezelést végez ezekkel kapcsolatban és hibásnak vélt adatokról JOptionPane ablakban láthatóvá teszi felhasználónak az adott hibá(ka)t és csak ezek javítása után halad tovább a program. Ezen funkciókhoz a Validálás osztály logikai típusú függvényeit hívom miután példányosítottam az osztályt ezeket pedig még egyszer megvizsgálom, hogy igaz, vagy hamis logikai értékkel térnek-e vissza. Ez határozza meg a felhasználóhoz kiadott üzenetet. Illetve felvitel műveletet is befolyásolhatja, ugyanis ha van már olyan nevű termék amit felkívánna vinni a felhasználó, ezt megakadályozza pusztán mennyiséget veszi át felületről és az adatbázisban már ezen létező termék többi adatával kiegészítve felviszi azt Beszerzés táblába és Termékek táblában pedig csak adott árucikk mennyisége növekszik a beszerzés alapján.

```
private boolean teljesEllenorzes() throws SQLException{  
    String hiba = "";  
    if (vizsgálat.Üres(nezet.getTfNév().getText())) {  
        hiba += "A névnek minimum 1 maximum 30 karakter hosszúnak  
        kelllennie!\n";  
    }  
    if(vizsgálat.egyezesVizsg(nezet.getTfNév().getText(), dao.select()))
```

```

if (vizsgalat.szamE(nezet.getTfMennyiség().getText())) {
    hiba += "Mennyiség nem szám vagy nem nagyobb 0";
}
if (vizsgalat.szamE(nezet.getTfMennyiség().getText())) {
    hiba += "Egységár nem szám vagy nem nagyobb 0";
}

if (!vizsgalat.ellenorizHossz(nezet.getTaLeírás().getText(), 30)) {
    hiba += "Leírás mező nem elég hosszú";
}
if (hiba.isEmpty()) {
    return true;
}
else {
    JOptionPane.showMessageDialog(nezet, hiba, "Hiba!",
        JOptionPane.ERROR_MESSAGE);
    return false;
}

```

Ez a teljes funkcionalitás [nézet](#) példányosításával kezdődik Controllerben itt teszem egyáltalán láthatóvá a felületet konstruktorban, fentebb említett funkciók példány gombot visszaadó getter metódusának visszatéréséhez fűzött eseménykezelő metódus belsejében valósul meg. Ha nincs hiba adatokkal, akkor tovább adjuk azokat adatbázis kapcsolatot implementáló, illetve adatbázis tartalmát befolyásoló műveleteket végző osztály példányának amit TermekDAO, azaz adat elérési objektum angol rövidítése néven deklaráltam, ennek konstruktorának. Az így inicializált TermekModell osztálypéldányt már képes átvenni TermekDAO adatbázisba való beszúrásért felelős metódusa.

```

@Override
public void insert(TermekModel obj) throws SQLException {

    ArrayList<KategoriaModell> kategLista=null;
    kategLista=kategoria.vegrehajtVisszaadSQL("Select * from
    Kategória where kategóriánév='"+obj.getKategoria()+"'");
    id=kategLista.get(0).getKategoriaID();
    String sql="InsertintoTermék(terméknév,kategóriaID,
    mennyiség,egységár,leírás)
    values('"+obj.getTermékNév()+"', '"+id+"', '"+
    obj.getMennyiség()+"', '"+obj.getEgységár()+"
    '"+obj.getLeírás()+"' ) ";

    vegrehajtSQL(sql);
}

```

Itt átvesszük Controllerben inicializált példányt paraméterként. Jelen esetben szükséges példányosítanunk kategóriához tartozó osztályt KategoriaModell-t, illetve az Ehhez tartozó Dao-t mivel adatbázisban Termék táblában kategóriát nem név szerint, hanem azonosítóval tároljuk és név szerint pedig ezen azonosítóval megegyező Kategória szótár tábla elsődleges kulcsára mutatunk ezen külső kulccsal és ,ha megegyezik kinyerjük a későbbiekben egy lekérdezéssel a felvitt termékek megjelenítése felület számára név szerint, nem pedig a termék táblában lévő kategória azonosítót. Visszatérve beszúráshoz a paraméter objektum getter metódusinak visszatérési értékeit

beleágyazzuk a `String` ként tárolt `sql` néven deklarált változóba ami a végrehajtandó scriptünket tartalmazza. Ezt pedig paraméterül adjuk `vegrehajtSQL`-nek ami `Beszúrás`, `törlés` és `frissítés` adatbázis műveletek végrehajtására van optimalizálva, lekérdezés esetében teljesen alkalmatlan a feladatra mivel az itt használt `PreparedStatement` interface képesség birtokában lévő objektumon hívott függvény logikai értékkel tér csak vissza, ezért csak azt tudhatjuk meg végrehajtás után, hogy ez sikeres, vagy sikertelen volt-e.

```
@Override
public void vegrehajtSQL(String sql) throws SQLException{

    Connection con = AdatbázisKapcsolat.getConnection();
    PreparedStatement st = con.prepareStatement(sql);
    st.execute();
    con.close();
}
```

végrehajtás végeztével természetesen zárjuk a kapcsolatot adatbázissal. Ha sikertelen volt hibát kapunk, ha nem felkerültek felhasználói felülettől kontroller által kezelt és Dao-nak átadott adatok felkerülnek az adatbázisba, amit felhasználó csak a következő termék készlet felületen tekinthet meg a kliense menüjéből kiválasztva azt. Illetve még ezen a felületen importálás gombhoz az előzőek alapján `Controller`ben csatolt eseménykezelés valósít meg, ennek hívása a `Controller` konstruktorában történik ugyan csak. Metódus belsejében példányosítjuk a megvalósításhoz az alábbiakat.

### 5.3 Fájl beolvasás

#### FajlValaszto.java

A különböző adat feltöltési felületeken elhelyezett nyomógombokhoz hozzá van rendelve egy eseménykezelés metódus amelynek tartalma az esemény bekövetkeztekor, azaz a gomb lenyomása esetén hajtódik csak végre. Az esemény elsőként egy fájlválasztó ablakot Javában név szerint `JFileChooser`-t hív amellyel a felhasználó grafikusán kiválaszthat egy a számára kívánt adatokat tartalmazó fájlt. Ezt a műveletet a `FajlKezeles` csomag `FajlValaszto` osztálya valósítja meg.

```
public class FajlValaszto {

    public String ablak() {
        String utvonal = "";

        UIManager.put("FileChooser.openDialogTitleText",
            "Forrásfájlmegnyitása");
        UIManager.put("FileChooser.lookInLabelText",
            "Aktuális mappa:");
        UIManager.put("FileChooser.openButtonText", "Megnyitás");
        UIManager.put("FileChooser.cancelButtonText", "Mégse");
    }
}
```

```

UIManager.put("FileChooser.fileNameLabelText", "Fájl neve:");
UIManager.put("FileChooser.filesOfTypeLabelText", "Fájl típusa:");

JFileChooser fc = new JFileChooser();
fc.addChoosableFileFilter(new FileNameExtensionFilter
("Szövegesfájlok", "csv", "txt", "xlsx"));
fc.setAcceptAllFileFilterUsed(false);
if (fc.showOpenDialog(fc) == JFileChooser.APPROVE_OPTION) {
    File fajl = fc.getSelectedFile();

    if (fajl != null) {
        utvonal = fajl.getAbsolutePath();
    }
}
return utvonal;
}

```

A kiválasztható fájlok formátumait megsűrjűk, csak txt, csv, xlsx formátumok láthatóak felhasználó számára ezért ezzel hibát nem követhet el, következésképpen hibaüzenet sem érkezik. Kiválasztás által megkapjuk az adott fájl útvonalát amit egy szöveges String változóban tárolunk el és adjuk tovább ExcelBeOlvasFajl metódusainak további kezelésre.

```

public class ExcelBeOlvasKiir {
    private List<String> nyersFajl=null;

    public void beOlvasNyersFajl(String beFajl){

        try {
            nyersFajl=Files.readAllLines(Paths.get(beFajl));
        } catch (IOException io) {
            JOptionPane.showMessageDialog(nezet,
                "Nem található fájl");
        }
    }

    public ArrayList<TermékModel> beOlvasTermek(String beFajl){
        TermékModel modell;
        beOlvasNyersFajl(beFajl);
        ArrayList<TermékModel> excelLista=new ArrayList<>();
        for (String string : nyersFajl) {
            String [] tört=string.split(";");
            String tNév=tört[0];
            String kateg=tört[1];
            int menny=Integer.parseInt(tört[2]);
            int egysÁr=Integer.parseInt(tört[3]);
            String leírás=tört[4];

            modell=new TermékModel(tNév, kateg, menny, egysÁr,
                leírás);
            excelLista.add(modell);
            System.out.println(modell);
        }
        return excelLista;
    }
}

```

Kiválasztott fájlban amennyiben ékezetes karaktereket is tartalmaz, mindenképpen UTF-8 kódolású kell hogy legyen, különben `Malformed input` kivétel hibaüzenet jelenik meg és a program ezen része nem fut tovább. Termékek, vevők, illetve beszállítók esetén is külön metódusok vannak létrehozva beolvasáshoz és beolvasott adatok továbbadásához az eltérő modellek miatt. Ezen modellek konstruktorán keresztül történő adatátadást leszámítva, minden művelet azonos. Többek között mindenki hívja `beOlvasNyersFajl` metódust ami egy paraméterként várja kiválasztott fájl relatív elérési útvonalát `String` típusú függvény visszatérési értékéből származik. A metódus megváltoztatja `nyersFajl` privát láthatóságú globális `String` típusú lista deklaráláskor `null`-ra inicializált értékét a beolvasott fájl sorainak tartalmára. Minden esetben pontos vessző alapján tagoljuk sor tartalmát és az így kapott adattagokat adjuk át szöveg, illetve egész szám típusú változóknak amelyeket adott modellnek konstruktorán kap meg paraméterként. Az adatokkal feltöltött példányt egy ilyen példány típusú listához fűzi, ami `for` ciklusban többször végrehajtódik, ezek számát `nyersFajl` mérete határozza meg. Ezután ciklus végén eljárás függvény végén visszatér az előbb feltöltött listával amit az őt hívó programnak át ad és véget ér metódus végrehajtása.

## 5.4 Termék készlet megtekintése

Hasonlóan az előző szerkezethez itt is példányosítjuk a [nézet](#), modell és adatbázis elérési objektum osztályait kezdetben `null` értékkel inicializálva, később konstruktorban új adott objektum típus alapértelmezett konstruktorával. `KezelController` nézete két féle lehet amely jogosultságtól függ azaz bizonyos funkciók nem elérhetőek rajta amit viszont `TermékKezelController` aki örököl ebből az osztályból tovább terjeszti a funkcionalitást. Azonban mindkettőben megtalálhatóak `TermekUrlap` adattagjai és metódusai amit az alapvető [TermekNezetDialog](#) örököl. Dolgozói jogosultságon kliens felület nézet menüjéhez beépítetten hozzáágyazott eseménykezelésre ezt a [TermekNezetDialog](#) nézetet névszerint `TermekNezetController` osztály példányosítja. `Termék` típusú `DAO` `adatokTablaTolt` metódusa végrehajt egy lekérdezést adatbázisból.

```
@Override
public ArrayList<TermékModel> select() throws SQLException {
    ArrayList<TermékModel> lekérdeLista=null;
    String sql="Select termékID,terméknév,kategórianév,mennyiség,
                egységár,leírás,[törölt-e]"
                + " from Termék Inner Join Kategória ON
```

```

        Termék.kategóriaID=Kategória.kategóriaID";
        lekérdLista=vegrehajtVisszaadSQL(sql);
        return lekérdLista;
    }

```

Ezt az SQL utasítást paraméterként átadjuk vegrehajtVisszaadSQL függvénynek.

**@Override**

```

public ArrayList<TermékModel>vegrehajtVisszaadSQL(Stringsql) throws
SQLException{

    Connection con=null;
    Statement st=null;
    ResultSet rs=null;
    ArrayList<TermékModel> lista=new ArrayList<>();

    con=AdatbazisKapcsolat.getKapcsolat();
    st=con.createStatement();

    rs=st.executeQuery(sql);

    while(rs.next()){
        if(rs.getBytes(7)==0){
            lista.add(newTermékModel(rs.getInt(1),rs.getString(2),
            rs.getString(3),rs.getInt(4),rs.getInt(5),rs.getString(6),
            rs.getBytes(7)));
        }
    }
    rs.close();
    st.close();
    con.close();

    return lista;}

```

Ez a függvény TermékModell típusú láncolt lista típusú. Connection típusú példányt inicializáljuk AdatbazisKapcsolat osztály getConnection() függvényével és ezáltal kapcsolat létrejött. Statement típusú példányunk értéke con példány createStatement függvényének visszatérése. Ezek után st-re hívjuk executeQuery metódust String sql paraméterrel és átadjuk egy ResultSet típusú példánynak. Ezt követően egy while ciklussal bejárjuk ResultSetet amíg van következő eleme. Ciklus belső részében inicializálok egy TermékModellt, paraméterek számára ResultSet i-edik eleme és pontos típus megadása a visszatéréshez lásd rs.getInt(1), Az így inicializált példányt hozzáfűzzük egy ilyen osztály típusú láncoltlistához. Ez a művelet try-catch blokkban áll, ha nem sikerül megfelelő kivételt dob. Ha sikeres következő pontban lezárjuk az adatbázis kapcsolódáshoz és SQL utasítás végrehajtásához szükséges folyamatokat, ezt is hibakezeléssel. Végül pedig függvény visszatér az objektumokkal feltöltött láncoltlistával.

```

public Object[][] adatokTablaba(ArrayList<TermékModel> lekerdezes)
throws SQLException{

    ArrayList<TermékModel> lista=lekerdezes;

```

```

for
(Iterator<TermékModel>iterator=lista.iterator();iterator.hasNext();)
{
    TermékModel next = iterator.next();
    if(next.getTörölt()!=0){
        iterator.remove();
    }
}
Object[][] sor = new Object[lista.size()][6];

for (int i = 0; i < lista.size(); i++) {
    sor[i][0] = lista.get(i).getTermékID();
    sor[i][1] = lista.get(i).getTermékNév();
    sor[i][2] = lista.get(i).getKategória();
    sor[i][3] = lista.get(i).getMennyiség();
    sor[i][4] = lista.get(i).getEgységár();
    sor[i][5] = lista.get(i).getLeírás();
}

return sor;}

```

DAO adatokTablaba object mátrix típusú függvény megkapja előző függvény visszatérését. Ezt átadjuk egy helyileg létrehozott de ugyanolyan láncoltlistának. Egy iterátorral bejárjuk ezt a listát és kivesszük belőle azokat az elemeket amelyek megfelelnek az elágazás feltételének, azaz objektum törölt nevű int típusú adattagját visszaadó getTörölt függvény visszatérési értéke nem nulla, vagyis logikailag törölt ezért nem jelenhet meg az adatokat megjelenítő táblázatban. Az így megszürt lista elemeinek getterei segítségével átadjuk az adatokat a mátrixnak, az egész ciklikusan láncolt lista végéig tart. Végül pedig visszatérünk sor nevű mátrix-szal. Controllerben meghívjuk nézet táblaszerkezetet visszaadó getter metódusát.

## 5.5 Keresés eredmények között

```

nezet.getBtnkeresés().addActionListener(e -> {
    try {

        if (!(nezet.getTfKeresMezo'().getText().isEmpty())) {
            System.out.println(dao.Szűrés((String)nezet.getJcKeres().
            getSelectedItem(), nezet.getTfKeresMezo'().getText()));
            tabla.tablaFeltolt(dao.adatokTablaba(dao.Szűrés
            ((String) nezet.getJcKeres().getSelectedItem(),
            nezet.getTfKeresMezo'().getText()),
            nezet.getJtTermekek());
        } else {
            tabla.tablaFeltolt(dao.adatokTablaba(dao.select()),
            nezet.getJtTermekek());
        }
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(nezet, ex);
    }
});

```

getter metódus segítségével elkérem [nézet](#)-től a Szűrés feliratú gombot, eseményt kezelő metódust rendelek hozzá. Ennek belsejében már fent említett `tablaMuveletek` osztály példányával hívjuk `tablaFeltolt()` metódusát ez előzőhöz képest más paraméterekkel. Ez az egész művelet egy elágazásban áll, amennyiben Szűrési mező üres és gomb lenyomott az alapvető paramétereket kap amelyek fentebb is látható, azaz töltse be minden terméket ami nem logikailag törölt. Különben pedig `TermékDao` osztály `Szűrés` nevű függvényének visszatérési értékével tölti fel a táblát, mely ugyebár a második paraméter.

`@Override`

```
public ArrayList<TermékModel> Szures(String oszlop,String
ertekek) throws SQLException {
    String sql = "";
    Pattern minta = Pattern.compile("[<=>]");
    Matcher egyezes = minta.matcher(ertekek);
    if (egyezes.find()) {
        sql = "Select * from Termék where " + oszlop + " " + ertekek +
        "";
    } else {
        sql="SelecttermékID,terméknév,kategórianév,mennyiség,egységár,
        leírás,[törölt-e]
        +\"fromTermék,KategóriawhereTermék.kategóriaID=Kategória.
        kategóriaID and \" + oszlop + \" Like '%" + ertekek + "%'";
    }
    ArrayList<TermékModel> lekérdeLista = null;

    lekérdeLista = vegrehajtVisszaadSQL(sql);
    return lekérdeLista;
}
```

Először is `Pattern` osztály `minta` nevű példányának értékül adjuk `Pattern` osztály `minta` nevű függvényének visszatérési értékét, lényegében rögzítjük benne a keresett mintákat `String`ként. Egészen pontosan reguláris kifejezések mintájának megfelelő formátumban. `Matcher` típusú `egyezes` nevezetű példány pedig az előbb felvitt minták és a mintára hívott `matcher` függvény, ami paraméterként átveszi `String` érték változó értékét ez a tartomány amiben végre kell hajtani a keresést. Elágazásban vizsgáljuk, hogy `egyezes.find()` logikai típusú függvény igaz, vagy hamis értékkel tér e vissza. Ez esetben ha igaz, akkor felhasználó által bevitt érték logikai operátort tartalmaz tehát `sql` lekérdezésben számhoz hasznát összehasonlítást át vesszük `nézet` szűrési mezőjétől. Másik lehetőség hogy szöveges keresésre van szükség ilyenkor `oszlop` paraméter ami azt jelenti ami alapján keressük nem egyenlő értékkel, hanem `" Like '%" SQL utasítás segítségével hasonló, adott értéket részletesen is tartalmazó eredményeket ad vissza. Eredményeinket szűrést követő vagy anélkül egyszerű egér kattintással kiválaszthatjuk, természetesen egyszerre csak egyet.`



```

public void mezokFeltolt() throws SQLException {

    int sorszam = nezet.getJtTermek().getSelectedRow();
    TermékModel táblaElem = lekerdezes.get(sorszam);

    nezet.getIdLbl().setText(nezet.getJtTermek().getValueAt(sorszam,
0).toString());
    nezet.getTfNév().setText(nezet.getJtTermek().getValueAt(sorszam,
1).toString());
    nezet.getScrKategória().setSelectedItem(táblaElem.getKategória());
    nezet.getTfMennyiség().setText(nezet.getJtTermek().getValueAt(
sorszam, 2).toString());
    nezet.getTfEgységÁr().setText(nezet.getJtTermek().
getValueAt(sorszam, 3).toString());
    nezet.getTaLeírás().setText(nezet.getJtTermek().getValueAt(
sorszam,4).toString());

}

```

A választott termék adatai amelyeket felhasználónak szabad módosítani automatikusan betöltődnek a [TermekHozzaad](#)ban használt [TermekUrlap](#) osztálytól megörökölt űrlap elemeibe.

## 5.6 Szállítólevél

Az előzőekhez hasonlóan [szállítólevél generálás felületen](#) is szűrhetünk és választhatunk táblák, nevezetesen egy vevők és egy termékek tábla tartalmai között ugyanazokkal a kódrészetekkel megvalósítva.

```

public void esemenyKezeles() {
    nezet.getBtnHozzaAdas().addActionListener(e -> {

        try {
            táblaElem();
            termékSegedtermekDao.Lekerdezes().get(nezet.getJTable2().
getSelectedRow());
            int mennyiség =Integer.parseInt(nezet.getTfMennyiség().getText());
            rendTétel = new RendelesModell(rendelesszam,
            termékSeged.getTermékID(),
            vevoDao.Lekerdezes().get(nezet.getTblVő'().getSelectedRow()).
getVevoID(), mennyiség, mennyiség * termékSeged.getEgységár());
            rendeles.Beszuras(rendTétel);
            termékSeged.setMennyiség(termékSeged.getMennyiség() - mennyiség);
            termék = termékSeged;
            termékDao.Frissites(termék);

            tábla.tablaFeltolt(termékDao.adatokTablaba(termékDao.Lekerdezes()),
            nezet.getJTable2());
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(nezet,ex,"hiba",
            JOptionPane.ERROR_MESSAGE);
        }
    });
}

```

Választás után hozzáadás gomb eseménykezelése mennyiség int típusú változóban rögzítjük mennyiség szövegmező String típusú tartalmát amit Integer csomagoló osztály Integer.parseInt() metódusának paraméterként átadva kapunk meg egész számértékként. rendTetel = new RendelesModell() rendTetel RendelesModell példányának új példányt adunk értékül, annak konstruktorának átadott értékekkel pedig módosítjuk eredeti null értékét. Ezen paraméterek rendelesszam amely értéke rendelesLista.get(rendelesLista.size()-1).getRendelesID()+1; amely Rendelés adatbázis tábla minden nem törölt rekordját tartalmazó láncolt lista utolsó rendelés objektum elemének rendelési azonosítót vagyis egy egész szám típusú értéket visszaadó metódus visszatérési értéke+1.Ezzel biztosítunk a következő rekord felvitele előtt számára egy generált kulcsot amit a programtól kap, nem pedig adatbázisból autoincrement által. Következő paramétere vevő adattáblából felhasználó által felületen elérhető táblázat kiválasztott elemének számával megegyező indexű láncoltlista elemből kinyert vevőt azonosítószám. Ezt követően az előbb feltüntetett mennyiség változó, illetve utoljára mennyiség szorozva segéd lekérdezésből ami termék típusú láncolt lista kinyert egységárral. Ezt követően ezt az objektumot paraméterül adjuk RendelésDao példányának Beszúrás nevű metódusának paramétereként. Ezen metódus a paraméterből visszanyeri az előzőekben Rendelés objektumba felvitt adatokat és ezeket felviszi adatbázis Rendelés táblájába új rekordként. Ezután termékSeged.setMennyiség() metódusával megváltoztatjuk ezen példány mennyiség adattagját, lényegében levonjuk a rendelt mennyiséget belőle. Ezt követően egy új Termék példánynak értékül adjuk azt, innentől pedig ezt az új példányt paraméterként adjuk termékDao objektum frissítés metódusának termékDao.Frissites(termek); ezzel Termék adatbázis táblában frissítjük termék készletet. Végül pedig frissítjük felületen található termék adatokat tartalmazó táblázat tartalmát

tabla.tablaFeltolt(termekDao.adatokTablaba(termekDao.Lekerdezes()),  
 nezet.getjTable2()); ,ezzel felhasználó számára is láthatóvá téve a változást.PDF generálás gombhoz rendelt eseménykezelésben pedig meghívjuk PDF metódust.

```
public void PDF() throws SQLException, IOException, DocumentException{

    int termékDarab = nezet.getjTable2().getSelectedRowCount();
    Document doc = new Document();

    PDFWriter kiIr =
    PDFWriter.getInstance(doc,newFileOutputStream("Szállítólevél.PDF"));
    doc.open();
```

```
Paragraph cim = new Paragraph("Szállítólevél", new
Font(Font.FontFamily.TIMES_ROMAN, 25f, 0));
cim.setAlignment(1);
doc.add(cim);
doc.add(new Chunk());
```

iText külső forrásból beimportált könyvtár metódusaival összeállítjuk a PDF dokumentum szerkezetét és ebbe felvisszük az előző részekben eltárolt rendelési adatokat. Elsőként `Document doc = new Document();` létrehozuk új dokumentumunkat. `PDFWriter kiIr = PDFWriter.getInstance(doc, new FileOutputStream("Szállítólevél.PDF"));` `PDFWriter` objektum `getInstance` metódusának paraméterül adjuk a dokumentumot és azt hogy hol és milyen néven jöjjön létre a PDF fájl. Jelen esetben az alapértelmezett könyvtárban fog létrejönni Szállítólevél néven. `doc.open();` segítségével úgymond megnyitjuk a dokumentumot vagyis indulhat a belső, avagy tartalmi szerkezet kiépítése. Kezdsnek címet állítunk be, illetve formáltuk azt, betűtípust adunk neki, középre pozicionáljuk. Mindez cím `Paragraph` típusú objektum tárolja majd ezt hozzáfűzzük dokumentumhoz, e nélkül az előző beállítások ellenére sem lenne látható a kész dokumentumban. Következőekben az alábbi kódrészlet segítségével

```
PDFPTable alanyokTablázat = new PDFPTable(2);
alanyokTablázat.setWidthPercentage(100);
alanyokTablázat.addCell(new Paragraph("Cégnév név: " +
elado.getVevonev() + "\n" + "Email-cím: " + elado.getEmail())
```

`PDFPTable` objektummal létrehozunk egy táblázatot 2 cellával melyek egyike céges adatokat, a másik cella pedig vevő adatait tartalmazza majd. Szállítólevelet kibocsájtó cég adatait egy szöveg, vagy excel fájlból olvassuk ki. `elado=excel.beOlvassVevo("cegesAdatok.txt").get(0);` eladó amit egy `vevőModell` típusú objektum eltárolja a beolvasott kezdetben `vevőModell` típusú láncolt listában tárolt objektumok 0.-ik elemét, hiszen csak ez az egy objektum és benne foglalt adatok lehetnek benne vagyis egy felhasználói cég adatai, azonban ennek adatai felhasználók által bármikor forrásfájlban módosíthatók. Ezen objektum adattagjait betöltjük a megfelelő cellába. Ugyanezen megismételjük tényleges vevő adattagjaival aminél a `vevőModell` objektum azonban a táblázatból kiválasztott sor adatai adják. Létrehozunk egy táblázat szerkezetet termékek adattagjainak számára is. Ennek tartalmát dinamikusán vesszük fel hiszen egy rendelésnél több terméket is rendelhet egy megrendelő.

```
for (int i = 0; i < termékLista.size(); i++) {
termekTablázat.addCell(termékLista.get(i).getTermékID()+"");
```

```

termeKTablázat.addCell(termeKLista.get(i).getTermékNév());
termeKTablázat.addCell(termeKLista.get(i).getKategória());
termeKTablázat.addCell(termeKLista.get(i).getLeírás());
termeKTablázat.addCell(termeKLista.get(i).getEgységár() + "");
termeKTablázat.addCell(nezet.getTfMennyiség().getText() + "");

int bruttoAr = Integer.parseInt(nezet.getTfMennyiség().getText())
* termeKLista.get(i).getEgységár();
bruttoAr = bruttoAr + (bruttoAr / 100) * 27;
teljesVegAR += bruttoAr;
termeKTablázat.addCell(bruttoAr + " FT");

}

```

Ezen dinamizmust egy számláló, vagy for ciklus biztosítja amely 0-tól `termeKLista.size()`; azaz eddig elmentett termékeket gyűjtő termék típusú láncolt lista méretéig növekszik. A megfelelő cellába felvisszük ezen lista *i*-edik termékének adatágjait egyesével a hozzájuk tartozó címsor alá. Az eddig eltárolt rendelési adatok mellett számolunk minden termékhez egy bruttó árat egységár és az adott termékből megrendelt mennyiség és 27%-os áfakulcs segítségével. `int bruttoAr = Integer.parseInt(nezet.getTfMennyiség().getText())` \* `termeKLista.get(i).getEgységár()`; .Illetve summázzuk ezek alapján az összes bruttó árat ezzel összegezve a végső kifizetendő bruttó összeget.

```

doc.addCreationDate();
DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
doc.add(new Paragraph("Kelt:" + dateFormat.format(new Date())));

```

Illetve alábbi kódrészlet segítségével a dokumentum végére fűzzük a kelezés dátumát. Ezt alapvetően `Date` osztály objektumától kapjuk meg ez azonban nem elegánsam megjeleníthető, ezért fentebbi kódrészlet első sorában megjelenő kóddal formázzuk azt. Ezzel pedig ha gombot amelyhez eseménykezelés hozzá van rendelve generálódik egy PDF fájl a megadott tartalmakkal.

## 5.7 Jelentések

```

if( nezet.getjCheckBox3().isSelected()) {
    lekerdezesGyujto.add( (ArrayList<Object>) (Object) raktarKeszlet());
    try {
        lekerdezes();
        generalPDF();
    } catch (FileNotFoundException ex) {
        JOptionPane.showMessageDialog(nezet, ex);
    } catch (DocumentException ex) {
        JOptionPane.showMessageDialog(nezet, ex);
    }
    catch (NullPointerException ex) {
        JOptionPane.showMessageDialog(nezet, ex);
    }
}

```

Eseménykezeléskor vizsgáljuk, hogy az adott gombot [felületen](#) kiválasztották-e, ha igen végrehajtjuk az adott lekérdezést, amely testreszabott minden kimutatáshoz visszkapott eredmények tekintetében egyébként megegyezik az előzőekben láthatott lekérdezési struktúrával például: Termék kezelése pont `TermekDao`. Illetve `generalPDF()` ; metódus strukturálisan megegyezik, csak adatok tekintetében tér el Szállítólevél generálás fentebb említett pont erre a célra használt metódusával.

## 6 Tesztelés

### Bejelentkezési felület tesztelése

Teszteset	Elvárt eredmény	Kapott eredmény
belépés megkísérlése adatbázisban nem szereplő felhasználónévvel és helyes jelszóval	Belépés sikertelen, hibaüzenet jelenik meg	Belépés sikertelen, hibaüzenet jelenik meg
belépés megkísérlése adatbázisban nem szereplő jelszóval és helyes felhasználónévvel	Belépés sikertelen, hibaüzenet jelenik meg	Belépés sikertelen, hibaüzenet jelenik meg
Belépés helyes felhasználónévvel és jelszóval, amik szerepelnek adatbázisban	Belépés sikeres üzenet jelenik meg, tovább dob megfelelő felületre	Belépés sikeres üzenet jelenik meg, tovább dob megfelelő felületre
Jogosultságkezelés, felületek közötti átjárhatóság meggátolása	Sikeres belépéskor menedzser felhasználót a hozzárendelt felület fogadja	Sikeres belépéskor eladó felhasználót a hozzárendelt felület fogadja
Jogosultságkezelés, felületek közötti átjárhatóság meggátolása	Sikeres belépéskor menedzser felhasználót a hozzárendelt felület fogadja	Sikeres belépéskor eladó felhasználót a hozzárendelt felület fogadja

### Űrlap adat vizsgálatok

Teszteset	Elvárt eredmény	Kapott eredmény
üres mező	hibát dob nem végez beszúrást adatbázisba	hibát dob nem végez beszúrást adatbázisba
nem elég hosszú adat	hibát dob nem végez beszúrást adatbázisba	hibát dob nem végez beszúrást adatbázisba
Szám ellenőrzés	hibát dob nem végez beszúrást adatbázisba	hibát dob nem végez beszúrást adatbázisba

egyezés vizsgálat	hibát dob nem végez beszúrást adatbázisba	nem végez beszúrást adatbázisba
Hozzáadás gomb lenyomása ha nincs semmilyen előző hiba	adatokat beszúrja adatbázisba	adatokat beszúrja adatbázisba

### Fájlválasztó

Teszteset	Elvárt eredmény	Kapott eredmény
gomb lenyomása fájl választó ablakot hoz elő	megjelenik fájlválasztó ablak	megjelenik fájlválasztó ablak
fájl szűrés	Csak előszűrt kiterjesztésű állományok választhatók	Csak előszűrt kiterjesztésű állományok választhatók
Ki nem választás esetén	nem dob hibaüzenetet	nem dob hibaüzenetet
kiválasztás esetén	megkapjuk fájl relatív elérési útját	megkapjuk fájl relatív elérési útját

### Fájl olvasás

Teszteset	Elvárt eredmény	Kapott eredmény
Kódolás	UTF-8 kódolástól eltérő fájlokat nem olvas	UTF-8 kódolástól eltérő fájlokat olvas, hibát dob
Több alany olvasása	képes kiolvasni több alany adatait fájlból	képes kiolvasni több alany adatait fájlból

### Termék/Partnerek Szűrése

Teszteset	Elvárt eredmény	Kapott eredmény
Szám mennyiségek keresése	PDF fájl generálódik	PDF fájl generálódott
Szám mennyiségek keresése logikai operátorokkal és összeggel kereső mezőben	Sikeres keresés és táblázatban kapott eredmények megjelenítése	Sikeres keresés és táblázatban kapott eredmények megjelenítése

Szám mennyiségek keresése logikai operátorok nélkül csak összeggel	Nem jelenik meg semmi táblázatban	Nem jelenik meg semmi táblázatban
Szöveg értékek keresése egyszerűen nevük beírásával szövegmezőbe	Sikeres keresés és táblázatban kapott eredmények megjelenítése	Sikeres keresés és táblázatban kapott eredmények megjelenítése
Módosítás előkészítése	Kiválasztott termék adatai megjelennek szövegmezőkben	Kiválasztott termék adatai megjelennek szövegmezőkben
Törlés	kiválasztás és törlés gomb lenyomására törlődik táblából a választott adat	kiválasztás és törlés gomb lenyomására törlődik táblából a választott adat
Módosítás	Kiválasztott termék adatait tartalmazó mezők átírását és gomb lenyomására táblában megváltoznak	Kiválasztott termék adatait tartalmazó mezők átírását és gomb lenyomására táblában megváltoznak

#### Szállítólevél generálás

Teszteset	Elvárt eredmény	Kapott eredmény
gomb lenyomása PDF-et generál	PDF fájl generálódik	PDF fájl generálódott
alany választás	Táblából kiválasztott egyed adatit megkapjuk	Táblából kiválasztott egyed adatit megkapjuk
PDF tartalmazás	PDF tartalmazza kiválasztott alanyok szükséges adatait	PDF tartalmazza kiválasztott alanyok szükséges adatait
PDF tartalmazás	PDF tartalmazza több termék szükséges adatait	PDF tartalmazza több termék szükséges adatait

#### Jelentések

Teszteset	Elvárt eredmény	Kapott eredmény
gomb lenyomása PDF-et generál	PDF fájl generálódik	PDF fájl generálódott



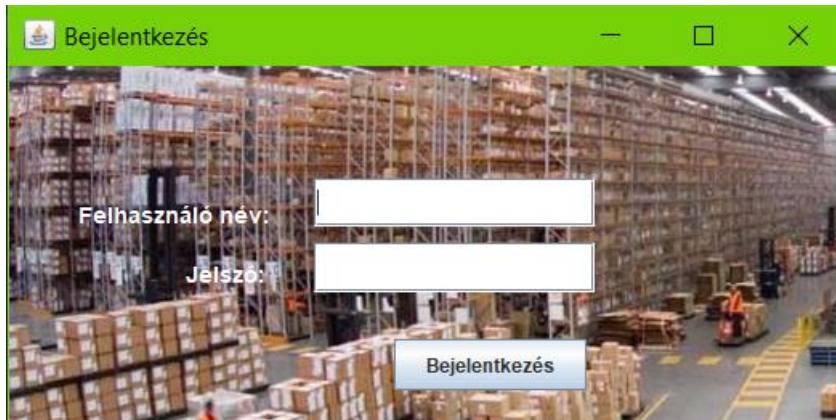
kiválasztott gombhoz tartozó jelentés	kiválasztott gombhoz tartozó jelentést tartalmazza PDF	kiválasztott gombhoz tartozó jelentést tartalmazza PDF
---------------------------------------	--	--

## 7 Felhasználói dokumentáció

### A program használata:

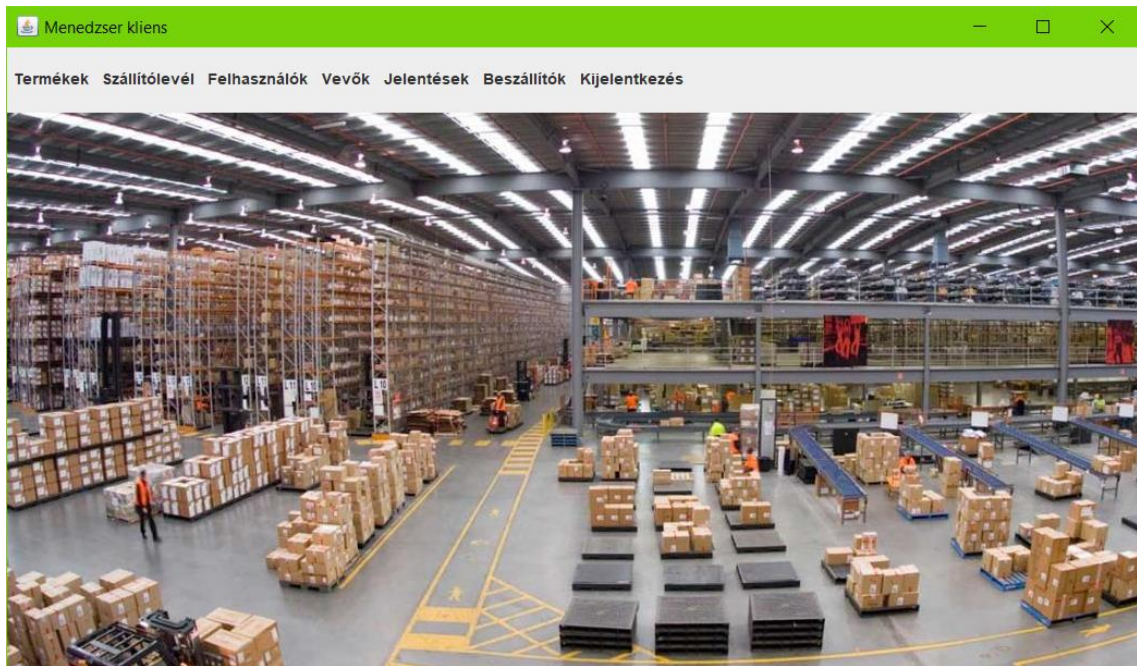
A felhasználók 2 különböző jogosultsági szinten ezek csökkenő sorrendben:( Raktár menedzser, Eladó) érhetnek el különböző feladat köreikhez szükséges funkciókat.

### 7.1 Bejelentkezés:



10. Bejelentkezési felület képernyőkép

A program indításakor a felhasználó elsőképpen a bejelentkező felülettel találkozik. Ehhez szüksége van egy adatbázisba felvett érvényes felhasználó névre és hozzá tartozó jelszóra. Mivel regisztrációra nincs lehetőség, hanem a legmagasabb jogosultságú felhasználók a Menedzserek rendelnek hozzá új felhasználókat (beleértve új menedzsereket is a kezdeti menedzser) különböző jogosultságokkal szabályozva az adott funkciókat. Ezen jogosultságok közé tartozik: menedzser, illetve az eladó/raktáros. Amennyiben rendelkezik ezen adatokkal és a lenyíló listából kiválasztja a 3 közül az ő fiókjához tartozó jogosultságot a rendszer tovább engedi az ehhez tartozó kliens felületre ahol a neki engedélyezett funkciókat képes használni.



11. Kliens felület képernyőkép

A felhasználók A kliens ablak felső részén elhelyezett menüből legördülő almenükben érik el a hozzájuk tartozó funkciókat amelyek:

## 7.2 Termék hozzáadása:

Menedzser jogosultságú felhasználó hozzáfér termékek menü első almenüjéhez melyben új termékeket vihet fel az adatbázisba űrlap segítségével megadva azok adatait: név leírás, mennyiség, egység, ár, illetve kategória választása. Beimportálás lehetőség gombnyomásra fájlválasztó ablakot nyit ahonnan grafikusán felhasználó kiválaszthatja a hozzáadni kívánt megadott kiterjesztésű fájlt. Ezen utóbbiak lehetőségek, de nem kötelező használatuk a felvitel sikeréhez.

### 7.3 Raktár készlet megtekintése:

ID:

Termék név:

Kategória: 

Item 1

Mennyiség:

Egységár:

Leírás:

Kép:

Termék neve:

Termék ID	Termék név	kategória	Mennyiség	Egységár	Leírás	Kategória

## 12. Kezelési felület képernyőkép

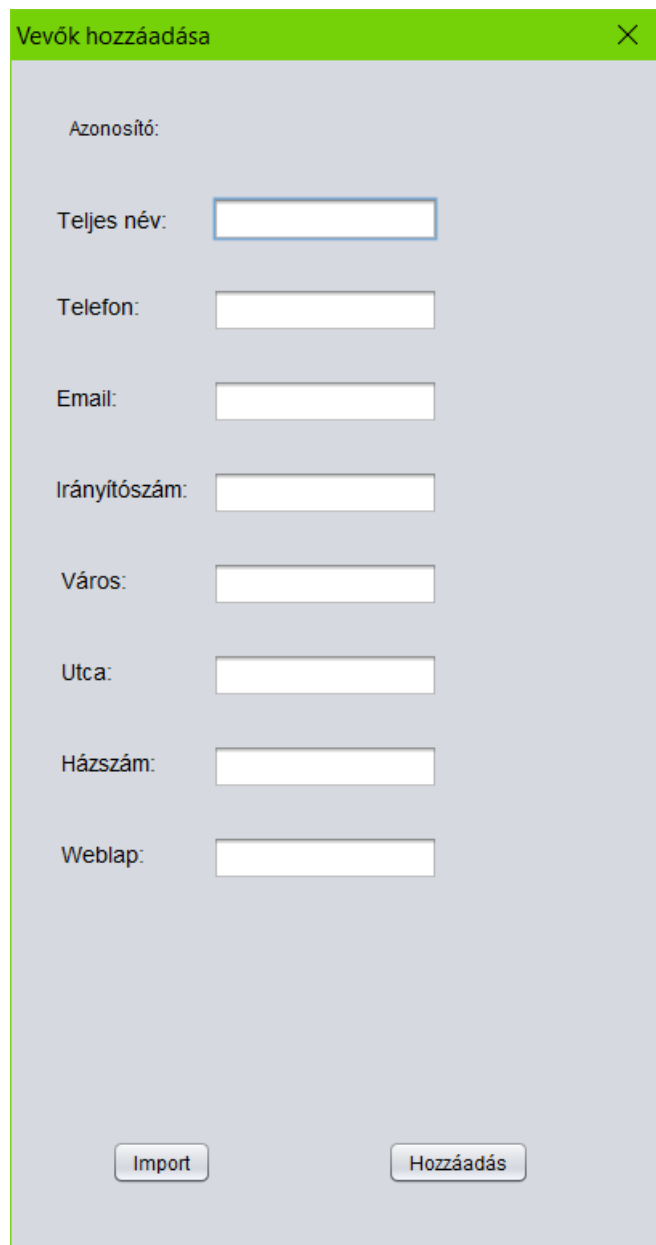
Minden felhasználó itt a már felvitt termékek előzően említett adatait, kiegészítve egy generált nem módosítható azonosítóval megtekintheti egy táblázatban. Szűrhet közöttük specifikusan név, kategória, mennyiség, egységár alapján amit csak be kell írni a kereső mezőbe és lenyomni a hozzárendelt gombot. Adatok módosítása az egérrel a táblázatból kiválasztott termék adatait űrlap mezőben átírva a hozzá rendelt gombbal végrehajtható. Ugyanezen kijelöléssel és véglegesítő gombbal termék törlése is lehetőség.

## 7.4 Szállítólevél készítés:

Kifelé irányuló forgalom esetére generálható szállítólevelet mind 2 kategóriába tartozó felhasználók elérhetik. Itt fel kell vinni a vevő számlázási, kiszállítási, elérhetőségi adatait, vagy ha visszatérő vásárló a már létező listából is kiválaszthatjuk egy gomb és egerünk segítségével egy táblázatból. Ugyan ezen módszerrel eladni kívánt termék(ek) választása és mennyiség megadása. Ezt követően egy gombbal véglegesítjük ennek hatására bekerül az adatbázisba, illetve generálódik egy PDF állomány, amely így nyomtatható.

Beszállítás esetén szinte azonos a felület, itt beszállító adatokra van szükségünk itt azonban csak kiválasztani lehet előre felvittek között, illetve ezt a funkciót az eladó nem érheti el.

## 7.5 Vevők kezelése:



Vevők hozzáadása

Azonosító:

Teljes név:

Telefon:

Email:

Irányítószám:

Város:

Utca:

Házzám:

Weblap:

Import Hozzáadás

### 13. Vevők feltöltése felület képernyőkép

Név szerint vevők menü két almenüje: hozzáad és nézet. Mindenki számára elérhető a szükséges adatok felvitelében nyilván eltér nevezetesen kell azonosító adatok, elérhetőségek és számlázási adatok, de csak úgy, mint termékek felvitelénél, illetve raktárkészlet menüpontoknál említett módszerekkel kivitelezhető a szükséges adatok felvitele/importálása és megtekintés ezen felületeken.

## 7.6 Beszállítók Kezelése:

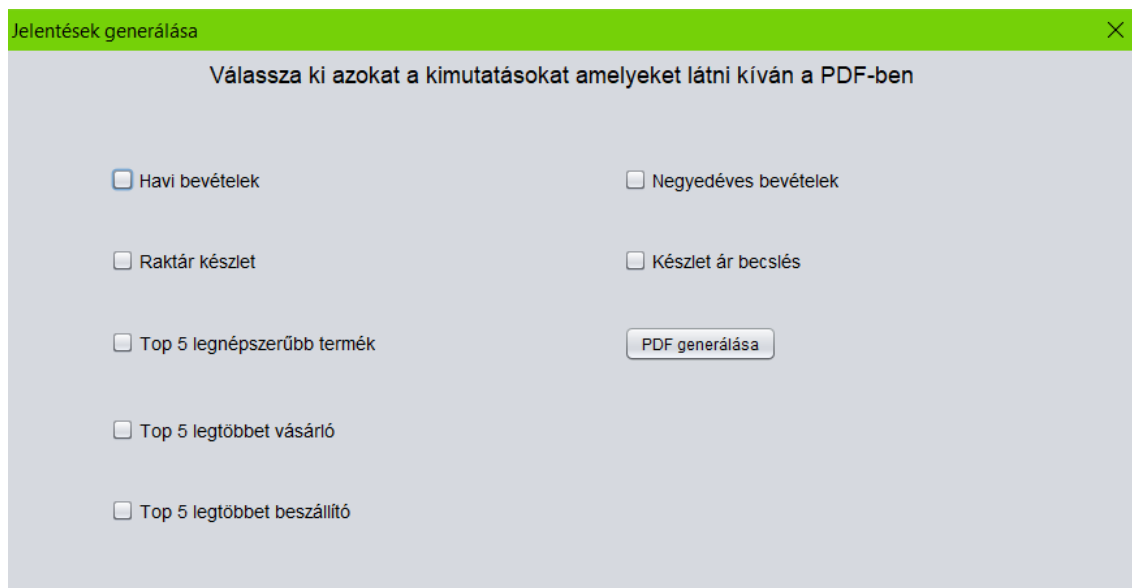
Hozzáadás és nézet Beszállítók menü két almenüje menedzser éri el mindkettőt eladó pedig csak nézetet. Hozzáadni kézzel, illetve importálás gombbal Excel fájlból vihető be, mint a másik kategóriájú partnereink a vevők és nézet is csak adatokban különbözik.

## 7.7 Felhasználók kezelése:

A kizárólag menedzser jogosultságú felhasználók számára használható a felhasználók adatainak felvitele személyes adatok, elérhetőségek, illetőleg felhasználó név és jelszó megadása űrlapon és jogosultság választása és adatbázishoz adása. Az itt felvitt adatok alapján az új felhasználó beléphet a rendszerbe.

Amennyiben valamilyen gond adódik, az illető jelszavával például az menedzser módosíthatja számára, beleértve más adatokkal együtt, illetve jogosultságát is változtathatja magasabb, vagy alacsonyabbra állítva azt. Ezen felül felhasználók végleges törlése ezzel kizárása a rendszerből is lehetséges. Ezen ablakok kinézete a már a többi felviteli, illetve nézet szerkezetével szinte megegyező.

## 7.8 Jelentések:



14. Jelentések felület képernyőkép

Jelentések felületen egyszerűen választó gombok segítségével kiválasztható egyszerre több hasznos kimutatás is. Ezek kiválasztása után pedig egyszerűen PDF generálása gomb lenyomásával alapértelmezett könyvtárba ahol a program is megtalálható generálódik egy PDF fájl ami tartalmazni fogja a kívánt kimutatásokat.

## **7.9 Kijelentkezés:**

Végül pedig természetesen kijelentkezés amire csak rá kell nyomni kilépünk a kliens ablakból, megjelenik a bejelentkező ablak, ha esetleg gyorsan vissza kell lépni. Ennek bezárásával pedig véget ér a program futása.

## **7.10 Programot készítette:**

Nagy Dávid

elérhetőségek:

email: [ndavid1226@gmail.com](mailto:ndavid1226@gmail.com)

Telefon:06-20-545-6681

## 8 Továbbfejlesztési lehetőségek

A szoftver véleményem szerint jelen verzióban képes eleget tenni egy raktárkezelő szoftver általános kívánalmainak, ezzel egy raktár teljes menedzseléséhez komoly háttér segítséget képes nyújtani az alkalmazottak számára. Azonban természetesen minden programot a végtelenségig lehetne tovább csiszolni, fejleszteni adott esetekre specifikálni. Véleményem szerint komolyabb hozzáadott értéke a következő felsorolt fejlesztések megvalósításának lenne a következő verziókban.

- **vonalkódok kezelése**

Vonalkódolvasását lehetővé tevő mobil applikáció megvalósítása amely hálózaton képes kommunikálni a készletnyilvántartó asztali alkalmazással. Vonalkód generálása kiválasztott termékek adataiból egy PDF fájlba, amelyből a PDF beépített lehetőségeivel kinyomtatjuk és később telefon beépített vonalkód Scannerével olvassuk azt. Így visszanyerhetjük az adatokat amelyet általunk írt applikáció továbbít az asztali alkalmazás termékek készlet felületéhez ahol automatikusan kiválasztódnak azon termék adatai amelyek a vonalkódban is szerepeltek. Így megvalósítva egy jóval gyorsabb és egyszerűbb, valamint pontosabb keresést a felhasználók számára. Valamint telefonos alkalmazásként költséghatékonyabb is mintha egy külön erre a célra kifejlesztett hagyományos vonalkód olvasó pisztolyt kellene bevásárolni minden munkaállomás számára.

- **Naplózás**

Egy kiterjedt naplózás megvalósítása a szoftverben amely nyilvántartaná minden felhasználó jogosultságtól független tevékenységeit bejelentkezett állapotban lévő idejük alatt. Ezen naplózások eredményét csak magasabb jogkörű menedzserek számára lehetne elérhető azaz dolgozók tevékenységeire lenne rálátásuk, illetve többi menedzserére is, így akár ellenőrizve is, segítve egymás munkáját. Igények és javaslatok függvényében elképzelhető lenne egy fő menedzser ez esetben egyedüli ként ő rendelkezne joggal ezen naplózások megtekintésére.



- **Jogkörök specifikusabb felosztása**

Dolgozók számára a menedzserek aprólékosabban oszthatnak ki hozzáféréseket adott funkciókhoz. egy választó gomb lenne minden felület minden lehetőségeivel amelyeket bepipálva növelheti, illetve ezeket kivéve csökkenteni lehetne egyesével az alárendelt felhasználók hozzáféréseit a funkciókhoz.

## 9 Összefoglalás

A záródolgozat írása, illetve a konkrét program fejlesztése közben rengeteg kihívással szembesültem melyeket hol önerőből, hol konzulensem segítségével sikerült kiküszöbölni és úgy hiszem komoly tapasztalatokra tettem szert ezen esetekből. Valamint sok új technológiával, módszertanokkal és szemléletmódokkal sikerült bővíteni programozói eszköztáram hála ennek a projektnek. Köztük adatbázis tervezési ismeretek, program, illetve adatbázis kapcsolata, együttműködésüknek sajátosságai elméleti és implementáció tekintetében. Az általam ezen projektben is alkalmazott MVC tervezési minta, mely átláthatóbbá és könnyebben karbantarthatóvá osztja a kódot. Ezen levont tapasztalatok kétségtelenül nagy segítséget jelentenek majd és fogom is alkalmazni őket a jövőbeli projektjeim fejlesztésénél.

## Irodalomjegyzék

[1] DAO:

<https://www.journaldev.com/16813/dao-design-pattern> 2019.03.31:14:48

[2] MVC:

[http://kaczursandor.hu/SZKI/2018-2019-1-2/java\\_alkalmazasok/MVC\\_mintak.ppt](http://kaczursandor.hu/SZKI/2018-2019-1-2/java_alkalmazasok/MVC_mintak.ppt) 2019.03.31:14:48

[3] MVC minta implementálása:

<https://www.youtube.com/watch?v=dTVVa2gfht8> 2019.03.31:14:49

[4] Adatbázis tervezés:

<https://etananyag.szamalk-szki.hu/course/view.php?id=131>

Rendelés adatbázis tervezéseTananyag 2019.03.31:14:50

[5] PDF könyvtár:

<http://www.java2s.com/Code/Jar/i/DownloaditextPDF540jar.htm> 2019.03.31:16:15

[6] Adatbázis kezelés Javában:

[http://kaczursandor.hu/SZKI/2018-2019-1-2/java\\_alkalmazasok/B-OracleHR.zip](http://kaczursandor.hu/SZKI/2018-2019-1-2/java_alkalmazasok/B-OracleHR.zip)  
2019.03.31:14:51

[7] Webáruház fázis2:

[https://drive.google.com/drive/folders/11M4uPcm5fLG0WDeV4uEHcYjqLpoNL\\_PP](https://drive.google.com/drive/folders/11M4uPcm5fLG0WDeV4uEHcYjqLpoNL_PP)  
2019.04.08:19:00

[8] háttérkép:

[https://www.google.com/search?rlz=1C1PRFI\\_enHU777HU777&biw=1707&bih=749&tbm=isch&sa=1&ei=i46rXJajMYbHwQKrnI\\_gBw&q=warehouse+nike&oq=warehouse+nike&gs\\_l=img.3..0i8i30i7.1945.2942...3676...0.0..0.111.367.3j1.....1....1..gws-wiz-img.....0i19j0i30i19j0i5i30i19j0i8i30i19.TzNogJ7SKl0#imgsrc=-MTiysqV2P4JZM](https://www.google.com/search?rlz=1C1PRFI_enHU777HU777&biw=1707&bih=749&tbm=isch&sa=1&ei=i46rXJajMYbHwQKrnI_gBw&q=warehouse+nike&oq=warehouse+nike&gs_l=img.3..0i8i30i7.1945.2942...3676...0.0..0.111.367.3j1.....1....1..gws-wiz-img.....0i19j0i30i19j0i5i30i19j0i8i30i19.TzNogJ7SKl0#imgsrc=-MTiysqV2P4JZM): 2019.04.08:20:10

## **A melléklet tartalma**

- A programkód .zip és .jar formátumban
- A dokumentáció .docx és .PDF formátumban

Az adatbázis forráskódja .sql formátumban

## **Köszönetnyilvánítás**

Szeretném megköszönni Kaczur Sándor tanár úrnak a rengeteg segítséget és hogy mindig elérhető volt a konzultációra. Kiváló mentorom volt, mindent felülmúló tudásával, jóindulatával és végtelen türelmével segített megküzdni a szakmai és időmenedzsmenti kihívásokkal.