

BGSzC Logisztikai és Kereskedelmi Technikum és Szakképző Iskola

1087 Budapest, Szörény utca 2-4.

# **Záró dolgozat**

## ***Bolti nyilvántartás***

Konzulens tanár:

Boros Sándor

Készítette:

Hummel Vendel, Hunka Róbert Emánuel

## Tartalom

1	Bevezetés .....	4
1.1	Feladat leírás .....	4
1.1.1	Fájlok előkészítése .....	4
1.1.2	Adatbázis és Szerver telepítése .....	4
1.1.3	Asztali felülethez való hozzáférés.....	5
1.1.4	Webes felülethez való hozzáférés .....	6
1.2	A felhasznált ismeretek .....	7
2	Felhasználói dokumentáció.....	8
2.1	A program általános specifikációja.....	8
2.2	Rendszerkövetelmények .....	8
2.2.1	Minimális Hardver követelmények.....	8
2.2.2	Szoftver követelmények .....	9
2.3	A program telepítése .....	11
2.4	A program használatának a részletes leírása .....	12
2.4.1	Weboldali funkciók: .....	12
2.4.2	Asztali funkciók: .....	16
2.4.3	Bevezető .....	16
2.4.4	Tétel hozzáadása.....	17
2.4.5	Fizetés és számla rögzítése .....	18
3	Fejlesztői dokumentáció .....	20
3.1	Az alkalmazott fejlesztői eszközök .....	20
3.1.1	Programozási és leíró nyelvek .....	20
3.1.2	• Fejlesztési környezetek .....	20
3.1.3	Adatbázis-kezelő rendszerek .....	20
3.1.4	Szerveroldali technológiák.....	20
3.1.5	Hardver és operációs rendszer .....	21

3.2	Adatmodell leírása.....	21
3.2.1	Bevezetés.....	22
3.2.2	Adattáblák leírása .....	22
3.2.3	Kapcsolatok az adattáblák között .....	24
3.3	Részletes feladat-specifikáció, algoritmusok.....	25
3.3.1	Bevezetés.....	25
3.3.2	Weboldal lényeges függvényeinek specifikációja.....	26
3.3.3	Összegzés.....	28
3.4	Tesztelési dokumentáció .....	28
3.4.1	Bevezetés.....	28
3.4.2	Tesztesetek .....	28
3.4.3	Összegzés.....	33
4	Összefoglalás.....	34
4.1	Önértékelés .....	34
4.1.1	Hummel Vendel önértékelése: .....	34
4.1.2	Hunka Róbert Emánuel önértékelése.....	35
4.2	Továbbfejlesztési lehetőségek.....	36
4.2.1	Továbbfejlesztői lehetőségek Hummel Vendel szerint.....	36
4.2.2	Továbbfejlesztői lehetőségek Hunka Róbert Emánuel szerint .....	37
5	Felhasznált irodalom .....	38
5.1	Felhasznált weboldalak: .....	38
6	Ábrajegyzék.....	39

## 1 Bevezetés

### 1.1 Feladat leírás

#### 1.1.1 Fájlok előkészítése

- Látogasson el a “<https://github.com/zenerra/vizsgaremek>” linken keresztül elérhető repositoryhoz
- A zöld ”Code” nevezetű gombot kattintva válassza ki a “Download ZIP” feliratot és töltsse le a fájlt a számítógépére majd bontsa ki.
- Győződjön meg róla, hogy a MySQL telepítve van és fut a rendszerén (a XAMPP használata nem kötelező; bármilyen MySQL környezet megfelelő). Importálja a *db\_nyilvantartas.sql* fájlt a MySQL adatbázisába olyan eszköz segítségével, mint a MySQL Workbench vagy a MySQL parancssor, mint például

```
1 mysql -u root -p db_nyilvantartas < path/to/db_nyilvantartas.sql
```

#### 1.1.2 Adatbázis és Szerver telepítése

- Futtassa le az alábbi parancsokat egy parancsor ablakban.

```
1 mkdir vizsgaremek
```

```
2 cd vizsgaremek
```

```
3 mkdir backend
```

```
4 cd backend
```

- Másolja be az alábbi fájlokat a backend mappából.

```
| app.js  
| db.js  
| db_nyilvantartas.sql  
| package-lock.json  
| package.json  
└── routes  
    alkalmazott.js  
    beszallito.js
```

Emánuel

cim.js  
szamla.js  
termek.js  
tetel.js

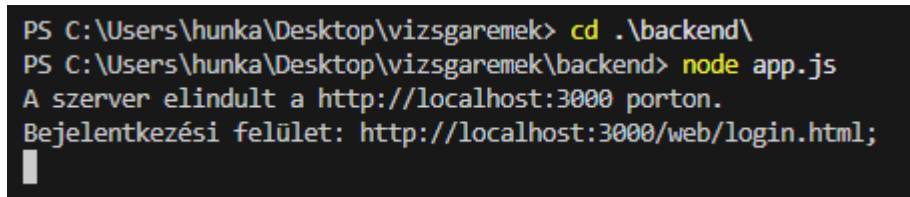
Hozzon létre egy .env fájlt. Az alábbi minta alapján, tegye a “backend” nevezetű mappába. Győződjön meg arról, hogy az adatbázis kacsolathoz megfelelő adatokat tartalmazzon.

```
1 DB_PASSWORD=  
2 DB_PORT=3306  
3 DB_NAME=db_nyilvantartas  
4 DB_USER=root  
5 DB_HOST=localhost
```

Futtassa le a következő parancsokat a backend mappa termináljában a szerver futtatásához.

```
1 npm install  
2 node app.js
```

Sikeres esetben elindult a szerver, ami az alábbi példában a “<http://localhost:3000>”-en található. A webes felületet pedig a “<http://localhost:3000/web/login.html>” URL-en érhet el.



```
PS C:\Users\hunka\Desktop\vizsgaremek> cd .\backend\  
PS C:\Users\hunka\Desktop\vizsgaremek\backend> node app.js  
A szerver elindult a http://localhost:3000 porton.  
Bejelentkezési felület: http://localhost:3000/web/login.html;  
█
```

ábra 1 Terminál tartalma sikeres telepítés esetén

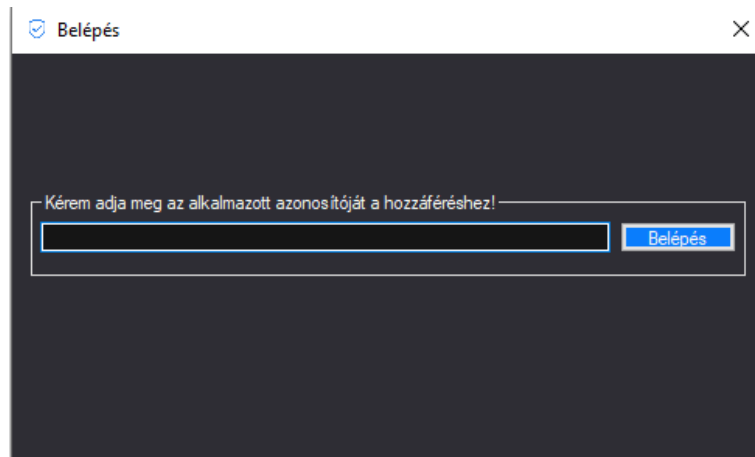
### 1.1.3 Asztali felülethez való hozzáférés

Töltse le a “desktopApp.zip” nevű fájlt a github repository “asztali” nevű mappájából. Eztán csomagolja ki a tartalmát egy tetszőleges helyre. Az asztali applikációt a “register.exe” nevű futtatható fájl futtatásával lehet elindítani.

Az alkalmazás futtatása után a belépési folyamat megfelel a következő (4.) pontban leírtaknak.

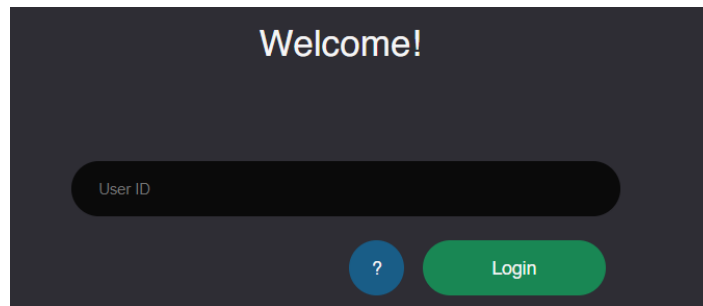
**Az applikációt, csak működő backend mellett indítsa el!!!**

Amennyiben elindította hamarabb, kövesse a 2.4.2-es pontban leírt útmutatót.



ábra 2 Asztali bejelentkező felület

#### 1.1.4 Webes felülethez való hozzáférés



ábra 3 Webes bejelentkező felület

Mindkét felület először a dolgozó azonosítóját kéri a funkciók eléréséhez, feltéve, hogy az adott dolgozónak van jogosultsága a felületre való belépéshez. Az azonosítót kizárólag az adatbázisból lehet lekérni. Az „1” azonosító mindkét felületre belépést biztosít.

## 1.2 A felhasznált ismeretek

Nagyban építkeztünk szakmai tanáraink tudására. Tóth Sándor oktató adta át az asztali alkalmazásokhoz szükséges tudását, ezzel megalapozva WinFormsos képességeinket Visual Studio-ban. Egy másik szakmai tanárunk, Boros Sándor, sokoldalúan támogatott minket a projekt során tudásával és tapasztalataival. Elsősorban a backend programozásban tanultunk tőle rengeteg újat. Probléma vagy kérdés esetén rá is támaszkodhattunk.

A projekt során API-t is használtunk, pontosabban a Chart.js JavaScript-alapú statisztikák megjelenítésére alkalmas könyvtárat. Ezt a hivatalos weboldalukon található dokumentációk alapján integráltuk.

Számos esetben kisebb-nagyobb hibakódokkal találkoztunk, amelyek megoldásához gyakran használtuk a Stack Overflow-t, egy nyilvános megoldáskereső oldalt programozók számára, valamint a YouTube videómegosztót.

Általános frontend tudásunkat az ágazati alapvizsga alapozta meg, így könnyedén felépítettük a weboldal külsejét.

### A felhasznált szoftverek

Az Office 365-öt használtunk, azon belül a Microsoft Word, Teams és PowerPoint programokat, valamint a Windows operációs rendszert a fejlesztés során.

Ezenkívül számtalan más szoftvert alkalmaztunk a projektben, erről a „3.1 Az alkalmazott fejlesztői eszközök” címsor alatt olvashat bővebb információt.

## 2 Felhasználói dokumentáció

### 2.1 A program általános specifikációja

E bolti nyilvántartó szoftver egy modern, integrált rendszer, amely hatékonyan segíti a készletek és tranzakciók kezelését bármelyik vegyeskereskedelmi kisboltnak. A szoftver két fő modulból áll: egy webes felületből és egy asztali applikációból. A webes felület lehetővé teszi a beérkező áruk és a raktárkészlet nyomon követését, a termékek adatainak szerkesztését, valamint részletes statisztikák és kimutatások készítését. A bolti alkalmazottak itt láthatják a termékek adatait, a készletmozgásokat és egyéb statisztikákat. Az asztali applikáció elsősorban a vásárlói tranzakciók feljegyzésére és nyugtázására szolgál.

Vásárlásokat és értékesítéseket lehet rögzíteni egy egyszerű, intuitív felületen, amely lehetővé teszi a nyugták automatikus formázását. Az alkalmazás offline módban is működik, a raktári mozgásokat pedig valós időben szinkronizálja a webes felülettel, amikor elérhető az internetkapcsolat. A rendszer többfelhasználós működést támogat, különböző jogosultsági szintekkel, és biztosítja a tranzakciók, valamint az adatmódosítások naplózását. A szoftver rugalmasan testreszabható, felhasználóbarát kialakítása pedig megkönnyíti a mindennapi munkavégzést a boltvezetők és az alkalmazottak számára.

### 2.2 Rendszerkövetelmények

#### 2.2.1 Minimális Hardver követelmények

A szerver futtatja a Node.js alapú backend alkalmazást (app.js), kezeli a HTTP kéréseket, és kapcsolódik a MySQL adatbázishoz. Az alábbi minimális konfiguráció biztosítja, hogy a rendszer alapvető funkcionalitása (tranzakciók lekérdezése, frissítése, az asztali alkalmazás, statikus fájlok kiszolgálása) működjön.

- **Processzor:** Kétféle CPU 2,0 GHz órajellel (például Intel Core i3 vagy azzal egyenértékű AMD processzor).



Emánuel

- **RAM:** 4 GB (a Node.js, a MySQL és az alapvető kéréskezelés futtatásához).
- **Tárhely:** 10 GB szabad lemezterület (a Node.js, az asztali app, a MySQL és az adatbázis tárolásához).

#### 2.2.1.1 További megjegyzések

**Adatbázis méret:** A MySQL adatbázis mérete a tranzakciók (szamla tábla) és tételek (tetel tábla) számával arányosan nő. A szerveroldali tárhelyet ennek megfelelően kell méretezni (például 1 GB 100 000 tranzakcióra, az adatok méretétől függően).

**Tárhelyhiány esetén:** Ha nincs több hely az adatbázis tárolására, a következő lépéseket teheti:

- **Régi adatok archiválása:** Az inaktív vagy kevésbé fontos tranzakciókat exportálja egy külső fájlba (például SQL dump vagy CSV formátumban), majd törölje azokat az adatbázisból.
- **Tárhely bővítése:** Telepítsen nagyobb kapacitású merevlemez vagy SSD-t a szerver tárhelyének növeléséhez.
- **Adatbázis optimalizálása:** Tisztítsa meg az adatbázist az elavult vagy felesleges rekordoktól, és használjon indexeket a tárolási igények csökkentésére. A `"DELETE FROM szamla WHERE skiallitas < DATE_SUB(CURDATE(), INTERVAL 15 YEAR);"` utasítással törölheti a 15 évnél régebbi tranzakciókat az adatbázisból.

## 2.2.2 Szoftver követelmények

### 2.2.2.1 Operációs rendszerek:

- A program szerveroldali komponensei a következő operációs rendszereken futnak:
  - Windows 10 / 11 (64 bites)
  - Ubuntu 18.04 LTS (vagy újabb, **asztali program nem futtatható**)
  - macOS 11 (vagy újabb, **asztali program nem futtatható**)

Emánuel

- A kliensoldali komponensek (webfelület) bármely, modern webböngészőt támogató operációs rendszeren elérhetők:
  - Windows 10 / 11 (64 bites)
  - macOS 10.15 (vagy újabb)
  - Ubuntu 18.04 LTS (vagy újabb)
  - Android 9.0 (vagy újabb)
  - iOS 14 vagy újabb (mobil eszközökhöz)

Ha mobil eszközén kívánja megtekinteni, csatlakoztassa a telefont ugyanahhoz a Wi-Fi hálózathoz, és használja a szerver helyi IP-címét (pl. `http://192.168.1.100:3000`). Az IP-címet a szerver gépen a *ipconfig* (Windows) vagy *ifconfig / ip addr* (Ubuntu/macOS) paranccsal tudja lekérdezni.

**Webböngésző:**

- Windows 10 / 11 operációs rendszer

**2.2.2.2 Webböngésző:**

- Bármely naprakész böngésző, például:
  - Google Chrome
  - Mozilla Firefox
  - Microsoft Edge
  - Safari

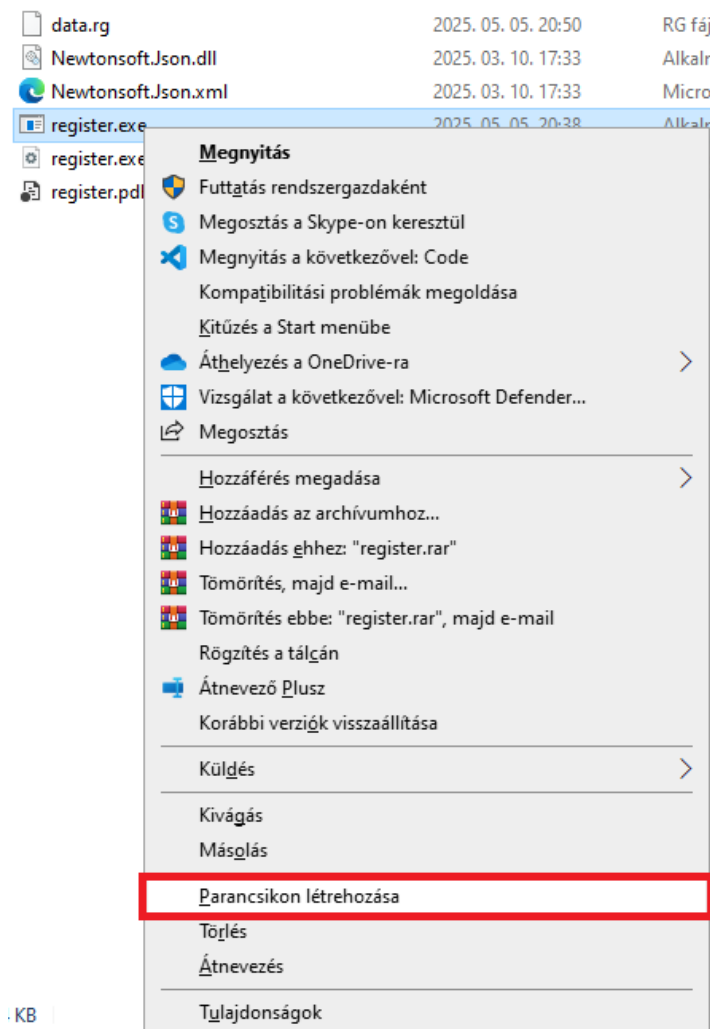
Az alkalmazás modern JavaScript funkciókat (például `fetch API`, `async/await`) használ. Régebbi böngészők esetén kompatibilitási problémák léphetnek fel, ezért a legfrissebb böngészőverzió használata erősen ajánlott.

### 2.2.2.3 Hálózat:

- A fájlok előkészítése miatt indokolt stabil internetkapcsolat a szoftverkomponensek (pl. Node.js, MySQL) letöltéséhez és a Github oldala betöltéséhez.

## 2.3 A program telepítése

Töltse le a “desktopApp.zip” nevű fájlt a github repository “asztali” nevű mappájából. Eztán csomagolja ki a tartalmát egy tetszőleges helyre. Az asztali applikációt a “register.exe” nevű futtatható fájl futtatásával lehet elindítani.



ábra 4 Parancsikont létrehozása

Szükség esetén a gördülékenyebb felhasználói élmény érdekében a “register.exe” nevű fájlra jobbegérgomb lenyomásával az alábbi opcióval hozhat létre parancsikont az alkalmazáshoz, amit utána szabadon testreszabhat és áthelyezhet tetszőleges helyre.

## 2.4 A program használatának a részletes leírása

### 2.4.1 Weboldali funkciók:

#### 2.4.1.1 Amennyiben módosításra lenne szükség a dizájnban

Az oldal színvilágát könnyedén módosíthatjuk a “frontend-->style” mappa és azon belül a “style.css” fájl módosításával. Ezáltal rengeteg féle variáció létezhet ugyanabból a programból.

```
:root {
  --backgroundcolor: #2E2D34;
  --signaturecolor: #007DFC;
  --white: #F5F5F5;
  --black: #0A0A0A;
  --grey: #151515;
  --success: #198754;
  --danger: #DC3545;
  --warning: #DAA520;
  scroll-behavior: smooth;
}
```

ábra 5 Színek, amiket a weboldalak használnak

Emellett a betűtípus is szerkeszthető, a fájl 14. kódsorában van erre lehetősége a felhasználónak.

Képeket ugyanazén mappában az “images”-en belül találhat, itt szintén könnyedén változtathatja meg az oldal ikonját, emblémáját ha egy új fájlal cseréli ki azt (Körülbelül 270 x 175 pixel méretű kép a támogatott a “logo.png” fájl cseréjekor.)

#### 2.4.1.2 Belépés & Rólunk oldal

Miután sikeresen elindította a szerveret és megnyitotta a weboldalt, az alábbi tartalom jelenik meg. A weboldalak láblécében két hivatkozás helyeződik el. Az első hivatkozás (“Mesterremek”) a projekt GitHubos repozitórájára, míg a jobb oldalon található második hivatkozás (“About us”) a fejlesztők személyes bemutatkozó oldalára vezet, ahol rövid ismertető található a csapat tagjairól, amelyeket az arcképei kísérnek. Az kártyáikon lévő Githubos ikont kattintva pedig egy külön oldalon a Github profiljukat érheti el a felhasználó.

Emánuel

A bejelentkezési oldal további elemei a következők: a cég logója, amely alatt egy üdvözlő szöveg található. Az üdvözlő szöveg alatt egy beviteli mező helyezkedik el, ahol a felhasználó a bejelentkezéshez szükséges felhasználói azonosítóját adhatja meg. A bejelentkezést a zöld színű "Login" gomb indítja el, és visz a főoldalra. Mellette egy kék "?" gomb található, amelyre kattintva egy információs szöveg jelenik meg. Ez a szöveg részletesen ismerteti a felhasználói azonosító beszerzésének lépéseit, például egy munkáltató számára.

Részlet az információs szövegből, magyarra fordítva: Az Ön UserID-je (rendszerünkben aazon néven ismert) egy egyedi azonosító, amelyet Önhöz mint alkalmazotthoz vagy engedélyezett felhasználóhoz rendelnek. Ez kapcsolódik az Ön profiljához a belső adatbázisunkban, és meghatározza az Ön hozzáférési jogait, beleértve a webes hozzáférést is.

#### *2.4.1.3 Főmenü oldal*

Sikeres bejelentkezés után a felhasználó a webes felület központi oldalára, a főmenü oldalra kerül. Ezen az oldalon keresztül könnyen elérhetők a legfontosabb aloldalak a navigációs sáv segítségével, amely minden oldalon megjelenik, kivéve a bejelentkezési és a bemutatkozó oldalakat. A „Dashboard” címsor alatt kártyák láthatóak, sorrendben magyarra fordítva: Legutóbbi Tranzakciók, Statisztikák, Termékek, Kijelentkezés, melyek gyorselérést biztosítanak a program kulcs funkcióihoz, melyeket elsősorban a munkavállaló fog használni. Ezeket a kártyákat egy egyszerű motiváló idézet követi, amelynek célja a munkavállalók elkötelezettségének és lelkesedésének fenntartása.

#### *2.4.1.4 Tranzakciók oldal*

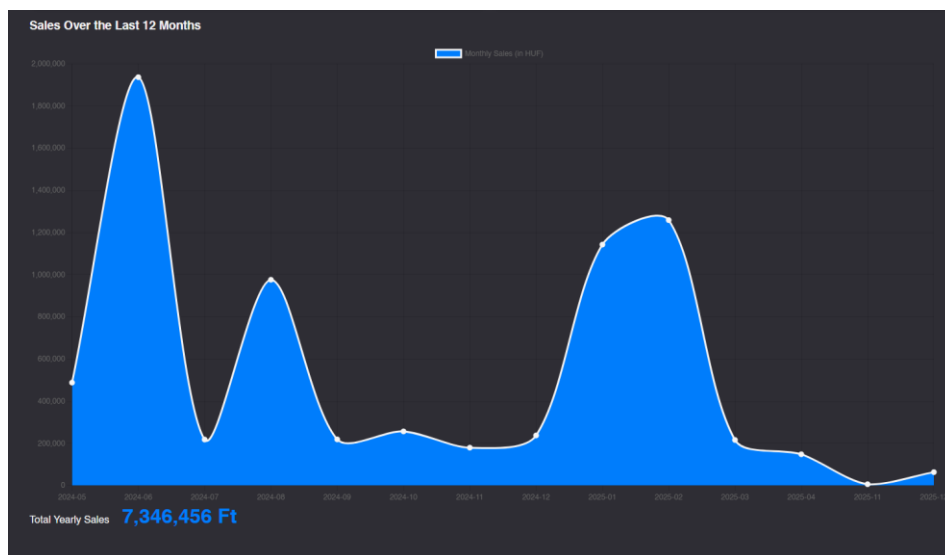
A "Transactions" gombra vagy kártyára (Latest Transactions) nyomva/kattintva elérhetőek a tranzakciók adatainak egy része. Pontosabban a tranzakció azonosítója, keletkezési dátuma, kassza azonosítója, kasszas dolgozó azonosítója, illetve a tételeknek az összege, azaz a számlának a teljes ára, mindezt egy dinamikus táblás elrendezésben, ami nem teszi ki a teljes képernyőt a görgethetési funkciónak köszönhetően. Ezáltal nem veszik el a felhasználó a tranzakciók akár százazrei között. Ezeket a tranzakciókat csak részletesen lehet módosítani vagy pedig teljesen eltörölni, ahogy a képen is látszik.

Emánuel

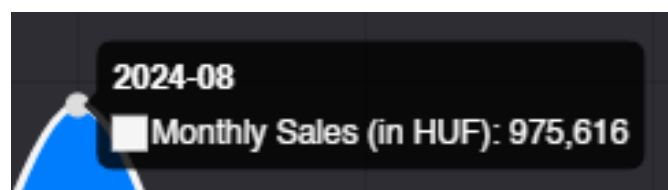
Az oldal egyik hátránya, hogy telefonos felületen nem elérhető, így a dolgozók kénytelenek gépen vagy nagyobb méretű tableten kihasználni ezeket a funkciókat a munkavégzéshez.

#### 2.4.1.5 Statisztikák

A statisztikák oldala rengeteg hasznos tartalommal bír. Felső bal sarokban a dolgozók teljesítményéről kaphatunk tisztább képet az oszlopdiagrammon keresztül. A program kimutatja, hányan dolgoznak a kasszas jogosultsággal a cégben, és kilistázza őket a grafikonon, teljesítmény alapján. Emellett betekintést nyerhetünk arról, ki milyen rég óta dolgozik aktívan a munkahelyén az alatta lévő listában. Ugyanitt jelenik meg az első sorban a cégvezető neve is, akinek szintén van jogosultsága a kasszas szoftverhez. A jobb sarokban leolvashatjuk milyen fizetési módot preferálnak leginkább a vásárlók, kördiagramm használatával. Egér ráhúzásával pontosabb adatok érhetőek el. Egészen pontosan hogy hány darab számla került nyugtázásra a kérdéses fizetési módban.



ábra 6 Vonaldiagramm az évi eladásokról



ábra 7 Pontos eladási összeg a 2024 augusztusi hónapból

A következő képek elárulják a vállalkozás havi eladásait, és annak évi összegét. Ha rávisszük az egerünket (telefonon ujjunkat) a pontokra, megjelenik a havi pontos összeg az adott hónapban.

Az oldal betöltésre történő animációkat is tartalmaz, emellett bizonyos grafikonokat a felettük lévő szövegre kattintva eltüntethetünk. Példa okán a kördiagramm egyik szeletét vagy az oszlopdiaagramm oszlopait, de a vonaldiagrammot is.

#### 2.4.1.6 *Termékek*

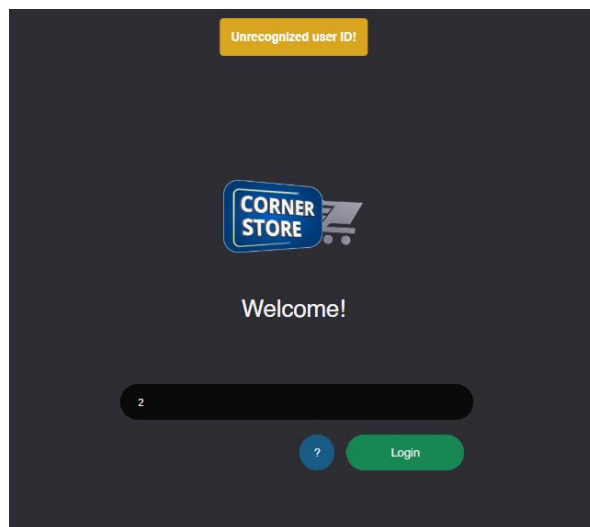
A termékek oldala szabadságot nyújt arra, hogy a termékek entitás minden recordját módosíthassa. A “Manipulate Data” gombbal gyorsan legördülhet a formhoz (vagy akár a jobb alsó sarokban megtalálható gombbal), ahol létrehozhat, szerkeszthet, törölhet adatokat. Egy extra gomb segítségével pedig új, üres form hozható elő.

A kártyákon látható gyorsgombokkal könnyen törölhet vagy módosíthat a termékek közül, míg a “Delete” egy felugró üzenet kíséretében felszabadítja az adatbázist a termék törlésével az azonosítója alapján, addig az “Edit” gomb csupán elviszi a felhasználót az oldal aljára a formhoz, ahol annak a kártyának a specifikus elemeit fogja beiterálni a mezőkbe, ezzel megkönnyítve a szerkesztés lehetőségét, ami mindig a termék azonosító alapján működik, beleértve a törlés funkciót. Egy kivétel a termék létrehozásakor történik, ugyanis azt az azonosítót a database generálja automatikusan. A háttérben minden megadott beviteli mező szigorúan vizsgálva van, hogy ne kerülhessen bele olyan adat amit esetleg az asztali app nem tud kezelni, illetve a felhasználó folyamatos visszajelzéseket kap, ha például üres mezőt ad meg hibásan, nem létező kategóriát akar bevinni a DevTools segítségével, stb. Ez jellemző a többi oldalon lévő űrlapokra is.

Az oldal egyik hátránya, hogy telefonos felületen nem elérhető, így a dolgozók kénytelenek gépen vagy nagyobb méretű tableten kihasználni ezeket a funkciókat a munkavégzéshez.

#### 2.4.1.7 Értésítések funkció

Termékek, tranzakciók és belépés oldalán van használva, sikeres vagy sikertelen beküldésekkor jelenik meg általában. Ezzel gyors és egyszerű választ adva a felhasználónak arról, ha valamit esetleg félregépet, hamis, téves vagy nem értelmezhető adatot dob be. A termékek oldalon minden mezőnek különböző üzenetei lehetnek. Például ha olyan terméket akarunk módosítani, arról egyértelműen visszajelez a szoftver. Illetve biztosítva van az is, hogy ne lehessen olyan kategóriát megadni egy terméknek, ami az adatbázisban hivatalosan nem létezik. Ezzel elkerülve a rosszindulatú adatbázis módosításokat.



ábra 8 Értésítési panel, ahogy megjelenik hibás bejelentkezés esetén

#### 2.4.2 Asztali funkciók:

#### 2.4.3 Bevezető

Ha az applikáció első futtatása nem működő vagy nem futó backend mellett történt, a következő futtatás előtt törölni szükséges az applikáció mappájában található “data.rg” tartalmát, vagy telepítse újra az applikációt.

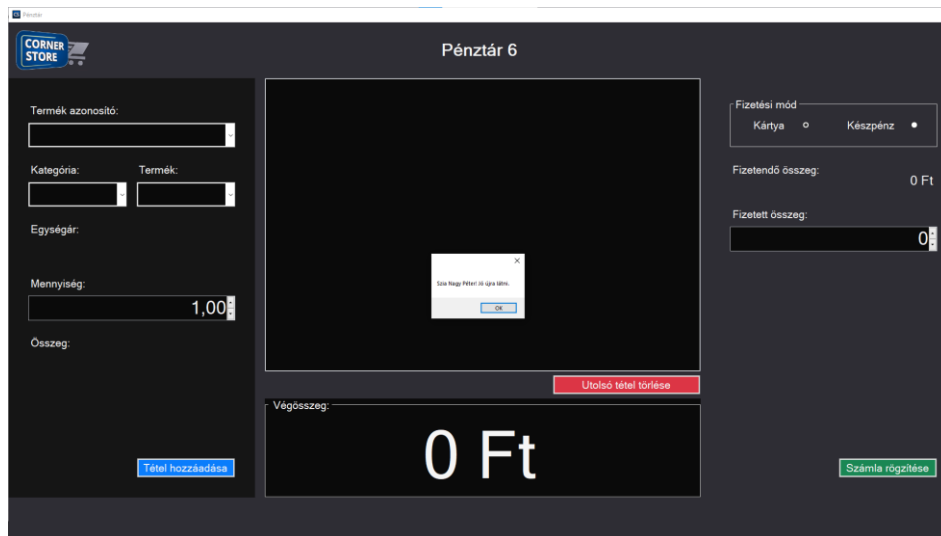
Amennyiben valamilyen okból új pénztárgép számot akar kiosztani a gének, szintén a “data.rg” tartalmát szükséges törölni. Ha pedig manuálisan módosítani szeretné azt akkor az abban szereplő számot kell átírni. Ezek a műveletek azonban nem ajánlottak valamint nem



Emánuel

szükségesek mivel normál esetben a rendszer magától osztja ki a következő még nem foglalt számot.

Az SQL szerver és backend elindítása után 1.1.3-as valamint az 1.1.4-es pontban leírt belépési folyamat után az alábbi felhasználó felület fogadja a felhasználót, valamint személyre szabottan köszönti:



ábra 9 Az asztali applikáció sikeres belépés után

Az applikáció felhasználó felülete vizuálisan és felhasználás szempontjából 3 fontos részre oszthatjuk:

- Baloldali panel - Tétel hozzáadásához használt felület.
- Jobboldali panel - Számla rögzítéséhez használt felület.
- Középső panel - Végösszeg és kosár tartalmának megjelenítése.

#### 2.4.4 Tétel hozzáadása

Válassza ki a termék kategóriáját, a program ezután feltölti a termékek listáját az adott kategóriába tartozó termékekkel.

Ezt követően válassza ki a terméket, majd az applikáció automatikusan kitölti a termék azonosító menüpontot és megjeleníti az ahhoz tartozó adatokat és a termékhez tartozó mennyiség/egységtől függően egész vagy tört módba állítja a mennyiség mezőt, ezeket

Emánuel

utána valós idejűen változtatja a mennyiség átírásakor, vagy másik termék kiválasztásakor.

Amennyiben olyan termék lett kiválasztva amit csak 18 éves kor felett lehet megvásárolni, automatikusan megjelennek a szükséges figyelmeztetések és megerősítési pontok.

Atételt az adatok kitöltése után a kék “Tétel hozzáadása” gombra kattintva adhatja a kosárhoz, aminek tartalma automatikusan frissül.

Tételt a piros “Utolsó tétel törlése” gombra kattintva lehet eltávolítani.



ábra 10 Az asztali applikáció főoldala két termékkel a kosárban

A kosár tartalmának változásakor a végösszeg és a jobboldali panelben megjelenített információk is automatikusan frissülnek.

#### 2.4.5 Fizetés és számla rögzítése

Alapértelmezetten a kártyás fizetés van kiválasztva, ilyenkor a fizetendő összeget nem kerekítjük 5 Ft-ra, a fizetett összeg pedig automatikusan megfelel a fizetendővel és nem jelenítjük meg a visszajáró összegét.

Készpénzes fizetésnél 5 Ft-ra kerekítünk és megadjuk a vásárló által fizetett összeget amiből a program kiszámítja és megjeleníti a visszajárót.

Számlát a zöld “Számla rögzítése” gombbal lehet rögzíteni. Ez azonban csak akkor lehetséges, ha a fizetendő összeg nem kisebb a fizetettnél, valamint egyes terméket tartalmazó kosarak további jóváhagyást igényelhetnek.

Számla rögzítése után az ablak alapállapotba helyezve újra megjelenik.

Az ablak ki nyomásával teljesen bezárja a programot.

## 3 Fejlesztői dokumentáció

### 3.1 Az alkalmazott fejlesztői eszközök

#### 3.1.1 Programozási és leíró nyelvek

- C# (a WinForms alkalmazás fejlesztéséhez)
- JavaScript (a kliensoldali logika és dinamikus elemek implementálásához, pl. transactions.js)
- HTML5 (a webes felület struktúrájához, pl. transactions.html)
- CSS (a webes felület dizájnjához és responszív megjelenítéséhez)
- Markdown (a dokumentáció részeként, például jegyzetekhez vagy formázott szövegekhez)

#### 3.1.2 • Fejlesztési környezetek

- Microsoft Visual Studio (a C# és WinForms kódolásához)
- Microsoft Visual Studio Code (a JavaScript, HTML és CSS szerkesztéséhez, valamint a Node.js projekthez)
- Microsoft Word (a dokumentáció véglegesítéséhez és formázásához)

#### 3.1.3 Adatbázis-kezelő rendszerek

- MySQL (az adatbázis kezelésére, pl. számla és tetel táblákhoz)

#### 3.1.4 Szerveroldali technológiák

- Node.js (a backend futtatásához, pl. app.js)  
felhasznált moduljai:
  - o Express.js (keretrendszer a szerveroldali útvonalakhoz)
  - o MySQL2 (a MySQL-lel való kommunikációhoz)
  - o npm (a Node.js csomagkezeléséhez, projektfüggőségek kezelésére)
  - o Body-parser (HTTP kérések adatfeldolgozásához (illetve néhol express.json()))
  - o Path (fájlrendszer elérési utak kezelésére a szerveren)
  - o URL (fileURLToPath, fájl-URL-ek konvertálására helyi elérési utakra, az url modul részeként)

Emánuel

- Dotenv (A szervertkonfiguráció és érzékeny adatok titkosítására.)

Verziókezelés:

- Git (a forráskód verziókezelésére és a projekt változásainak nyomon követésére)
- GitHub (Távoli tárolás és csapatmunka közreműködésére)

Projektmenedzsment és kommunikációs eszközök:

- Trello (a feladatok szervezésére és követésére)
- Discord (a csapat hangalapú kommunikációjához)
- Microsoft Teams (a gyakran használt jegyzetek, képanyagok tárolására, egyszerű csetelésre)

Dizájn és grafikai eszközök:

- Figma (az oldal elrendezésének, grafikai elemek és a dizájnnyelv megtervezéséhez)
- Webes erőforrások (betűtípusok és színpaletta kiválasztásához, pl. Dafontfree és Coolers)

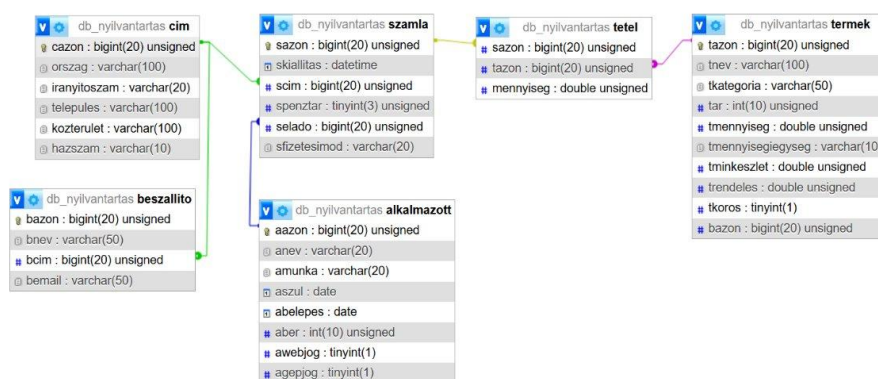
Külső API-k és könyvtárak:

- Chart.js (diagramok és statisztikák megjelenítéséhez a webes felületen)

### 3.1.5 Hardver és operációs rendszer

Windows alapú számítógép (a fejlesztés elsődleges környezeteként)

## 3.2 Adatmodell leírása



ábra 11Kép a táblák relációjáról

### 3.2.1 Bevezetés

A készletkezelő rendszer egy MySQL adatbázisra épül, amely a termékek, beszállítók, tranzakciók és kapcsolódó entitások adatait tárolja. Az adatmodell célja a strukturált adatkezelés, a kapcsolatok biztosítása és a hatékony lekérdezések támogatása. Az alábbiakban részletezem az adattáblákat, a közöttük lévő kapcsolatokat, valamint egy diagramot is bemutatok az átláthatóság érdekében.

### 3.2.2 Adattáblák leírása

Az adatbázis hat táblát tartalmaz, amelyek a rendszer működéséhez szükséges adatokat kezelik. Minden tábla oszlopait, adattípusait és korlátozásait részletezzük.

#### 3.2.2.1 1.1. *db\_nyilvantartas.cim* tábla (Címek)

- **Leírás:** Tárolja a címeket, amelyeket a beszállítók vagy az üzlet használ.
- **Oszlopok:**
  - `cazon` (BIGINT(20) UNSIGNED, PRIMARY KEY, AUTO\_INCREMENT): A cím egyedi azonosítója, a bolt mindig '1'.
  - `orszag` (VARCHAR(100), NOT NULL): Ország neve.
  - `iranyitoszam` (VARCHAR(20), NOT NULL): Irányítószám.
  - `telepules` (VARCHAR(100), NOT NULL): Település neve.
  - `kozterulet` (VARCHAR(100), NOT NULL): Közterület neve.
  - `hazszam` (VARCHAR(10), NOT NULL): Házzám.

#### 3.2.2.2 1.2. *db\_nyilvantartas.szamla* tábla (Tranzakciók)

- **Leírás:** Tárolja a tranzakciók (számlák) adatait, például kiállítási időpontot és fizetési módot.
- **Oszlopok:**
  - `sazon` (BIGINT(20) UNSIGNED, PRIMARY KEY, AUTO\_INCREMENT): A tranzakció egyedi azonosítója.
  - `skiallitas` (DATETIME, NOT NULL): A számla kiállításának időpontja.
  - `sfizetesimod` (VARCHAR(20), NOT NULL): Fizetési mód (pl. "Készpénz").

- spenztar (VARCHAR(20), NOT NULL): Pénztár neve.
- selado (BIGINT(20) UNSIGNED, NOT NULL): Az eladó azonosítója, az alkalmazott táblára hivatkozik.

### 3.2.2.3 1.3. db\_nyilvantartas.tétel tábla (Tranzakciós tételek)

- **Leírás:** Tárolja a tranzakciókhoz tartozó tételeket, például a vásárolt termékeket és mennyiségeket.
- **Oszlopok:**
  - szon (BIGINT(20) UNSIGNED, FOREIGN KEY, NOT NULL): A tranzakció azonosítója, a számla táblára hivatkozik.
  - tazon (BIGINT(20) UNSIGNED, FOREIGN KEY, NOT NULL): A termék azonosítója, a termék táblára hivatkozik.
  - mennyiség (DOUBLE UNSIGNED, NOT NULL): A vásárolt mennyiség.
  - **Összetett elsődleges kulcs:** (szon, tazon).

### 3.2.2.4 1.4. db\_nyilvantartas.termek tábla (Termékek)

- **Leírás:** Tárolja a termékek adatait, például nevüket, készletüket és árukat.
- **Oszlopok:**
  - tazon (BIGINT(20) UNSIGNED, PRIMARY KEY, AUTO\_INCREMENT): A termék egyedi azonosítója.
  - tnev (VARCHAR(100), NOT NULL): A termék neve.
  - tkategoria (VARCHAR(50)): A termék kategóriája (pl. "Gyümölcs").
  - tar (INT(10) UNSIGNED, NOT NULL): A termék ára (forintban).
  - tmennyiség (DOUBLE UNSIGNED, NOT NULL): Aktuális készlet.
  - tmennyisegiegység (VARCHAR(10), NOT NULL): Mennyiségi egység (pl. "db", "kg").
  - tminkeszlet (DOUBLE UNSIGNED, NOT NULL): Minimális készletszint.
  - tutankeszlet (DOUBLE UNSIGNED, NOT NULL): Utántöltési mennyiség.
  - tkoros (TINYINT(1), NOT NULL): Korhatárhoz kötött-e (0: nem, 1: igen).

- `bazon` (BIGINT(20) UNSIGNED, NOT NULL): A beszállító azonosítója, a `beszallito` táblára hivatkozik.

### 3.2.2.5 1.5. *db\_nyilvantartas.beszallito* tábla (Beszállítók)

- **Leírás:** Tárolja a beszállítók adatait, például nevüket, emailjüket és címüket.
- **Oszlopok:**
  - `bazon` (BIGINT(20) UNSIGNED, PRIMARY KEY, AUTO\_INCREMENT): A beszállító egyedi azonosítója.
  - `bnev` (VARCHAR(50), NOT NULL): A beszállító neve.
  - `bemail` (VARCHAR(50), NOT NULL): A beszállító email címe.
  - `bcim` (BIGINT(20) UNSIGNED, NOT NULL): A cím azonosítója, a `cim` táblára hivatkozik.

### 3.2.2.6 1.6. *db\_nyilvantartas.alkalmazott* tábla (Alkalmazottak)

- **Leírás:** Tárolja az alkalmazottak adatait, például nevüket és jogosultságukat.
- **Oszlopok:**
  - `aazon` (BIGINT(20) UNSIGNED, PRIMARY KEY, AUTO\_INCREMENT): Az alkalmazott egyedi azonosítója.
  - `anev` (VARCHAR(20), NOT NULL): Az alkalmazott neve.
  - `amunka` (DATE): Az alkalmazott belépésének dátuma.
  - `abelepes` (DATE): Az alkalmazott belépésének dátuma (duplikátum, ellenőrizendő).
  - `aber` (INT(10) UNSIGNED): Egyéb azonosító (célja nem egyértelmű).
  - `awebjog` (TINYINT(1)): Webes jogosultság (0: nem, 1: igen).
  - `agepjog` (TINYINT(1)): Gépi jogosultság (0: nem, 1: igen).
- **Megjegyzés:** Az `amunka` és `abelepes` oszlopok duplikálódnak, ami redundanciát jelez. Javasolt az egyik eltávolítása.

## 3.2.3 Kapcsolatok az adattáblák között

Az adattáblák közötti kapcsolatokat idegen kulcsok (FOREIGN KEY) biztosítják:

- **cim ↔ beszallito:**



Emánuel

- Kapcsolat: A beszallito.bcim idegen kulcs a cim.cazon-ra hivatkozik. Egy címhez több beszállító is tartozhat (1:N kapcsolat).
- Példa: A beszallito.bcim: 1 a cim.cazon: 1 rekordjára mutat.
- **beszallito ↔ termék:**
  - Kapcsolat: A termék.bazon idegen kulcs a beszallito.bazon-ra hivatkozik. Egy beszállítóhoz több termék tartozhat (1:N kapcsolat).
  - Példa: A termék.bazon: 101 a beszallito.bazon: 101 rekordjára mutat.
- **termék ↔ tétel:**
  - Kapcsolat: A tétel.tazon idegen kulcs a termék.tazon-ra hivatkozik. Egy termékhez több tétel tartozhat egy tranzakcióban (1:N kapcsolat).
  - Példa: A tétel.tazon: 1 a termék.tazon: 1 rekordjára mutat.
- **szamla ↔ tétel:**
  - Kapcsolat: A tétel.sazon idegen kulcs a szamla.sazon-ra hivatkozik. Egy tranzakcióhoz több tétel tartozhat (1:N kapcsolat).
  - Példa: A tétel.sazon: 1 a szamla.sazon: 1 rekordjára mutat.
- **szamla ↔ alkalmazott:**
  - Kapcsolat: A szamla.selado idegen kulcs az alkalmazott.aazon-ra hivatkozik. Egy alkalmazott több tranzakciót is kezelhet (1:N kapcsolat).
  - Példa: A szamla.selado: 1 az alkalmazott.aazon: 1 rekordjára mutat.

### 3.3 Részletes feladat-specifikáció, algoritmusok

#### 3.3.1 Bevezetés

Ez a dokumentum a készletkezelő rendszer kulcsfontosságú függvényeinek és metódusainak specifikációját, valamint az algoritmizálható részek leírását tartalmazza. A specifikáció részletezi a funkciókat, paramétereket, visszatérési értékeket, és pszeudokód formájában mutat be.

### 3.3.2 Weboldal lényeges függvényeinek specifikációja

#### 3.3.2.1 1.1. *fetchSuppliers()* függvény (*products.js*)

Leírás: Lekéri a beszállítók azonosítóit a /server/beszallito végpontról, és frissíti a globális validSupplierIds tömböt.

Paraméterek: Nincs paraméter.

Visszatérési érték: *Array<number>* – A beszállítók azon azonosítóit tartalmazó tömb, üres tömb ([]) hibás esetben.

```
async function fetchSuppliers() {
  try {
    const response = await fetch('/server/beszallito');
    if (!response.ok) throw new Error(`Failed to fetch suppliers: ${response.status}`);
    const suppliers = await response.json();
    validSupplierIds = suppliers.map(s => s.bazon);
    return validSupplierIds;
  } catch (error) {
    console.error('Error fetching suppliers:', error);
    showNotification(`Error loading suppliers: ${error.message}`, 'warning');
    validSupplierIds = [];
    return [];
  }
}
```

ábra 12A *fetchSuppliers()* függvény a *products.js*-en belül

Magyarázat: A függvény aszinkron módon hívja a server végpontot, kezeli a hibákat, és a kapott bazon értékeket ([101, 102]) menti a validSupplierIds-be validációhoz.

#### 3.3.2.2 *WUpdateTermek(termek)* függvény (*db.js*)

Leírás: Frissíti egy meglévő termék rekordját a termék táblában az adatbázisban a megadott paraméterek alapján. A függvény *SQL UPDATE* parancsot használ, amely a tazon azonosítóval azonosított rekordot módosítja.

- **Paraméterek:**

- *termek* (Object): Egy objektum, amely a frissítendő termék adatait tartalmazza. A következő mezőkkel kell rendelkeznie:
  - *tnev* (string): A termék neve.
  - *tkategoria* (string): A termék kategóriája.
  - *tar* (number): A termék ára.
  - *tmennyiség* (number): A termék készlete.

- **tmennyisegiegység (string):** A készlet egysége (pl. "kg", "db", stb.).
  - **tkoros (boolean):** A termék korhatáros-e (0 vagy 1).
  - **bazon (number):** A beszállító azonosítója.
  - **tminkeszlet (number):** A minimum készlet szint.
  - **trendeles (number):** Az utántöltési mennyiség.
  - **tazon (number):** A frissítendő termék azonosítója (WHERE feltétel).
- **Visszatérési érték:** Object – A MySQL execute metódus visszatérési értéke, amely a frissítés eredményét tartalmazza (pl. az érintett sorok számát: { affectedRows: 1, ... }). Hibás esetben kivételt dobhat.
  - **Pszeudó kódja:**

*FÜGGVÉNY WUpdateTermek(termek)*

*KÖZLEP: Kezdés*

*VÁLTOZÓK:*

```
sql = "UPDATE termék SET tnev = ?, tkategoria = ?, tar =
      ?, tmennyiség = ?, tmennyiségiegység = ?, tkoros = ?, bazon
      = ?, tminkeszlet = ?, trendeles = ? WHERE tazon = ?"
paraméterek = [termek.tnev, termék.tkategoria, termék.tar,
               termék.tmennyiség, termék.tmennyiségiegység, termék.tkoros, t
               ermek.bazon, termék.tminkeszlet,
               termék.trendeles, termék.tazon, termék.tazon]
```

*PRÓBÁL*

```
result ← connection.execute(sql, paraméterek) // SQL parancs
futtatása
```

*VISSZA result[0] // Az eredmény első eleme*

*KIVÉTEL (error)*

*DOBJ hibát(error) // Hiba továbbdobása*

*VÉGE PRÓBÁL*

*KÖZLEP: Vége*

*VÉGE FÜGGVÉNY*

### 3.3.3 Összegzés

A specifikáció részletezi a kulcsfontosságú függvényeket (fetchSuppliers, WGetBeszallito, /beszallito), amelyek a beszállítók lekérdezését és validációját biztosítják. Az algoritmusok pszeudokód és struktogram formájában ábrázolják a logikát, míg a kódrészek magyarázata segít megérteni a működést. A teljes forráskód a repositoryban (<https://github.com/zenerra/vizsgaremek>) érhető el, a dokumentáció pedig a lényeges részekre fókuszál.

## 3.4 Tesztelési dokumentáció

### 3.4.1 Bevezetés

A tesztelési dokumentáció célja a készletkezelő rendszer funkcionalitásának ellenőrzése különböző felhasználói forgatókönyvek során, a program reakcióinak és üzeneteinek elemzése, valamint a hibák azonosítása. A tesztelés során fekete doboz és fehér doboz módszereket alkalmaztunk, hogy átfogó képet kapjunk a rendszer működéséről.

### 3.4.2 Tesztesetek

#### 3.4.2.1 *Teszteset: Termék szerkesztés érvénytelen adatokkal (Normál teszteset, Fekete doboz módszer)*

**Leírás:** Egy terméket szándékosan nem támogatott adatokkal próbálunk feltölteni.

**Lépések:**

1. Navigálunk a products.html oldalra (<http://localhost:3000/web/products.html>).
2. Az űrlapot kitöltjük valótlan, értelmezhetetlen adatokkal.
3. Elküldjük / módosítani kísérelünk az “Edit” / “Create Product” gombokkal

**Eredmény:** Az oldal visszautasítja a kérést, nem hajtja végre. A showNotification függvény segítségével megtudakolja a rendszer velünk, hogy hol akadt el pontosan a kérésünk a weboldal tetején, ami akár láncszerűen is meg tud jelenni, így a felhasználó azonnal tudni fogja mi mindent csinált rosszul. Kódpélda:

```

if (!tnev || tnev.length > 100) errors.push('Product name is required and must be ≤ 100 characters.');
```

```

if (isNaN(tar) || tar < 0) errors.push('Price must be a non-negative number.');
```

```

if (isNaN(tmennyiseg) || tmennyiseg < 0) errors.push('Stock must be a non-negative number.');
```

```

if (isNaN(tminkeszlet) || tminkeszlet < 0) errors.push('Minimum quantity must be a non-negative number.');
```

```

if (isNaN(trendeles) || trendeles < 0) errors.push('Refill size must be a non-negative number.');
```

```

if (isNaN(bazon) || bazon <= 0) errors.push('Supplier ID must be a positive integer.');
```

```

if (!validSupplierIds.includes(bazon)) errors.push('Supplier ID does not exist in the suppliers list.');
```

```

if (errors.length > 0) {
  showNotification(errors.join(' '), 'warning');
  return false;
}
return true;

```

ábra 13A validálási függvény lecsekkolja, hogy minden bevitt adat az űrlapról megfelel a feltételeknek, ha nem, azt üzenetben adja vissza

### 3.4.2.2 *Tesztet: Szerver válaszok naplózása és tesztelése az app.js-ben(Normál tesztet, Fehér doboz módszer)*

Az app.js-ben aktívan teszteljük a szerver válaszait egy middleware segítségével, amely minden kéréshez naplózza a szerver válaszádatait és a státuszkódot. Ez a middleware átírja a res.json függvényt, hogy minden válasz előtt kiírja a kérés módszerét (pl. GET, POST), az URL-t (pl. /server/termek/all), a válasz tartalmát (pl. a visszaküldött JSON adatot), valamint a státuszkódot (pl. 200 OK). Így a szerver konzolon keresztül könnyen nyomon követhetjük a válaszokat, ami segít a hibakeresésben és a végpontok helyes működésének ellenőrzésében, például ha egy végpont váratlan státuszkódot (pl. 500) ad vissza. A middleware kódja:

```

app.use((req, res, next) => {
  const originalJson = res.json;
  res.json = function (data) {
    console.log(`Response for ${req.method} ${req.url}:`, data, `Status: ${res.statusCode}`);
    return originalJson.call(this, data);
  };
  next();
});

```

ábra 14A kód ami kiírja a szerver parancssorába a válaszait az app.js fájlban

Ez a naplózás különösen hasznos volt a /server/beszallito végpont hibás státuszkódjának (201 Created) azonosításában, amit később 200 OK-ra javítottunk.

### 3.4.2.3 *Teszteset: Tranzakciók lekérdezése üres adatbázissal (Normál teszteset, Fehér doboz módszer)*

**Leírás:** A tranzakciók listázását teszteljük egy üres adatbázisban, hogy ellenőrizzük, hogyan kezeli a rendszer a hiányzó adatokat. A fehér doboz módszert alkalmaztuk, mivel ismertük a transactions.js kódját, és tudtuk, hogy a fetchTransactions függvény üres lista esetén egy "No transactions available." üzenetet jelenít meg.

**Lépések:**

1. Üresítjük a számla, tetel, alkalmazott, és termék táblákat az adatbázisban:
2. Navigálunk a transactions.html oldalra (<http://localhost:3000/web/transactions.html>).
3. Megvárjuk, hogy a fetchTransactions függvény lefusson.

**Várt eredmény:** A rendszer nem talál tranzakciókat, és egy üzenetet jelenít meg.

**Kapott eredmény:** A transactionList elemében megjelenik a "No transactions available." üzenet. A böngésző konzolban nem jelenik meg hiba.

**Üzenet kezelése:** Tranzakciót kell rögzíteni az adatbázisba.

Ezután újra kell tölteni az oldalt.

**Értékelés:** A teszt sikeres, a rendszer megfelelően kezeli az üres adatbázis állapotát. Ez az eljárás működik a webes felület összes lekérdezésekor.

### 3.4.2.4 *Teszteset: Tömeges érvénytelen adatbevitel extrém mennyiségű termékkel (Extrém teszteset, Bolondbiztosság tesztelése, Fekete doboz módszer)*

**Leírás:** A rendszert extrém terhelésnek tesszük ki azzal, hogy egyszerre próbálunk meg 10 000 érvénytelen adatot tartalmazó terméket létrehozni (pl. negatív árakkal vagy nem létező beszállító azonosítóval), hogy teszteljük a validáció határait és a szerver stabilitását.

**Lépések:**

1. Navigálunk a products.html oldalra (<http://localhost:3000/web/products.html>).

Emánuel

2. Egy automatizált szkriptet (pl. JavaScript) használunk az űrlap tömeges kitöltésére, amely 10 000 rekordot generál az alábbi adatokkal:
  - a. Terméknév: Random szöveg (pl. "Termék\_X")
  - b. Kategória: "Egyéb"
  - c. Ár: -100 (negatív érték)
  - d. Készlet: 0
  - e. Egység: "db"
  - f. Minimum készlet: -5 (negatív érték)
  - g. Utántöltési mennyiség: 0
  - h. Korhatár: Igen (checked)
  - i. Beszállító ID: 999 (nem létező beszállító)
3. A szkript elküldi az adatokat a szervernek a "Save" gomb szimulált megnyomásával.

**Várt eredmény:** A rendszer minden rekordot elutasít, hibát jelez minden érvénytelen bemenetnél, és nem omlik össze a szerver vagy a kliens oldali felület.

**Kapott eredmény:** A rendszer a showNotification függvényen keresztül többször megjeleníti az alábbi üzeneteket:

- "Price must be a non-negative number." (minden negatív ár esetén)
- "Supplier ID does not exist in the suppliers list." (minden 999-es beszállító ID esetén)
- "Minimum quantity must be a non-negative number." (minden negatív minimum készlet esetén)

A szerver és a kliens oldali felület stabil maradt, nem történt összeomlás.

**Üzenet kezelése:**

- A felhasználónak ellenőriznie kell az űrlap adatait, és csak érvényes, nem negatív értékeket (pl. ár: 0 vagy nagyobb, minimum készlet: 0 vagy nagyobb) kell megadnia.
- A nem létező beszállító ID helyett a /server/beszallito végpont válaszában található érvényes bazon értékeket (pl. 2) kell használni.
- Ha a rendszer lassulást mutat, csökkenteni kell a egyszerre küldött rekordok számát, vagy optimalizálni a kliensoldali szkriptet.

**Értékelés:** A teszt sikeres, a rendszer helyesen elutasította az érvénytelen adatokat, és nem omlott össze a tömeges terhelés alatt.

*3.4.2.5 Teszteset: Asztali applikáció felhasználói autentikálásának tesztelése*

<b>Teszteset</b>	<b>Példa bemenet</b>	<b>Várt eredmény</b>	<b>Valós eredmény</b>
<b>Nem numerikus érték bevitele</b>	“admin”	Sikertelen belépés	Sikertelen belépés
<b>Nem létező azonosító bevitele</b>	“200”	Sikertelen belépés	Sikertelen belépés
<b>Nem engedélyezett azonosító bevitele</b>	“8”	Sikertelen belépés	Sikertelen belépés
<b>Megfelelő azonosító bevitele</b>	“1”	Sikeres belépés	Sikeres belépés



Emánuel

**Értékelés:** A teszt sikeres, a rendszer minden tesztnél megfelelően reagált és minden sikertelen belépési próbálkozáskor egyértelmű egyéni hibaüzenettel reagál a felhasználó kérésére.

### 3.4.3 Összegzés

A tesztelés során fekete doboz és fehér doboz módszereket alkalmaztunk, hogy különböző forgatókönyveket vizsgáljunk. A normál tesztesetek (pl. termék létrehozása, tranzakciók lekérdezése) igazolták a rendszer alapvető működését, míg az extrém teszteset a bolondbiztosságot tesztelte. A hibák azonosítása és javítása során a rendszer stabilitása jelentősen javult, biztosítva a felhasználók számára a megbízható működést.

## 4 Összefoglalás

### 4.1 Önértékelés

#### 4.1.1 Hummel Vendel önértékelése:

Próbáltam olyan felületet kidolgozni, amely praktikus eszközt kínál a hétköznapi kihívások kezelésére, például egy kisvállalkozó számára a pénztárgépek kezelésénél. A hangsúlyt a rugalmas működésre és a felhasználók igényeire helyeztem, hogy gördülékenyebbé tegyem a munkavégzést. Remélem, az alkalmazás valóban megkönnyíti a mindennapokat azok számára, akik használják.

Az asztali applikációhoz a későbbiekben tervezek telepítő programot is létrehozni, de ezt azonban azért nem éreztem szükségesnek, mivel az applikáció felhasználási élményébe nem tartozik bele annak telepítése, mivel a telepítést nem az azt felhasználó pénztárosok hajtják végre. Azonban a jelenlegi telepítési folyamatot is kellően egyszerűnek találom.

Azt gondolom, hogy a magam elé kitűzött céljaim nagy részét megfelelően teljesítettem, és bár minden szoftvert tovább lehet fejleszteni és nincs olyan ami 100%-ban minden felhasználási mód tesztelve lenne, ennek ellenére véleményem szerint elégedetten adhatjuk ki kezeinkből ezt az üzleti nyilvántartás és a hozzá tartozó felhasználói felületeket.

Mindazon által kétségtelen, hogy ha ma kezdeném el az applikáció fejlesztését, sok mindent máshogy csinálnék amire azóta találtam jobb megoldást. Ez az applikáció forráskódjában is felfedezhető, hogy egyes folyamatok kódjai bár hasonlóan, de egyre szofisztikáltabban lettek megvalósítva, ahogy új dolgokat tanultam és tapasztalatokat szereztem.

#### 4.1.2 Hunka Róbert Emánuel önértékelése

##### - Probléma (1):

A kliens oldalon egy form elküldése után az input mezők kattintásra nem reagáltak, mert egy üzenetdoboz (message box) hibája blokkolta őket.

Ok és hogyan történt : A probléma abból adódott, hogy a Node.js nem támogatja megfelelően ezt a funkciót, emiatt az üzenetdoboz zavarokat idézett elő a kliensoldali JavaScriptben.

Megoldás: Idő és egyértelmű megoldás hiányában végül nem használtam üzenetdobozokat, helyette egy értesítési rendszert építettem fel, ami megoldotta a problémát.

##### - Probléma (2):

Amikor megpróbáltam frissíteni egy adatot a számla táblában, kaptam egy hibát, ami miatt nem ment a művelet.

Ok és hogy történt : A hiba onnan eredt, hogy MySQL-ben a tetel tábla szon mezője idegen kulcsként (foreign keys) hivatkozott a számla táblára, és a frissítés megsértette ezt a korlátozást. Olyan rekordot próbáltam módosítani, amit a tetel tábla még használt.

Megoldás: Először ellenőriztem a kapcsolódó rekordokat a tetel táblában, és frissítettem őket, hogy ne legyen korlátozási hiba. Utána már működött a számla tábla frissítése.

- Miért történt : A MySQL-ben az idegen kulcsok (foreign keys) biztosítják az adatbázis integritását. Ha egy számla rekordot próbáltál módosítani vagy törölni, de a tetel tábla még hivatkozott rá, a MySQL hibát dobott, mert nem engedélyez változásokat, amelyek megsértenék a kapcsolatot.

##### - Céljaim:

Céлом volt egy olyan UI-t építeni, amit szép, barátságos és közérthető használni egy átlagos embernek. Emiatt animációk mellett Figmát (egy kollaboratív felülettervező szoftver) is használtam a letisztult kinézet elérése érdekében. Igyekeztem olyan

Emánuel

programit készíteni, ami akár valóban segítségül szolgálna például egy kiskereskedő vállalkozó számára. Ugy gondolom, ezek bizonyos mertekben teljesültek is, amire büszke vagyok

A projekt során elsajátítottam az adatbázis és a backend közötti kapcsolat kialakítását, amely korábban nehézséget okozott számomra. A termékek CRUD felépítésének sikeres kidolgozása jelentős előnyt jelentett, mivel lehetővé tette, hogy a tranzakciók szerkesztését ugyanilyen hatékonyan valósítsam meg.

## 4.2 Továbbfejlesztési lehetőségek

### 4.2.1 Továbbfejlesztői lehetőségek Hummel Vendel szerint

1: A projekt végső szakaszában rájöttem, hogy a termékeknek érdemes lenne egy „elérhető-e” tulajdonsággal rendelkezniük, amely jelzi, hogy egy termék kapható-e az üzletben, vagy csak az adatbázisban szerepel (például limitált kiadású termékként).

A probléma az, hogy a termékek értékeinek (például árának) módosítása befolyásolná a korábbi tranzakciók összegét, ami jelentős hibákat okozhat, például a statisztikák megjelenítésekor.

Megoldásként egy külön entitás létrehozását javaslom a múltbéli értékek tárolására, így a tranzakciók mentése zavartalanul történhetne, anélkül hogy a termékárak vagy más változók problémát okoznának. Jelenleg átmeneti megoldásként a felhasználó létrehozhat egy új terméket hasonló névvel és eltérő árazással, miközben a régi termék készletét nullára csökkenti. Ez megakadályozza a statisztikák torzulását és a duplikált termékek megjelenítését az asztali felületen. A jövőben érdemes lenne ezen a megoldáson továbbfejleszteni a társammal.

Valamint életszerű használatban, megfelelő eszközökkel rendelkezve bővíteni lehetne vonalkód olvasó rendszerrel és automatikus nyugta nyomtatással. Ezek jelentősen egyszerűsítene a termékek gyors azonosítását és tranzakciók nyugtázását, valamint gördülékenyebb felhasználó élményt nyújtanának.

#### **4.2.2 Továbbfejlesztői lehetőségek Hunka Róbert Emánuel szerint**

1: Szerettem volna lehetőséget biztosítani a statisztikák fájlba történő exportálására, hogy a szerveren kívül is használhatóak legyenek, de időhiány miatt ez nem valósult meg.

2. Eleinte tervben volt egy admin felület is, amiben testreszabhatók az egyes alkalmazottak jogosultságai, adatai (például fizetés), és erősebb biztonságot biztosít a bejelentkezési oldalon, de ezt az iskolai évek alatt nem tanultuk meg alaposan, így nem került rá sor.

3. A tranzakciók nem teljesen szerkeszthetők a webes felületen, mert a termékek CRUD fejlesztése sok időt vett igénybe, ezért egy kisebb, kompakt verziót készítettem a tranzakciók szerkesztésére a teljes körű szerkeszthetőség helyett, hiszen a termék tablához hasonlóan rengeteg entitást érintett egyszerre.

## 5 Felhasznált irodalom

### 5.1 Felhasznált weboldalak:

- W3Schools (2025.05.02) - <https://www.w3schools.com>
- Boros Sandor GitHub repository (2025.04.22) - [https://github.com/borossandor27/orai\\_anyagok](https://github.com/borossandor27/orai_anyagok)
- CSS Unit Converter (2025.04.25) - <https://cssunitconverter.vercel.app/rem-to-em>
- ⓘ Character (2025.04.16) - <https://www.compart.com/en/unicode/U+24D8>
- Helvetica Font (2024.11.17) - <https://font.download/font/helvetica-255>
- Chart.js API (2025.03.27) - <https://www.chartjs.org/docs/latest/charts/bar.html>
- MySQL Node.js Express videó Sam Meech-Wardtól (2024.12.02) - [https://youtu.be/Hej48pi\\_lOc?si=Vav7bRaEriQxE7SA](https://youtu.be/Hej48pi_lOc?si=Vav7bRaEriQxE7SA)
- Chat GPT adatok automatikus feltöltésére (2024.11.28) - <https://chatgpt.com/>
- Colors (2024.11.17) - <https://coolors.co>
- Munkavállalói idézet Charles Dickenstől (2025.05.02) - <https://www.vantagecircle.com/en/blog/employee-motivational-quotes/>
- Stack Overflow - (2025.11.18) <https://stackoverflow.com/>

## 6 Ábrajegyzék

ábra 1 Terminál tartalma sikeres telepítés esetén .....	5
ábra Asztali bejelentkező felület .....	6
ábra Webes bejelentkező felület .....	6
ábra Parancsikon létrehozása .....	11
ábra 5 Színek, amiket a weboldalak használnak .....	12
ábra Vonaldiagramm az évi eladásokról .....	14
ábra Pontos eladási összeg a 2024 augusztusi hónapból .....	14
ábra Értesítési panel, ahogy megjelenik hibás bejelentkezés esetén.....	16
ábra Az asztali applikáció sikeres belépés után .....	17
ábra Az asztali applikáció főoldala két termékkel a kosárban .....	18
ábra Kép a táblák relációjáról .....	21
ábra A fetchSuppliers() függvény a products.js-en belül.....	26
ábra A validálási függvény lecsekkolja, hogy minden bevitt adat az űrlapról megfelel a feltételeknek, ha nem, azt üzenetben adja vissza.....	29
ábra A kód ami kiírja a szerver parancssorába a válaszait az app.js fájlban.....	29