BGSZC Logisztikai és Kereskedelmi Technikum és Szakképző Iskola

1087 Budapest, Szörény utca 2-4.

1. Záró dolgozat

MTG Card Shop

Konzulens tanár: Készítették:

Boros Sándor Masznyik Márton, Berényi Benedek

Tartalom

[1 Bevezetés 2](#_Toc879919144)

[1.1 Feladat leírás 3](#_Toc1269499038)

[1.2 A felhasznált ismeretek 4](#_Toc238747000)

[1.3 A felhasznált szoftverek 5](#_Toc1571574783)

[2 Felhasználói dokumentáció 6](#_Toc64205971)

[2.1 A program általános specifikációja 7](#_Toc848403734)

[2.2 Rendszerkövetelmények 7](#_Toc1621509621)

[2.3 3. A program telepítése 8](#_Toc1405238500)

[2.4 A program használatának a részletes leírása 14](#_Toc2006876319)

[3 MTG Card Shop – Webes alkalmazás Felhasználói dokumentáció 14](#_Toc1140699594)

[1. Bevezetés 15](#_Toc1372963528)

[2. Program működése 15](#_Toc29262641)

[MTG Card Shop – Asztali alkalmazás Felhasználói dokumentáció 18](#_Toc805722337)

[1. Bevezetés 19](#_Toc977565894)

[2. A program indítása 19](#_Toc220921783)

[3. Események kezelése (Events) 20](#_Toc1838767929)

[4. Felhasználók kezelése (Customers) 21](#_Toc920156500)

[5. Termékek kezelése (Products) 22](#_Toc883155060)

[6. Megjelenési módok: Light és Dark Mode 24](#_Toc377246720)

[7. Adatkezelés és validálás 25](#_Toc1302750177)

[8. Mentés és frissítés 26](#_Toc919918172)

[9. Telepítési tudnivalók 26](#_Toc1042451609)

[10. Hibakezelés és támogatás 27](#_Toc909253547)

[4 Fejlesztői dokumentáció 27](#_Toc670839197)

[4.1 Az alkalmazott fejlesztői eszközök 28](#_Toc1593006219)

[4.2 Adatmodell leírása 29](#_Toc474427542)

[4.3 Részletes feladatspecifikáció, algoritmusok 31](#_Toc1368478855)

[Webes Alkalmazás: 31](#_Toc1888185363)

[4.4 Web alkalmazás specifikációk: 31](#_Toc622595746)

[Asztali Alkalmazás: 33](#_Toc964680022)

[1. Program függvényeinek és osztályainak metódusainak specifikációja 33](#_Toc970068341)

[2. Algoritmusok 38](#_Toc280265050)

[3. A kód fontosabb részei – Magyarázattal 39](#_Toc2121435217)

[4. UML aktivitásdiagram – Termék frissítése 41](#_Toc996500418)

[Asztali alkalmazás tesztek: 42](#_Toc2002666506)

[Teszteset 3: Extrém adatok termék hozzáadásakor 44](#_Toc391910108)

[5 Összefoglalás 46](#_Toc1865171257)

[5.1 Önértékelés 47](#_Toc1047046261)

[5.2 Továbbfejlesztési lehetőségek 48](#_Toc688296493)

[6 Felhasznált irodalom 49](#_Toc1235058256)

# Bevezetés

## Feladat leírás

A szakmai érettségi projektünk célja egy olyan komplex alkalmazás fejlesztése volt, amely egy Magic: The Gathering (MTG) témájú kártyabolt működését támogatja, mind webes, mind asztali alkalmazás formájában. A projekt címe: MTG Card Shop.

A Magic: The Gathering egy világszerte ismert és elismert gyűjtögetős kártyajáték, amely 1993 óta van jelen a piacon, és az évtizedek alatt számos különféle kiadással és játékmóddal bővült. A játékosok egyedi vagy előre elkészített paklikkal, különféle versenyformátumokban mérhetik össze tudásukat. Az MTG világában kártyák ezrei léteznek, amelyek értéke jelentősen változhat, attól függően, hogy milyen gyakoriak, mennyire erősek, vagy éppen mennyire keresettek a játékosközösségben.

Projektünk célja, hogy egy könnyen kezelhető, naprakész információkat biztosító, korszerű felhasználói felülettel rendelkező alkalmazást hozzunk létre, amely lehetőséget nyújt:

Termékek böngészésére, vásárlására és eladására

A készlet és árak kezelésére adminisztrátori oldalon

Az új kiadások naprakész nyomon követésére

A bolt háttérrendszerének átlátható és hatékony menedzselésére

A program használatához, telepítéséhez és beüzemeléséhez részletes felhasználói dokumentáció készült. Ennek célja, hogy segítséget nyújtson minden leendő felhasználónak a rendszer használatának elsajátításában, valamint megkönnyítse a telepítési és működtetési folyamatokat. A dokumentáció lépésről lépésre bemutatja a rendszer funkcióit, lehetőségeit, és választ ad a felmerülő gyakorlati kérdésekre is.

* A projekt két fő komponensből épül fel:

Webes alkalmazás

A felhasználók ezen a felületen tudnak:

* böngészni a MTG hivatalos termékei között
* Termékeket rendelni és eseményekre jelentkezni
* értesülni az új eseményekről és esetleges árváltozásokról

A webalkalmazás frontendje modern technológiákkal (React, HTML, CSS, JavaScript) készült, a backend Node.js és Express alapokra épül. A webes felület a felhasználói élményt helyezi előtérbe, gyors betöltéssel, reszponzív designnal és logikus navigációval.

Asztali alkalmazás

Ez a rész a bolt adminisztratív tevékenységeit támogatja:

* készletkezelés (új termékek felvétele, meglévők módosítása)
* Események nyomon követése
* statisztikák és forgalmi adatok megtekintése

Az asztali alkalmazást C# nyelven, WinForms keretrendszerben fejlesztettük, amely stabil és egyszerű megoldást biztosít a helyi ügyintézéshez.

Miért ezt a feladatot választottuk?

A választásunk nem véletlen: csapattagként mindketten hosszú évek óta rajongói vagyunk az MTG-nek, így a téma nemcsak szakmai kihívást, hanem valódi érdeklődést is jelentett számunkra. Célunk volt egy olyan rendszert készíteni, amely a való életben is működőképes lenne egy MTG-specializált kártyabolt vagy magán személy számára, és amelyet akár tovább is lehetne fejleszteni egy teljes értékű, nyílt kereskedelmi platformmá.

Ezzel a projekttel nemcsak a programozási tudásunkat, hanem a csapatmunkában, verziókezelésben, adatbázis-tervezésben és felhasználói felület kialakításban szerzett ismereteinket is gyakorlatba ültethettük. Fontos volt számunkra, hogy olyan dokumentáció készüljön hozzá, amely világosan bemutatja a működést, és minden szinten támogatja a felhasználókat a rendszer megismerésében.

## A felhasznált ismeretek

A projekt elkészítése során több olyan tantárgyi és gyakorlati ismeretre is szükség volt, amelyeket az iskolai tanulmányok, valamint önálló tanulás során szereztünk meg. A program elkészítésében fontos szerepet játszott az **asztali alkalmazás fejlesztése**, a **backend és frontend programozás**, illetve az **adatbázis-kezelés** ismeretei. Ezek közül több témával már találkoztunk a korábbi gyakorlati feladatok során is.

A projekt technológiai megvalósításához használt nyelvek és keretrendszerek ismerete elengedhetetlen volt. Ezek közé tartozik:

* **C#** – az asztali alkalmazás elkészítéséhez
* **JavaScript, HTML, CSS** – a webes felülethez
* **React** – a webes frontend modern, komponens-alapú kialakításához
* **Node.js és Express** – a szerveroldali működéshez és REST API-k létrehozásához
* **MySQL** – az adatbázis kezeléséhez, a kapcsolatok és lekérdezések kialakításához
* **Bootstrap** – a reszponzív, letisztult megjelenés megvalósításához

Ezeket az ismereteket **részben az iskolai oktatás során**, részben pedig **különböző online forrásokon** keresztül sajátítottuk el, például dokumentációk, oktatóvideók és fejlesztői fórumok segítségével.

A fejlesztés során **önállóan sajátítottuk el** a **GitHub** és **Git** használatát, melyek nélkülözhetetlenek voltak a közös munka során. Továbbá fontos volt a **szakmai dokumentáció elkészítése**, amelyben szintén fejlődtünk a projekt alatt.

Külön figyelmet kellett fordítani az **Express** keretrendszer és a **MySQL** adatbázis-motor működésének mélyebb megértésére is, mivel ezek biztosították a backend és az adatkezelés alapját.

## A felhasznált szoftverek

A fejlesztés során többféle szoftverkörnyezetet és eszközt alkalmaztunk annak érdekében, hogy hatékonyan tudjunk dolgozni a projekt különböző részein. A rendszer felépítése miatt mind **webes**, mind **asztali környezetekben** dolgoztunk, így szükség volt több platform és technológia összehangolt használatára.

A projekt fejlesztése **elsősorban Windows operációs rendszeren** zajlott, azonban előfordult, hogy nem állt rendelkezésre PC-s környezet, így bizonyos backend munkákat **MacOS rendszeren** végeztünk el. Ez különösen hasznos volt a platformfüggetlen fejlesztési szemlélet kialakításában.

A konkrét fejlesztői környezetek és eszközök a következők voltak:

* **Visual Studio (C# WinForms)** – az asztali alkalmazás fejlesztéséhez
* **Visual Studio Code** – a webes alkalmazás frontend és backend részének fejlesztéséhez (React, Node.js, Express)
* **XAMPP** – a helyi szerver környezet biztosításához és a MySQL adatbázis futtatásához
* **phpMyAdmin** – az adatbázis kezelőfelületének használatához, amely segítséget nyújtott a táblák, kapcsolatok és lekérdezések vizuális kezelésében

Az adatbázis-kezeléshez **MySQL-t** használtunk, amely a **XAMPP környezet részeként** biztosította a szükséges szerveres háttért. Az adatbázist phpMyAdmin segítségével adminisztráltuk, így gyorsan és átláthatóan tudtuk módosítani a szerkezetet, illetve tesztelni a lekérdezéseket.

A projekt verziókezeléséhez a **GitHub** szolgáltatását vettük igénybe. A Git-es munkafolyamat lehetővé tette, hogy a csapat minden tagja nyomon kövesse a változtatásokat, hibajavításokat és fejlesztéseket. A GitHub lehetőséget biztosított a párhuzamos munkavégzésre is, és több mint **150 commit** készült a fejlesztés során, ami jól mutatja az aktív csapatmunkát.

Ezek a szoftverek együttesen biztosították, hogy a fejlesztés folyamatos, stabil és jól szervezett legyen, valamint lehetőséget adtak a rendszer működésének gyors tesztelésére és finomhangolására.

# Felhasználói dokumentáció

## A program általános specifikációja

Az MTG CardShop egy többplatformos alkalmazás, amely a Magic: The Gathering (MTG) kártyajáték tematikájára épülő kártyabolt működését digitalizálja. A program célja, hogy modern, felhasználóbarát felületet biztosítson a kártyák böngészéséhez, vásárlásához és értékesítéséhez, valamint események kezeléséhez.

A rendszer két fő modulból áll:

* **Webes alkalmazás:** Ezen a felületen a felhasználók böngészhetnek az aktuálisan elérhető MTG kártyák között, vásárlásokat bonyolíthatnak le, valamint regisztrálhatnak különböző eseményekre (pl. versenyekre). Emellett naprakész információkat kapnak az újdonságokról, akciókról és a közelgő kiadásokról is.
* **Asztali alkalmazás:** Ez a felület az üzlet működtetéséért felelős adminisztrátorok számára készült. Lehetővé teszi az adatbázisban tárolt termékek kezelését (új kártyák felvétele, meglévők szerkesztése, törlése), az események nyomon követését, valamint a statisztikák és forgalmi adatok megtekintését.

Az alkalmazás több szempontból is törekszik a hatékonyságra és az egyszerű használhatóságra: átlátható kezelőfelülettel, naprakész adatbázissal és stabil háttértechnológiával támogatja a felhasználói élményt, legyen szó vásárlóról vagy rendszergazdáról.

## Rendszerkövetelmények

### Hardver követelmények

Minimális követelmények:

* Processzor: Dual-core 1.8 GHz
* Memória: 4 GB RAM
* Tárhely: Legalább 500 MB szabad hely az alkalmazások és adatbázis számára
* Kijelző: Minimum 1366x768 felbontás
* Internetkapcsolat: Szükséges a webes alkalmazás használatához és adatfrissítésekhez

Ajánlott konfiguráció:

* Processzor: Quad-core 2.5 GHz vagy gyorsabb
* Memória: 8 GB RAM vagy több
* Tárhely: Legalább 1 GB szabad hely
* Kijelző: Full HD (1920x1080) vagy nagyobb felbontás
* Internetkapcsolat: Stabil, szélessávú internetkapcsolat

Az alkalmazás nem igényel különleges hardverelemeket, a legtöbb mai irodai vagy otthoni gépen zökkenőmentesen futtatható.

### Szoftver követelmények

Webes alkalmazás futtatásához:

* Operációs rendszer: Windows 10/11, MacOS vagy bármilyen Linux disztribúció
* Böngésző: Legfrissebb verziójú Google Chrome, Mozilla Firefox, Microsoft Edge vagy Safari
* Frontend futtatásához: React kompatibilis környezet
* Backend kiszolgálásához:
* Node.js (14-es verzió vagy újabb)
* Express.js keretrendszer
* MySQL szerver (XAMPP csomagon belül ajánlott)

Asztali alkalmazás futtatásához:

* Operációs rendszer: Windows 10 vagy újabb
* Futtatókörnyezet: .NET Framework (ajánlott: .NET 4.7 vagy újabb)
* Fejlesztőkörnyezet (ha módosítani akarjuk a programot): Microsoft Visual Studio

Adatbázis-kezeléshez:

* MySQL
* phpMyAdmin (XAMPP csomagon belül)

## 3. A program telepítése

### Üzembe helyezés:

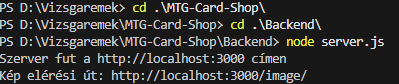
* A weboldalon lévő adatok betöltéséhez és az asztali alkalmazáshoz is szükséges egy webszer, ami lehet például a XAMPP webszerver és MySQL kiszolgáló.
* Ha a XAMPP-ot telepítettük akkor el kell indítani az Apache és a MySQL szolgáltatásokat.
* A XAMP működéséhez be kell importálni „cardshop.sql” fájlt az adatbázisba. Tartalmazza a teljes adatbázist minden beállításával együtt.

A fájlt a [https://github.com/AsuraMTG/MTG-Card-Shop](https://github.com/AsuraMTG/MTG-Card-Shop“) linken lehet elérni a “Database” nevű mappában.

A további fájlokat is ezen a linken található mappákból kell elérhetővé tenni.

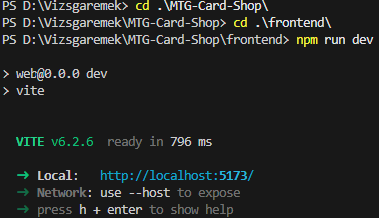
### Adatbázis és szerver telepítése:

* Futtassa az alábbi parancsokat egy parancssor ablakban, ha már a fájlok elérhetőek a “Backend” mappából.
  + *cd MTG-Card-Shop*
  + *cd Backend*
* Hozzon létre egy .env fájlt az alábbi minta alapján. Tegye bele a “Backend” mappába azt.
  + DB\_HOST=localhost
  + DB\_PORT=3306
  + DB\_USER=root
  + DB\_PASSWORD=
  + DATABASE=cardshop
* Futtassa le a következő parancsokat a “Backend” mappa termináljában a szerver futtatásához
  + npm i
  + node server.js
* Miután sikeresen elindult, a terminálban látható, hogy a szerver fut.



### Web alkalmazás:

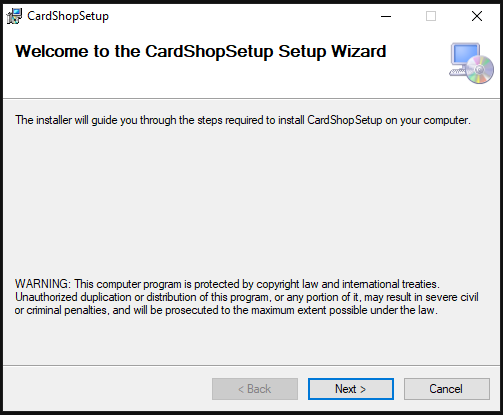
* Futtassa az alábbi parancsokat egy parancssor ablakban, ha a fájlok már elérhetőek a “frontend” mappából
  + *cd MTG-Card-Shop*
  + *cd frontend*
  + *npm i*
  + *npm run dev*
* Miután elindult egy localhost linket kell, hogy kapjunk, amin a weboldal fut.



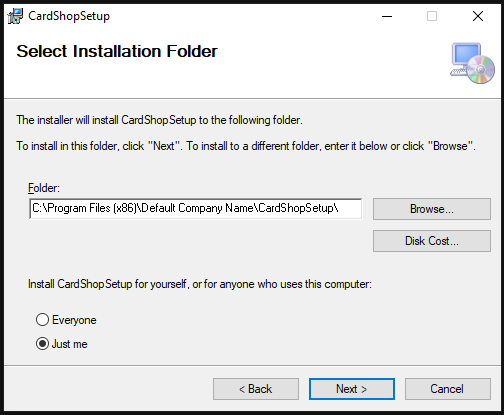
### Asztali alkalmazás:

Az MTG-Card-Shop\Desktop\CardShopSetup\Release  
 A mappán belül megtalálod a CardShopSetup.msi fájlt.

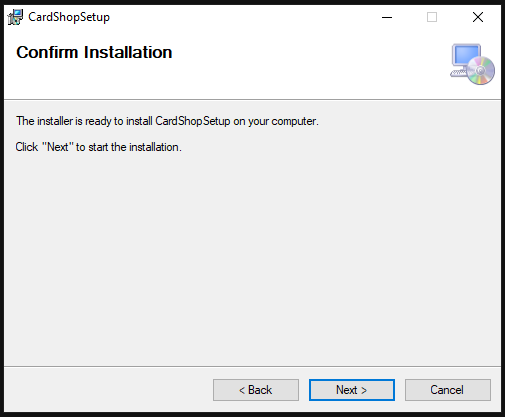
Ezt a fájlt futtatva elindul a telepítő.

1. Ablak 

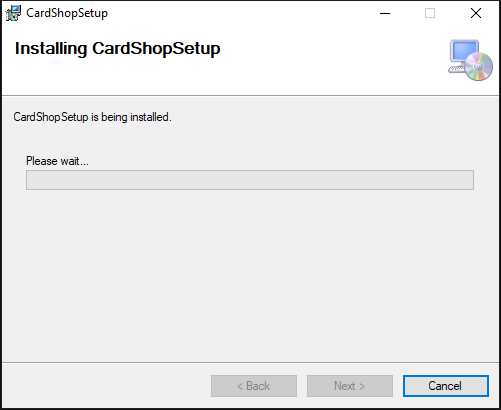
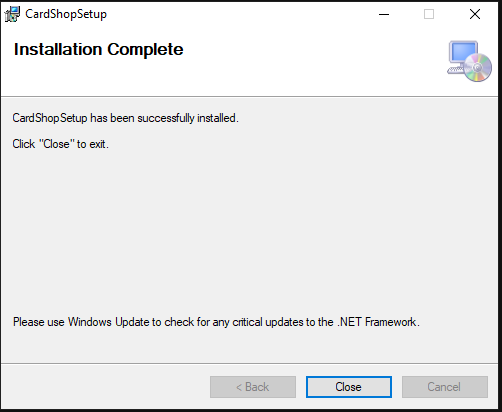
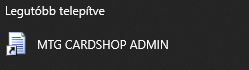
A felhasználó a "Next" gombra kattint, ami a következő részre viszi.

1. Ablak 

A felhasználó itt a „Browse” gombra kattintva kiválaszthatja, hova szeretné telepíteni az alkalmazást, ha az ajánlott útvonal nem felel meg neki. Ezutan nyomj a „Next” gombra.

1. Ablak

A telepítő készen áll a telepítésre, és a „Next” gombra kattintva el is indul a folyamat.

1. Ablak  Rövid várakozás után az alkalmazás telepítése befejeződik, és az utolsó ablak jelenik meg.
2. Ablak  Végül a „Close” gombra kattintva a telepítés befejeződik.
3. Az alkalmazást a számítógépen *MTG CARDSHOP ADMIN* néven találja meg. 

## A program használatának a részletes leírása

# MTG Card Shop – Webes alkalmazás Felhasználói dokumentáció

**Platform**: Windows (React JavaScript, CSS, HTML)  
**Nyelv**: Angol, Magyar  
**Célközönség**: Kártyázók, gyűjtők

## 1. Bevezetés

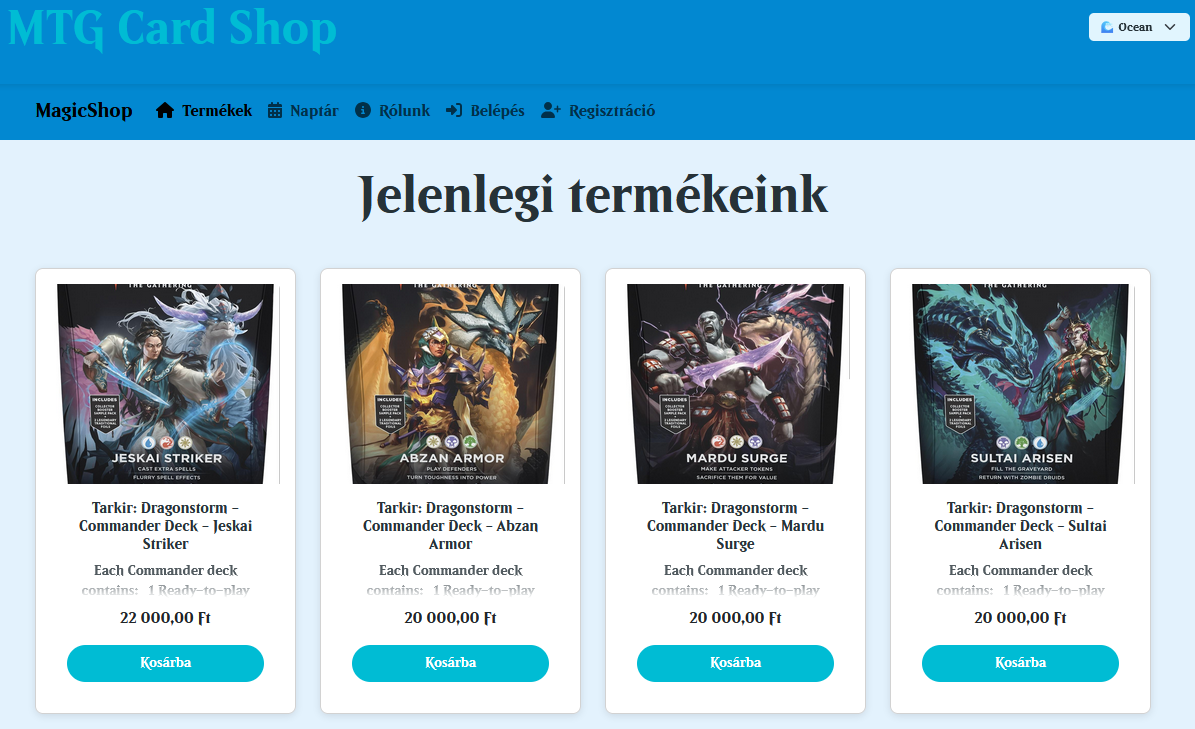
Ez a kézikönyv a *MTG Card Shop* nevű web alkalmazás használatához készült. A szoftver célja, hogy egy adott személy kártyagyűjteményét nyilvánosságra hozza olyanok számára, akik érdekeltek lehetnek benne.

A dokumentáció célja, hogy lépésről lépésre bemutassa a szoftver használatát, laikus felhasználók számára is egyértelmű módon.

## 2. Program működése

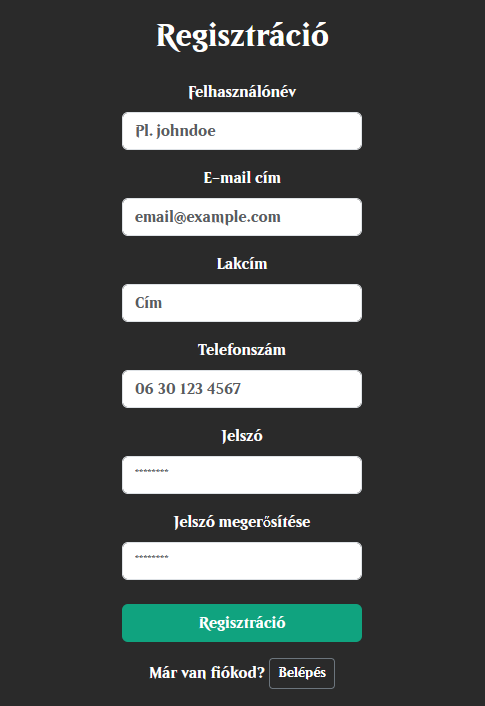
### 2.1 Termékek oldal

* Az oldal megnyitása után a “termékek” oldal nyílik meg. A jobb fölső sarokban lévő menü-vel lehet változtatni az oldal kinézetét.



* A felhasználó a “Kosárba” gomb kattintásával tudja a kosárba helyezni a termékeket, de csakis akkor kerül a nevéhez és láthatja a terméket, hogyha regisztrál és/vagy bejelentkezik egy felhasználó névvel.

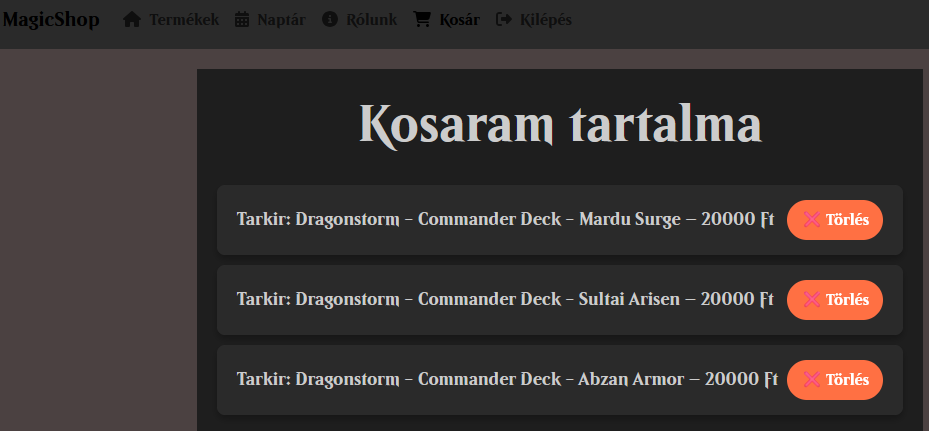
### 2.2 Bejelentkezés és regisztráció:



* A navigációs sávon kattintson a “Regisztráció” menüpontra. Adja meg a következőd adatokat:
  + Felhasználónév
  + E-mail cím
  + Lakcím
  + Telefonszám
  + Jelszó
  + Jelszó újra
  + Kattintson a “Regisztráció” gombra
* Hogyha a látogatónak már van fiókja a “Belépés” gombbal már át is lesz irányítva a bejelentkező felületre.
* Sikeres regisztráció után a “Belépés” oldalra irányítjuk a felhasználót, ahol az előbb megadott Felhasználónévvel és jelszóval tud belépni.

### 2.3 Kosár

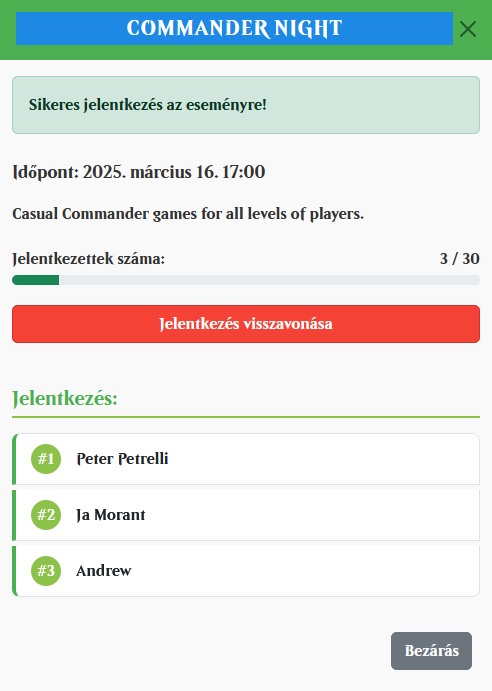
* Sikeres bejelentkezés követően a navigációs sáv megváltozik és megjelenik a “Kosár”, illetve “Kilépés” menüpont, hogyha a felhasználó befejezni kívánja a böngészést.



* A törlés gomb megnyomásával eltávolíthatja a kívánt terméket a kosár listájából a felhasználó.

### 2.4 Naptár

* A naptár oldalon az összes év minden napját látni lehet hónapokra felosztva.
* Válasszon egy napot, amin van esemény
  + Az eseményre kattintva megjelennek az esemény részletei és az eddig regisztrált jelentkezők név szerint.
* Hogyha a felhasználó “belépett” akkor lehetősége van jelentkezni a “Jelentkezés” gombbal bármilyen eseményen, így mindenki tudja, hogy hány emberrel számoljon és hogy mennyi szabad hely van még.
* Hogyha a felhasználó meggondolja magát vissza tudja vonni a jelentkezését a későbbiekben a “Jelentkezés visszavonása” gombbal.



# **MTG Card Shop – Asztali alkalmazás Felhasználói dokumentáció**

### **Adminisztrációs rendszer**

**Platform**: Windows (C# WinForms)  
**Verzió**: 1.0  
**Nyelv**: Angol  
**Célközönség**: Üzletvezetők, rendszergazdák

## **1. Bevezetés**

Ez a kézikönyv a *MTG Card Shop* nevű Windows-alapú adminisztrációs rendszer használatához készült. A szoftver célja, hogy lehetőséget biztosítson az üzletvezetők vagy rendszergazdák számára események, felhasználók és termékek kezelésére. A program egy grafikus felhasználói felületen (GUI) keresztül működik, és telepíthető Windows rendszeren.

A dokumentáció célja, hogy lépésről lépésre bemutassa a szoftver használatát, laikus felhasználók számára is egyértelmű módon. Minden funkció, amely a programban elérhető, szerepel ebben a kézikönyvben.

## **2. A program indítása**

A programot a telepítés után az alábbi módon indíthatjuk el:

1. Nyissa meg a **Start menüt**.
2. Keresse meg az alkalmazások között a **„MTG CARDSHOP ADMIN** nevet.
3. Kattintson a program ikonjára a megnyitáshoz.

A program első megnyitásakor automatikusan a Events jelenik meg, amelyből az alábbi három fő funkció érhető el:

* **Events** (Események kezelése) itt kezd a felhasználó
* **Customers** (Felhasználók kezelése)
* **Products** (Termékek kezelése)

A főmenü jobb oldalán található egy **Light/Dark Mode** váltó gomb, amely lehetővé teszi a világos vagy sötét megjelenés kiválasztását.

## **3. Események kezelése (Events)**

Az *Events* menüpont alatt eseményeket hozhatunk létre, módosíthatunk vagy törölhetünk. Minden esemény az alábbi mezőkből áll:

* **Name** – Az esemény neve (szöveges mező).
* **Date** – Az esemény időpontja (dátumválasztó használatával).
* **Max participants** – A résztvevők maximális száma.
* **Current participants** – Jelenleg regisztrált résztvevők száma. (ez nem változtatható)
* **Description** – Rövid leírás az eseményről.

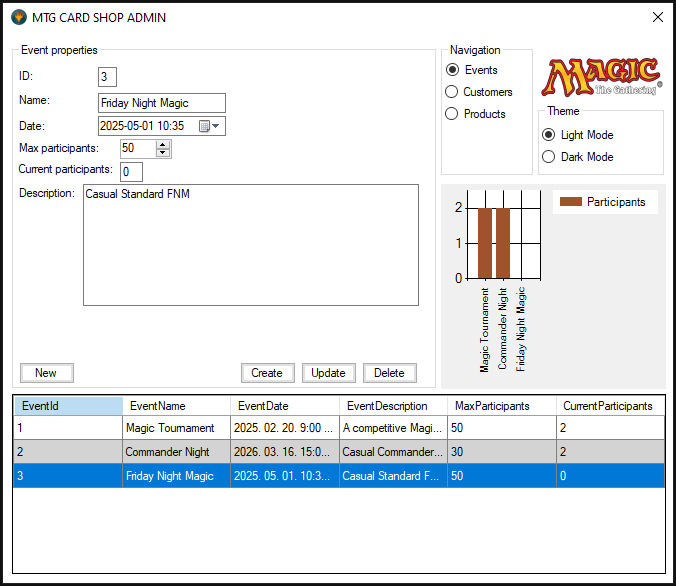
### **3.1 Új esemény hozzáadása**

1. Nyissa meg az *Events* szekciót.
2. Töltse ki a fenti mezőket.
3. Kattintson a **„Create”** gombra.

### **3.2 Létező esemény módosítása vagy törlése**

1. Válassza ki a listából a módosítani kívánt eseményt.
2. Végezze el a szükséges változtatásokat.
3. Kattintson a **„Update”** gombra a mentéshez.

Ha törölni szeretne egy eseményt:

1. Jelölje ki a listából.
2. Kattintson a **„Delete”** gombra.

## **4. Felhasználók kezelése (Customers)**

A *Customers* menüpont lehetővé teszi regisztrált felhasználók adatainak megtekintését, módosítását és törlését.

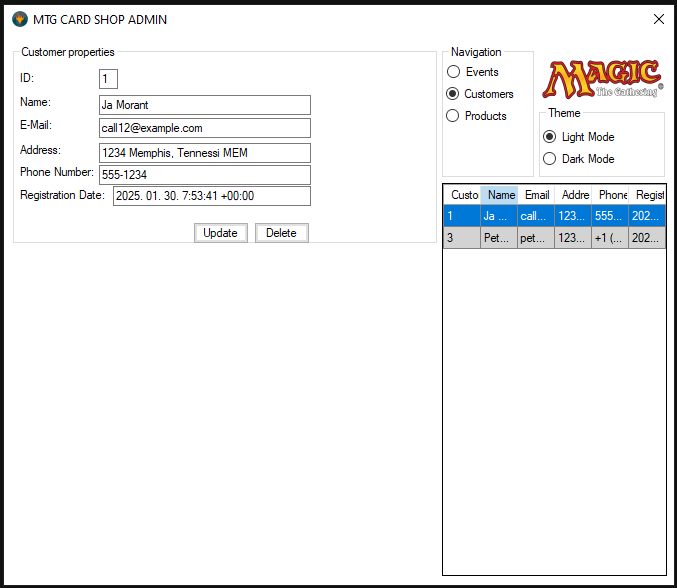
### **4.1 Felhasználó kiválasztása**

1. Válasszon ki egy felhasználót a listából.
2. Az adatok megjelennek az alábbi mezőkben:
   1. **Name**
   2. **E-mail**
   3. **Address**
   4. **Phone number**

### **4.2 Felhasználói adatok módosítása**

1. Végezze el a kívánt módosításokat a megjelenő mezőkben.
2. Kattintson a **„Update”** gombra.

### **4.3 Felhasználó törlése**

1. Jelölje ki a törölni kívánt felhasználót a listában.
2. Kattintson a **„Delete”** gombra. 

## **5. Termékek kezelése (Products)**

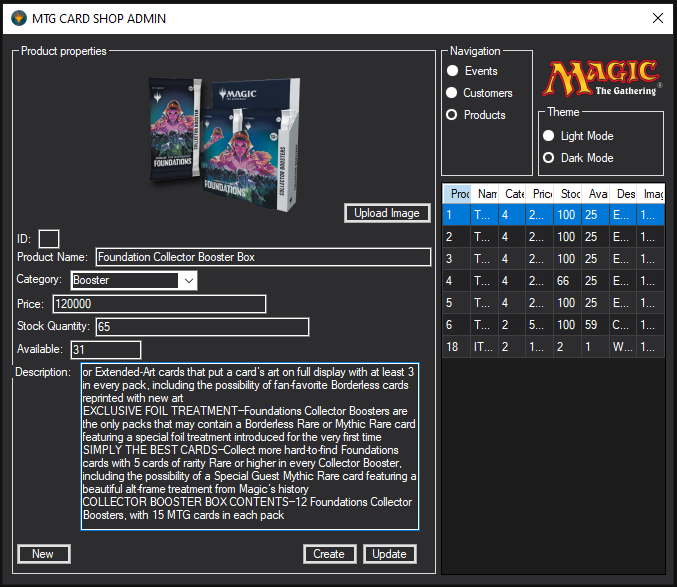
A *Products* szekcióban új termékeket vehetünk fel, illetve meglévő termékeket módosíthatunk.

**Fontos:** Terméket törölni nem lehet.

### **5.1 Új termék hozzáadása**

1. Kattintson az **„New”** gombra.
2. Töltse ki az alábbi mezőket:
   1. **Product name**
   2. **Category** *(választható lista - ComboBox)*
   3. **Price**
   4. **Stock quantity**
   5. **Available** *(jelölőnégyzet)*
   6. **Description**
3. Töltsön fel egy képet a termékről a **„Upload Image”** gombbal.
4. Kattintson a **„Create”**gombra.

### **5.2 Meglévő termék módosítása**

1. Válassza ki a terméket a listából.
2. Módosítsa a szükséges mezőket.
3. Kattintson a **„Update”** gombra. 

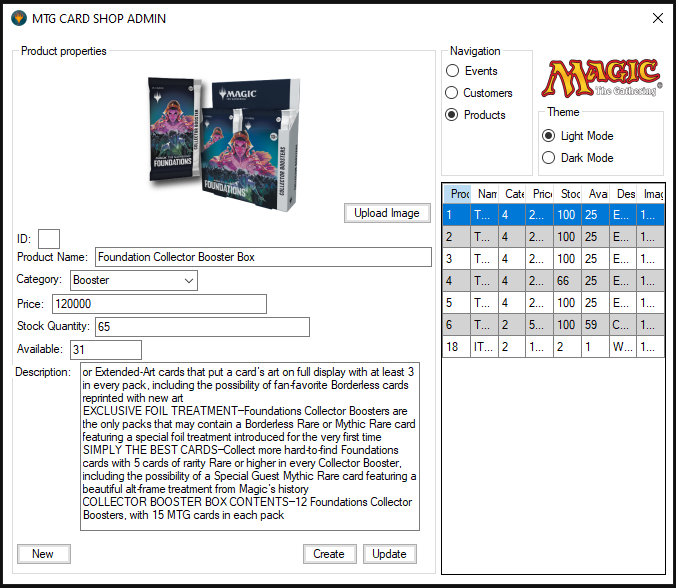
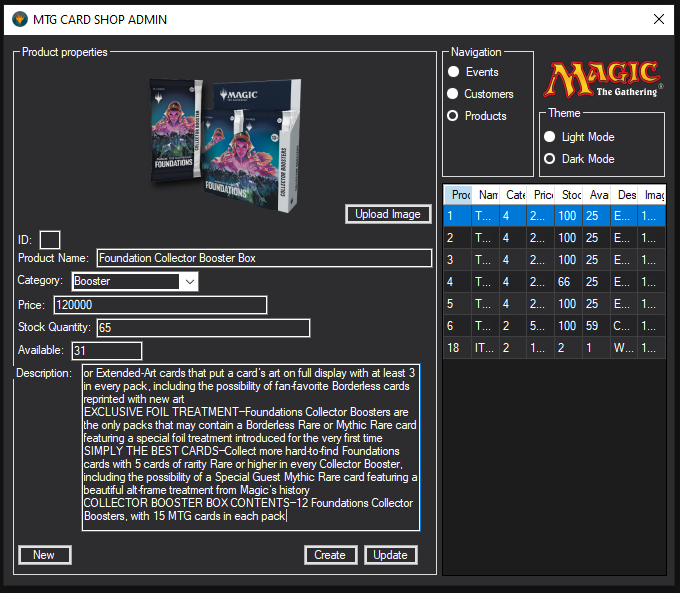
## **6. Megjelenési módok: Light és Dark Mode**

A *MTG Card Shop* alkalmazás lehetőséget biztosít a felhasználói felület megjelenésének testreszabására két színséma között:

* **Light Mode (Világos mód)** – Alapértelmezett világos színek, amelyek nappali fényviszonyok között jól olvashatók.
* **Dark Mode (Sötét mód)** – Sötét háttér világos betűszínnel, ami kíméli a szemet és ideális sötétebb környezetben való használathoz.

### **6.1 Megjelenési mód váltása**

1. A program minden főablakának jobb oldalan található egy gomb a következő címkével:
   1. „**Light Mode**” vagy „**Dark Mode**” – attól függően, hogy melyik mód van éppen bekapcsolva.
2. Kattintson a gombra a módosításhoz.
3. A felület azonnal átvált a másik színsémára, nincs szükség újraindításra.



## **7. Adatkezelés és validálás**

A program minden adatbevitel során ellenőrzi a mezők helyes kitöltését, hogy megelőzze a hibás vagy hiányos adatok rögzítését.

### **7.1 Kötelező mezők**

Az alábbi mezők minden esetben kötelezőek:

#### **Events:**

* Name
* Date
* Max participants

#### **Customers:**

* Name
* Email (érvényes e-mail formátum szükséges)
* Phone number (csak számokat fogad el)

#### **Products:**

* Product name
* Category
* Price (pozitív szám)
* Stock quantity (egész szám)

### **7.2 Hibajelzések**

Ha bármely kötelező mező üres, vagy hibás értéket tartalmaz, a program figyelmeztető üzenetet jelenít meg, például:

* „Please fill out all required fields.”
* „Invalid email address.”
* „Price must be a positive number.”

A hibák javítása után ismételten el kell végezni a mentést.

## **8. Mentés és frissítés**

Minden módosítás, amelyet a felhasználó végez (új esemény, felhasználó vagy termék létrehozása, meglévő adatok módosítása), azonnal elmentésre kerül az adatbázisba, amint a mentés vagy frissítés gombra kattint.

* **Sikeres mentés után** a program egy visszajelző üzenetet jelenít meg: „Sikeres feltöltés.”
* **Mentés sikertelensége esetén** a rendszer hibaüzenetet ad, például: „Hiba történt.”

## **9. Telepítési tudnivalók**

A program telepíthető önálló .msi telepítőfájlból, amely már tartalmazza a szükséges fájlokat és könyvtárakat. A telepítés lépései:

1. Nyissa meg a telepítőfájlt: [CardShopSetup.msi](https://github.com/AsuraMTG/MTG-Card-Shop/blob/main/Desktop/CardShopSetup/Release/CardShopSetup.msi).
2. Kövesse a telepítő utasításait.

A program csak **Windows operációs rendszeren** futtatható.

## **10. Hibakezelés és támogatás**

Bár a program stabil működésre készült, előfordulhatnak kisebb hibák. Az alábbi hibák gyakoriak lehetnek:

|  |  |  |
| --- | --- | --- |
| **Hiba típusa** | **Leírás** | **Teendő** |
| Hibás adatbevitel | Helytelen formátum vagy hiányzó mező | Ellenőrizze az összes kitöltött mezőt |
| Nincs kiválasztva elem a listából | Frissítés vagy törlés próbálkozása esetén | Előbb jelölje ki az elemet a listából |
| Kép nem töltődik fel | Nem támogatott fájltípus vagy nagy méret | Használjon .jpg vagy .png fájlokat |

# Fejlesztői dokumentáció

A fejlesztői dokumentáció célja, hogy a segítse program logikájának, illetve a program kódjának a megértését, illetve a program továbbfejlesztését. Szakemberek számára készül, elvárás tehát a pontosság és a szakmai jellegű stílus.

## Az alkalmazott fejlesztői eszközök

A projekt során több különböző fejlesztői eszközt és környezetet alkalmaztunk a program elkészítéséhez, illetve a dokumentáció kidolgozásához. A választott eszközök és nyelvek mind ingyenesen elérhetőek, nyílt forráskódúak vagy oktatási célra jogszerűen használhatók.

Programozási nyelvek:

* C# – A WinForms-alapú asztali alkalmazás megvalósításához
* JavaScript – A webes alkalmazás dinamikus működéséhez
* HTML és CSS – A felhasználói felület szerkezetének és stílusának kialakításához
* SQL – Az adatbázis műveletekhez (lekérdezések, adatmanipulációk)

Fejlesztői környezetek és platformok:

* Visual Studio – A C# alapú asztali alkalmazás fejlesztésére
* Visual Studio Code – A frontend és backend fejlesztéshez (React, Node.js, Express)
* XAMPP – A MySQL adatbázis és a phpMyAdmin kezelőfelület helyi futtatására
* phpMyAdmin – Az adatbázis grafikus kezelőfelülete

Backend és adatbázis:

* Node.js + Express.js – A REST API-k fejlesztéséhez és szerveroldali logika megvalósításához
* MySQL – Az alkalmazás adatainak tárolásához és kezeléséhez

Frontend:

* React – A webes felület komponensalapú felépítéséhez
* Bootstrap – A reszponzív megjelenés biztosításához, kezdetleges kinézethez.
* Styled-components - Több irányból megközelített kinézet miatt
* Axios – Http kérések a szerver felé
* Figma – kezdetleges kinézet és drótvázhoz

Verziókezelés és csapatmunka:

* Git + GitHub – Verziókezelés és kollaborációs feladatokhoz.
* Microsoft Teams + Discord – Kommunikációs és projektkoordinációs eszközök

Dokumentáció készítéséhez használt szoftverek:

* Microsoft Word / Google Docs – A dokumentáció megírására és szerkesztésére
* Greenshot / Lightshot – A képernyőképek készítéséhez, amelyek a program működését illusztrálják
* Paint – Egyszerű képszerkesztési feladatokhoz

Külső modulok és csomagok:

* MySQL2 (npm csomag) – A Node.js és MySQL közötti kapcsolat biztosítására
* React Router – A frontend oldali navigáció kezeléséhez

A használt külső könyvtárak és csomagok mind nyílt forráskódúak, és a hivatalos npm registry-n keresztül letölthetők, így jogszerűen alkalmazhatók oktatási célú projektekben.

## Adatmodell leírása

A projekt során egy relációs adatbázis került kialakításra, amely a rendszer működéséhez szükséges összes entitást és azok kapcsolatát tartalmazza. Az adatbázis neve: cardshop. Az adatbázis célja, hogy hatékonyan és biztonságosan tárolja a felhasználói, termék-, esemény- és vásárlási adatokat.

Az adattáblák és funkcióik

customers  
 Ez a tábla tárolja a regisztrált felhasználók adatait. Minden felhasználó rendelkezik egyedi customer\_id-val, valamint olyan mezőkkel, mint name, email, password, address és registration\_date.

products  
 A bolt kínálatában szereplő termékeket tartalmazza. Minden termék egyedi product\_id-val azonosítható, és tartalmazza a name, price, description, stock\_quantity valamint a category\_id mezőket, amely utóbbi a termék típusát határozza meg.

categories  
 A termékkategóriák (pl. booster, display, commander deck stb.) itt kerülnek tárolásra. Az egyedi azonosító a category\_id, emellett tartalmaz category\_name és category\_description mezőket is.

orders  
 A felhasználók által leadott rendelések adatait tárolja. A customer\_id mező jelöli, hogy ki adta le a rendelést. Az order\_date, total\_amount és status mezők tartalmazzák a vásárlás időpontját, összegét és állapotát.

order\_items  
 Ez a tábla a rendelések részleteit tartalmazza, vagyis, hogy egy adott rendelés során milyen termékek kerültek a kosárba. Az order\_id és product\_id mezők kapcsolják össze a többi táblával, a quantity és price\_at\_order a rendeléskori mennyiséget és árat tárolják.

events  
 Az MTG eseményeket tartalmazza, amelyeken a felhasználók részt vehetnek. Minden esemény egyedi event\_id-val azonosítható, tartalmaz nevet, leírást, időpontot és részvételi adatokat (max\_participants, current\_participants).

registrations  
 Ez a tábla a felhasználók eseményekre történő regisztrációit rögzíti. Az event\_id és customer\_id mezők együttesen jelzik, ki melyik eseményre jelentkezett. A registration\_date tárolja a jelentkezés időpontját.

A táblák közötti kapcsolatok

Az adatbázisban több idegen kulcs (foreign key) kapcsolat is létrejött:

A customers tábla kapcsolódik az orders és registrations táblákhoz.

Az orders tábla kapcsolódik az order\_items táblához.

Az order\_items tábla kapcsolódik a products táblához.

A products tábla kapcsolódik a categories táblához.

A registrations tábla kapcsolódik az events táblához.

Ez a kapcsolati struktúra biztosítja az adatok logikus szervezését, és lehetővé teszi a komplex lekérdezések egyszerű és hatékony végrehajtását (pl. egy adott felhasználó megrendeléseinek listázása vagy eseményre jelentkezők száma).

Adatkonzisztencia és adatintegritás

Az idegen kulcsos kapcsolatok biztosítják az adatintegritást és a hivatkozások érvényességét. Például, ha egy termék törlésre kerülne, és az szerepel egy rendelésben, a rendszer hibaüzenetet adna vissza, vagy csak a kapcsolt adatok törlésével együtt engedélyezné a műveletet.

Az adatbázisban található trigger is, amely automatikusan frissíti az események résztvevőinek számát, ha új regisztráció történik. Ez különösen fontos az eseményeknél a maximális létszám betartása érdekében.

## Részletes feladatspecifikáció, algoritmusok

## Webes Alkalmazás:

## Web alkalmazás specifikációk:

A weboldal megnyitásakor megjelenik egy navigációs sáv, egy stílus változtató menü, az összes termék, amit az admin felületen hozzá adtunk a listához és egy lábléc.

ezek láthatóak minden egyes oldalon, de a navigációnál meg van különböztetve, hogy bejelentkezés nélkül mihez lehet hozzáférni.

### Navigációs sáv (bejelentkezés nélkül):

* MagicShop felirat (vissza a termékekhez)
* Termékek oldal --> hozzá lehet adni a kosárhoz, de a felhasználó nem látja
* Naptár oldal
* Rólunk oldal
* Regisztráció oldal
* Bejelentkezés oldal

### Navigációs sáv (bejelentkezés után):

* MagicShop felirat (vissza a termékekhez)
* Termékek oldal
* Naptár oldal
* Rólunk oldal
* Kosár oldal (itt látszik, hogy milyen termékeket adtunk hozzá a listához)
* Kijelentkezés (Visszatér a bejelentkezés előtti állapotba)

### Termékek oldal tartalma:

* Termék képe
  + Ha a felhasználó rákattint az egyik képre akkor annak a terméknek az egyedi oldalára kerül.
* Termék leírás
* Kosárba gomb

#### **Termék egyedi oldala:**

* Termék képe (nagyban)
* Termék leírás
* Kosárba gomb

### Naptár oldal tartalma:

* Mai nap gomb
  + Ha a felhasználó rákattint erre a gombra akkor a naptár azt az évet, hónapot fogja mutatni a címében. A mai napot pedig észrevehetővé teszi
* Hónap előre / hátra gombok
  + A naptár szélein az egyik [<] gomb a hónapokat időrendi sorrendben visszafelé látjuk, a [>] gombbal pedig időrendi sorrendben előre.
* Esemény gomb
  + Azok az események, amik fel lettek töltve az admin által a naptárban azon a napon jelennek meg amikor feltöltik. Gombnyomásra az Esemény oldalra kerül a felhasználó.

### Esemény oldal tartalma:

#### **Bejelentkezéssel:**

* Jelentkezés gomb
  + A felhasználó, ha jelentkezik ott lesz a neve az esemény mellett az adatbázisban is.
* Jelentkezés visszavonása gomb
  + Ha már jelentkezett az adott eseményre egy felhasználó, akkor lehetősége van lemondani a jelentkezését.
* Bezárás gomb (vissza a naptárba)

#### **Bejelentkezés nélkül:**

A felhasználó csak látja az eddigi jelentkezőket

* Bezárás gomb (vissza a naptárba)

### Kosár oldal tartalma:

* Termék törlése gomb [x]
  + Kitörli a terméket a kosár listájából, ha a felhasználó mégsem szeretné azt az adott terméket.

### **Rólunk oldal tartalma**

* Kép változás
  + A kép megnyomására egy másik kép veszi át a helyét
* moxfield.com logo gomb

## Asztali Alkalmazás:

## **1. Program függvényeinek és osztályainak metódusainak specifikációja**

### **1.1. Customer (Vevő) funkciók**

#### **1.1.1. getCustomers()**

* **Leírás**: Ez a metódus lekéri az összes vevő adatokat az API-ból egy HttpClient segítségével.
* **Paraméterek**: Nincsenek paraméterek.
* **Visszatérési érték**: Task (Aszinkron metódus).
* **Működés**:
  + HTTP GET kérés küldése a customersBaseURL URL-re.
  + Ha sikeres a válasz (IsSuccessStatusCode), akkor a választ JSON formátumban kiolvassa és deszerializálja a Customer típusú objektummá.

#### **1.1.2. dataGridViewCustomers\_CellClick()**

* **Leírás**: Ezt a metódust akkor hívjuk meg, amikor egy cellát választanak ki a DataGridView-ben. A vevő adatai automatikusan betöltődnek a megfelelő mezőkbe.
* **Paraméterek**:
  + sender: Az eseményt kiváltó objektum (a DataGridView).
  + e: Az esemény argumentumai, amelyek tartalmazzák a kiválasztott cella indexét.
* **Visszatérési érték**: Nincs visszatérési érték.
* **Működés**:
  + Kiválasztott sor alapján az összes fontos adat (pl. név, cím, email, telefonszám) betöltése a megfelelő szövegdobozokba.

#### **1.1.3. buttonCustomerUpdate\_Click()**

* **Leírás**: Ez a metódus a vevő adatainak frissítésére szolgál. A felhasználó által megadott adatokat frissíti az adatbázisban.
* **Paraméterek**:
  + Nincsenek paraméterek.
* **Visszatérési érték**: Nincs visszatérési érték.
* **Működés**:
  + Az adatokat kiolvassa a szövegdobozokból, majd a felhasználótól megerősítést kér, hogy biztosan frissíteni szeretné-e az adatokat.
  + Ha a felhasználó megerősíti, akkor az updateCustomer() metódus kerül meghívásra.

#### **1.1.4. updateCustomer()**

* **Leírás**: Frissíti a vevő adatokat az API-ban HTTP PUT kérés segítségével.
* **Paraméterek**:
  + id: A vevő azonosítója.
  + name: A vevő neve.
  + email: A vevő email címe.
  + address: A vevő lakcíme.
  + phoneNumber: A vevő telefonszáma.
* **Visszatérési érték**: Task (Aszinkron metódus).
* **Működés**:
  + A frissítendő adatokat JSON formátumban küldi el a szervernek.
  + Ha a frissítés sikeres, frissíti a vevők listáját és az adatokat a felületen.

#### **1.1.5. buttonCustomerDelete\_Click()**

* **Leírás**: Ez a metódus lehetővé teszi a felhasználó számára, hogy töröljön egy vevőt az adatbázisból.
* **Paraméterek**:
  + Nincsenek paraméterek.
* **Visszatérési érték**: Nincs visszatérési érték.
* **Működés**:
  + A vevő azonosítóját ellenőrzi, és ha érvényes, akkor törlésre kerül az deleteCustomer() metódussal.

#### **1.1.6. deleteCustomer()**

* **Leírás**: Törli a vevőt az API-ból HTTP DELETE kérés segítségével.
* **Paraméterek**:
  + id: A törlendő vevő azonosítója.
* **Visszatérési érték**: Task (Aszinkron metódus).
* **Működés**:
  + Törlés történik a megadott id alapján.
  + Ha sikeres, akkor frissíti a vevői listát és kiüríti a beviteli mezőket.

### **1.2. Product (Termék) funkciók**

#### **1.2.1. getProducts()**

* **Leírás**: Ez a metódus lekéri az összes terméket az API-ból.
* **Paraméterek**: Nincsenek paraméterek.
* **Visszatérési érték**: Task (Aszinkron metódus).
* **Működés**:
  + HTTP GET kérés küldése a productsBaseURL URL-re.
  + A válaszban kapott JSON adatokat deszerializálja és a products listába tárolja.

#### **1.2.2. dataGridViewProducts\_CellClick()**

* **Leírás**: A kiválasztott termék adatainak megjelenítése a szövegdobozokban és a kép megjelenítése.
* **Paraméterek**:
  + sender: Az eseményt kiváltó objektum.
  + e: Az esemény argumentumai.
* **Visszatérési érték**: Nincs visszatérési érték.
* **Működés**:
  + A kiválasztott termék adatai megjelennek a megfelelő szövegdobozokban, és a termék képe betöltődik a PictureBox-ba.

#### **1.2.3. buttonProductCreate\_Click()**

* **Leírás**: Új termék létrehozása.
* **Paraméterek**:
  + Nincsenek paraméterek.
* **Visszatérési érték**: Task (Aszinkron metódus).
* **Működés**:
  + Az új termék adatait JSON formátumban elküldi a szervernek. Ha van kép, azt is mellékeli.

#### **1.2.4. UpdateProduct()**

* **Leírás**: Frissíti a termék adatait a szerveren.
* **Paraméterek**:
  + id: A termék azonosítója.
  + name: A termék neve.
  + category: A termék kategóriája.
  + price: A termék ára.
  + stock: A termék készlete.
  + available: A termék elérhetősége.
  + description: A termék leírása.
* **Visszatérési érték**: Task (Aszinkron metódus).
* **Működés**:
  + A frissített adatokat JSON formátumban küldi a szervernek, és ha sikeres, frissíti a termékek listáját.

### **1.3. Dark and Light Mode funkciók**

#### **1.3.1. SetDarkMode()**

* **Leírás**: Beállítja a sötét módot az alkalmazás számára. Minden UI elem háttere sötét színű, a szöveg világos színű.
* **Paraméterek**: Nincsenek paraméterek.
* **Visszatérési érték**: Nincs visszatérési érték.
* **Működés**:
  + A form és az összes vezérlő (Button, TextBox, DataGridView, stb.) színeinek módosítása.

#### **1.3.2. SetLightMode()**

* **Leírás**: Beállítja a világos módot az alkalmazás számára. Minden UI elem háttere világos, a szöveg sötét színű.
* **Paraméterek**: Nincsenek paraméterek.
* **Visszatérési érték**: Nincs visszatérési érték.
* **Működés**:
  + A form és az összes vezérlő (Button, TextBox, DataGridView, stb.) színeinek módosítása világos módra.

## **2. Algoritmusok**

### **2.1. Adatok frissítése (updateCustomer)**

Pszeudo-kód:

Bemenet: id, name, email, address, phoneNumber

Ha a felhasználó megerősítette a frissítést:

Készíts JSON objektumot a paraméterekből

Küldj HTTP PUT kérést a szervernek a termék frissítéséhez

Ha sikeres:

Frissítsd a termék listát

Frissítsd a UI-t

Kiüríteni a szövegdobozokat

Ha hiba történt:

Hibaüzenetet jeleníts meg

### 2.2. Új termék létrehozása (buttonProductCreate\_Click)

Pszeudo-kód:

Bemenet: name, category, price, stock, available, description, image

Ha a felhasználó megadta az összes adatot:

Készíts egy MultipartFormDataContent objektumot

Add hozzá a fájlt és a termék adatokat

Küldj HTTP POST kérést a termék létrehozásához

Ha sikeres:

Frissítsd a termék listát

Kiüríteni a szövegdobozokat

Ha hiba történt:

Hibaüzenetet jeleníts meg

## **3. A kód fontosabb részei – Magyarázattal**

### **3.1. Dark Mode beállítása**

A SetDarkMode() metódus beállítja a teljes felhasználói felület (UI) sötét színvilágát. Ez főként esztétikai és használhatósági okokból hasznos, különösen alacsony fényviszonyok között.

Fontosabb részlet:

foreach (Control control in this.Controls)

{

if (control is Button)

{

Button button = control as Button;

button.BackColor = Color.FromArgb(66, 66, 66);

button.ForeColor = Color.White;

}

else if (control is TextBox || control is RichTextBox)

{

control.BackColor = Color.FromArgb(28, 28, 28);

control.ForeColor = Color.White;

}

...

}

#### **Magyarázat:**

A foreach ciklus minden vezérlőelemet (Control) bejár az űrlapon, és típus szerint módosítja a megjelenést (háttér- és betűszín).

### **3.2. Light Mode beállítása**

A SetLightMode() metódus ezzel szemben világos színeket állít be minden vezérlőre.

Fontosabb részlet:

if (control is DataGridView)

{

DataGridView dgv = control as DataGridView;

dgv.BackgroundColor = Color.White;

dgv.DefaultCellStyle.BackColor = Color.White;

dgv.DefaultCellStyle.ForeColor = Color.Black;

...

}

#### **Magyarázat:**

Ez különösen fontos a táblázatok (DataGridView) esetén, mivel ott külön kell kezelni a cellák és fejlécek megjelenését is.

### **3.3. Termék létrehozása képfeltöltéssel**

A buttonProductCreate\_Click() metódus többfunkciós: a termék adatok mellett képet is feltölt a szerverre multipart/form-data formában.

Fontosabb részlet:

using (var formData = new MultipartFormDataContent())

{

var fileContent = new ByteArrayContent(File.ReadAllBytes(filePath));

fileContent.Headers.ContentType = MediaTypeHeaderValue.Parse("image/jpeg");

formData.Add(fileContent, "image", Path.GetFileName(filePath));

formData.Add(new StringContent(name), "name");

...

}

#### **Magyarázat:**

A MultipartFormDataContent osztály lehetővé teszi bináris fájl (pl. kép) és szöveges mezők egyidejű elküldését a szervernek.

### **3.4. DataGridView adat betöltés eseményre**

Amikor a felhasználó rákattint egy sorra a táblázatban, a megfelelő adatok automatikusan betöltődnek a mezőkbe.

Fontosabb részlet:

DataGridViewRow selectedRow = dataGridViewProducts.Rows[index];

pictureBoxProductImage.Load($"http://localhost:3000/image/{selectedRow.Cells["ImageUrl"].Value}");

#### **Magyarázat:**

A DataGridView kiválasztott sorából betöltött értékek alapján automatikusan frissül a kép (PictureBox) és a beviteli mezők.

## 4. UML aktivitásdiagram – Termék frissítése

+-------------------------+

| buttonProductUpdate\_Click |

+-------------------------+

|

v

+-------------------------+

| Kiolvassa a mezőkből az adatokat |

+-------------------------+

|

V

+-------------------------+

| Ellenőrzi, hogy a név üres-e |

+-------------------------+

| NO

v

+-------------------------+

| Leírás formázása (újsor) |

+-------------------------+

|

v

+-------------------------+

| Meghívja: UpdateProduct() |

+-------------------------+

# Asztali alkalmazás tesztek:

### **Teszteset 1: Normál esemény létrehozása (pozitív teszt)**

**Típus:** Fekete doboz teszt  
 **Cél:** Egy új esemény létrehozása megfelelő adatokkal

#### **Lépések:**

1. Navigáltunk az *Events* szekcióhoz.
2. Kitöltöttük a következő mezőket:
   1. Name: "Friday Draft"
   2. Date: 2025-06-01
   3. Max participants: 16
   4. Current participants: 0
   5. Description: "Standard booster draft."
3. Kattintottunk az **Add Event** gombra.

#### **Várt eredmény:**

* Az esemény bekerül az eseménylistába.
* Megjelenik egy visszajelző üzenet: **"Event created successfully."**

#### **Valós eredmény:**

A program a várt módon viselkedett. Az új esemény azonnal megjelent a listában.

#### **Teendő:**

Nincs hiba. A funkció megbízhatóan működik normál bemenet esetén.

### **Teszteset 2: Hibás e-mail cím bevitele felhasználó módosításakor**

**Típus:** Fekete doboz teszt (hibakezelési teszt)  
 **Cél:** Validáció ellenőrzése hibás e-mail formátum esetén

#### **Lépések:**

1. Kiválasztottunk egy létező felhasználót a *Customers* listából.
2. Módosítottuk az e-mail mezőt a következő értékre: "user[at]mail"
3. Rákattintottunk az **Update Customer** gombra.

#### **Várt eredmény:**

* Hibaüzenet: **"Invalid email address."**
* A módosítás nem kerül mentésre.

#### **Valós eredmény:**

A rendszer megakadályozta a mentést, és az alábbi üzenetet kaptuk:  
 **"Invalid email address."**

#### **Teendő:**

A felhasználónak javítania kell az e-mail mezőt a helyes formátumra (pl. [user@mail.com](mailto:user@mail.com)).

# Teszteset 3: Extrém adatok termék hozzáadásakor

**Teszt típusa:** Fehér doboz + Fekete doboz teszt  
 **Cél:** Annak vizsgálata, hogy a rendszer hogyan reagál szélsőséges vagy hibás adatbevitelre a termék létrehozása során.

#### **Lépések:**

Megnyitottuk a *Products* szekciót.

Az „New” gombra kattintva megnyitottuk az új termék felvitelére szolgáló űrlapot.

A következő adatokat adtuk meg:

**Product name:** ChaosCard

**Category:** *nincs kiválasztva* (üres maradt a ComboBox)

**Price:** -99999 (negatív szám)

**Stock quantity:** 999999999 (irreálisan magas érték)

**Available:** bekapcsolva (✓)

**Description:** *üres* (nem adtunk meg leírást)

**Kép feltöltés:** *kihagyva* (nem töltöttünk fel képet)

Kattintottunk a **„Create”** gombra.

#### **Elvárt eredmény:**

A program azonnal hibaüzenetet ad a negatív ár miatt.

Figyelmeztet, ha nincs kiválasztva kategória vagy nincs megadva leírás.

A kép hiánya esetén szintén jelzi a problémát.

A mentés nem történik meg, amíg a hibák javítva nincsenek.

#### **Valós eredmény:**

A program több hibaüzenetet is megjelenített:

* 1. **„Please fill out all fields.”**
  2. **„Price must be a positive number.”**

A mentés meghiúsult.

A rendszer nem engedte elmenteni az űrlapot, így nem került hibás termék az adatbázisba.

#### **Teendő:**

Az űrlapon javítani kell a hibás mezőket:

Pozitív számot kell megadni az árhoz (pl. 9.99).

Kategóriát ki kell választani a listából.

A leírás mezőt kötelező kitölteni.

Célszerű egy termékképet is feltölteni a vizuális megjelenítéshez.

Ezután a mentés funkció sikeresen működik.

1. Legalább 3 különböző teszteset részletes bemutatása.

* különböző felhasználó tevékenységek esetén hogyan reagált a program
* milyen üzeneteket kaptunk
* mi a teendő az egyes üzenetek esetében

1. Normál teszteset, extrém teszteset (bolondbiztosság tesztelése)
2. A tesztelés során kiderült hibák felsorolása A tesztelési dokumentációból derüljön ki, hogy ismered a különböző tesztelési módszereket (pl. fekete doboz, fehér doboz módszer)

Ajánlott terjedelem: a feladat jellegétől függően 2-5 oldal.

# Összefoglalás

## Önértékelés

A projekt kezdeti célja az volt, hogy létrehozzunk egy teljeskörű, felhasználóbarát webes és asztali alkalmazást, amely lehetőséget ad a felhasználók számára a Magic: The Gathering (MTG) termékek böngészésére, vásárlására, eseményekre való jelentkezésére, míg az adminisztrátorok számára egy egyszerű kezelőfelületet biztosít a készlet és események nyomon követésére. Ezen célok elérése érdekében mindketten különböző feladatkörökben dolgoztunk, szorosan együttműködve, hogy a projekt minden szempontja működőképes és átlátható legyen.

A fejlesztés során számos technikai kihívással szembesültünk, mint például az adatbázis struktúrájának kialakítása, a frontend és backend közötti kommunikáció, valamint a felhasználói interakciók kezelésének megoldása. Az adatbázis kapcsolatok megtervezésében és az SQL lekérdezések optimalizálásában fontos szerepet játszottunk, hogy biztosítsuk a rendszer hatékony működését. Különösen az események és regisztrációk kezelésénél, ahol valós idejű adatfrissítésekre volt szükség, számos technikai problémával kellett szembenéznünk. Az ilyen típusú problémák megoldásában egymást segítve dolgoztunk, hogy minden funkció gördülékenyen működjön.

A saját fejlődésünket több területen is észleltük a projekt során. Az egyik legfontosabb tapasztalatunk a **backend fejlesztés** terén szerzett tudásunk mélyítése volt, különösen az Express.js és MySQL alapú adatbázis-kezelésben. A backend felépítése, a REST API-k kialakítása és a megfelelő adatkezelés technikai szempontból jelentős fejlődést hozott számunkra. A frontend fejlesztés során, amelyben React és HTML/CSS használatával készítettük el az alkalmazás felületét, sokat tanultunk az interaktív webalkalmazások fejlesztéséről és optimalizálásáról. Emellett mindketten elmélyültünk a verziókezelő rendszerek, mint a Git és GitHub használatában, melyek biztosították a projekt gördülékeny és hatékony együttműködését.

A felmerült problémák és azok megoldásai is hozzájárultak a fejlődésünkhöz. A különböző technikai kihívások, mint a **backend és frontend közötti integráció**, vagy a **rugalmas adatbázis struktúra kialakítása**, folyamatos tanulásra és kísérletezésre késztettek minket. Az ilyen típusú problémák kezeléséhez mélyebb megértést nyertünk az adatbázisok működéséről, a RESTful API-k tervezéséről és a komplex felhasználói interakciók kezeléséről is.

Összességében úgy érezzük, hogy a projekt során jelentős fejlődést értünk el mind a technikai tudásunkban, mind a projekt menedzselésében. A munkánk nemcsak a konkrét technikai megoldásokban, hanem az együttműködés és a problémamegoldás terén is sokat fejlődött, amely nagyban hozzájárult a sikeres projekt megvalósításához.

## Továbbfejlesztési lehetőségek

A projekt során rengeteg olyan ötlet merült fel, amelyek izgalmasak voltak, de sajnos nem fért bele az időnkbe, vagy technikai korlátok miatt nem valósultak meg. Ugyanakkor ezek mind olyan irányok, amelyek a jövőben továbbfejleszthetők, és amelyek jelentősen hozzájárulhatnának az alkalmazás funkcionalitásához és felhasználói élményéhez.

Nem megvalósított ötletek:

Felhasználói visszajelzés rendszer: Eredetileg terveztük, hogy a felhasználók értékelhetik a megvásárolt termékeket és az eseményeken való részvételt, így javítva a vásárlói élményt és segítve a többi vásárlót a döntéseikben. Bár a funkció elképzelése megvolt, az időhiány miatt nem sikerült implementálni.

Webshop fizetési integrációk: Terveztük, hogy a rendszerhez különböző fizetési lehetőségeket integrálunk, például PayPal-t vagy bankkártyás fizetést, hogy a felhasználók könnyebben, gyorsabban tudjanak vásárolni. A háttérrendszer alapja elkészült, de az ilyen típusú integrációk időigényesek, és nem volt rá elég időnk a fejlesztés során.

Események automatikus értesítése: Szerettük volna, hogy az eseményekről a felhasználók automatikusan értesítést kapjanak e-mailben, vagy akár SMS-ben, amint új esemény kerül meghirdetésre, vagy egy esemény már csak néhány résztvevővel elérhető. Ennek a fejlesztésére sem volt elegendő időnk.

Termékajánlók és személyre szabott ajánlatok: A felhasználói vásárlási előzmények és preferenciák alapján egy olyan ajánló rendszert szerettünk volna építeni, amely személyre szabottan ajánl termékeket. Ez a funkció különösen fontos lehet egy webshop számára, hogy növelje a vásárlói elköteleződést és a vásárlások számát, de nem volt elegendő idő az ilyen algoritmusok kidolgozására és tesztelésére.

Jövőbeli fejlesztési lehetőségek:

Adminisztrátori elemzések és riportok: Bár már készültek statisztikai funkciók, a rendszer tovább fejleszthető lenne a backend adminisztrátorok számára hasznos riportokkal, például a termékek forgalmának részletesebb elemzésével, a vásárlói viselkedésről készült statisztikákkal, valamint az események népszerűségének és a résztvevői visszajelzések mérésével.

Termékkészlet és árképzés optimalizálása: A készletkezelés tovább fejleszthető lenne automatikus riasztó rendszerekkel, amelyek figyelmeztetik az adminisztrátort, amikor egy termék készlete alacsony, vagy ha az árat módosítani kell a piaci igények függvényében. A dinamikus árazás, amely az időszakos akciókhoz és vásárlói kereslethez igazodik, szintén érdekes lehetőség lenne a rendszer számára.

Social media integráció: A felhasználók számára lehetőséget biztosíthatnánk a termékek és események megosztására közösségi média platformokon (Facebook, Instagram, Twitter), ami segíthetne a márka ismertségének növelésében, valamint az események népszerűsítésében. Ez a funkció különösen fontos lehet, hogy növelje a rendszer elérhetőségét és a vásárlói bázist.

AI-alapú termékajánlók fejlesztése: A jövőben lehetőség lenne mesterséges intelligenciát alkalmazni a vásárlói előzmények és a termékek tulajdonságai alapján történő dinamikus termékajánlásokra. Az ilyen típusú személyre szabott ajánlórendszerek képesek lennének növelni a vásárlási hajlandóságot és a felhasználói élményt.

# Felhasznált irodalom

Használtunk saját előző munkáinkból kódokat, amik jó kezdőpontot adtak nekünk.

* <https://chatgpt.com/> és <https://grok.com/> - Új irányok felfedezéséhez, tesztelés során bekövetkezett hibák kijavításához.
* <https://www.metagames.hu/> - Hogy megtudjuk, hogyan is működik egy kártya webshop
* <https://magic.wizards.com/en> - Az MTG hivatalos oldala, ahonnan a termékeket szedtük.
* <https://github.com/Saeris/typeface-beleren-bold> - Az MTG hivatalos betűkészletéhez
* <https://stackoverflow.com/questions> - Stack Overflow, ha kérdéseink voltak egyes problémákkal kapcsolatban.
* <https://www.w3schools.com/> - Alap kérdések, ha valamiben nem voltunk biztosak.