győri szakképzési centrum

Jedlik Ányos
Gépipari és Informatikai
Technikum és Kollégium

9021 Győr, Szent István út 7.
+36 (96) 529-480
+36 (96) 529-448

OM: 203037/003
jedlik@jedlik.eu
www.jedlik.eu

Záródolgozat feladatkiírás

Tanuló(k) neve:	Somlói Dávid, Trifusz Huba, Verba Viktor
Képzés:	nappali
Szak:	5 0613 12 03 Szoftverfejlesztő és -tesztelő technikus

A záródolgozat címe:

Elevate

Konzulens:	Sándor László
Beadási határidő:	2025. 04. 15.

Győr, 2025. 02. 01

Módos Gábor
igazgató

győri szakképzési centrum

Jedlik Ányos
Gépipari és Informatikai
Technikum és Kollégium

9021 Győr, Szent István út 7.
+36 (96) 529-480
+36 (96) 529-448

OM: 203037/003
jedlik@jedlik.eu
www.jedlik.eu

Konzultációs lap

	A konzultáció		Konzulens aláírása
	ideje	témája	
1.	2025.02.15.	Témaválasztás és specifikáció	
2.	2025.03.14.	Záródolgozat készültségi fokának értékelése	
3.	2025.04.15.	Dokumentáció véglegesítése	

Tulajdonosi nyilatkozat

Ez a dolgozat a saját munkánk eredménye. Dolgozatunk azon részeit, melyeket más szerzők munkájából vettünk át, egyértelműen megjelöltük.
Ha kiderülne, hogy ez a nyilatkozat valótlan, tudomásul vesszük, hogy a szakmai vizsgabizottság a szakmai vizsgáról kizár minket és szakmai vizsgát csak új záródolgozat készítése után tehetünk.

Győr, 2025. április 15.

_____
tanuló aláírása

_____
tanuló aláírása

_____
tanuló aláírása

Tartalomjegyzék

1. A projektről

A téma kiválasztásánál arra törekedtünk, hogy egy, a hétköznapi élet során alkalmazható szoftvert készítsünk. Több opció is felmerült, azonban végül egy szokásformáló felület mellett döntöttünk, amit Elevate-nek neveztünk el, az egészséges, felemelő életmód jegyében. Az Elevate ösztönzi a felhasználókat, hogy új, pozitív szokásokat vezessenek be, miközben hatékonyan követhetik saját fejlődésüket, emellett hozzájárul életminőségük javításához és a fenntartható fejlődéshez.

1.1 Az Elevate célja és funkciói

A szoftver célja, hogy a kliens az általa kívánt szokásokat fejlessze, vagy újakat építsen be a napirendjébe. Például, ha a felhasználó a dohányzásról szeretne leszokni, akkor monitorozni tudja a fogyasztását, nyomon követheti haladását a felállított célja felé. Nem csak a rossz szokások követését biztosítja az applikáció, pozitív célokat is ki lehet tűzni, mint “Napi 10 fekvőtámasz" vagy “Hetente kitakarítani”. Új szokásokat napi, heti, havi vagy egyéni rendszerességgel lehet felvenni. Az egyéni opció választása esetén a felhasználó megadhatja, hogy a hét melyik napjain szeretné elvégezni a feladatot. Egy szokás tartásához elengedhet-etlen, hogy a beállított gyakorisággal teljesítsük a kitűzött kihívásokat. Ennek megkönnyítése érdekében az Elevate egy naptárszerű nézetben, színkódolva jeleníti meg a teendőket és emlékeztet azok elvégzésére. Minden elvégzett feladat után növekedik az adott szokáshoz tartozó streakje, ezzel ösztönözve arra, hogy megszakítás nélkül, konzisztensen küzdjön a céljáért. Ezen felül barátokat is hozzáadhatunk, akikkel kihívásokat indíthatunk, így még inkább motiválva egymást a célok elérésére.

2. Weboldal

3. Mobil Applikáció

4. Adatbázis

4.1 Adatbázis tervezés

Az Elevate két különböző adatbázis rendszert támogat a különböző környezetekben való futtatáshoz:

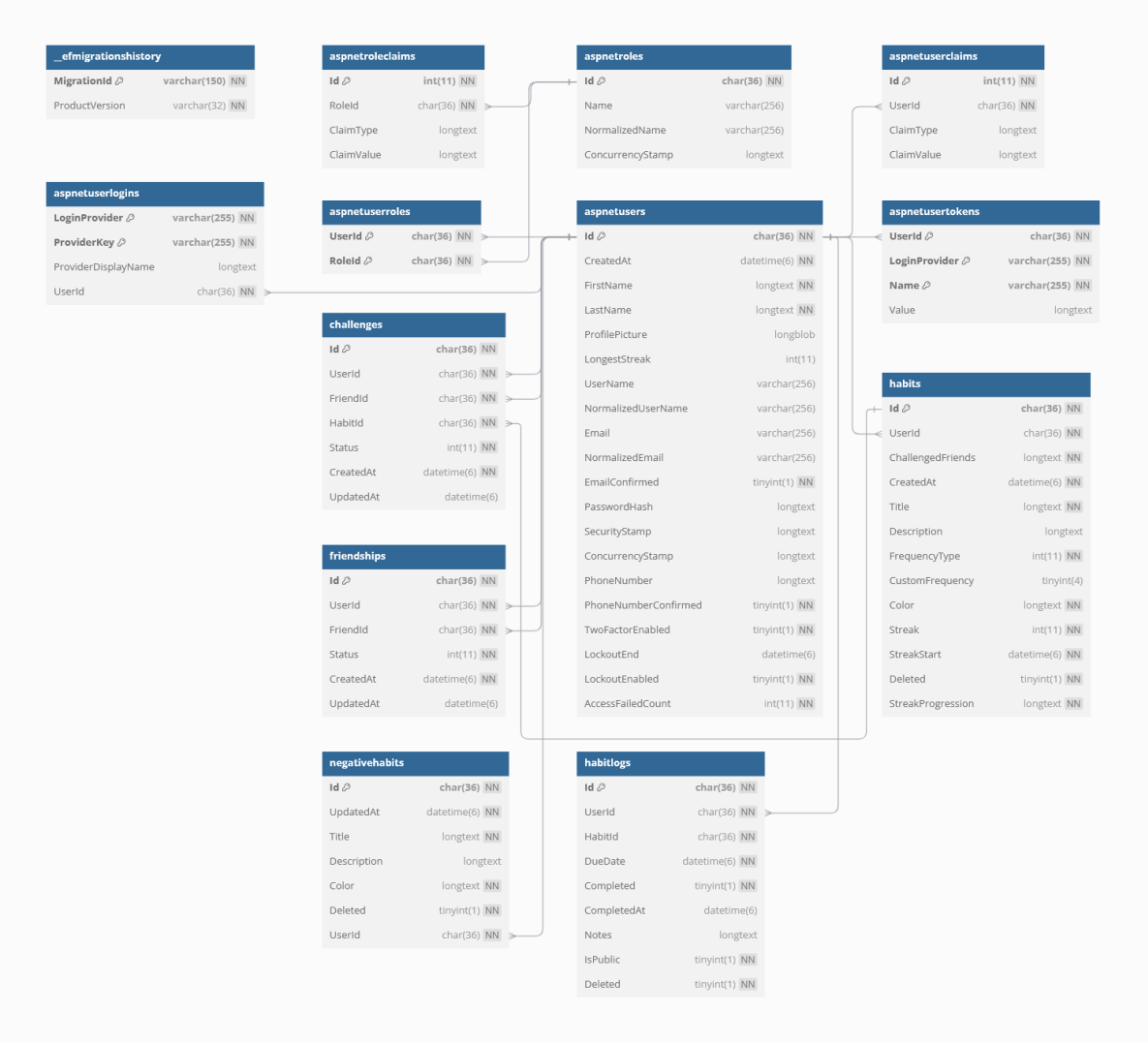
- Fejlesztési környezetben: MySQL
- Production környezetben: PostgreSQL

Erre azért van így, mert a fejlesztést MySQL-el kezdtük, majd a Koyeb-re való publikáláshoz szükségessé vált PostgreSQL kompatibilitás. Az adatbázis migrációk kezelésére az Entity Framework Core migrációs rendszerét használjuk, amely lehetővé teszi a séma verziókövetését és az adatbázis automatikus frissítését, ez kisebb módosításokkal mindkét adatbázis környezettel megfelelően működik.

4.2 Entitások és kapcsolatok

Az adatbázis séma a következő fő táblákat tartalmazza:

Emellett az ASP.NET Core Identity által biztosított felhasználói táblák is megtalálhatóak, amelyek a felhasználók kezeléséért felelősek. A teljes adatbázis séma így néz ki:



4.3 Adatbázis biztonság

- A jelszavak hash-elve tárolódnak az adatbázisban (ASP.NET Core Identity)
- Adatbázis migrációk verziókövető rendszerben tárolva
- A kapcsolatok integritása constraint-ekkel biztosítva
- Indexek használata a gyakori lekérdezések optimalizálására

4.4 Adatbázis elérés

Az adatbázis elérését a DbConnectionManager osztály biztosítja az alábbi módon:

DbConnectionManager.cs

```
24  public DbConnection GetOpenConnection()
25  {
26      if (Environment.GetEnvironmentVariable("ASPNETCORE_ENVIRONMENT") ==
        ↪ "Production")
27      {
28          var connection = new NpgsqlConnection(GetConnectionString());
29          connection.Open();
30          return connection;
31      }
32      else
33      {
34          var connection = new MySqlConnection(GetConnectionString());
35          connection.Open();
36          return connection;
37      }
38  }
```

Majd az így kapott kapcsolattal a DbContext osztály alkot egy, a későbbiekben feldolgozható adatszerkezetet.

ElevateDbContext.cs

```
61      base.OnModelCreating(modelBuilder);
62
63      if (DbConnectionManager.IsProduction())
64      {
65          modelBuilder.UseIdentityAlwaysColumns();
66
67          modelBuilder.Entity<ApplicationUser>().ToTable("aspnetusers");
68          modelBuilder.Entity<IdentityRole<Guid>>().ToTable("aspnetroles");
69          modelBuilder.Entity<IdentityUserRole<Guid>>().ToTable("aspnetuserroles");
70          modelBuilder.Entity<IdentityUserClaim<Guid>>().ToTable("aspnetuserclaims");
71          modelBuilder.Entity<IdentityUserLogin<Guid>>().ToTable("aspnetuserlogins");
72          modelBuilder.Entity<IdentityRoleClaim<Guid>>().ToTable("aspnetroleclaims");
73          modelBuilder.Entity<IdentityUserToken<Guid>>().ToTable("aspnetusertokens");
74
75          modelBuilder.Entity<HabitModel>().ToTable("habits");
76          modelBuilder.Entity<HabitLogModel>().ToTable("habitlogs");
77          modelBuilder.Entity<ChallengeModel>().ToTable("challenges");
78          modelBuilder.Entity<FriendshipModel>().ToTable("friendships");
79          modelBuilder.Entity<NegativeHabitModel>().ToTable("negativehabits");
80      }
81
82      modelBuilder.Entity<ApplicationUser>(b =>
83      {
84          b.HasKey(u => u.Id);
85          b.HasIndex(u => u.Email).IsUnique();
86      });
```

5. Backend

5.1 Technológia

Az Elevate backend rendszere ASP.NET Core alapú, Entity Framework Core ORM-mel. Az adatbázis és a backend kapcsolata model first elv alapján lett létrehozva. Az API RESTful elvek alapján lett kialakítva és a CRUD (Create, Read, Update, Delete) műveleteket valósítja meg.

5.2 Architektúra

A backend a következő komponensekből épül fel:

- **Modellek** - Az adatmodelleket és adatbázis entitásokat reprezentálják
- **DTO-k (Data Transfer Objects)** - Adatok átvitelére szolgáló objektumok a rétegek között, illetve a kliens és szerver között
- **Repository-k** - Az adatbázissal való kommunikációért felelősek, CRUD műveletek végrehajtása
- **Kontrollerek** - A kérések feldolgozása, autentikáció és authorizáció kezelése, valamint a válaszok generálása
- **Service-k** - Az üzleti logika megvalósítása
- **Middleware** - Kivételek kezelése és egyéb előfeldolgozási feladatok
- **Segédosztályok** - Általános funkciók és segédszolgáltatások

5.3 Végpontok

A Backend API részletes dokumentációja a Swagger felületen érhető el. Az alábbiakban a főbb végpontok láthatóak:

- **Autentikáció**
  - Regisztráció (POST /api/auth/register)
  - Bejelentkezés (POST /api/auth/login)
- **Felhasználó**
  - Felhasználó adatainak lekérése email alapján (GET /api/user)
  - Felhasználó adatainak lekérése id alapján (GET /api/user/:id)

  - Felhasználó adatainak frissítése (PATCH /api/user/:id)

- **Szokások**
  - Szokások listázása (GET /api/habit)
  - Szokás lekérése azonosító alapján (GET /api/habit/:id)
  - Új szokás létrehozása (POST /api/habit)
  - Szokás módosítása (PATCH /api/habit/:id)
  - Szokás törlése (DELETE /api/habit/:id)
  - Negatív szokások listázása (GET /api/habit/negative/:userId)
  - Negatív szokás léztrehozása (POST /api/habit/negative)
  - Negatív szokás módosítása (PATCH /api/habit/negative/:id)
  - Negatív szokás törlése (DELETE /api/habit/negative/:id)

- **Szokás napló**
  - Szokás naplók listázása (GET /api/habitlog)
  - Szokás napló lekérése azonosító alapján (GET /api/habitlog/:id)
  - Napi szokás naplók lekérése (GET /api/habitlog/:dueDate)
  - Szokás napló frissítése (PATCH /api/habitlog/:id)

- **Kihívások**
  - Kihívások lekérése felhasználó azonosító alapján (GET /api/challenge/:userId/challenges)
  - Kihívás meghívók listázása (GET /api/challenge/:userId/challenge-invites)
  - Elküldött kihívás meghívók listázása (GET /api/challenge/:userId/sent-challenge-invites)
  - Új kihívás létrehozása (POST /api/challenge)
  - Kihívás státuszának frissítése (PATCH /api/challenge)
  - Kihívás törlése (DELETE /api/challenge)

- **Feed**
  - Feed bejegyzések lekérése (GET /api/feed)

- **Barátok**
  - Barátok listázása (GET /api/friendship/:userId/friends)
  - Beérkezett barátkérések lekérése (GET /api/friendship/:userId/fried-requests)
  - Küldött barátkérések lekérése (GET /api/friendship/:userId/friend-requests-sent)
  - Barátkérés küldése (POST /api/friendship)
  - Barátkérés elfogadása/elutasítása (PATCH /api/friendship)
  - Barátság törlése (DELETE /api/friendship)

5.4 Autentikáció és biztonság

Az API biztonságos használatához JWT (JSON Web Token) alapú autentikáció van implementálva. A működése:

- A felhasználó bejelentkezéskor egy JWT tokent kap(aszimmetrikus titkosítással)
- A token érvényességi ideje korlátozott
- A védett végpontok eléréséhez a tokent minden kérés fejlécében el kell küldeni

A biztonság további rétegei:

- Input validáció
- CORS védelem (A mobil alkalmazás miatt enyhített)
- Jelszó titkosítás

6. Tesztelés

6.1 Unit tesztelés

A backend tesztelésére az xUnit keretrendszert használtuk, amellyel a Service réteg metódusait teszteltük mockolt adatokkal.

7. Források

- Microsoft Docs - Entity Framework Core (2025.04.15.)
- Koyeb Documentation (2025.04.15.)
- Swagger UI Documentation (2025.04.15.)