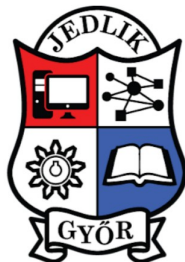




győri szakképzési centrum

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.

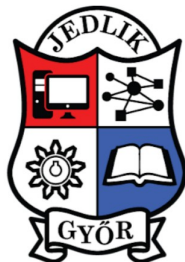
Győr, 2025. 02. 01

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	5
1.1	Az Elevate célja és funkciói	5
2	Weboldal	6
2.1	Projekt Beállítása	6
2.1.1	Előfeltételek	6
2.1.2	Telepítés	6
2.2	Mappastruktúra	7
2.2.1	Áttekintés a mappastruktúráról	7
2.3	Oldalak	7
2.3.1	Dashboard	7
2.3.2	Friends	9
2.3.3	Habits	9
2.3.4	Feed	10
2.4	Komponensek	10
2.4.1	Áttekintés	10
2.4.2	Funkcionalitás	11
2.5	Stílusok	12
2.5.1	Változók (variables.scss)	12
2.5.2	Űrlapok és Input Mezők (inputs.scss)	13
3	Mobil Applikáció	14
4	Adatbázis	15
4.1	Adatbázis tervezés	15
4.2	Entitások és kapcsolatok	15
4.3	Adatbázis biztonság	16
4.4	Adatbázis elérés	16
5	Backend	18
5.1	Technológia	18
5.2	Architektúra	18
5.3	Végpontok	18
5.3.1	Autentikáció	18
5.3.2	Felhasználó	19
5.3.3	Szokások	19
5.3.4	Szokás napló	19
5.3.5	Kihívások	19
5.3.6	Feed	19

5.3.7	Barátok	20
5.4	Autentikáció és biztonság	20
6	Tesztelés	21
6.1	Weboldal tesztelése	21
6.1.1	Bejelentkezési oldal tesztek (<code>login.spec.ts</code>)	21
6.1.2	Fiók létrehozása oldal tesztek (<code>create-account.spec.ts</code>)	21
6.1.3	Dashboard oldal tesztek (<code>task-view.cy.ts</code>)	21
6.1.4	Barátok oldal tesztek (<code>friends-page.cy.ts</code>)	22

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 elengedhetetlen, 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

2.1 Projekt Beállítása

2.1.1 Előfeltételek

A projekt futtatásához szükséges előfeltételek, eszközök:

- Node.js ($\geq 14.x$)
- npm ($\geq 6.x$)

2.1.2 Telepítés

1. Klónozd a repozitóriumot:

```
git clone https://github.com/hubatrifusz/elevate.git
```

2. Telepítsd a függőségeket:

```
cd elevate  
cd web-frontend  
npm install
```

3. Indítsd el a fejlesztői szerveret:

```
ng serve -o
```

Az alkalmazás elérhető lesz a `http://localhost:4200` címen

2.2 Mappastruktúra

2.2.1 Áttekintés a mappastruktúráról

```
/web-frontend
|-- /public          # Publikus fájlok (pl. képek)
|-- /src
|   |-- /components  # Újrahasználható UI komponensek
|   |-- /guards      # Biztonsági őrök (pl. autentikáció)
|   |-- /models      # Adatmodellek
|   |-- /pages       # Oldal komponensek (pl. HomePage, Dashboard)
|   |-- /services    # API hívások és üzleti logika
|   |-- index.html   # Alkalmazás belépési pontja
|   |-- main.ts      # Alkalmazás belépési pontja
|   |-- styles.scss  # Globális stílusok
|-- /assets          # Statikus fájlok (pl. képek, betűtípusok)
|   |-- /fonts       # Betűtípusok
|   |-- /styles      # Stílus fájlok
|-- /environments    # Környezeti konfigurációk
|   |-- environment.ts
|-- angular.json      # Angular konfigurációs fájl
|-- package.json      # Projekt metaadatok és függőségek
```

2.3 Oldalak

2.3.1 Dashboard

Leírás

A Dashboard oldal a felhasználó napi teendőit és a negatív szokásokat jeleníti meg.

Fő komponensek

- **Task View** – A napi feladatok listázása, új szokások felvétele, meglévő szokások törlése vagy módosítása.
- **Negative Habits** – (külön komponens, részletezése később következik).

Task View funkciói

- **Napi dátum kezelése** – A felhasználó az aktuális, előző vagy következő napra navigálhat.
- **Napi szokások listázása** – A mai napra rögzített szokások megjelenítése.
- **Új szokás hozzáadása** – Pozitív vagy negatív szokás rögzítése úrlapon keresztül.
- **Szokások törlése és módosítása** – A meglévő szokások frissítése vagy eltávolítása.

Task komponens (Pozitív szokás)

A napi feladatokat az `TaskComponent` jeleníti meg. Ez a komponens a következőket tartalmazza:

- **Feladat bejelölése** – A felhasználó checkbox segítségével jelezheti, hogy teljesítette-e az adott szokást.
- **Szokás adatok** – A cím, leírás, szín és aktuális streak megjelenítése.
- **Leírás részletezése** – Az egyes szokásokhoz tartozó részletes leírás lenyitható.
- **Jegyzet mentése** – A felhasználó személyes jegyzeteket fűzhet a napi szokásnaplóhoz.
- **Állapot frissítése** – A szokás teljesítésekor a feladat stílusa inaktívvá válik, és a streak számláló növekszik.

A komponens működése:

- Ha a feladatot bejelölik, a háttér szín halványodik és egy API hívás történik a napló frissítésére.
- Ha a felhasználó megváltoztatja a jegyzetet, mentésre kerül az adatbázisba.
- A feladat részletei külön gomb segítségével lenyithatók vagy bezárhatók.

Űrlap mezők

- Cím (kötelező)
- Leírás (kötelező)
- Színválasztás (kötelező)
- Pozitív/negatív szokás megkülönböztetése
- Gyakoriság kiválasztása (pozitív szokásoknál kötelező)

Állapotok kezelése

- Új szokás létrehozása sikeres vagy sikertelen esetén figyelmeztető üzenetet jelenít meg.
- Betöltés közbeni spinner animáció.
- Üres napi lista esetén: “No tasks for today!” üzenet megjelenítése.

Negative Habits (Negatív szokások)

A negatív szokásokat a `NegativeHabitsComponent` kezeli. Minden negatív szokás egy külön `NegativeHabitComponent` komponensben jelenik meg.

- **Lista betöltése** – A felhasználó negatív szokásai lekérésre kerülnek a szerverről.
- **Napok számlálása** – Minden szokás esetén látható, hogy hány nap telt el az utolsó visszaesés óta.
- **Kördiagram megjelenítése** – A napok számát arányosan kitöltött színes körgrafika ábrázolja.
- **Automatikus frissítés** – A számláló 10 percenként automatikusan újraszámolódik.

A kördiagram kitöltésének logikája:

- A háttér egy `conic-gradient` segítségével töltődik ki.
- A kitöltés aránya: `differenceInDays / 3.65 %`.

2.3.2 Friends

Leírás

A Barátok oldal lehetőséget ad:

- Barátok keresésére e-mail cím alapján
- Barátkérések elfogadására vagy elutasítására
- Barátok törlésére
- Barátok listájának megtekintésére

Függvények

- **searchUsers()** – Felhasználók keresése e-mail cím alapján.
- **getFriends()** – Barátok lekérése az adatbázisból.
- **sendFriendsRequest(friendId, event)** – Barátkérés küldése egy másik felhasználónak.
- **getFriendRequests()** – Beérkezett barátkérések lekérése.
- **getSentFriendRequests()** – Elküldött barátkérések lekérése.
- **acceptFriendRequest(friend)** – Barátkérés elfogadása.
- **declineFriendRequest(friend)** – Barátkérés elutasítása.
- **deleteFriend(friend)** – Barát törlése a listából.
- **showOptions(event, friend)** – Opciók megjelenítése (pl. törlés gomb).
- **hideOptions(event)** – Opciók elrejtése kattintásra.

2.3.3 Habits

Leírás

A Szokások oldal segítségével a felhasználók kezelhetik saját szokásaikat, kihívásokat küldhetnek barátoknak, elfogadhatják vagy visszautasíthatják a kihívásokat.

Függvények

- **ngOnInit()** – A komponens inicializálása, szokások és kihívások betöltése.
- **loadHabits()** – A felhasználó szokásainak lekérése és megjelenítése.
- **loadChallengeInvites()** – A felhasználónak érkezett kihívások lekérése.
- **openChallengeModal(habit)** – Kihívás küldési modal ablak megnyitása adott szokásnál.
- **closeChallengeModal()** – A kihívásküldő modal bezárása.
- **challengeFriend(friendId)** – Kihívás küldése egy barátunk egy szokás alapján.

- **acceptChallenge(challenge)** – Egy beérkezett kihívás elfogadása.
- **declineChallenge(challenge)** – Egy beérkezett kihívás visszautasítása.
- **isFriendAlreadyChallenged(friendId)** – Ellenőrzi, hogy egy barát már ki lett-e hívva adott szokásra.
- **isHabitOwner(habit)** – Ellenőrzi, hogy a szokás tulajdonosa-e a bejelentkezett felhasználó.
- **deleteHabit(habit)** – Egy szokás törlése a listából.

2.3.4 Feed

Leírás

A Feed oldal publikus posztokat jelenít meg a felhasználók számára. Ha nincsenek elérhető posztok, azt üzenetben jelzi.

Függvények

- **constructor()** – A komponens inicializálásakor automatikusan meghívja a feed betöltését.
- **loadFeed()** – Lekéri az aktuális felhasználó számára elérhető publikus posztokat. Hiba esetén kezeli az üres feed állapotát.

2.4 Komponensek

2.4.1 Áttekintés

Újrahasználható Komponensek: A projektben számos újrahasználható komponens található, amelyek a különböző oldalakon és funkciókban alkalmazhatók, hogy javítsák a kód újrahasználhatóságát, karbantarthatóságát és olvashatóságát. Ilyen komponensek például:

- **Loading spinner:** Egy betöltési animáció, amely jelzi, hogy az alkalmazás adatokat tölt be vagy folyamatban lévő műveletet végez.
- **Validation message:** Egy üzenetkomponens, amely tájékoztatja a felhasználót a hibás adatokkal kapcsolatban, például hiányzó mezők vagy érvénytelen formátum esetén.
- **Navbar:** Egy navigációs sáv, amely lehetővé teszi a felhasználó számára a különböző oldalakon való navigálást.

Oldal Komponensek: Ezek a komponensek az alkalmazás egyes oldalaihoz tartoznak. Minden oldalhoz saját komponens tartozik, amelyek kezelik a megjelenítést és a dinamikus adatokat. Részletesebben feljebb lehet őket megtekinteni. Példák:

- **Dashboard:** A felhasználói irányítópult komponense, amely összegzi az alkalmazás legfontosabb információit.
- **Habits:** A felhasználó pozitív és negatív szokásait megjelenítő és kezelő oldal.

2.4.2 Funkcionalitás

Fő Funkciók

- **Felhasználói Hitelesítés:** A felhasználó hitelesítése a rendszerbe való belépéskor történik. Az alkalmazás a bejelentkezés és a regisztráció során JWT tokeneket használ, hogy biztosítsa a felhasználó hitelesítését és jogosultságát.
- **Űrlapok kezelése:** A szokások hozzáadásához és módosításához űrlapokat használunk, amelyek tartalmazzák a szokás címét, leírását, színválasztását és egyéb paramétereket. Az űrlap validálása Angular form-control segítségével történik. A sikeres vagy sikertelen műveletek érdekében hibakezelés és figyelmeztető üzenetek is megjelennek.
- **API Integráció:** Az API hívásokat az Angular szolgáltatásain keresztül kezeljük. Az Angular beépített HTTP modulját használjuk az API-k elérésére. A `UserService` és `HabitService` szolgáltatások segítségével történnek a felhasználói és szokásadatok lekérése és frissítése.
- **Állapotkezelés:** Az alkalmazás állapotát az Angular szolgáltatások kezelik, amelyek az adatokat és a felhasználói interakciókat menedzselik. A `UserService` és `HabitService` biztosítják, hogy a felhasználói és szokás adatokat a komponensek között megfelelően osszuk meg. Az állapot frissítése minden interakció során automatikusan megtörténik, és a komponensek az Angular változásfigyelő mechanizmusain keresztül frissülnek.

API Hívás Példa

API hívások az Angular HTTP Client használatával:

```
HabitService.ts
13 export class HabitService {
14     private apiUrl = environment.apiUrl;
15
16     constructor(private http: HttpClient, private authService: AuthService) { }
17
18     private getAuthHeaders() {
19         const token = this.authService.getToken();
20         return { Authorization: `Bearer ${token}` };
21     }
22
23     getHabits(): Observable<Habit[]> {
24         const userId = this.authService.getUserId();
25         const params = new HttpParams()
26             .set('userId', userId as string)
27             .set('pageNumber', 1)
28             .set('pageSize', 20);
29
30         return this.http.get<Habit[]>(`${this.apiUrl}/habit`, {
31             params,
32             headers: this.getAuthHeaders(),
33         });
34     }
}
```

A fenti kód segítségével történnek az API hívások az `HttpClient` modulon keresztül. A `getHabits()` metódus lekéri a szokásokat, míg a `updateHabitLog()` frissíti a meglévő szokás adatokat.

2.5 Stílusok

A projekt stílusait SCSS fájlokban definiáljuk, hogy rugalmasan kezelhessük a különböző megjelenési elemeket, mint például a színeket, betűméreteket és elrendezéseket. Az alábbiakban bemutatásra kerülnek a legfontosabb változók és stílusok, amelyek az alkalmazás kinézetét meghatározzák.

2.5.1 Változók (`variables.scss`)

A stílusok alapja a `variables.scss` fájlban található változók, amelyek a különböző színeket, betűméreteket, valamint az alapvető dimenziókat tartalmazzák.

Színek

A színek különböző kategóriákra vannak bontva, mint például az akcentus színek, fő színek, szövegszínek, figyelmeztető színek és feladat színek:

- **Akcentus színek:** A projekthez használt hangsúlyos színek, mint például `$accent-color-100`, `$accent-color-200` stb.
- **Fő színek:** A háttérszínek, mint például `$main-color-100`, `$main-color-200` stb.
- **Szövegszínek:** A szövegekhez használt színek, például `$text-color-100`, `$text-color-200` stb.
- **Figyelmeztető színek:** A hibák és figyelmeztetések megjelenítéséhez használt színek, mint például `$alert-color-100`, `$alert-color-200` stb.
- **Feladat színek:** A különböző típusú feladatokhoz rendelt színek, mint például `$task-color-red`, `$task-color-blue`, `$task-color-green`.

Betűtípusok és Méretek

A `variables.scss` fájlban található a projekt által használt betűtípusok és betűméretek definíciója is:

- **Betűtípus:** Az Inter betűtípus van használatban, amely a következő módon van importálva:

```
@font-face {
  font-family: 'Inter';
  src: url('../fonts/Inter-VariableFont_opsz\,wght.ttf') format('truetype');
}
```

- **Betűméretek:** A különböző szövegekhez rendelt betűméretek a következők:

- `$font-size-xs: 12px;`
- `$font-size-sm: 14px;`
- `$font-size-base: 16px;`
- `$font-size-md: 18px;`

- `$font-size-lg: 20px;`
- `$font-size-xl: 24px;`
- `$font-size-2xl: 32px;`
- `$font-size-3xl: 48px;`

Dimenziók

A különböző méretek, például a szélességek, magasságok és a határok is a `variables.scss` fájlban találhatóak:

- `$sidebar_width: 5rem;`
- `$border_radius: 10px;`
- `$input_height: 2rem;`

2.5.2 Űrlapok és Input Mezők (inputs.scss)

Az alkalmazásban az űrlapok és bemeneti mezők stílusait az `inputs.scss` fájlban definiáljuk. A bemeneti mezők, mint például a szöveg-, email-, jelszó-, kereső- és szövegdobozok egyedi stílusokat kaptak.

Input Mezők

A bemeneti mezők általános stílusai a következőképpen vannak megadva:

- A mezők háttérszíne `$main-color-100`, a szöveg színe pedig `$text-color-900`.
- A mezők határvonalai `$main-color-600` színűek, és a mezők lekerekítettek `$border_radius` értékkel.
- A mezők fókuszálásakor a határvonal `$accent-color-400`-ra változik, és az éppen kitöltött mezők címkéje átméreteződik és a `$accent-color-400` színre vált.
- A checkbox inputok egyedi vizuális stílust kaptak, és csak akkor jelenítenek meg egy pipát, ha az adott checkbox be van jelölve.

Feladatok Stílusai (task component)

A feladatok megjelenítése a következő stílusokkal van definiálva az `task_container` osztályban:

- A feladatok konténere `display: flex;` és `flex-direction: column;` beállításokkal van struktúrába rendezve.
- A feladatok tetején található fejlécek a `task_header_container` osztályban vannak elhelyezve, amely tartalmazza a bal és jobb oldali információkat. A színük a `$main-color-100` háttérszínre kapja, a szövegek színe pedig `$text-color-900`.
- A feladatok részletei a `task_details_container` osztályban jelennek meg, amely az animációs hatásokkal bővített magasságot használ.
- A feladatokhoz kapcsolódó nem mai szokások esetén egy egyszerű stílusú cím és határidő jelenik meg, amelyek szintén a változókkal vannak testreszabva.

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

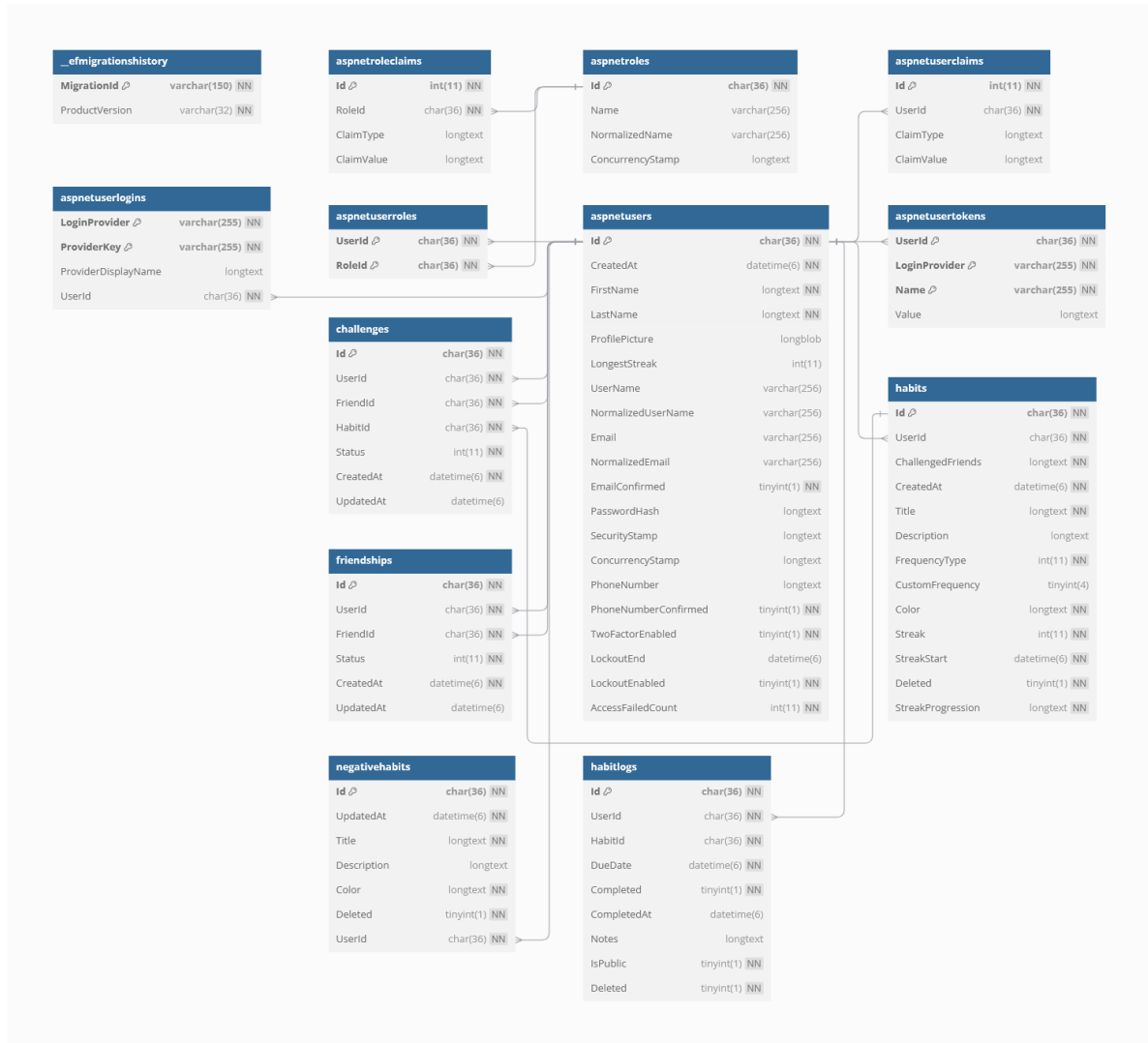
Erre azért van így, mert a fejlesztést MySQL-el kezdtük, majd a Koyeb-re való publikáláshoz szükségessé vált PostgreSQL kompatibilitás. Az adatbázis migrációk kezelésére az Entity Framework Core migrációs rendszerét használjuk, amely lehetővé teszi a séma verziókövetését és az adatbázis automatikus frissítését, 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") ==
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 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 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:

5.3.1 Autentikáció

- Regisztráció (**POST** /api/auth/register)
- Bejelentkezés (**POST** /api/auth/login)

5.3.2 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)

5.3.3 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 létrehozása (**POST** /api/habit/negative)
- Negatív szokás módosítása (**PATCH** /api/habit/negative/:id)
- Negatív szokás törlése (**DELETE** /api/habit/negative/:id)

5.3.4 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)

5.3.5 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)

5.3.6 Feed

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

5.3.7 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-t 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 Weboldal tesztelése

A frontend tesztelése Cypress segítségével történt. A tesztek célja a helyes működés és a validációk biztosítása.

6.1.1 Bejelentkezési oldal tesztek (`login.spec.ts`)

- **Sikeres bejelentkezés** – Érvényes e-mail és jelszó megadása esetén a rendszer beenged.
- **Hiányzó e-mail mező** – A mező üresen hagyása validációs hibaüzenetet eredményez.
- **Hiányzó jelszó mező** – A mező üresen hagyása validációs hibaüzenetet eredményez.
- **Helytelen e-mail formátum** – Hibás e-mail formátum esetén megjelenik a validációs hiba.
- **Jelszó megjelenítésének váltása** – A szem ikonra kattintva a jelszó láthatóvá válik.
- **Navigáció a jelszó-émléktető oldalra** – Az „Elfelejtetted a jelszavad?” link megfelelő oldalra irányít.
- **Navigáció a regisztrációs oldalra** – A „Fiók létrehozása” gombra kattintva a regisztrációs oldalra navigál a felhasználó.

6.1.2 Fiók létrehozása oldal tesztek (`create-account.spec.ts`)

- **Sikeres fiók létrehozása** – Minden mező helyes kitöltése esetén a rendszer 201-es választ ad.
- **Üres mezők esetén hibaüzenet** – A regisztrációs mezők üresen hagyása validációs hibákat okoz.
- **Hibás e-mail formátum** – Érvénytelen e-mail cím esetén hibaüzenet jelenik meg.
- **Nem egyező jelszavak** – A két jelszó eltérése esetén figyelmeztetés jelenik meg.
- **Jelszó láthatóság kapcsolása** – A szem ikon funkcióját teszteli.
- **Visszatérés a bejelentkezési oldalra** – A vissza gombra kattintva a login oldalra navigálás történik.

6.1.3 Dashboard oldal tesztek (`task-view.cy.ts`)

- **Dátum helyes megjelenítése** – Ellenőrzi, hogy az aktuális nap dátuma helyesen jelenik meg a felületen, a hét napja és a hónap/nap formátumban.
- **Új feladat hozzáadási űrlap megnyitása és bezárása** – Teszteli, hogy az „Új feladat hozzáadása” gombra kattintva megjelenik az űrlap, majd az űrlap bezárására szolgáló ikonra kattintva eltűnik.

- **Előző nap gomb inaktívvá tétele mai nap esetén** – Ellenőrzi, hogy az előző napra léptető gomb nem elérhető (le van tiltva) ha az aktuális dátumot nézzük.
- **„Nincs feladat mára” üzenet megjelenése** – Szimulált adat hiányában ellenőrzi, hogy megjelenik a „Nincs feladat mára!” és a „Kattints az 'Add' gombra új szokás létrehozásához.” szöveg.
- **Pozitív szokás létrehozása** – Teszteli, hogy egy új pozitív szokást helyesen lehet hozzáadni: cím, leírás, szín és gyakoriság megadásával.

6.1.4 Barátok oldal tesztek (friends-page.cy.ts)

- **Betöltési animáció megjelenése barátok lekérése közben** – Teszteli, hogy az adatok betöltése során a spinner látható.
- **Üzenet megjelenése, ha nincsenek barátok** – Ellenőrzi, hogy ha nincs visszaküldött barát adat, akkor egy „nincsenek barátaid” üzenet jelenik meg.
- **Barátok listájának megjelenítése** – Valós adatok esetén ellenőrzi, hogy a barátok neve és e-mail címe megjelenik a listában.
- **Barát törlése** – Teszteli, hogy egy barát törlése után a felhasználó eltűnik a listából.
- **Felhasználók keresése e-mail alapján** – Ellenőrzi, hogy a keresőmező segítségével megadható e-mail cím alapján megjelennek a találatok.
- **Barát felkérés küldése** – Teszteli, hogy a keresési találat melletti „követés” ikonra kattintva barátfelkérés küldése történik, és az ikon „elfogadva” állapotra vált.
- **Barátkérések megjelenítése** – Ellenőrzi, hogy a beérkező barátkérések listája helyesen jelenik meg az oldalon.
- **Barátkérés elfogadása** – Teszteli, hogy a beérkező barátkérés elfogadása után az adott kérés eltűnik a listából.
- **Barátkérés elutasítása** – Teszteli, hogy a barátkérés elutasítása után az adott kérés eltűnik a listából.