

Elevate Dokumentáció

Somlói Dávid

Trifusz Huba

Verba Viktor

April 15, 2025

Tartalomjegyzék

1	A projektről	2
1.1	Az Elevate célja	2
2	Weboldal	3
3	Mobil Applikáció	4
4	Adatbázis	5
4.1	Adatbázis tervezés	5
4.2	Entitások és kapcsolatok	5
4.3	Adatbázis biztonsági megfontolások	6
4.4	Adatbázis elérés	6
5	Backend	7
5.1	Technológia	7
5.2	Architektúra	7
5.3	Végpontok	7
5.4	Autentikáció és biztonság	8
6	Tesztelés	10

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

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 és különféle jutal-makat kap, ha tartja a felállított célját. Nem csak a rossz szokások követését biztosítja az applikáció, pozitív célokat is ki lehet tűzni, mint "Napi 10 fekvőtámasz" vagy "Hetente kitakarítani". Egy szokás tartásához elengedhetetlen, hogy a beállított gyakorisággal teljesítsük a kitűzött kihívásokat. Ennek megkönnyítése érdekében az Elevate egy naptárszerű nézetben jeleníti meg a teendőket és emlékeztet azok elvégzésére.

2. Weboldal

3. Mobil Applikáció

4. Adatbázis

4.1 Adatbázis tervezés

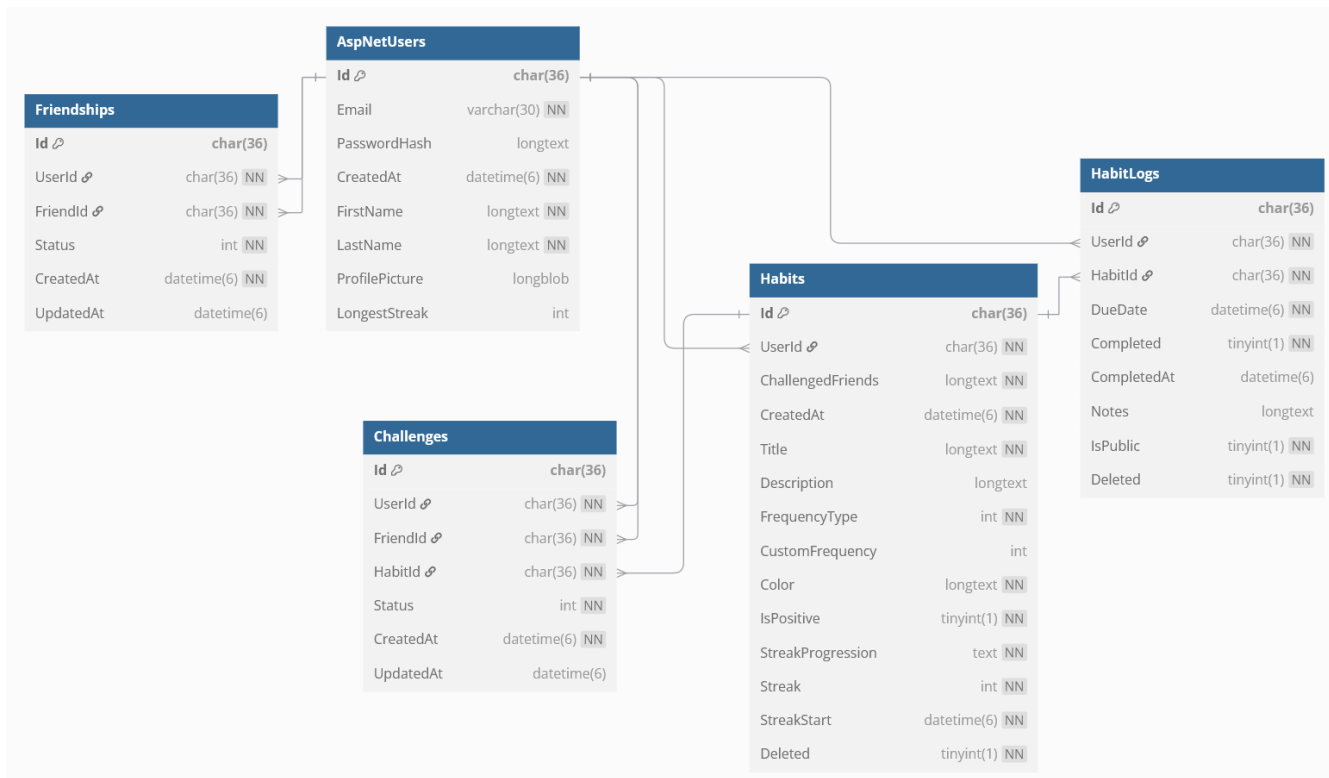
Az Elevate két különböző adatbázis rendszert támogat a különböző környezetekben való futtatáshoz:

- 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:



4.3 Adatbázis biztonsági megfontolások

- 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") ==
27         "Production")
28     {
29         var connection = new NpgsqlConnection(GetConnectionString());
30         connection.Open();
31         return connection;
32     }
33     else
34     {
35         var connection = new MySqlConnection(GetConnectionString());
36         connection.Open();
37         return connection;
38     }
39 }
```

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

```
ElevateDbContext.cs

61 protected override void OnModelCreating(ModelBuilder modelBuilder)
62 {
63     base.OnModelCreating(modelBuilder);
64
65     if (DbConnectionManager.IsProduction())
66     {
67         modelBuilder.UseIdentityAlwaysColumns();
68
69         modelBuilder.Entity<ApplicationUser>().ToTable("aspnetusers");
70         modelBuilder.Entity<IdentityRole<Guid>>().ToTable("aspnetroles");
71         modelBuilder.Entity<IdentityUserRole<Guid>>().ToTable("aspnetuserroles");
72         modelBuilder.Entity<IdentityUserClaim<Guid>>().ToTable("aspnetuserclaims");
73     }
74 }
```

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 autorizáció 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)
- 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/friend-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 token-t kap (aszimmetrikus titkosítással)

- A token érvényességi ideje korlátozott
- A védett végpontok eléréséhez a token minden kérés fejlécében el kell küldeni

A biztonság további rétegei:

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

6. Tesztelés