



EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

MÉDIA- ÉS OKTATÁSinFORMATIKA TANSZÉK

## Pénzügyeket rendszerező webes alkalmazás

*Témavezető:*

Dr. Bende Imre

egyetemi adjunktus

*Szerző:*

Németh Zsófia Réka

programtervező informatikus BSc

*Budapest, 2025*

# Tartalomjegyzék

<b>1. Bevezetés</b>	<b>3</b>
<b>2. Felhasználói dokumentáció</b>	<b>4</b>
2.1. Rendszerkövetelmények . . . . .	4
2.2. Felhasználói esetek . . . . .	5
2.2.1. Fiók . . . . .	5
2.2.2. Pénzügyek . . . . .	8
2.2.3. Csoportok . . . . .	14
<b>3. Fejlesztői dokumentáció</b>	<b>17</b>
3.1. Architektúra . . . . .	17
3.1.1. Külső függőségek, technológiák . . . . .	18
3.1.2. Külső függőségek, technológiák . . . . .	18
3.1.3. Architektúra leírása . . . . .	21
3.2. Backend . . . . .	22
3.2.1. appsettings.json . . . . .	22
3.2.2. Program.cs . . . . .	22
3.2.3. Modellek . . . . .	22
3.2.4. API . . . . .	22
3.3. Frontend . . . . .	23
3.3.1. Layout . . . . .	23
3.3.2. static . . . . .	25
3.4. Adatbázis . . . . .	25
3.4.1. Szerkezet . . . . .	25
3.4.2. Backend - Adatbázis kapcsolat . . . . .	27
3.4.3. Migrations . . . . .	29
3.5. Tesztek . . . . .	29

<b>4. Összegzés</b>	<b>37</b>
<b>5. Továbbfejlesztési lehetőségek</b>	<b>38</b>
5.1. Statisztikák . . . . .	38
5.2. Tranzakciók . . . . .	38
5.3. Csoportok . . . . .	39
5.4. UI . . . . .	39
<b>Irodalomjegyzék</b>	<b>40</b>
<b>Ábrajegyzék</b>	<b>40</b>
<b>Táblázatjegyzék</b>	<b>41</b>
<b>Forráskódjegyzék</b>	<b>43</b>

# 1. fejezet

## Bevezetés

A választott témám egy személyes pénzügyeket rendszerező webalkalmazás. A dolgozat a full-stack webfejlesztés témaköréhez kapcsolódik, mivel frontend-, backend- és adatbáziskezelést is magában foglal. A backendet és a frontendet egy projekt részeként, egy C# alapú ASP.NET Web App (Razor Pages) (.NET 8) keretrendszerrel valósítottam meg, amelyhez egy MySQL relációs adatbázist csatoltam az Entity Framework segítségével. A .NET lehetőséget nyújt különböző frontend keretrendszerek használatára is, azonban a webalkalmazásomat jelenleg egy egyszerű HTML/CSS/JavaScript kombináció alkotja, Bootstrap (CSS framework) elemek felhasználásával.

Az alkalmazás fő lényege, hogy a felhasználó költségeit és bevételeit nyomon tudja követni, megtakarításait kezelhesse akár egyénileg, akár csoportokban, mindezt egy felhasználóbarát felületen. Az alkalmazás az alap funkciókon kívül még kimutatásokat is készít a felhasználó pénzkezelési szokásairól, melyeket akár PDF formátumban is le lehet tölteni.

Véleményem szerint a már erre a célra létrehozott megoldások egy része túlságosan leegyszerűsített, másik része pedig annyira összetett és funkciógazdag, hogy hétköznapi felhasználók számára nehezen átláthatóvá válik. Jelen alkalmazás ezt az ellentétet próbálja kiegyensúlyozni, egy egyszerű, de funkcionális megoldást kínálva.

A következő fejezetekben részletesen bemutatom a projektet a felhasználói és fejlesztői dokumentációkon keresztül.

## 2. fejezet

# Felhasználói dokumentáció

A következő fejezetben fogom bemutatni az alkalmazás elérését, egyes komponenseit, illetve felhasználási lehetőségeit.

Ez a pénzügyeket rendszerező alkalmazás alapvetően magánszemélyeknek készült, személyes felhasználásra, de mivel lehetőséget nyújt csoportos használatra is, ezáltal akár egy kisebb vállalat igényeit is elláthatja.

A főbb funkciók közé tartozik, hogy bevételeket és kiadásokat lehet rögzíteni, kategóriák szerint csoportosítva, melyeket utána egy összefoglaló - rendezhető és kereshető - adattáblázatban lehet látni. Az alkalmazás készít ezekről a pénzügyi szokásokról kimutatásokat, melyeket PDF formátumba is lehet exportálni. Az alap funkciókon kívül még kialakításra került egy megtakarításokkal foglalkozó oldal, ahol megadható tetszőleges mennyiségű pénzügyi gyűjtés, és utána ezeket lehet kezelni és folyamatosan monitorozni. Az imént felsorolt valamennyi funkció nem csupán egyéni felhasználásra áll rendelkezésre, hanem csoportok számára is. Csoportokból is tetszőleges mennyiségűt lehet létrehozni.

### 2.1. Rendszerkövetelmények

Mivel egy webes alkalmazásról van szó, ezért különleges gépigeny nem szükséges. Szinte az összes böngésző támogatott, ahogyan azt a 2.1. táblázat is mutatja.<sup>1</sup>

---

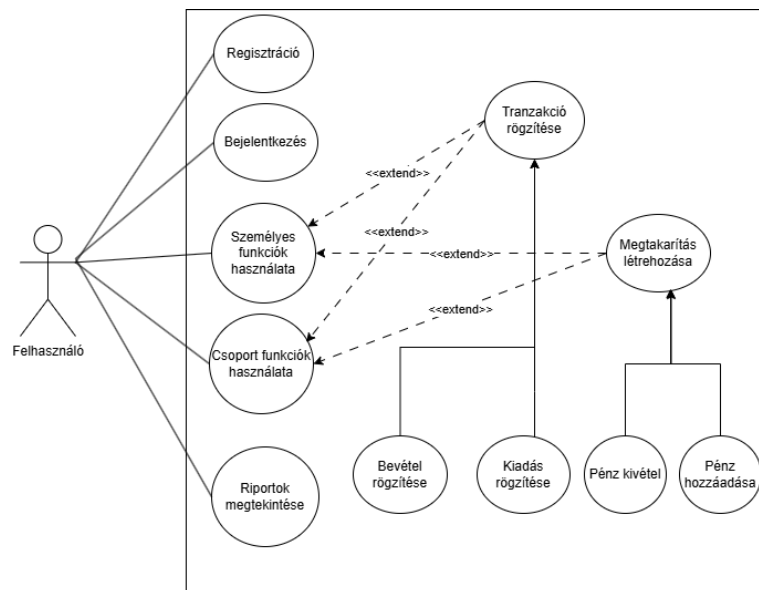
<sup>1</sup>Megjegyzés: Mivel a Microsoftnak nem található hivatalos adata a böngészők legfrissebb támogatott verzióira (a legújabb támogatott verzió ezek közül mindig a "current", vagyis az éppen legfrissebb), ezért a táblázat csupán egy általános modern webapplikáció böngésző kritériumait mutatja.

Böngésző	Minimum Verzió	Támogatás
Google Chrome	70+	Teljes
Mozilla Firefox	68+	Teljes
Microsoft Edge (Chromium)	79+	Teljes (a régi Edge (EdgeHTML) nem támogatott)
Safari	13+	Teljes
Opera	57+	Teljes
Internet Explorer	Nem alkalmazható	Nem támogatott

2.1. táblázat. Böngésző támogatás

## 2.2. Felhasználói esetek

Az alkalmazás felhasználói eseteit ez a bejegyzés fogja részletesen taglalni, képernyőképekkel kiegészítve. A felhasználói esetek bemutatását a 2.1. ábra mutatja be, és az utána következő fejezetek fejtik ki.



2.1. ábra. Használati eset diagram: Általános

### 2.2.1. Fiók

A felhasználói fiókok kezelése a szokásos Regisztráció – Bejelentkezés – Profil funkciókra épül. A felhasználónak a webalkalmazás elérése érdekében regisztrálnia kell, majd sikeres regisztrációt követően bejelentkezhet a felületre. A belső oldalra érve elérhető egy "Profil" oldal a bal menüsávból is, illetve a jobb felső sarokban az avatar ikonra kattintva a legördülő menüből is kiválasztható ez a menüpont. Szintén

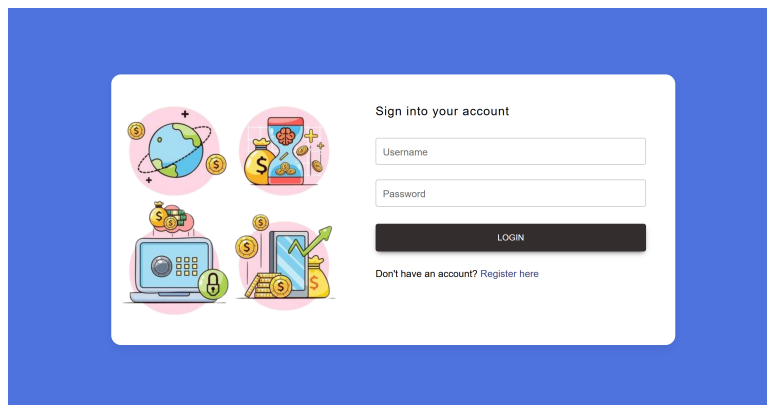
ezen a két helyen találjuk a kijelentkezés gombot is, amellyel megszüntethetjük az adott munkamenetet és kiléphetünk a fiókból. Ezen funkciók részletes leírását a 2.2. táblázat foglalja össze.

### Fiók kezelés

Funkció	Leírás
<i>Bejelentkezés</i>	Bejelentkezés egy egyedi felhasználónévvel és egy jelszóval lehetséges. (2.3. ábra)
<i>Regisztráció</i>	Regisztrálni lehet az oldalra a következő adatok megadásával: e-mail cím, felhasználónév (egyedi), teljes név, jelszó. Jelszó kritériumok: legalább 8 karakter, nagybetűt, kisbetűt, számot és egy speciális karaktert is tartalmaznia kell. (2.2. ábra)
<i>Profil</i>	A Profil oldalon lehet a fiókhoz tartozó adatokat módosítani: a felhasználónevet, a teljes nevet, illetve az e-mail címet is. Itt elérhető még egy "Change Pass" gomb is, amely elnavigálja a felhasználót a jelszó megváltoztató felületre. A profil oldal használatát a 2.3. táblázat fejti ki bővebben. (2.4. ábra)
<i>Jelszó módosítás</i>	A Profil oldalról lehet a jelszó változtató felületet elérni a "Change Pass" gomb megnyomásával. Itt meg kell adni a következő adatokat: régi jelszó, új jelszó, új jelszó megerősítése.
<i>Kijelentkezés</i>	Ha a felhasználó befejezte kívánt tevékenységét, akkor a bal oldali menüsáv alján található "Logout" gomb megnyomásával tud kijelentkezni. A kijelentkező gomb még elérhető a jobb felső sarokban található ember ikonra kattintva legördülő menüben is.

2.2. táblázat. Fiók műveletek

2.2. ábra. Screenshot: Regisztrációs felület

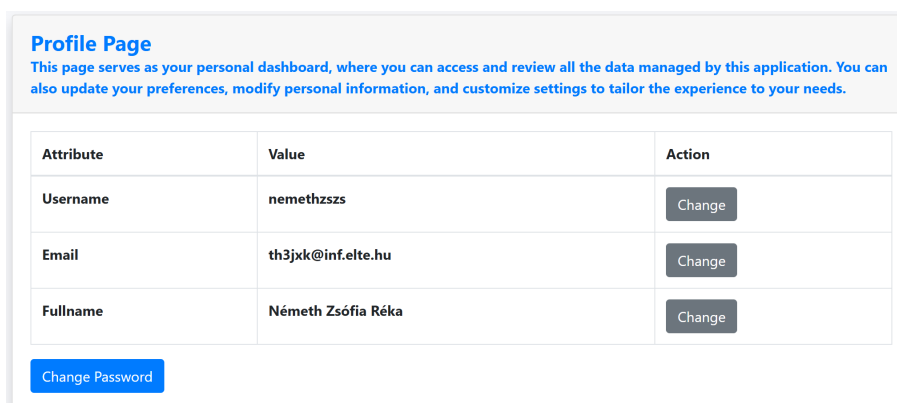


2.3. ábra. Screenshot: Bejelentkező felület

## Profil

Funkció	Leírás
<i>Adatok megtekintése</i>	A regisztrációnál megadott személyes adatait a felhasználó itt tekintheti meg.
<i>Adatok módosítása</i>	A személyes adatok módosítására is itt van lehetőség, egyszerű beviteli mezőkkel lehet módosítani a felhasználónevet (egyedi), teljes nevet és e-mail címet. A "Change" gombra kattintva előugrik egy beviteli mező egy "Save" és "Cancel" gombbal kiegészítve. Az előbbi megpróbálja végrehajtani a kért módosítást, és jelzi az esetlegesen fellépő hibát (érvénytelen e-mail cím, foglalt felhasználónév, stb.) a felhasználó felé.
<i>Jelszó módosítás</i>	Ha a felhasználó módosítani kívánja a jelszavát, akkor a "Change Password" gombra kattintva megjelenik egy új felület, három jelszó beviteli mezővel (rég, új, új megerősítés). Jelszó kritériumok: legalább 8 karakter, nagybetűt, kisbetűt, számot és egy speciális karaktert is tartalmaznia kell.

2.3. táblázat. Profil oldal



2.4. ábra. Screenshot: Profil felület



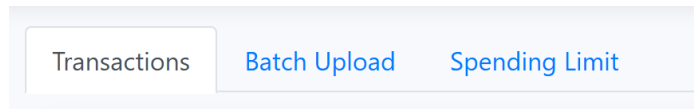
### 2.2.2. Pénzügyek

A pénzügyeket kezelő oldalakat a kezdőképernyőről, illetve a baloldalon lévő menüből érhetjük el. A "Finance" cím alatt 3 különböző lap került kialakításra.

- Tracker - Bevételek és kiadások vezetése, havi költési limit kezelése és előzmények megtekintése
- Savings - Megtakarítások megtekintése és kezelése
- Reports - Kimutatások megtekintése, személyre szabása és exportálása

#### Tracker felület

A Tracker felület foglalja magában a bevételek és kiadások vezetését, havi költési limit módosítását és monitorozását, illetve az előzmények megtekintését és korábbi tranzakciók eltávolítását. Az oldal 3 különböző lapból áll, ezek között lehet navigálni az oldal tetején lévő gombok segítségével (2.5. ábra).



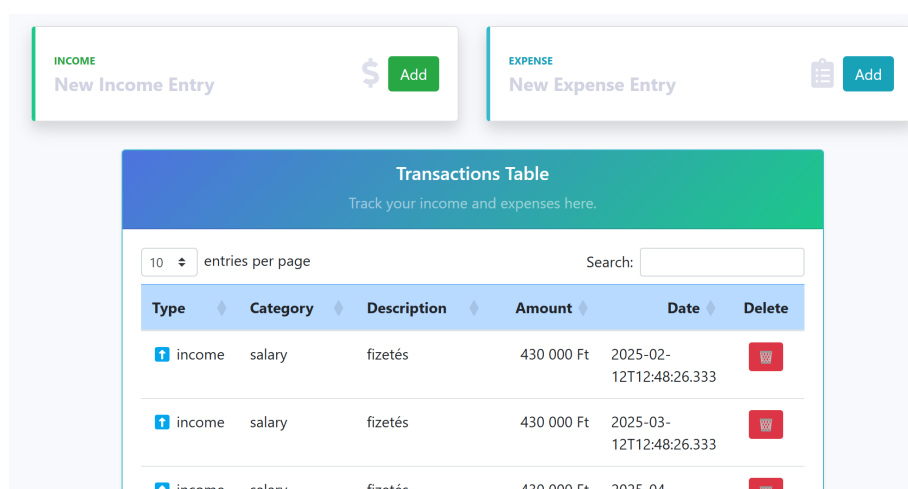
2.5. ábra. Screenshot: Tracker felület navigációs sáv

- Transactions (2.4. táblázat és 2.6. ábra) (Bevételek és kiadások vezetése, és előzmények megtekintése)
- Batch Upload (2.5. táblázat és 2.7. ábra) (Tömeges bejegyzés-feltöltés)
- Monthly Spending Limit (2.6. táblázat és 2.8. ábra) (Költési limit kezelése)

*Transactions* A tranzakciókkal kapcsolatos funkciók leírását a 2.4. táblázat tartalmazza.

Funkció	Leírás
<i>Kiadás/bevétel rögzítése</i>	Külön dobozokban lehet rögzíteni a kiadásokat, és a bevételeket. Egyszerre egyet lehet bevinni, melynek adni kell egy összeget, egy leírást, opcionálisan lehet kategóriát is megadni. Ha kiadást rögzítünk, akkor egy felugró ablak jelzi minden alkalommal, hogy mennyi maradt még a havi költési limitből, illetve ha már túl lett lépve, akkor egy figyelmeztető ablak ugrik fel.
<i>Előzmények</i>	A költési és bevételi előzményeket is itt lehet megtekinteni egy összesítő táblázatban. A táblázatban felül a "Search" mező módosításával lehet szűrni címszó alapján, azaz tudunk bármire (összegre, címre, kategóriára, stb.) keresni. Illetve az oszlopok címekre kattintva rendezni is lehet őket. A táblázat utolsó ("Delete") oszlopa arra szolgál, hogy egy adott sorban a kuka ikonra rákattintva, az a bejegyzés törlődik.

2.4. táblázat. Transactions lap



2.6. ábra. Screenshot: Transactions lap

*Batch Upload* A tömeges feltöltéssel kapcsolatos funkciókat a 2.5. táblázat tartalmazza.

Funkció	Leírás
<i>CSV feltöltés</i>	A felhasználó számára lehetőség adott tömeges tranzakció bejegyzés feltöltésére. Egy ("," vagy ";" karakterekkel tagolt) .csv kiterjesztésű fájl feltöltésével automatikusan rögzítésre kerülnek a benne szereplő tranzakciók. A fájl helyes felépítése a 2.1. forráskód ábrán látható. Minden mező kitöltése kötelező, illetve kategóriából csak a felületen már létező kategória érték adható meg. Az általános sor struktúra: [típus (1 = bevétel, 2 = kiadás), kategória (vagy "-" ha nem kívánja megadni), leírás, összeg (pozitív egész szám), dátum (formátum: YYYY-MM-DD)]

2.5. táblázat. Batch upload lap

```

1      2, family, vacsi petivel, 10500, 2024-10-12
2      2, family, botanikus kert, 12300, 2024-11-15
3      1, -, ruha, 4500, 2024-09-24

```

2.1. forráskód. Tömeges feltöltés: csv fájl struktúra

**Upload Data (CSV)**

Upload a CSV file with the following row format:

- [type (1 = income, 2 = expense), category (or "-" if none), description, amount (positive integer), date (YYYY-MM-DD)]
- Example: [2, family, dinner with parents, 10500, 2024-10-23]

Fájl kiválasztása Nincs fájl kiválasztva

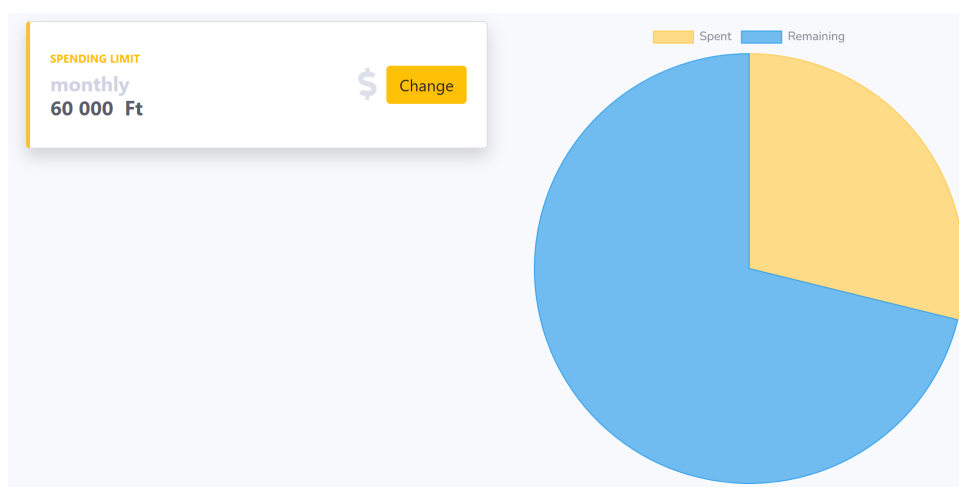
Upload

2.7. ábra. Screenshot: Tömeges feltöltés lap

*Spending Limit* A havi költési limittel kapcsolatos funkciók leírását a 2.6. táblázat tartalmazza.

Funkció	Leírás
<i>Havi költési limit módosítása</i>	Itt tudja a felhasználó a havi költési limitjét módosítani. A "Change" gombra kattintva előugrik egy beviteli mező és egy "Save" (mentés) gomb.
<i>Diagram</i>	A költési limit vizuális nyomon követését segíti a lapon elhelyezett kördiagram, amely egyszerű módon mutatja, hogy mennyit költött már a felhasználó a limitből.

2.6. táblázat. Monthly Spending limit lap



2.8. ábra. Screenshot: Monthly Spending Limit lap

## Megtakarítás felület

*Gyűjtés létrehozása* A felhasználó új megtakarítási célt hozhat létre, például egy konkrét vásárlás vagy vészhelyzeti alap létrehozásához. A cél tartalmazhat megnevezést, opcionális leírást, célösszeget és határidőt. A cél a főoldalon egy külön kártyaként jelenik meg. Egy új gyűjtést a 2.9. ábrán látható felület kitöltésével tud hozzáadni a felhasználó, amely az "Add new saving goal" gomb megnyomásával ugrik fel.

Create a Savings Goal

Title

Description

Goal Amount (Optional)

Deadline (Optional)

éééé. hh. nn.

Save

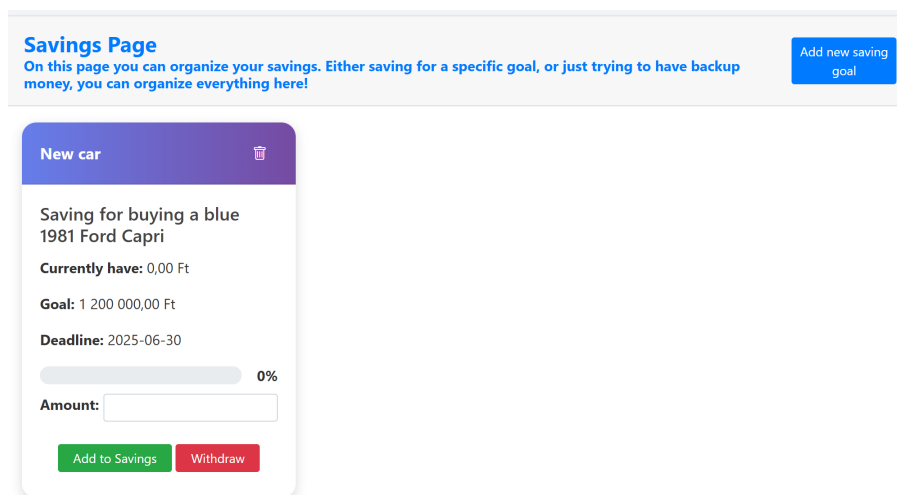
Cancel

2.9. ábra. Screenshot: Gyűjtés létrehozása

*Gyűjtések kezelése* A gyűjtéseket az egyes kártyákon belül elhelyezett gombok, és beviteli mezők segítségével lehet kezelni. Ezeket a funkciókat a 2.7. táblázat.

Funkció	Leírás
<i>Összeg hozzáadása a gyűjtéshez</i>	Lehetővé teszi, hogy a felhasználó megtakarított összeget adjon hozzá egy kiválasztott célhoz. A kártyán található beviteli mező és "Add to Savings" gomb segítségével történik a művelet. Az aktuális megtakarítás értéke és a haladás százalékos formában is megjelenik.
<i>Összeg kivonása a gyűjtésből</i>	Amennyiben a felhasználó el kíván távolítani egy összeget a gyűjtésből (pl. téves bevétel miatt), azt megteheti a "Withdraw" gomb segítségével. Ez a funkció csökkenti a gyűjtött összeget, de nem törli a célt.
<i>Gyűjtés törlése</i>	A gyűjtés jobb felső sarkában elhelyezett kuka ikon segítségével a felhasználó teljesen eltávolíthat egy megtakarítási célt. Ez véglegesen törli az adott kártyát és a hozzá tartozó adatokat.
<i>Cél elérése vizualizáció</i>	Ha a megtakarított összeg eléri vagy meghaladja a kitűzött célt, a kártya vizuálisan zöld színre vált.

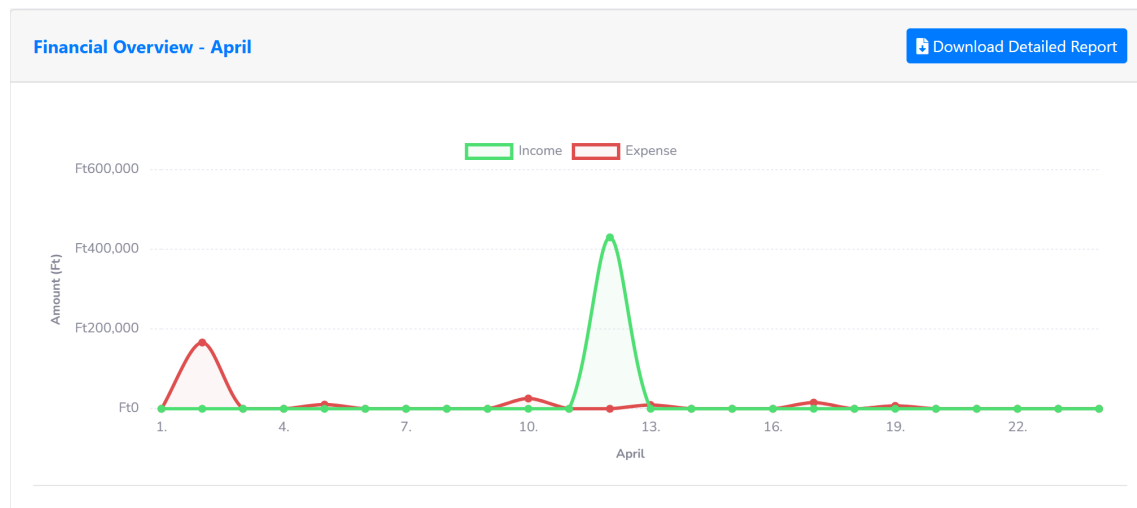
2.7. táblázat. A Savings oldal



2.10. ábra. Screenshot: Savings felület

## Kimutatások felület

*Havi riport* Az oldal első szekciója egy vonaldiagramot jelenít meg (2.11. ábra), amely napi bontásban mutatja az adott hónap bevételeit és kiadásait. A diagram segít a felhasználónak átlátni a pénzmozgásokat, észrevenni a kiugró értékeket. A színek jól elkülönítik a bevételt (zöld) és a kiadást (piros).



2.11. ábra. Screenshot: Aktuális havi riport

*Letöltés* A "Download Detailed Report" gomb lehetőséget ad arra, hogy a felhasználó exportálja a részletes pénzügyi riportját az adott hónapra (PDF formátumban). Ez hasznos lehet adminisztrációhoz vagy hó végi összesítéshez.

*Éves riport* Egy oszlopdiagram (2.12. ábra) összehasonlítja az egyes hónapok bevételi és kiadási értékeit. A felhasználó egy legördülő listából kiválaszthatja az

évet (de alapértelmezetten az idei év adatai láthatóak), amely alapján frissül a diagram. Ez segít az éves pénzügyi trendek nyomon követésében.



2.12. ábra. Screenshot: Éves összesítő riport

### 2.2.3. Csoportok

A csoportokban valamennyi funkció elérhető, melyek a személyes használat esetében is. Ezeket az előző fejezetek taglalták részletesebben. A következő bekezdésben a csoportok kezelését és elérését fogom bemutatni.

#### Csoport létrehozása

Az 2.13. ábrán látható felület a baloldali menüsávból érhető el ("Create Group" menüpont). Itt a következő adatok megadása szükséges: Title (csoport neve), Type (csoport típusa, ez csak a felhasználó számára egy rendszerezési lehetőség, jelentősége nincsen), Description (csoport részletesebb leírása). Ezután a "Create Group" gombra kattintva már kész is a csoport.

2.13. ábra. Screenshot: Create Group oldal

## Saját csoportok

*Megtekintés* Az 2.14. ábrán látható felület a baloldali menüsávból érhető el ("My Groups" menüpont). Az oldalon belül minden csoport külön lapokon tekinthető meg, ezek között lehet navigálni az oldal tetején lévő gombok segítségével. Minden gomb felirata az adott csoport neve. A lapok tetején láthatóak az alap adatok, mint a csoport neve, leírása, tagok és a felhasználó szerepköre ("Admin" = ha ő hozta létre, "Member" = azaz "tag", ha a felhasználó csak meghívva lett a csoportba)

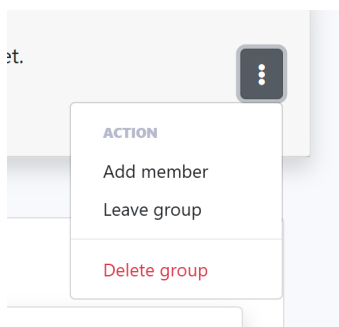
2.14. ábra. Screenshot: My Group oldal

*Funkciók* A csoportos funkciók megegyeznek a személyes funkciókkal (bevételek/-kiadások rögzítése és törlése, előzmények megtekintése, riport megtekintése,



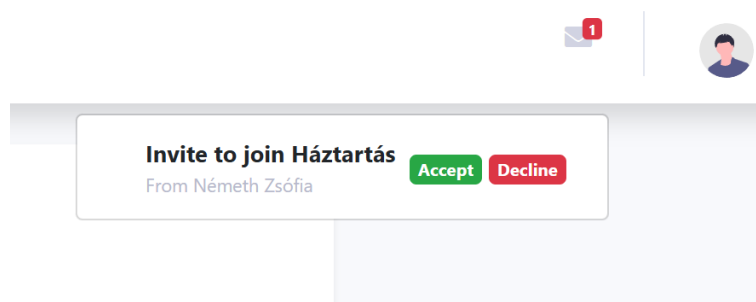
megtakarítások létrehozása és kezelése). Ezek részletes bemutatásra már megtörtént korábbi fejezetekben.

*Kezelés* Egy adott csoport fejlécének jobb oldalán található 3 pont megnyomása egy leugró menüt nyit meg. Itt lehet meghívót küldeni a csoportba egy másik felhasználónak ("Add member"), elhagyni a csoportot ("Leave group"), vagy ha Admin szerepkörrel rendelkezik az adott felhasználó akkor törölni is lehet a csoportot.



2.15. ábra. Screenshot: Screenshot: Csoport műveletek

*Meghívó* A meghívó küldése után a meghívott felhasználónak érkezni fog egy értesítése (mely az oldal tetején lévő levél ikon megnyomásával válik láthatóvá), amely tartalmazza a következő információkat: csoport neve, felhasználó teljes neve, aki meghívta oda. Itt az 2.16 ábrán látható módon egy "Accept" (elfogadás) és egy "Deny" (elutasítás) gomb található. Ha elfogadja a felhasználó a meghívást, akkor tagja lesz a csoportnak.



2.16. ábra. Screenshot: Értesítések

## 3. fejezet

# Fejlesztői dokumentáció

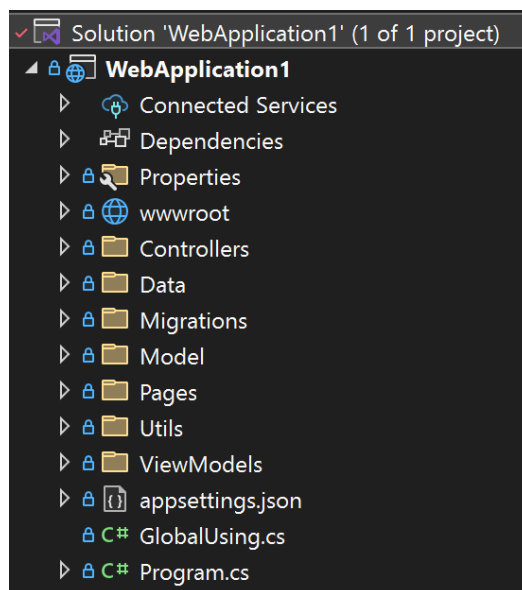
A következő fejezetben az alkalmazást fejlesztői szemszögből mutatom be. Részletesen kitérek az alkalmazás felépítésére, a használt technológiákra, az adatbázis-struktúrára, valamint a fejlesztés során követett elvekre és megoldásokra.

### 3.1. Architektúra

- Frontend: HTML<sup>1</sup>/CSS/JavaScript (és Bootstrap)
- Backend: C# (ASP.NET Web App (Razor Pages))
- Adatbázis: MySQL relációs adatbázis

---

<sup>1</sup>Mivel a C# projekten belül lett létrehozva a frontend is, ezért a .html helyett .cshtml kiterjesztésű fájlok vannak. Ezek annyiban különböznek a HTML-től, hogy vannak bizonyos tagek, kulcsszavak, melyekkel különleges dolgokat tudunk csinálni. A Frontend című fejezetben kerül ez a téma bővebb kifejtésre.



3.1. ábra. Projekt struktúra

### 3.1.1. Külső függőségek, technológiák

Az alkalmazás fejlesztése során az alábbi külső csomagok és könyvtárak kerültek felhasználásra:

### 3.1.2. Külső függőségek, technológiák

Csomagok hozzáadása a projekthez történhet a NuGet Package Manager vagy CDN segítségével is. Az alkalmazás fejlesztése során az alábbi külső csomagok és könyvtárak kerültek felhasználásra:

- **Entity Framework Core (EF Core)** – ORM technológia, amely lehetővé teszi az adatbázissal való objektumorientált munkát. Kiemelten hasznos a LINQ támogatása és a migrációs rendszer.<sup>2</sup>
- **Pomelo.EntityFrameworkCore.MySql** – Az EF Core és a MySQL adatbázis közötti kapcsolatot biztosító NuGet csomag.
- **MySqlConnection** – Könnyűsúlyú, aszinkron támogatással rendelkező ADO.NET driver MySQL-hez.
- **Swashbuckle.AspNetCore** – A Swagger UI integrálására szolgáló csomag, amely lehetővé teszi az API-k dokumentálását és tesztelését.

---

<sup>2</sup>A migrációk működéséről az adatbázissal foglalkozó fejezet fog részletesebb leírást nyújtani.

- **Bootstrap** – Reszponzív front-end keretrendszer, amely előre definiált CSS osztályokat és JavaScript komponenseket biztosít a felhasználói felület egységes és gyors kialakításához. Az alkalmazás a Bootstrap 4-es verzióját használja, amely lehetővé teszi különböző vizuális elemek (például gombok, táblázatok, rácsszerkezetek) egyszerű alkalmazását csupán megfelelő class attribútumok beállításával. Nem csak az esztétikus megjelenésben nyújt segítséget a Bootstrap, hanem interaktív viselkedések (pl. modális ablakok, lenyíló menük) hozzáadásában is, beépített JavaScript funkciók segítségével.
- **FontAwesome (CDN)** – Ikonok használatához alkalmazott ikoncsomag.
- **QuestPDF** -Az alkalmazás lehetőséget nyújt az adatok exportálására nyomtatható formátumban, egy PDF fájl formájában. Erre a célra a QuestPDF nevű nyílt forráskódú .NET könyvtárat használtam.

A QuestPDF lehetővé teszi professzionális minőségű PDF fájlok generálását teljes mértékben C# nyelvben, deklaratív stílusban. Előnyei:

- Reszponzív dokumentumelrendezés
- Könnyen tanulható, C#-os fluent API
- Támogatja a táblázatokat, stílusokat, színeket, képeket

A generálás egy `Document.Create(...)` hívással történik, ahol meghatározásra kerül a dokumentum szerkezete, stílusa, tartalma. Például:

```
1      Document.Create(container =>
2      {
3          container.Page(page =>
4          {
5              page.Content()
6                  .Column(col =>
7                  {
8                      col.Item().Text("Kiadasi osszesito");
9                      col.Item().Text("Osszes tranzakcio: 42");
10                 });
11             });
12     }).GeneratePdf("output.pdf");
```

3.1. forráskód. PDF generálás példa

A PDF fájl generálása szerveroldalon történik, a Utils/PDFGenerator.cs osztályban megvalósított logika segítségével. A kész fájl ezután egy HTTP végpont (Controller) segítségével érhető el és tölthető le a felhasználó által.

```
1 [HttpGet("DownloadReportForCurrent")]
2 public async Task<IActionResult> DownloadReport()
3 {
4     // get data
5
6     var pdf = new PdfGenerator(user.Fullname, transactions);
7     var pdfBytes = pdf.GeneratePdf();
8
9     return File(pdfBytes, "application/pdf", $"report_{user.
        Fullname}_{DateTime.Now:yyyyMMdd}.pdf");
10 }
```

3.2. forráskód. PDF generálás API végpont

- **Chart.js** – Az alkalmazás az adatok vizuális megjelenítésére a népszerű JavaScript könyvtárat használja, amely interaktív, reszponzív grafikonokat képes kirajzolni HTML canvas elemeken keresztül.

Két típusú diagram került megvalósításra, melyek közül az utóbbi csoportokban nem elérhető:

- Napi bontású bevétel és kiadás alakulása az aktuális hónapban (vonaldiagram)
- Havi bontású bevétel–kiadás összehasonlító diagram (oszlopdiaagram)

A grafikonok kliensoldalon, JavaScript segítségével jönnek létre. Az adatok aszinkron módon kerülnek lekérésre az API végpontokról, a felhasználó azonosítója alapján. A lekért tranzakciós adatok formázása és csoportosítása után a rendszer automatikusan diagramot generál azok vizuális megjelenítésére.

Az alábbi példában látható egy egyszerűbb diagram kirajzolása:

```
1 const ctx = document.getElementById("finance-chart");
2 // <canvas> elem
3 const monthlySpendingChart = new Chart(ctx, {
4     type: 'bar',
5     data: {
```

```
6      labels: months, // Jan - Dec
7      datasets: [{
8          label: "Income",
9          data: incomeData // from backend
10     }]
11 }
12 });
```

3.3. forráskód. Chart.js alapú havi diagram kirajzolása

### 3.1.3. Architektúra leírása

Ahogy a 3.1. ábrán is látható, az egész webalkalmazás egy projekten belül lett kialakítva. A .NET-es integrált frontend és backend fejlesztés hatalmas előnye a rendszerezettség, a szabályszerű kommunikáció az egyes rétegek között (illetve ezen rétegek helyes elkülönölése), és a rengeteg beépített segédfüggvény/konfiguráció/cshtml tag. Kiemelném még a modellek használatát is, amelyek egyszerűbb és biztonságosabb (például SQL injection elleni védelem) adatbázis kezelést biztosítanak.

Az egyes route-ok konfigurálása is meglehetősen letisztult ebben a keretrendszerben. A "Pages" mappa<sup>3</sup> fájlstruktúrája alapján automatikusan létrejönnek a route-ok, ha a "Program.cs"-ben megadjuk a programnak, hogy hozza létre őket:

```
1 app.UseRouting();
```

3.4. forráskód. Route-ok konfigurálása

A frontend és a backend közötti hagyományos JavaScript segítségével történő kommunikáción kívül, a cshtml formátum miatt lehetőség nyílik egyszerűbb esetekre közvetlen kapcsolatot is létesíteni a backend és a frontend között. Például:

```
1 <h1>Username</h1>
2 <p>@Model.UserData.Username</p>
```

3.5. forráskód. Frontend modell egy alkalmazása

Itt ugye mint látható a modellből tudunk adatot lekérdezni. De mi is a "Model" pontosan? Ugyebár az ASP .NET Web App keretrendszer úgy működik, hogy minden oldal (Razor page) egy .cshtml és egy .cs fájl együtteséből áll össze. A .cs kiterjesztésű fájl az adott oldal modellje. Itt definiálhatunk adatszerkezeteket és

---

<sup>3</sup>A "Pages" mappa, ahogy a neve is mutatja, tartalmazza a weboldal egyes oldalait. Bővebb kifejtés a Frontend című fejezetben.

függvényeket, mint például az `OnGet()` és `OnPost()`, amelyek beépített (opcionális) metódusok, és ahogy a nevük is mutatja, az oldalról érkező GET és POST requesteket kezelik. Tehát például, ha az adott oldalon egy darab form-unk van, amit POST metódussal be akarunk küldeni a szervernek, ezt JS (JavaScript) kód írása nélkül biztonságosan meg tudjuk tenni.

## 3.2. Backend

A következő fejezetben az alkalmazás backend architektúráját mutatom be, kifejtve az egyes elemek funkcióit és használatát.

### 3.2.1. appsettings.json

Az `appsettings.json` egy konfigurációs fájl. Ebben lehet alkalmazásszintű beállításokat tárolni, például: adatbáziskapcsolati stringeket, API kulcsokat, logolási beállításokat, vagy bármilyen egyedi, fejlesztő által definiált értéket. Az értékeket a program bármely részén könnyen ki lehet olvasni (szótár szintaktika használatával) (pl. `Configuration["Kulcs"]`). Ez segít elkülöníteni a kódot a konfigurációtól, így könnyebb a karbantartás, és egyszerűen kezelhető a környezetenkénti eltérés (pl. fejlesztői vs. éles környezet).

### 3.2.2. Program.cs

Ez a fő program fájl, és egyben az alkalmazás belépési pontja. Ebben készítjük elő a webalkalmazásunk tulajdonságait (persze a Razor Pages egyszerűségének hála ez nem sok plusz feladattal jár, csupán a helyes függvényeket kell meghívni helyes sorrendben, attól függően, hogy hogyan akarjuk konfigurálni az alkalmazást). Itt lehet beállítani a korábban említett routing-ot, az autentikációt, a cookie-kat, a session-t, a fejlesztői és éles környezeteket, és még sok minden mást. Itt tudjuk felkonfigurálni a modellt, az adatbázist és az oldalakat (Pages) is. ezután az `app.Run()` függvény hívással tudjuk ténylegesen elindítani a weboldalt.

### 3.2.3. Modellek

Minden adatbázis táblának létrehozunk egy modellt (bővebb kifejtés az "adatbázis" pontban). Ezek mind `C#` osztályok táblaként, és minden oszlop egy külön

propertynek feleltethető meg.

### 3.2.4. API

Az ASP.NET-es struktúra úgy működik, hogy miután a Program.cs-ben felkonfiguráltuk a szükséges dolgokat, megalkottuk a modelleket, létrehoztuk a HTML lapokat, már csak egy fontos dolog maradt hátra. A frontend-backend kommunikáció. Említés esett már arról, hogy egyszerűbb esetekben ezt a HTML-ben bizonyos tagekkel meg tudjuk tenni, de a nem triviális esetek megoldására is egyszerű módot kínál a .NET. A Controllers mappában minden modellnek (vagyis minden adatbázis táblának) létrehozunk egy Controller fájlt is. Ebben fogunk API-végpontokat definiálni, melyeket a frontenden JavaScript segítségével tudunk meghívni, és ezáltal adatot közvetíteni a felhasználó és a szerver között (oda-vissza). Az API-endpointok tesztelésére és leírására egy hasznos, szintén beépített, megoldást használhatunk, a Swagget. Ezt is a Program.cs fájlban állíthatjuk be.<sup>4</sup> A Swagger az alkalmazás API-végpontjainak dokumentációja, amely lehetőséget biztosít az egyes végpontok közvetlen tesztelésére is.

## 3.3. Frontend

Térjünk át a frontend réteg alkotóelemeire, és ezek működésére. Itt is komponensenként fogom bemutatni a felhasznált technológiákat.

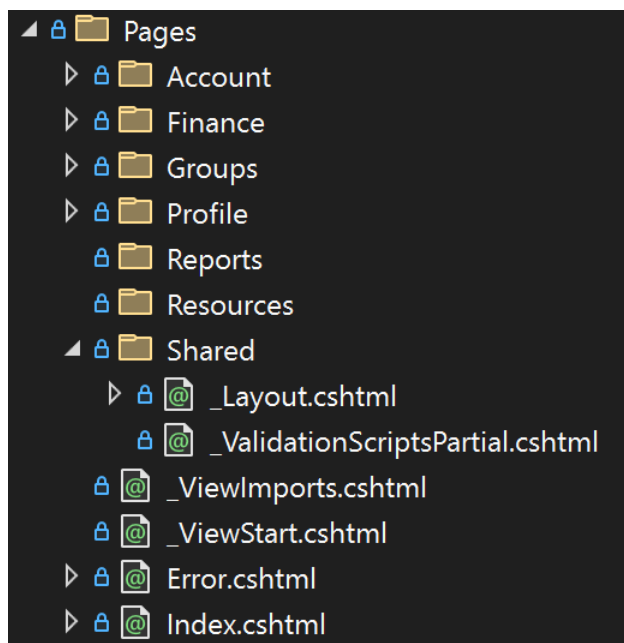
### 3.3.1. Layout

A "Pages" mappa tartalmazza az oldalakat. A keretrendszer ugyebár úgy működik, hogy minden oldal (Razor page) egy .cshtml és egy .cs fájl együtteséből áll össze. A Pages mappa ezeket a párosokat tartalmazza, további kisebb mappákra bontva a funkció csoportok alapján. Az alább mellékelt ábrából (3.2. ábra) látható tehát, hogy például a bejelentkező felületet a következő linken tudjuk elérni: "localhost:«port»/Account/Login".

---

<sup>4</sup>Szigorúan csak Development módban jelenjen meg, adatvédelmi/adat hitelességi okokból.





3.2. ábra. Pages struktúra

Találhatók még ebben a mappában egyéb segédfájlok is, mint például az `Error.cshtml`, ami az esetleges hibák esetén jelenik meg. Kiemelendő még a `Shared` mappa, amiben található egy `Layout.cshtml` fájl is (modell nélkül). Ez, ahogy a neve is mutatja egy alap sablont biztosít az alkalmazás felületének. A weboldal egy (a képernyő baloldán lévő) főmenüből, egy kis (az oldal tetején található) menüsávból, a fő tartalmi részből áll, illetve egy footer részből áll. A footer és a menüsávok kódját tartalmazza a `Layout.cshtml` között a tartalomnak "kihagyott" rész:

```
1 <div class="container-fluid">
2 @RenderBody()
3 </div>
```

### 3.6. forráskód. Layout extension ASP.NET keretrendszerben

A `RenderBody()` függvény (szintén beépített .NET metódus) behelyezi az adott oldal HTML kódját ebbe a `div` HTML elembe. Ez automatikusan megtörténik az összes oldal esetében amelyre navigál a felhasználó. Ha valami miatt éppen nem ezt a sablont akarjuk követni egy oldalunkkal (például: Bejelentkező felület), akkor az adott oldal HTML kódjába beírhatjuk hogy ne sablont kövessen:

```
1 @{
2     Layout = null; // Remove layout if you want a standalone page
3 }
```

### 3.7. forráskód. Layout extension ASP.NET keretrendszerben

### 3.3.2. static

A "wwwroot" mappában található az összes statikus erőforrás (static resource). Itt a hagyományosan használt mappastruktúrát követtem az alkalmazás készítésekor, azaz létre lettek hozva a következő mappák:

- "js": JavaScript kódok
- "css": CSS stylesheetek
- "img": Képek

A frontend rész formázását Bootstrap keretrendszer felhasználásával valósítottam meg. Ezt úgy importáltam a projektbe, hogy a kész CSS/JS fileok vannak letöltve ugyanide, a wwwroot mappába, ezen kívül egy külső hivatkozás van egy betűtípusra.

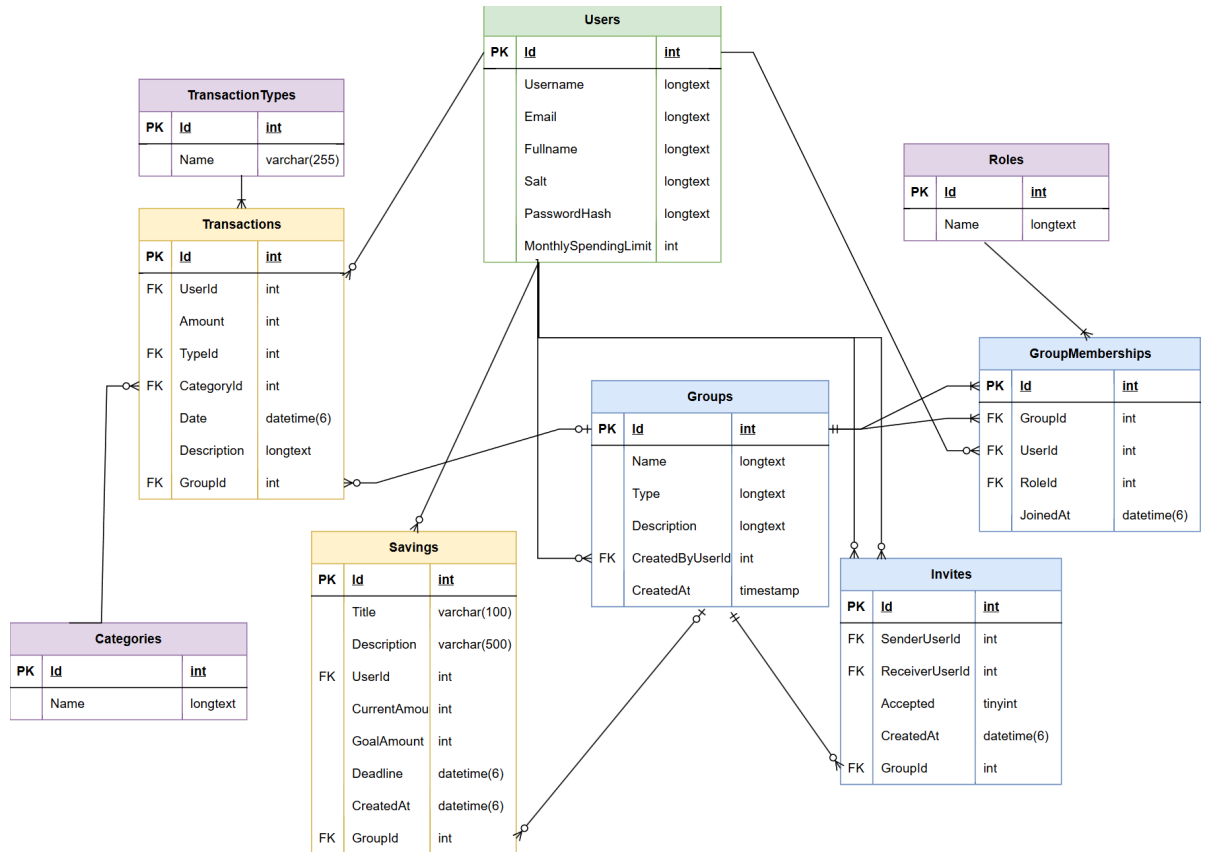
## 3.4. Adatbázis

Az alkalmazás működésének egyik alapvető pillére az adatbázis, amely a rendszerben tárolt információk strukturált kezelését, hosszú távú perzisztenciáját és elérhetőségét biztosítja. Az adatbázis szerkezete úgy került kialakításra, hogy az hatékonyan támogassa az alkalmazás funkcionális igényeit, miközben jól skálázható és könnyen karbantartható maradjon.

Az adatbázis relációs modellre épül, és az adatok külön táblákban kerülnek tárolásra, melyeket kulcsok és kapcsolat típusok kötnék össze.

### 3.4.1. Szerkezet

Az alkalmazás adatbázisa egy localhoston futó MySQL adatbázis. A MySQL-t sok szempont miatt szokták kisebb alkalmazásokhoz használni, ezek közé tartozik például az a nem elhanyagolható indok, hogy teljesen ingyenes a használata. Illetve szintén az egyszerűségét tudnám kiemelni, és a kompatibilitását az ASP.NET keretrendszerrel. Egy darab adatbázis lett létrehozva, azon belül több tábla található, melyek természetesen kapcsolódnak egymáshoz. Az adatmodell a relációs adatbázisok normalizálási elvein alapul, és a harmadik normálformáig (3NF) került kialakításra. A szerkezetet és a logikai kapcsolatokat az egyes táblák között a 3.3. ábra szemlélteti. A táblák részletes leírását pedig a 3.1. táblázat tartalmazza.



3.3. ábra. Adatbázis struktúra

Tábla	Leírás
Users	A rendszer felhasználóit tartalmazza. Itt tároljuk az alapvető adatokat (felhasználónév, e-mail, teljes név), a hitelesítéshez szükséges információkat (salt és jelszó-hash), valamint a havi költési limitet is.
Groups	A csoportokat kezeli, amelyekbe a felhasználók beléphetnek. Minden csoportnak van neve, típusa (pl. család, baráti társaság, ez szabad szöveges érték), leírása, létrehozó felhasználója és létrehozási dátuma.
GroupMemberships	A felhasználók csoporttagságait rögzíti. Megmutatja, hogy ki melyik csoporthoz tartozik, milyen szerepkörben (pl. admin vagy tag), és mióta tagja a csoportnak.

Tábla	Leírás
Roles	Az egyes csoporttagságokhoz rendelhető szerepköröket definiálja, például adminisztrátor vagy sima tag. A szerepkörök szabályozzák a felhasználók jogosultságait a csoportban. A csoport létrehozójának "Admin" jogköre lesz automatikusan, a felvett tagnak pedig "Member".
Transactions	A felhasználók által rögzített pénzügyi tranzakciókat tárolja. Minden tranzakcióhoz tartozik összeg, típus (bevétel vagy kiadás), kategória (pl. étel, szórakozás), dátum, leírás, és adott esetben kapcsolódó csoport is. A "GroupId" értéke null, ha személyes a tranzakció.
TransactionTypes	A tranzakciók típusait (bevétel, kiadás) határozza meg.
Categories	A tranzakciókhoz tartozó kategóriákat tartalmazza, mint például élelmiszer, közlekedés, szórakozás stb. Segíti a kiadások és bevételek részletesebb elemzését.
Savings	A felhasználók által kitűzött megtakarítási célokat tárolja. A célösszeg és a határidő opcionális adatok, tehát lehet null értékük. A "GroupId" értéke is null, ha személyes megtakarítás.
Invites	A csoportokba történő meghívásokat kezeli. Menti, hogy ki küldött meghívót kinek, melyik csoportba, és hogy a meghívás elfogadásra került-e. Az elfogadott meghívók "Accepted" értéke 1-re vált a default 0-ról. Elutasítás esetén a meghívó törlődik a táblából.

3.1. táblázat. Adatbázis táblák magyarázata

### 3.4.2. Backend - Adatbázis kapcsolat

A backend és az adatbázis kapcsolata relatív egyszerűen megvalósítható az ASP.NET keretrendszerben. A NuGet package managerből az EntityFramework egyes csomagjait, illetve a MySQL Connector csomagot letöltve már alig van dolgunk. Az appsettings.json fájlban tudjuk a connection stringet definiálni.

```
1 "ConnectionStrings": {  
2   "MySQLConnection": "server=localhost;port=3306;database=  
   betterspend;user=root;password=<<password>>;"  
3 }
```

### 3.8. forráskód. Adatbázis Connection String

majd a Program.cs fájlban tudunk ténylegesen csatlakozni:

```
1 // Get MySQL connection string from configuration  
2 string connectionString = builder.Configuration.  
   GetConnectionString("MySQLConnection");  
3  
4 // Add DB Context to the application  
5 builder.Services.AddDbContext<ApplicationDbContext>(options =>  
6 {  
7   options.UseMySQL(connectionString, ServerVersion.AutoDetect(  
       connectionString));  
8 });
```

### 3.9. forráskód. Program.cs: Adatbázis csatlakozás

A tényleges adatbázis modellje:

```
1 using WebApplication1.Model;  
2  
3 namespace WebApplication1.Data  
4 {  
5   public class ApplicationDbContext : DbContext  
6   {  
7     public ApplicationDbContext(DbContextOptions<  
       ApplicationDbContext> options)  
8       : base(options)  
9     {  
10    }  
11  
12     public DbSet<User> Users { get; set; }  
13     public DbSet<Transaction> Transactions { get; set; }  
14     public DbSet<Category> Categories { get; set; }  
15     public DbSet<Role> Roles { get; set; }  
16     public DbSet<GroupMembership> GroupMemberships { get; set; }  
17     public DbSet<Group> Groups { get; set; }  
18     public DbSet<Invite> Invites { get; set; }
```

```
19     public DbSet<Saving> Savings { get; set; }
20     public DbSet<TransactionType> TransactionTypes { get; set; }
21
22 }
23 }
```

### 3.10. forráskód. Adatbázis modell

Ezután már az adatbázisunk össze van kötve teljesen az alkalmazással. ASP.NET-ben közvetlenül az adatbázishoz (lekérdezésekkel, parancsokkal) nem tudunk hozzáférni. Amihez hozzá tudunk férni azok a modell fájlok (pl. Model/User). Ezeket tudjuk módosítani LINQ segítségével egyszerűen, és ez szinkronizálni fogja a tényleges adatbázisunkkal.

### 3.4.3. Migrations

A későbbi fejlesztések során migrációkat használhatunk az adatbázis verziókövetésére és a változtatások biztonságos, konzisztens bevezetésére. A migrációk használata Entity Framework Core segítségével történik, ahol a DbContext osztály definiálja az adatbázis szerkezetét. A módosításokat a dotnet ef migrations add paranccsal lehet rögzíteni, majd a dotnet ef database update parancs segítségével alkalmazni az adatbázisra. Ez lehetővé teszi a séma változásainak verziókövetését és biztonságos frissítését.

## 3.5. Tesztek

<b>Teszteset azonosító</b>	TC001
<b>Funkció</b>	Regisztráció
<b>Bemenet</b>	Valid e-mail cím, egyedi felhasználónév, teljes név, jelszó
<b>Elvárt eredmény</b>	Sikeres regisztráció, átirányítás a bejelentkezés felületre, visszajelző üzenet a felhasználónak
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.2. táblázat. Sikeres regisztráció

<b>Teszteset azonosító</b>	TC002
<b>Funkció</b>	Regisztráció
<b>Bemenet</b>	Üres mezők beküldése vagy invalid e-mail cím, vagy nem elég erős jelszó
<b>Elvárt eredmény</b>	Hibajelzés, regisztráció sikertelen
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.3. táblázat. Sikertelen regisztráció

<b>Teszteset azonosító</b>	TC003
<b>Funkció</b>	Bejelentkezés
<b>Bemenet</b>	Helyes felhasználónév - jelszó páros megadása
<b>Elvárt eredmény</b>	Belépteti a felhasználót az oldalra
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.4. táblázat. Sikeres bejelentkezés

<b>Teszteset azonosító</b>	TC004
<b>Funkció</b>	Bejelentkezés
<b>Bemenet</b>	Üres mezők beküldése, vagy helytelen felhasználónév-jelszó páros
<b>Elvárt eredmény</b>	Hibajelzés, a bejelentkezés sikertelen
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.5. táblázat. Sikertelen bejelentkezés

<b>Teszteset azonosító</b>	TC005
<b>Funkció</b>	Profil oldal - adatok módosítása
<b>Bemenet</b>	Új, egyedi felhasználónév megadása VAGY új, valid e-mail cím megadása VAGY új teljes név megadása
<b>Elvárt eredmény</b>	Sikeres módosítás
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.6. táblázat. Személyes adatok sikeres módosítása

<b>Teszteset azonosító</b>	TC006
<b>Funkció</b>	Profil oldal - adatok módosítása
<b>Bemenet</b>	Helytelen e-mail cím megadás, foglalt felhasználónév megadás, üres mező beküldése
<b>Elvárt eredmény</b>	Hibajelzés, a módosítás nem lép érvénybe
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.7. táblázat. Személyes adatok sikertelen módosítása

<b>Teszteset azonosító</b>	TC007
<b>Funkció</b>	Profil oldal - Jelszó módosítása
<b>Bemenet</b>	Helyes régi jelszó, új jelszó, új jelszó megerősítése egyezően, új jelszó elég erős
<b>Elvárt eredmény</b>	Sikeres jelszováltás, visszajelzés a felhasználónak
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.8. táblázat. Sikeres jelszó módosítás

<b>Teszteset azonosító</b>	TC008
<b>Funkció</b>	Profil oldal - Jelszó módosítása
<b>Bemenet</b>	Helytelen régi jelszó, új jelszó és megerősítés nem egyező, nem elég erős új jelszó
<b>Elvárt eredmény</b>	Hibajelzés, módosítás nem lép érvénybe
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.9. táblázat. Sikertelen jelszó módosítás

<b>Teszteset azonosító</b>	TC009
<b>Funkció</b>	Új bevétel rögzítése
<b>Bemenet</b>	Helyes amount (pl. 5000) és description (pl. "Fizetés")
<b>Elvárt eredmény</b>	Sikeres mentés, üzenet: "Transaction saved!"
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.10. táblázat. Új bevétel helyes adatokkal

<b>Teszteset azonosító</b>	TC010
<b>Funkció</b>	Új bevétel rögzítése
<b>Bemenet</b>	Üres vagy hibás amount/description mezők
<b>Elvárt eredmény</b>	Hibajelzés: "Failed to create transaction!" (Tranzakció mentése sikertelen.)
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.11. táblázat. Új bevétel hibás adatokkal

<b>Teszteset azonosító</b>	TC011
<b>Funkció</b>	Kiadás rögzítése, ha havi limit = 0
<b>Bemenet</b>	Új kiadás mentése 0 költségű limit mellett
<b>Elvárt eredmény</b>	Üzenet: "Transaction saved!" (limit státusz nélkül)
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.12. táblázat. Kiadás rögzítése havi limit nélkül



<b>Teszteset azonosító</b>	TC012
<b>Funkció</b>	Kiadás rögzítése, ha van havi limit
<b>Bemenet</b>	Új kiadás mentése havi limit mellett
<b>Elvárt eredmény</b>	Limit státusz megjelenik: zöld (ha van keret), sárga (ha túllépés történt)
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.13. táblázat. Kiadás rögzítése havi limittel

<b>Teszteset azonosító</b>	TC013
<b>Funkció</b>	Előzmény táblázat ellenőrzése
<b>Bemenet</b>	Új bevétel/kiadás rögzítése
<b>Elvárt eredmény</b>	Az új tranzakció megjelenik az előzmények között
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.14. táblázat. Új tranzakció megjelenése az előzmények között

<b>Teszteset azonosító</b>	TC014
<b>Funkció</b>	CSV batch feltöltés
<b>Bemenet</b>	Helyes formátumú CSV fájl feltöltése (típus, kategória, leírás, összeg, dátum)
<b>Elvárt eredmény</b>	Minden sor külön tranzakcióként mentésre kerül, üzenet: "Transactions saved!" (Tranzakciók elmentve!)
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.15. táblázat. CSV batch feltöltés helyes fájjal

<b>Teszteset azonosító</b>	TC015
<b>Funkció</b>	CSV batch feltöltés
<b>Bemenet</b>	Helytelen formátumú CSV fájl feltöltése (típus, kategória, leírás, összeg, dátum)
<b>Elvárt eredmény</b>	Hibaüzenet: "Error! «a pontos hiba»" (Hiba!)
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.16. táblázat. CSV batch feltöltés helytelen fájjal

<b>Teszteset azonosító</b>	TC016
<b>Funkció</b>	Havi költési limit módosítása
<b>Bemenet</b>	Új pozitív egész költési limit megadása (pl. 100000)
<b>Elvárt eredmény</b>	Sikeres módosítás, limit frissítése az adatbázisban
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.17. táblázat. Sikeres havi költési limit módosítása

<b>Teszteset azonosító</b>	TC017
<b>Funkció</b>	Havi költési limit módosítása
<b>Bemenet</b>	Érvénytelen költési limit megadása (pl. negatív szám)
<b>Elvárt eredmény</b>	Sikertelen módosítás
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.18. táblázat. Sikertelen havi költési limit módosítása

<b>Teszteset azonosító</b>	TC018
<b>Funkció</b>	Új megtakarítás létrehozása helyes adatok megadásával
<b>Bemenet</b>	Név, leírás, célösszeg (opcionális), határidő megadása (opcionális)
<b>Elvárt eredmény</b>	Sikeres megtakarítás létrejötte a megadott adatokkal
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.19. táblázat. Új megtakarítás létrehozása

<b>Teszteset azonosító</b>	TC019
<b>Funkció</b>	Összeg hozzáadása megtakarításhoz
<b>Bemenet</b>	Megtakarításhoz adott összeg megadása (pl. 10000)
<b>Elvárt eredmény</b>	Az összeg hozzáadódik a megtakarításhoz, frissül az aktuális összeg
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.20. táblázat. Összeg hozzáadása megtakarításhoz

<b>Teszteset azonosító</b>	TC020
<b>Funkció</b>	Összeg elvétele megtakarításból
<b>Bemenet</b>	Megtakarításból elvont összeg megadása (pl. 5000)
<b>Elvárt eredmény</b>	Az összeg levonásra kerül, frissül az aktuális összeg
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.21. táblázat. Összeg elvétele megtakarításból

<b>Teszteset azonosító</b>	TC021
<b>Funkció</b>	Megtakarítás törlése
<b>Bemenet</b>	Meglévő megtakarítás kártyán lévő kuka ikon megnyomása és a törlés megerősítése
<b>Elvárt eredmény</b>	A kiválasztott megtakarítás törlésre kerül
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.22. táblázat. Megtakarítás törlése

<b>Teszteset azonosító</b>	TC022
<b>Funkció</b>	Részletes riport letöltése
<b>Bemenet</b>	Letöltés gombra
<b>Elvárt eredmény</b>	A részletes riport fájl letöltésre kerül
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.23. táblázat. Részletes riport letöltése

<b>Teszteset azonosító</b>	TC023
<b>Funkció</b>	Év váltása az éves összefoglaló riporton
<b>Bemenet</b>	Másik év kiválasztása az éves riport nézetben
<b>Elvárt eredmény</b>	Az adott évre vonatkozó adatok jelennek meg
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.24. táblázat. Év váltása éves összefoglaló riporton

<b>Teszteset azonosító</b>	TC024
<b>Funkció</b>	Új csoport létrehozása
<b>Bemenet</b>	Csoportnév, típus, leírás megadása, létrehozás gomb megnyomása
<b>Elvárt eredmény</b>	A csoport sikeresen létrejön és megjelenik a listában
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.25. táblázat. Új csoport létrehozása

<b>Teszteset azonosító</b>	TC025
<b>Funkció</b>	Csoport törlése
<b>Bemenet</b>	Meglévő csoport törlésének megerősítése
<b>Elvárt eredmény</b>	A kiválasztott csoport törlésre kerül a listából
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.26. táblázat. Csoport törlése

<b>Teszteset azonosító</b>	TC026
<b>Funkció</b>	Csoportban új tranzakció rögzítése
<b>Bemenet</b>	Összeg és leírás megadása
<b>Elvárt eredmény</b>	A tranzakció megjelenik a táblázatban a felhasználó nevével és helyes adatokkal
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.27. táblázat. Csoportban tranzakció rögzítése

<b>Teszteset azonosító</b>	TC027
<b>Funkció</b>	Csoport elhagyása
<b>Bemenet</b>	Csoportból kilépés megerősítése
<b>Elvárt eredmény</b>	A felhasználó kikerül a csoportból
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.28. táblázat. Csoport elhagyása

<b>Teszteset azonosító</b>	TC028
<b>Funkció</b>	Tag meghívása csoporthoz
<b>Bemenet</b>	Új tag felhasználónevének megadása, meghívó küldése
<b>Elvárt eredmény</b>	Meghívó elküldve
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.29. táblázat. Tag meghívása csoporthoz - sikeres

<b>Teszteset azonosító</b>	TC029
<b>Funkció</b>	Tag meghívása csoporthoz
<b>Bemenet</b>	Nem létező felhasználónév, üres mezők megadása
<b>Elvárt eredmény</b>	Hibajelzés, meghívó nem kerül küldésre
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.30. táblázat. Tag meghívása csoporthoz - sikertelen

<b>Teszteset azonosító</b>	TC030
<b>Funkció</b>	Meghívó elfogadása
<b>Bemenet</b>	Meghívó elfogadása az értesítések fülön
<b>Elvárt eredmény</b>	A felhasználó bekerül a csoportba
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.31. táblázat. Meghívó elfogadása

<b>Teszteset azonosító</b>	TC031
<b>Funkció</b>	Meghívó elutasítása
<b>Bemenet</b>	Meghívó elutasítása az értesítések fülön
<b>Elvárt eredmény</b>	A felhasználó nem kerül be a csoportba, meghívás megszűnik
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.32. táblázat. Meghívó elutasítása

<b>Teszteset azonosító</b>	TC032
<b>Funkció</b>	Csoport megtakarítás hozzáadása
<b>Bemenet</b>	Új megtakarítás név és cél megadása csoport szinten
<b>Elvárt eredmény</b>	A csoport megtakarítás létrejön és megjelenik a listában
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.33. táblázat. Csoport megtakarítás hozzáadása

<b>Teszteset azonosító</b>	TC033
<b>Funkció</b>	Összeg elvétele csoport megtakarításból
<b>Bemenet</b>	Összeg megadása, amely levonásra kerül a csoport megtakarításból
<b>Elvárt eredmény</b>	Az aktuális összeg csökken a megadott összeggel
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.34. táblázat. Összeg elvétele csoport megtakarításból

<b>Teszteset azonosító</b>	TC034
<b>Funkció</b>	Összeg berakása csoport megtakarításba
<b>Bemenet</b>	Összeg megadása, amely hozzáadódik a csoport megtakarításhoz
<b>Elvárt eredmény</b>	Az aktuális összeg nő a megadott összeggel
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.35. táblázat. Összeg berakása csoport megtakarításba

<b>Teszteset azonosító</b>	TC035
<b>Funkció</b>	Csoport megtakarítás törlése
<b>Bemenet</b>	Kártya melletti kuka ikonra kattintás
<b>Elvárt eredmény</b>	A kiválasztott csoport megtakarítás törlődik
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.36. táblázat. Csoport megtakarítás törlése

<b>Teszteset azonosító</b>	TC036
<b>Funkció</b>	Kijelentkezés
<b>Bemenet</b>	Baloldali menüsávban található kijelentkezés gomb megnyomása
<b>Elvárt eredmény</b>	Munkamenet befejezése, átirányítás a bejelentkező felületre
<b>Tényleges eredmény</b>	Megfelel az elvárt eredménynek

3.37. táblázat. Kijelentkezés

## 4. fejezet

# Összegzés

Összegezve tehát az alkalmazás elsősorban a spórolást, tudatos pénzköltést szeretné promótálni, és erre kínál magánszemélyeknek vagy háztartásoknak/családoknak egy hasznos eszközt. A fő funkció a költségeink és bevételeink vezetése, amely az első lépés a tudatosság és a pénzügyi szabadság irányába. Ezután a következő lépésnek tekinthetjük azt, amikor már az okosabb költekezés következményeképpen többlet bevételünk is lesz, melyet elkezdünk félrerakni. A megtakarításainkat is nyomon tudjuk követni az alkalmazáson belül, sőt még kimutatásokat is láthatunk, melyek rendszeres tanulmányozásával különböző mintázatokat vélhetünk felfedezni pénzügyi szokásainkból, amelyekből a tanulságokat levonva, tovább tudjuk optimalizálni az anyagiak kezelését.

A webalkalmazást egy C# alapú keretrendszerrel (ASP.NET Web App, Razor Pages) valósítottam meg, amely egy adatbázissal is természetesen összeköttetésben áll. A fejlesztés során végig törekedtem a szabványos és jól strukturált felépítésre, a Clean Code elveinek, valamint az adatbázistervezés bevált gyakorlatainak követésére. A felhasználói felület modern és letisztult megjelenést kapott, így nem csupán az élményre helyeztem a hangsúlyt, hanem az alkalmazás jövőbeni továbbfejleszthetőségére, a kód olvashatóságára és könnyű karbantarthatóságára is.

## 5. fejezet

# Továbbfejlesztési lehetőségek

Egy alkalmazást véleményem szerint gyakorlatilag soha nem lehet teljesen befejezettnek tekinteni. Mindig lehet hova fejleszteni, és mindig lesznek új ötletek hozzá, illetve ha más nem is, újabb technológiák egészen biztosan. És egy programozó is folyamatosan tanul, egyre jobb és jobb termékeket igyekszik kiadni a kezei közül.

Ezen webalkalmazás továbbfejlesztésére számos olyan ötletem van, amelyek megvalósítása már nem esett jelen projekt keretei közé. A következő fejezetben ezeket fogom röviden kifejteni.

### 5.1. Statisztikák

A "Reports" oldal jelenleg 2 darab egyszerű, összegző diagramot tartalmaz csak, de itt azt gondolom sokkal nagyobb potenciál is lehetne. Akár AI modellekkel előállítani statisztikákat, elemezni a költségi trendeket, és ezáltal személyre szabott pénzügyi tanácsadást biztosítani a felhasználónak.

Megjegyezném, hogy AI integráció nélkül is érdemes lenne a statisztika sokszínű eszközeit jobban kihasználni, és az adattengerből hasznos következtetéseket levonni.

### 5.2. Tranzakciók

A tranzakciók lekövetésére és megjelenítésére is számos jobb módszer lehetne, mint amit én ebben a projektben megvalósítottam. A felhasználónak például jelentősen megkönnyítené a dolgát, ha nem manuálisan kellene bevinnie mindig a kiadásait. Erre nyújthat megoldást egy beépített AI technológia, amely egy lefotózott

blokk/számla tartalmát értelmezni tudja, és automatikusan, tételenként, rögzíti az alkalmazásba. Ezen felül még lehetne ismétlődő bevételeket és kiadásokat beállítani (például fizetés, lakbér, stb.), ezzel is csökkentvén a felhasználói beavatkozást.

### 5.3. Csoportok

A csoportos funkciókban szerintem rengeteg potenciál bújkik meg. A közös megtakarítások lehetőséget adnak arra, hogy barátok, családtagok vagy akár munkatársak együtt kezeljék bizonyos pénzügyeiket, megtakarításaikat. Az alap koncepción nem sokat változtatnék, inkább a kivitelezésen és a plusz funkciókon. A csoportos takarékoskodás nemcsak anyagi szempontból lehet hatékonyabb, hanem pszichológiai szempontból is motiváló. Ha mások is részt vesznek benne, az ösztönözheti az embert, hogy ő is kitartóbb legyen. Ehhez kapcsolódhatnak különféle motivációs taktikák, mint például haladási grafikonok, ranglisták, vagy gamification-elemek (jelvények, célkitűzések, stb.).

Illetve a jövőben érdemes lehet olyan funkciókat is beépíteni, mint például a csoportos üzenetküldés és kommentelés, és automatikus értesítések a tagoknak a határidőkről, befizetésekről.

Ez a modul nemcsak a közösségi élményt erősítené, hanem a felhasználók aktivitását és elköteleződését is jelentősen növelhetné.

### 5.4. UI

A felhasználói élmény szempontjából kiemelten fontos, hogy egy alkalmazás átlátható, letisztult és esztétikus legyen. A modern webes elvárások világában a vanilla HTML, CSS és JavaScript (még Bootstrap-tel kiegészítve is) már nem feltétlenül elegendő ahhoz, hogy igazán igényes, vizuálisan is kiemelkedő felhasználói felületet hozzunk létre. Emellett például egy React vagy más frontend keretrendszer alkalmazása nemcsak látványosabb UI-t tesz lehetővé, de strukturáltabb, könnyebben karbantartható kódot is eredményez. Ezért érdemes lehet a felhasználói felületet (UI<sup>1</sup>) egy modern frameworkben újraalkotni.

---

<sup>1</sup>UI = User Interface (felhasználói felület)



# Ábrák jegyzéke

2.1. Használati eset diagram: Általános . . . . .	5
2.2. Screenshot: Regisztrációs felület . . . . .	6
2.3. Screenshot: Bejelentkező felület . . . . .	7
2.4. Screenshot: Profil felület . . . . .	7
2.5. Screenshot: Tracker felület navigációs sáv . . . . .	8
2.6. Screenshot: Transactions lap . . . . .	9
2.7. Screenshot: Tömeges feltöltés lap . . . . .	10
2.8. Screenshot: Monthly Spending Limit lap . . . . .	11
2.9. Screenshot: Gyűjtés létrehozása . . . . .	12
2.10. Screenshot: Savings felület . . . . .	13
2.11. Screenshot: Aktuális havi riport . . . . .	13
2.12. Screenshot: Éves összesítő riport . . . . .	14
2.13. Screenshot: Create Group oldal . . . . .	15
2.14. Screenshot: My Group oldal . . . . .	15
2.15. Screenshot: Screenshot: Csoport műveletek . . . . .	16
2.16. Screenshot: Értesítések . . . . .	16
3.1. Projekt struktúra . . . . .	18
3.2. Pages struktúra . . . . .	24
3.3. Adatbázis struktúra . . . . .	26

# Táblázatok jegyzéke

2.1. Böngésző támogatás . . . . .	5
2.2. Fiók műveletek . . . . .	6
2.3. Profil oldal . . . . .	7
2.4. Transactions lap . . . . .	9
2.5. Batch upload lap . . . . .	10
2.6. Monthly Spending limit lap . . . . .	11
2.7. A Savings oldal . . . . .	12
3.1. Adatbázis táblák magyarázata . . . . .	27
3.2. Sikeres regisztráció . . . . .	29
3.3. Sikertelen regisztráció . . . . .	30
3.4. Sikeres bejelentkezés . . . . .	30
3.5. Sikertelen bejelentkezés . . . . .	30
3.6. Személyes adatok sikeres módosítása . . . . .	30
3.7. Személyes adatok sikertelen módosítása . . . . .	30
3.8. Sikeres jelszó módosítás . . . . .	31
3.9. Sikertelen jelszó módosítás . . . . .	31
3.10. Új bevétel helyes adatokkal . . . . .	31
3.11. Új bevétel hibás adatokkal . . . . .	31
3.12. Kiadás rögzítése havi limit nélkül . . . . .	31
3.13. Kiadás rögzítése havi limittel . . . . .	32
3.14. Új tranzakció megjelenése az előzmények között . . . . .	32
3.15. CSV batch feltöltés helyes fájlal . . . . .	32
3.16. CSV batch feltöltés helytelen fájlal . . . . .	32
3.17. Sikeres havi költési limit módosítása . . . . .	32
3.18. Sikertelen havi költési limit módosítása . . . . .	33
3.19. Új megtakarítás létrehozása . . . . .	33
3.20. Összeg hozzáadása megtakarításhoz . . . . .	33

3.21. Összeg elvétele megtakarításból . . . . .	33
3.22. Megtakarítás törlése . . . . .	33
3.23. Részletes riport letöltése . . . . .	33
3.24. Év váltása éves összefoglaló riporton . . . . .	34
3.25. Új csoport létrehozása . . . . .	34
3.26. Csoport törlése . . . . .	34
3.27. Csoportban tranzakció rögzítése . . . . .	34
3.28. Csoport elhagyása . . . . .	34
3.29. Tag meghívása csoporthoz - sikeres . . . . .	34
3.30. Tag meghívása csoporthoz - sikertelen . . . . .	35
3.31. Meghívó elfogadása . . . . .	35
3.32. Meghívó elutasítása . . . . .	35
3.33. Csoport megtakarítás hozzáadása . . . . .	35
3.34. Összeg elvétele csoport megtakarításból . . . . .	35
3.35. Összeg berakása csoport megtakarításba . . . . .	35
3.36. Csoport megtakarítás törlése . . . . .	36
3.37. Kijelentkezés . . . . .	36

# Forráskódjegyzék

2.1. Tömeges feltöltés: csv fájl struktúra . . . . .	10
3.1. PDF generálás példa . . . . .	19
3.2. PDF generálás API végpont . . . . .	19
3.3. Chart.js alapú havi diagram kirajzolása . . . . .	20
3.4. Route-ok konfigurálása . . . . .	21
3.5. Frontend modell egy alkalmazása . . . . .	21
3.6. Layout extension ASP.NET keretrendszerben . . . . .	24
3.7. Layout extension ASP.NET keretrendszerben . . . . .	24
3.8. Adatbázis Connection String . . . . .	28
3.9. Program.cs: Adatbázis csatlakozás . . . . .	28
3.10. Adatbázis modell . . . . .	28