**Projekt dokumentálása**

**TopForm webalkalmazás**

****

Premontrei Szakgimnázium és Technikum

Keszthely

2025

Szoftverfejlesztő és tesztelő szak

5-0613-12-03

Készítette:

Fülöp Krisztián Kristóf, Lóki Dániel,

Lucz Boldizsár Károly

**Tartalom**

[Bevezetés 4](#_Toc195297475)

[A projektben használt alkalmazások: 5](#_Toc195297476)

[Projekt megvalósítási terv: 5](#_Toc195297477)

[A projekt felhasználói felületének a felépítése: 6](#_Toc195297478)

[A projekt fő funkciói: 7](#_Toc195297479)

[A csapatmunka és a csapat tagok feladatai: 8](#_Toc195297480)

[A projekt működési elve: 10](#_Toc195297481)

[Front End 11](#_Toc195297482)

[Kezdő oldal: 11](#_Toc195297483)

[Betöltő oldal: 12](#_Toc195297484)

[Bejelentkezési oldal 13](#_Toc195297485)

[Regisztráció oldal 2 14](#_Toc195297486)

[Étrendnapló oldal: 15](#_Toc195297487)

[Főoldal: 16](#_Toc195297488)

[Edzésnapló olda 17](#_Toc195297489)

[Ranglista oldal 18](#_Toc195297490)

[Profil oldal 19](#_Toc195297491)

[Beállítások oldal 20](#_Toc195297492)

[A felhasználó nincs bejelentkezve oldal 21](#_Toc195297493)

[Hibás URL oldal 22](#_Toc195297494)

[Admin Dashboard 23](#_Toc195297495)

[Admin adatbázis oldal 24](#_Toc195297496)

[Admin edzés hozzá adás oldal 25](#_Toc195297497)

[Back End 26](#_Toc195297498)

[ApplicationDbContext 26](#_Toc195297499)

[Modellek 27](#_Toc195297500)

[Felhasználói felület kontrollerei: 28](#_Toc195297501)

[DietController 28](#_Toc195297502)

[GenerateWorkoutControler 29](#_Toc195297503)

[GetDietController 30](#_Toc195297504)

[GetWorkoutController 31](#_Toc195297505)

[LeaderboardController 32](#_Toc195297506)

[LoginController 33](#_Toc195297507)

[ProfileController 34](#_Toc195297508)

[Registration2Controller 35](#_Toc195297509)

[RegistrationController 36](#_Toc195297510)

[UserController 37](#_Toc195297511)

[WorkoutController 38](#_Toc195297512)

[Admin Kontrollerek 39](#_Toc195297513)

[AdminTableController 39](#_Toc195297514)

[AdminUserTableController 39](#_Toc195297515)

[Mesterséges Intelligencia 40](#_Toc195297516)

[A projekt tesztelése 42](#_Toc195297517)

[Swagger tesztelés 43](#_Toc195297518)

[Selenium tesztelés 43](#_Toc195297519)

[NUnit tesztek 44](#_Toc195297520)

[Adatbázis 46](#_Toc195297521)

[Reflexió 49](#_Toc195297522)

[Források 50](#_Toc195297523)

# Bevezetés

A képen szöveg, képernyőkép, tervezés látható

Automatikusan generált leírásA projektünk neve **TopForm**, egy edzéskövetést és egészséges életmódot támogató webalkalmazás. Témaválasztásunkat az indokolta, hogy a csoport minden tagja jártas a témában, érdeklődést mutat iránta, és szívesen foglalkozik vele. Magyarországon a lakosság több mint 58%-a túlsúlyos vagy elhízott, és ez az arány a mai napig folyamatosan növekszik. Az érintettek közül sokan küzdenek a diabétesz valamely típusával (1-es, 2-es vagy 3-as típus), ami azt jelenti, hogy szinte minden tizedik magyar felnőtt érintett a cukorbetegség valamely formájában.

Projektünk célja egy olyan innovatív webalkalmazás kifejlesztése volt, amely játékos, motiváló formában támogatja a felhasználókat a rendszeres testmozgásban és az egészségesebb életmód kialakításában. A platform lehetőséget biztosít arra, hogy a felhasználók személyre szabott edzéstervet és étrendet állítsanak össze, valamint folyamatosan nyomon követhessék fizikai állapotuk és teljesítményük alakulását. A felhasználói élményt vizuális elemek, statisztikák és visszajelzések teszik még ösztönzőbbé. Az alkalmazás külön figyelmet fordít a felhasználók egyéni igényeire és céljaira – legyen szó fogyásról, izomtömeg-növelésről, állóképesség-fejlesztésről vagy egyszerűen az egészséges életmód fenntartásáról. A rendszer rugalmassága lehetővé teszi, hogy bárki, bármilyen edzettségi szinttel, saját tempójában haladjon. Az alkalmazást minden olyan személy számára ajánljuk, aki nyitott saját testi és egészségi állapotának tudatos fejlesztésére, függetlenül attól, hogy kezdőként vagy haladóként vág bele ebbe az életmódváltásba. Meggyőződésünk, hogy a digitalizáció nyújtotta lehetőségeket kihasználva hatékonyan tudjuk támogatni a felhasználókat céljaik elérésében és egy hosszú távon fenntartható, egészséges életvitel kialakításában.

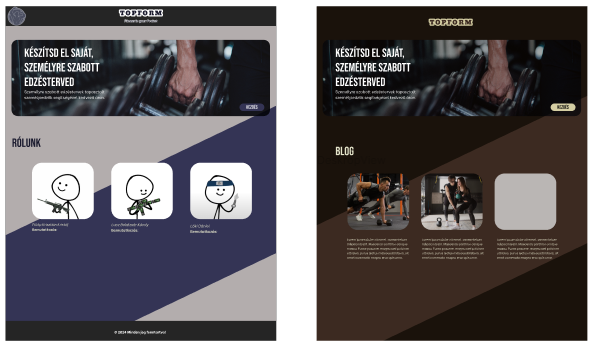
A projektet háromfős fejlesztőcsapatunk valósította meg: Fülöp Krisztián Kristóf, Lóki Dániel és Lucz Boldizsár Károly, akik mindannyian aktív érdeklődést mutatnak a témakör iránt, és különféle területeken jártasak a fejlesztésben.

## A projektben használt alkalmazások:

* Discord: a közös kommunikációt biztosító platform
* GitHub: a projekt fájlokat tároljuk itt
* Figma: kezdeti design ötletek megtervezésére használjuk
* Visual Studio 2022: Az ASP részét itt szerkesztjük
* Visual Studio Code: A REACT részét itt szerkesztjük
* TeleportHQ: A véglegesített design elkészítésének segítségére használtuk
* Deepseek: A kód egy adott részébe segítség gyanánt használtuk

Ezek a programok támogatták a közös munkát, a programozást, a folyamatos kommunikációt és ötletelést, valamint egymás segítését.

## Projekt megvalósítási terv:

A projekt megvalósításához a Microsoft fejlesztők számára készült integrált fejlesztőkörnyezetét (IDE) használtuk. A felhasználói felület kezdeti terveit a Figma alkalmazásban készítettük el, majd ezek alapján valósítottuk meg a reszponzív és a mai designtrendeknek megfelelő felületet React keretrendszer segítségével.

A vizuális élmény fokozása érdekében 3D modelleket is alkalmaztunk, amelyeket a Meshy AI generatív mesterséges intelligencia segítségével készíttettünk el. A projekt során egy mesterséges intelligencia integrációt is megvalósítottunk Python nyelven, amely támogatást nyújt a felhasználók számára az egyéni edzéstervek automatikus generálásában. A front-end fejlesztéshez a React keretrendszert választottuk, melyet oktatóink és szakmai forrásaink is ajánlottak. A főoldal HTML5 struktúrában készült, míg az aloldalak JSX fájltípusokban kerültek megvalósításra. Az animációkat CSS és JavaScript segítségével hoztuk létre. A vizuális komponensek kialakításához a Material UI könyvtárból és a React különféle csomagjaiból használtunk előre definiált elemeket, amelyeket igény szerint testre szabtunk. A 3D modellek megjelenítésére a THREE.js könyvtárat alkalmaztuk, amelyet további egyedi JavaScript kódokkal egészítettünk ki. A felület tartalmaz egy külső API-t is, amely edzéssel és fitnesztrendekkel kapcsolatos híreket szolgáltat a felhasználók számára. A back-end fejlesztést ASP .NET környezetben, C# nyelven végeztük, mivel ez a technológia a tanulmányaink során kiemelten szerepelt, és lehetővé tette a front-end, az adatbázis, valamint a helyileg futtatott mesterséges intelligencia közötti hatékony kommunikációt. Az API végpontokat RESTful architektúra szerint alakítottuk ki, mely biztosítja a JSON-alapú adatcserét a rendszer komponensei között. Az adatbázist MySQL alapokon építettük fel, összesen öt fő táblával és egy kapcsolótáblával. A fejlesztés során folyamatos manuális tesztelést végeztünk, melyet kiegészítettünk NUnit egységtesztekkel, Selenium alapú automatizált tesztekkel, valamint a Swagger által biztosított dokumentált tesztfelülettel.

## A projekt felhasználói felületének a felépítése:

1. Kezdő oldal
2. Beléptető oldal
3. Regisztráció oldal
4. Főoldal
5. Edzésnapló oldal
6. Étrendnapló oldal
7. Ranglista oldal
8. Profil oldal
9. Beállítások oldal
10. Admin oldal
11. A felhasználó nincs bejelentkezve oldal
12. Hibás url oldal
13. Az oldalak betöltésére szánt oldal

## A projekt fő funkciói:

1. Bejelentkezés
2. Regisztráció
3. Elfelejtett jelszó
4. Manuális edzésterv készítése és mentése
5. Korábbi edzéstervek lekérdezése
6. AI edzésterv generálása
7. Étrend megtervezése, és mentése
8. Korábbi étrendek lekérdezése
9. A rang rendszer
10. Az összes felhasználó megjelenítése a ranglista oldalon
11. A motivációs szövegek
12. Az edzéssel kapcsolatos hírek megjelenítése
13. A profil kép módosítása
14. A felhasználónév és jelszó módosítása
15. A felhasználói adatok lekérdezése
16. Az admin felület AI API megjelenítése
17. Az admin felület összes adatbázis tábla megjelenítése
18. Az admin felületen a felhasználó törlése
19. Az admin felületen a felhasználói adatok módosítása
20. Az admin felületen új típusú edzés mód hozzá adása

## A csapatmunka és a csapat tagok feladatai:

A kommunikációt Discordon és Messengeren valósítottuk meg, a közös egyidejű programozást GitHub és a Visual Studio 2022 által nyújtott Live Share-el valósítottuk meg.

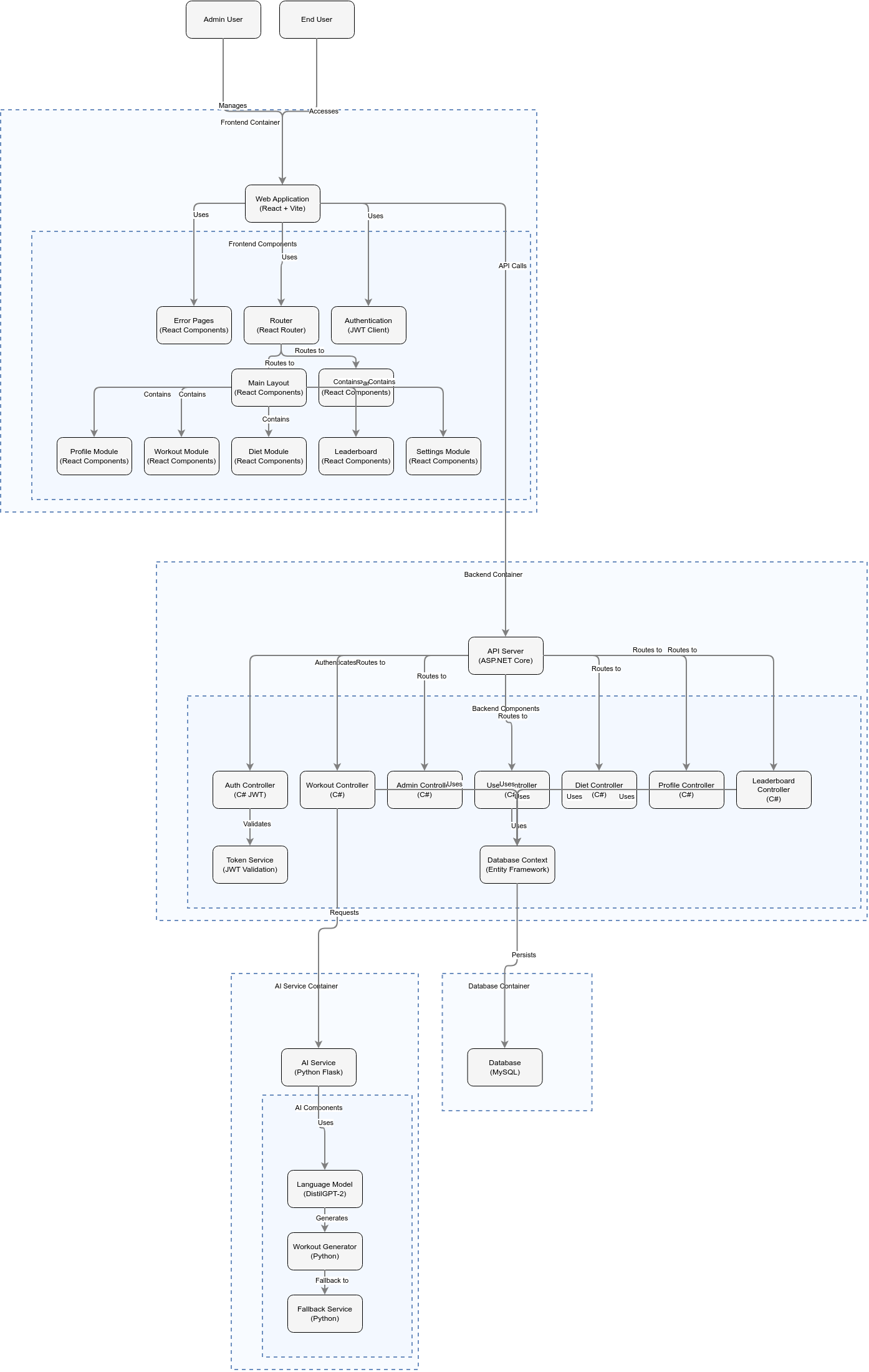
A következő táblázatban látható, hogy melyik csapat tag melyik oldalt design-olta:

|  |  |  |
| --- | --- | --- |
| **Lucz Boldizsár Károly** | **Fülöp Krisztián Kristóf** | **Lóki Dániel** |
| Beléptető oldal | Kezdő oldal | Főoldal |
| A felhasználó nincs bejelentkezve oldal | Hibás url oldal | Regisztráció oldal |
| Étrendnapló oldal | Edzésnapló oldal | Ranglista oldal |
| Profil oldal | Az oldalak betöltésére szánt oldal | Beállítások oldal |
| Admin oldal | Admin oldal | Admin oldal |

A következő táblázatban látható, hogy melyik csapat tag melyik funkció létrehozásában látott el szerepet, de sokszor, ha valamelyikünk elakadt valahol ki segítettük egymást:

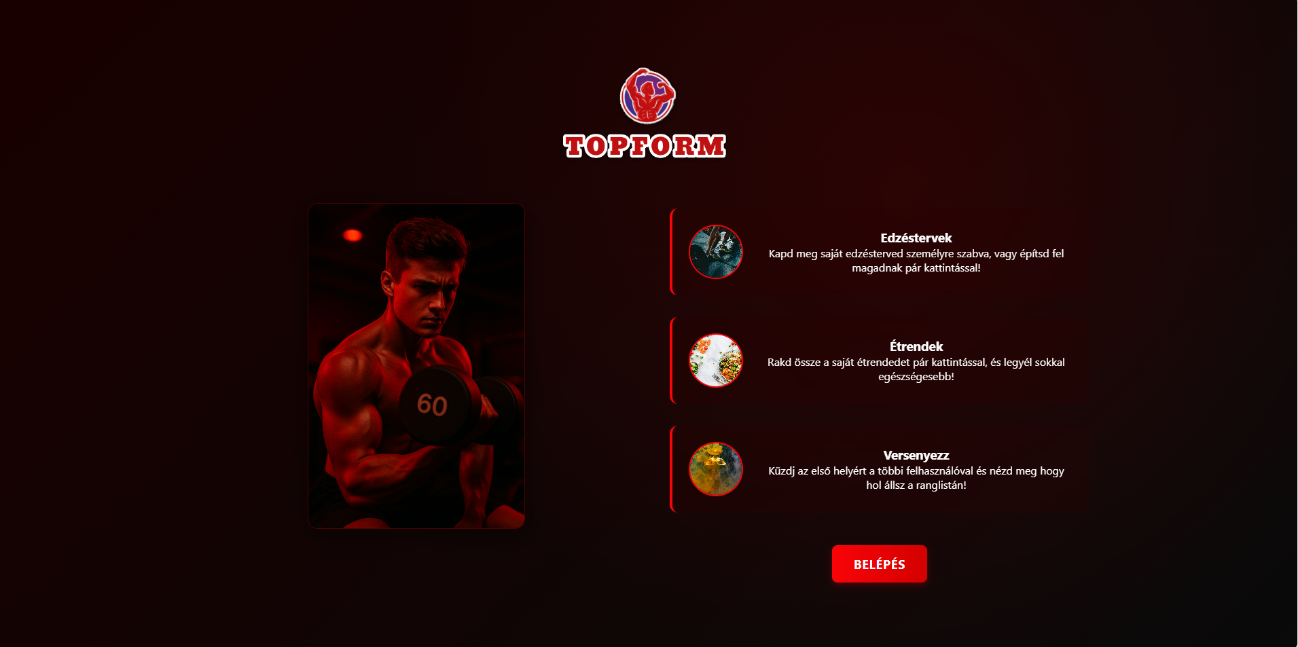
|  |  |  |
| --- | --- | --- |
| **Lucz Boldizsár Károly** | **Fülöp Krisztián Kristóf** | **Lóki Dániel** |
| Regisztráció 1.szakasz | Regisztráció 2.szakasz | Bejelentkezés |
| Korábbi edzéstervek lekérdezése | Elfelejtett jelszó | Manuális edzésterv mentése |
| Étrend mentése | AI edzésterv készítése | Étrend lekérdezése |
| A motivációs szövegek | A ranglista lekérdezése | A rang rendszer |
| A hírek megjelenítése | A profil kép módosítása | A felhasználónév és felszó módosítása |
| A felhasználói adatok lekérdezése | Admin AI API | Admin adatbázis tábla megjelenítése |
| Admin Új edzés hozzá adása | Admin felhasználói adatok módosítása | Admin felhasználók törlése |

## A projekt működési elve:

A következő elvi ábra, csak egy kis szeletét tükrözi a teljes projektnek, de egy jó átlátást nyújt arra hogyan működnek az API végpontok, meg minden más.

# Front End

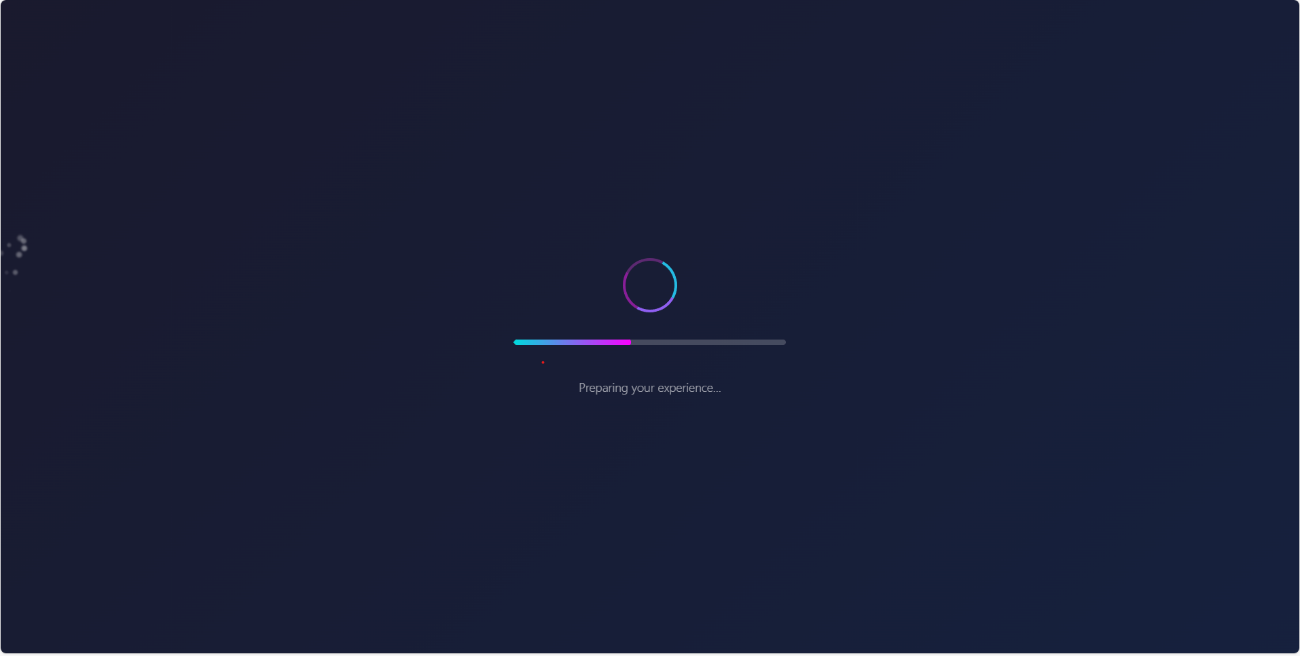
**(REACT)**

****Kezdő oldal: Ez az oldal szolgál első benyomásként mind az új, mind a visszatérő felhasználók számára, ezért kiemelt célja a figyelemfelkeltés, valamint az alkalmazás legfontosabb funkcióinak rövid és hatékony bemutatása. A felület modern, reszponzív kialakítással rendelkezik, amely illeszkedik a mai webdesign trendekhez, és vizuálisan is erős jelenléttel bír. Az oldalon két logó jelenik meg: egy teljes logó, illetve egy ikon formájú változat. A központi elem egy edzőtermi hangulatot idéző kép, amely célzottan érzelmi kapcsolatot teremt a felhasználóval az edzés és életmód témájában. Az alkalmazás főbb funkcióit kártyaszerű vizuális elemekben jelenítjük meg, amelyeket rövid, figyelemfelkeltő szövegek és ikonképek egészítenek ki. A sötétebb árnyalatú háttér, valamint a képi tartalom dominanciája lendületes, fiatalos megjelenést kölcsönöz az oldalnak. A kezdőlap fő interakciós eleme egyetlen gomb, amely közvetlenül a bejelentkezési oldalra navigálja a felhasználót, ezzel biztosítva az egyszerű és gyors továbblépést az alkalmazás használatához.

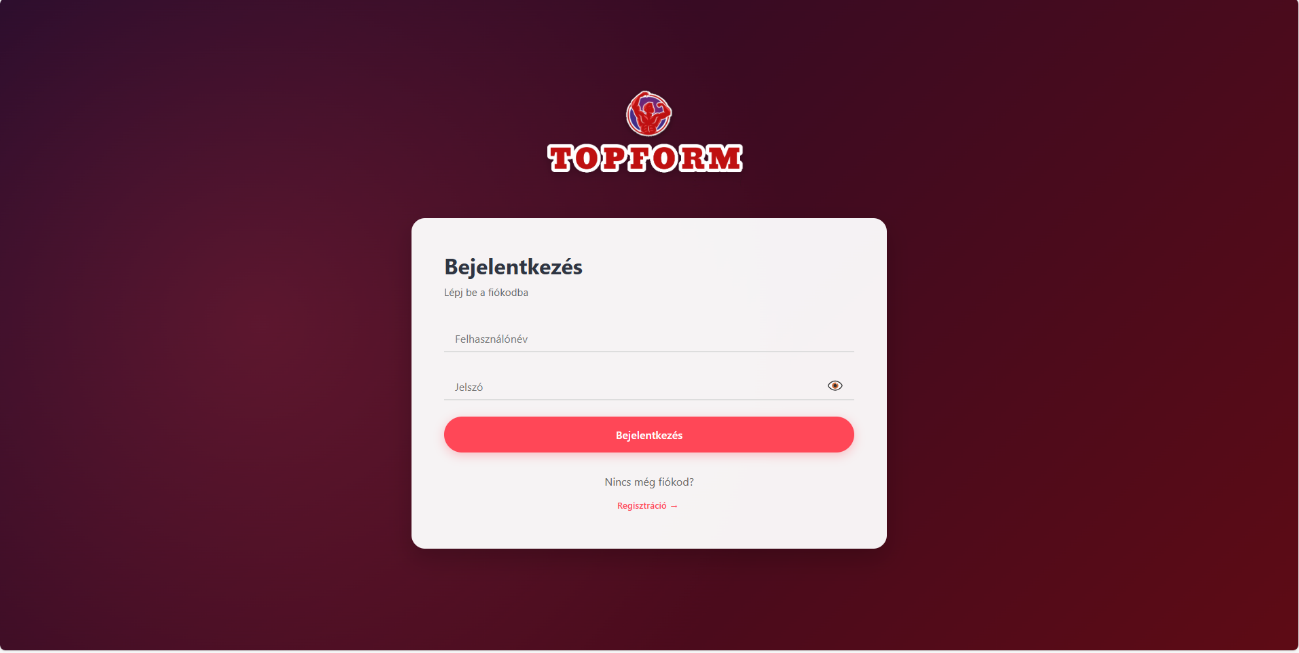
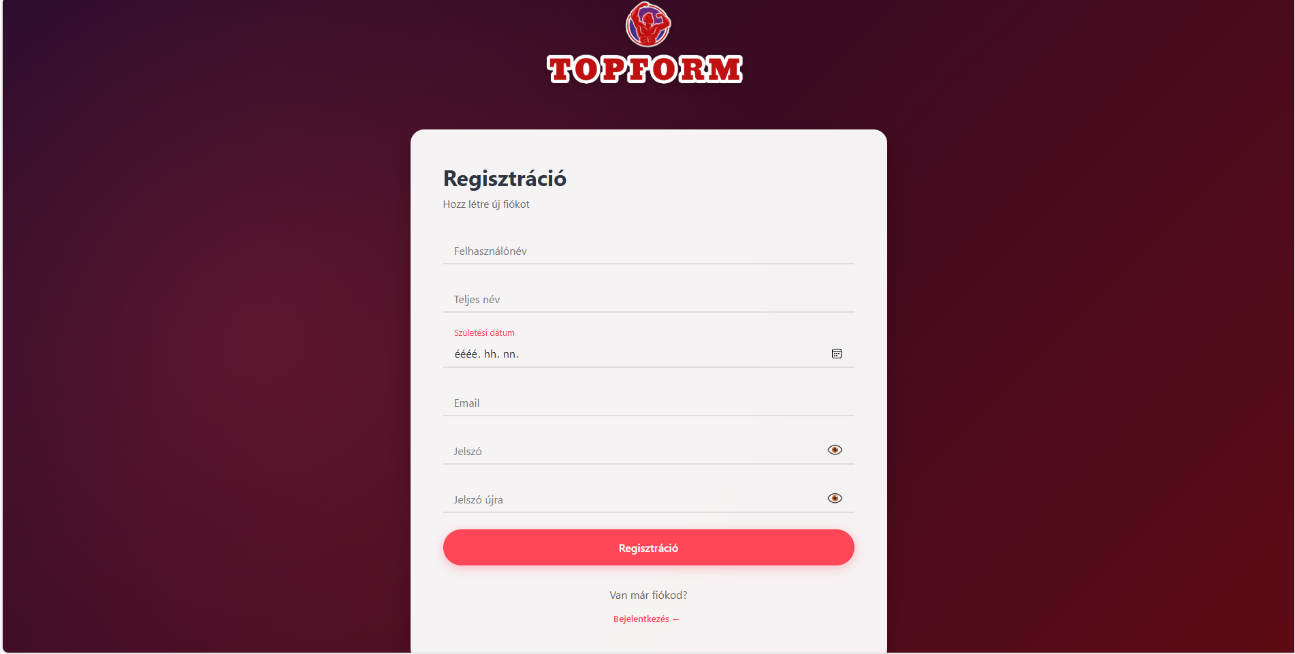
*Forrás fájl:* \TopForm\reactapp1.client\src\components/WelcomePage.jsx

Betöltő oldal: Ez a komponens egy dinamikus, animált betöltőfelületet jelenít meg, amelynek fő eleme egy fokozatosan töltődő haladási sáv (progress bar). A betöltőoldal célja, hogy vizuálisan megerősítse a felhasználónak: a rendszer dolgozik, az élmény készül. Akkor jelenik meg, amikor a felhasználó bejelentkezett, vagy amikor egy oldal tartalma még nem áll teljesen rendelkezésre.

A progress bar töltése során egy forgó spinner és lebegő partikulák teszik látványossá az élményt, miközben egy halványan pulzáló „Preparing your experience...” felirat jelzi, hogy a háttérben folyamatban van az adatlekérés. Amikor a töltés eléri a 100%-ot, megjelenik egy „Ready to go!” üzenet egy animált pipával, amely lendületes mozdulattal húzódik ki – ez az interakció az azonnali visszajelzést szolgálja. Ezt követően az oldal automatikusan átirányítja a felhasználót a főoldalra.

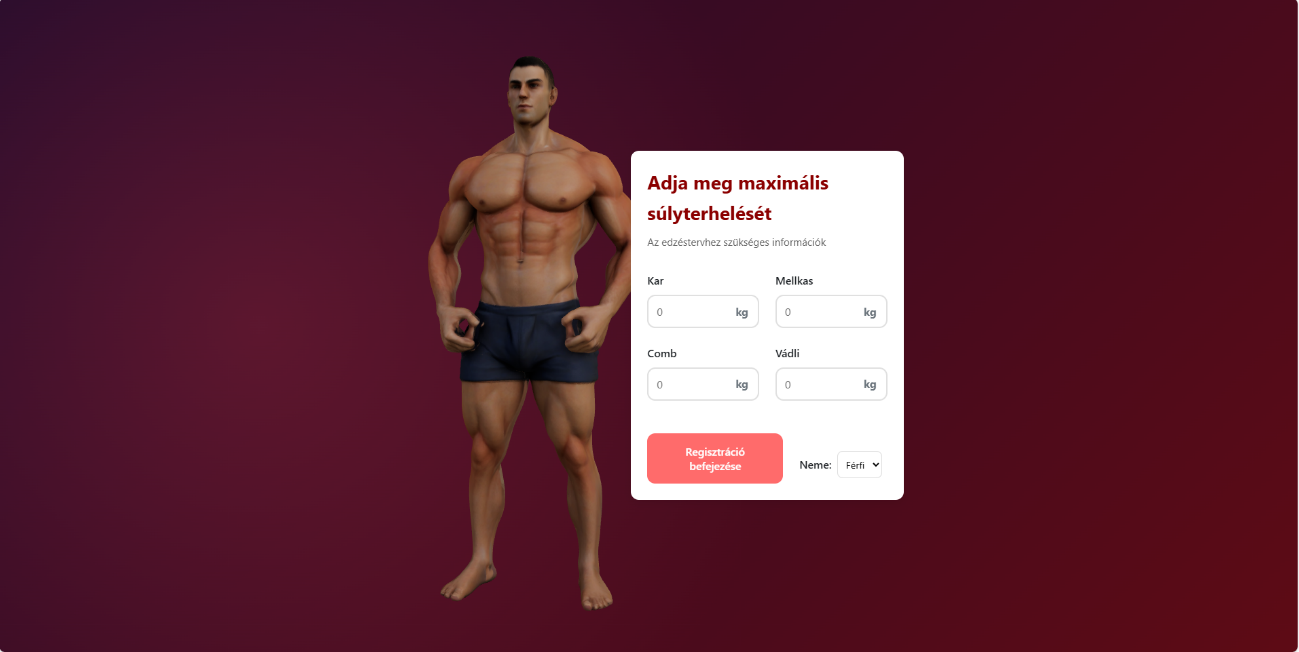
Az egész komponens framer-motion és styled-components segítségével készült, modern, reszponzív dizájnnal és finom animációkkal, melyek biztosítják a gördülékeny, esztétikus felhasználói élményt.

*Forrás fájl:* \TopForm\reactapp1.client\src\ErrorPages/loadingPage.jsx

Bejelentkezési oldal: Ez a komponens egy animált autentikációs felületet biztosít, amely lehetőséget ad a felhasználóknak bejelentkezésre és regisztrációra is. Az oldal a useLogin és useRegister hookok segítségével végzi el az API-hívásokat. Támogatja a jelszó láthatóságának váltását, a mezők validálását, a hibakezelést, valamint snackbar értesítések megjelenítését, melyek a Material UI komponens részei. A felület stílusát egyedi CSS és animált panelek biztosítják, vizuálisan elkülönítve a bejelentkezési és regisztrációs nézeteket.

*Forrás fájl:* \TopForm\reactapp1.client\src\components/loginRegister.jsx

Regisztráció oldal 2: Ez a komponens a regisztrációs folyamat második szakaszát valósítja meg, amely során a felhasználónak meg kell adnia az egyes izomcsoportok maximális terhelhetőségét. A 3D modell vizuális megjelenítéséért a WebGL technológia felelős, melyet a @react-three/fiber és @react-three/drei csomagok segítségével valósít meg.

A modell megjelenítéséhez és interakcióhoz a komponens a Canvas elemet használja, amely először betölti a megfelelő GLB fájlt (3D modell), majd folyamatosan forgatja azt. Amennyiben a WebGL nem elérhető a felhasználó eszközén, a rendszer figyelmeztetést jelenít meg. A regisztrációs form kitöltésekor, ha hibák lépnek fel, azok piros színnel kerülnek kiírásra a formon belül, így egyértelműen jelezve a felhasználónak, hogy hol történt hiba. Ha a regisztráció sikeres, a felhasználó automatikusan visszairányításra kerül a bejelentkezési oldalra. Az adatok küldését az useCompleteRegister hook biztosítja, amely az API-val kommunikál a felhasználói információk továbbításához.

*Forrás fájl:* \TopForm\reactapp1.client\src\components/BodyDetails.jsx

Étrendnapló oldal: Ez a React-alapú komponens egy korszerű, interaktív étrendkezelő felületet valósít meg, ahol a felhasználók intuitív módon szerkeszthetik és rendszerezhetik napi étkezéseiket. A drag and drop műveletek zökkenőmentes implementációja lehetővé teszi az ételtételek gyors mozgatását az egyes étkezések között, míg a dátum alapú szűrés segít a korábbi étrendek böngészésében és újratervezésében.

A komponens hátterében olyan modern React eszközök dolgoznak, mint a hook-alapú állapotkezelés (useState, useEffect), az egyéni hook-ok (pl. useDiet, useFetchDiet) és az aszinkron adathozzáférés, melyek együttesen biztosítják a robusztus és skálázható architektúrát. A komponens intelligens mentési mechanizmusa felismeri a mentetlen módosításokat, és automatikusan elmenti a lokális vázlatokat a localStorage segítségével, biztosítva a felhasználó munkájának védelmét.

A vizuális élményt az MUI komponensek modern dizájnja és a Framer Motion animációi teszik élővé, míg az értesítési rendszer (Snackbar, Alert) folyamatos visszacsatolást nyújt a felhasználó minden lépésére.

A képen képernyőkép, szöveg, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Ez az oldal nem csupán egy eszköz, hanem egy társ a mindennapi egészséges életmódhoz vezető úton: tárol, elemez, segít dönteni – és mindeközben megtartja a rugalmasságot, amire a modern felhasználónak szüksége van.

*Forrás fájl:* \TopForm\reactapp1.client\src\components/dietPage.jsx

Főoldal: Ez az oldal a felhasználó főoldalát jeleníti meg, amely modern és vizuálisan kiemelkedő felhasználói felületet biztosít, így az első pillanattól kezdve vonzó és könnyen navigálható élményt nyújt. A komponens célja, hogy üdvözölje a bejelentkezett felhasználót, és hozzáférést biztosítson a fő funkciókhoz, mint például az edzéstervek, étrendtervek és ranglisták, mindezt egy dinamikus és vizuálisan gazdag dizájnban. A főoldalon a felhasználó azonnal elérheti a személyre szabott edzéstervét, napi étkezési tervét, illetve a legfrissebb ranglistát, amely az alkalmazásban elért eredmények alapján mutatja be a legjobb teljesítményt nyújtókat. A lebegő profil menü tartalmazza a profil és beállítások oldalt, amely lehetővé teszi a felhasználó számára, hogy gyorsan módosítsa személyes adatait, vagy átállítsa a preferenciáit, mindezt intuitív módon. A navigációs menü a többi oldal elérhetőségét kínálja, és egy finom animációval bővül ki, amikor az egér az adott menüpontra kerül, így megjelennek az oldalak nevei, biztosítva a felhasználó számára a könnyed eligibilitást a kívánt funkciók elérésére. A navigációs sávban lehetőség van a kijelentkezésre is, amely a felhasználót biztonságosan kijelenti az alkalmazásból, és ha ismét használni kívánja, újra be kell jelentkeznie. Ezen kívül a felhasználói élményt segíti az élő értesítések megjelenítése, amelyek az alkalmazás fontos eseményeiről vagy státuszváltozásairól tájékoztatják a felhasználót, miközben egy dinamikusan frissülő háttér biztosítja a gördülékeny működést. Az oldal tehát nem csupán információkat kínál, hanem egy letisztult, interaktív központot biztosít, amely zökkenőmentesen navigálható, A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

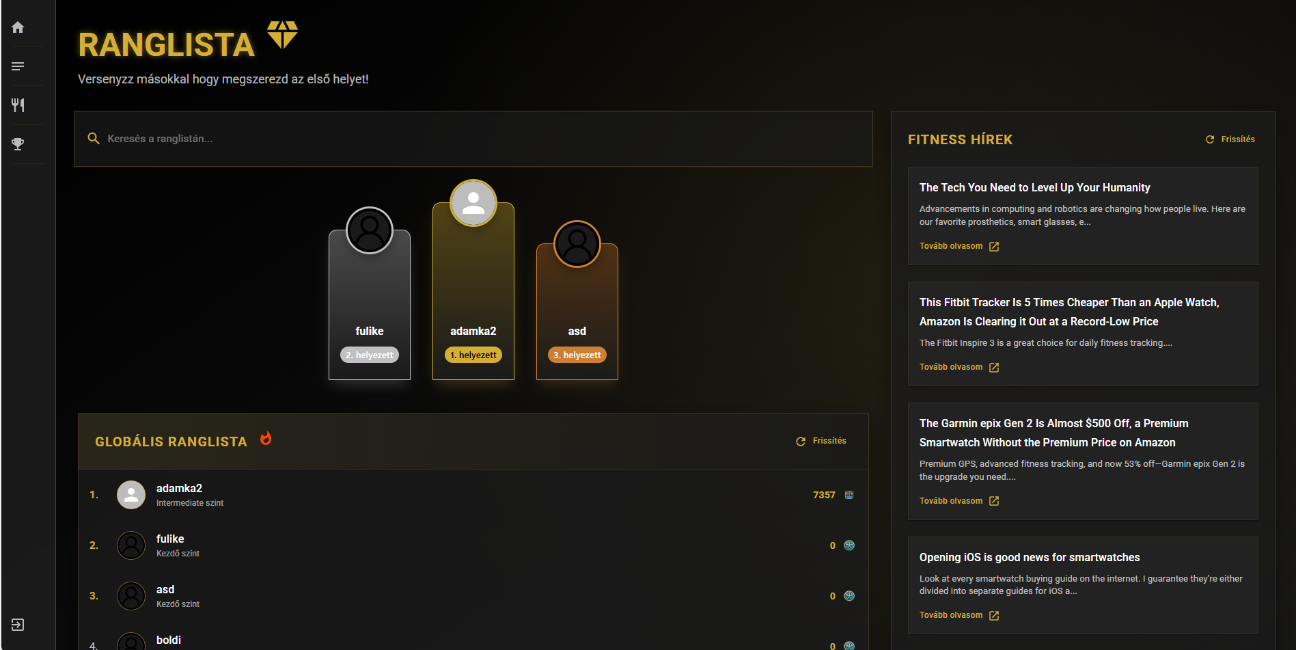
Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.így a felhasználók gyorsan elérhetik a számukra legfontosabb tartalmakat.

*Forrás fájl:* \TopForm\reactapp1.client\src\components/mainPageContent.jsx

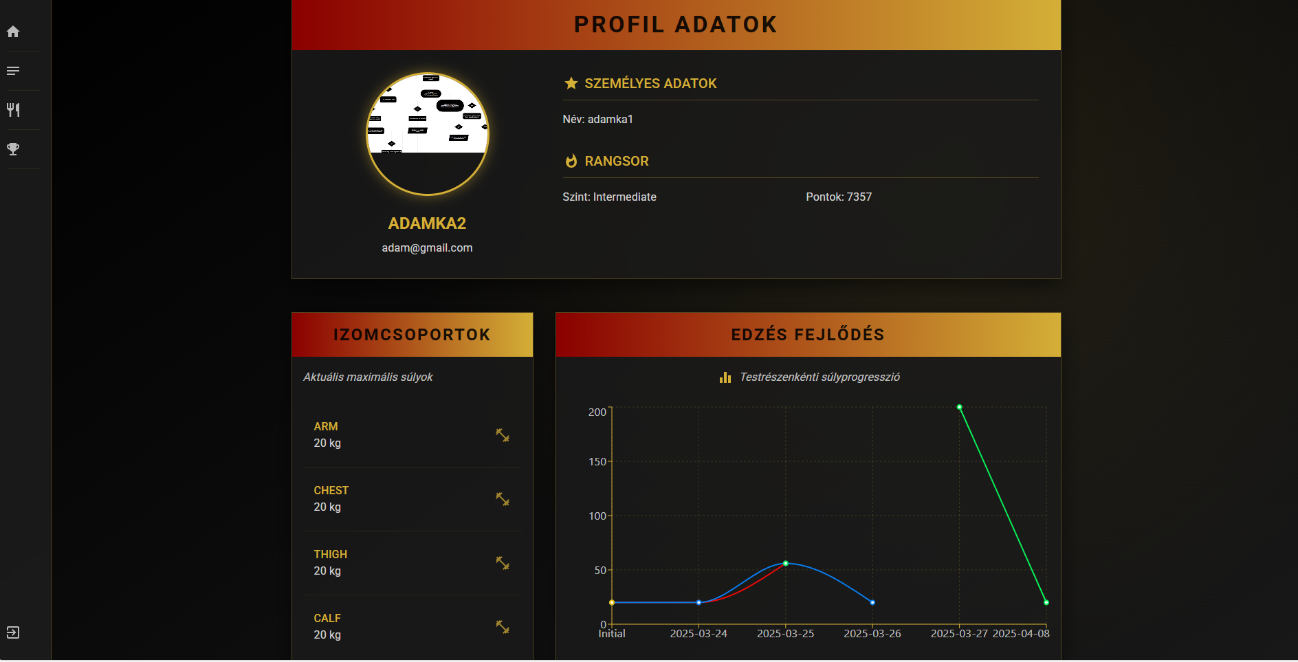
A képen képernyőkép, Multimédiás szoftver, szöveg, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Edzésnapló olda**l**: Ez a komponens szolgál az edzés követésére, lehetőséget biztosítva a felhasználóknak az edzéstervek létrehozására, mentésére és a mentett edzések visszanézésére, így minden egyes edzésfolyamat könnyen nyomon követhető. A főbb funkciók közé tartozik az edzéstervek létrehozása, amely lehetővé teszi a felhasználóknak, hogy saját igényeik szerint alakítsák ki az edzésprogramjukat, valamint a mentett edzések dátum szerinti megtekintése, így a felhasználók könnyedén visszakereshetik korábbi edzéseiket. Az oldal mesterséges intelligencia segítségével generál edzésterveket, személyre szabott ajánlásokat adva a felhasználóknak a fejlődésük és céljaik alapján, így biztosítva, hogy az edzéstervek mindig a felhasználó aktuális szükségleteihez igazodjanak. A komponens az **useWorkout** és a **useFetchWorkout** hookokat használja annak érdekében, hogy a felhasználói interakciók során minden adat automatikusan frissüljön, és az edzések adatainak mentése zökkenőmentesen történjen. Az egyik különlegessége, hogy ha a felhasználó átlépne egy másik oldalra, akkor sem veszik el az edzése, amit még nem mentett el, biztosítva, hogy a felhasználó ne veszítse el a folyamatban lévő munkát. Ezzel a funkcióval az oldal még inkább felhasználóbarát lesz, hiszen bármikor visszatérhetnek, és folytathatják a munkát anélkül, hogy adatvesztés történne. Az oldal dizájnja úgy lett kialakítva, hogy egyszerű és átlátható legyen, minden szükséges eszközt egyetlen képernyőn biztosítva, így a felhasználók gyorsan és kényelmesen hozzáférhetnek minden funkcióhoz, amelyre szükségük van az edzéseik sikeres nyomon követéséhez.

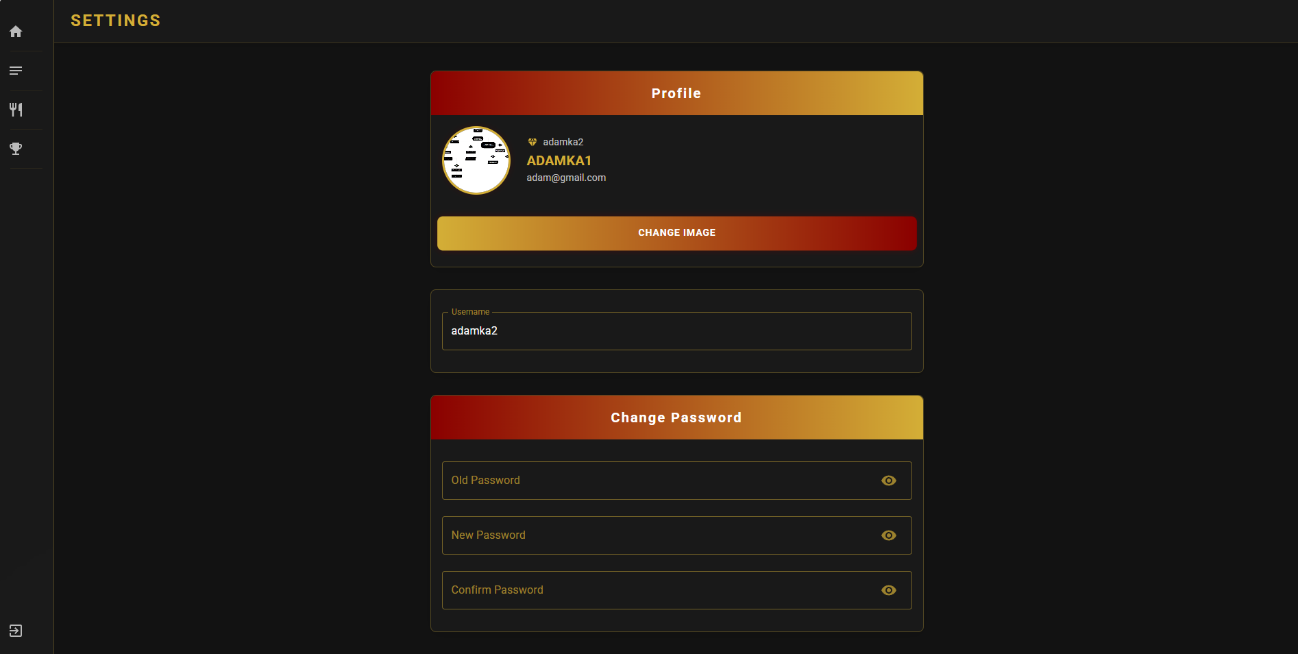
*Forrás fájl:* \TopForm\reactapp1.client\src\components/workoutPage.jsx

Ranglista oldal: Ez a komponens a felhasználók edzési teljesítménye alapján történő rangsorolásra szolgál, lehetőséget biztosítva a felhasználóknak, hogy összehasonlíthassák saját eredményeiket másokkal, így motiválva őket a további fejlődésre. Az oldal három fő tartalmi részből áll: a ranglistából, amely folyamatosan frissül az új edzésadatok alapján; a részletes statisztikákból, amelyek a felhasználók edzési eredményeit és fejlődését mutatják, így minden felhasználó nyomon követheti saját haladását; valamint a fitness hírekből és napi motivációs idézetekből, amelyek friss információkat és inspirációt biztosítanak a felhasználók számára. A hírek megjelenítéséhez külső API hívást használunk, hogy naprakész információkat nyújtsunk a legújabb fitness trendekről, eseményekről és egyéb releváns témákról, így a felhasználók mindig a legfrissebb tartalmakkal találkozhatnak. A napi motivációs idézetek pedig segítenek fenntartani a felhasználók lelkesedését, ösztönözve őket, hogy folytassák edzéseiket és érjék el céljaikat. Az oldal dizájnja intuitív, a három szekció jól elkülönül egymástól, így a felhasználók könnyen navigálhatnak az oldalon és élvezhetik a tartalmat, miközben motiválva érzik magukat a további fejlődésre.

*Forrás fájl:* \TopForm\reactapp1.client\src\components/leaderboard.jsx

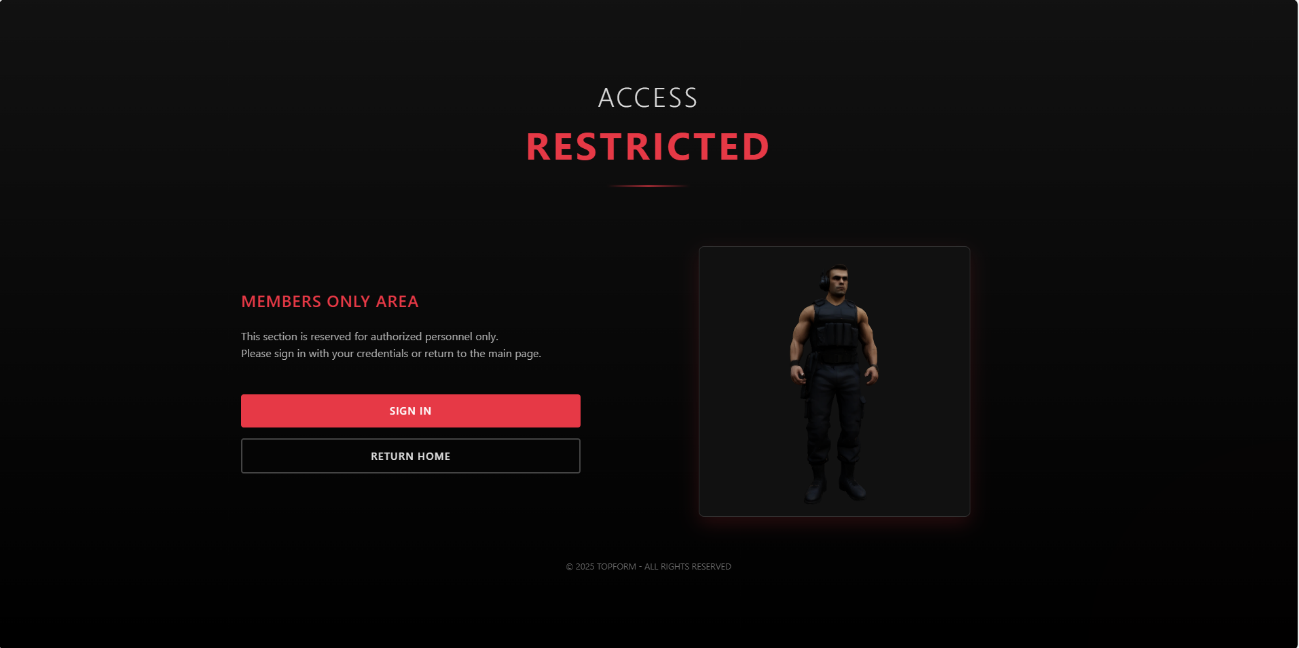
Profil oldal: Ez a komponens egy dinamikus és vizuálisan vonzó felhasználói profiloldalt biztosít, amely számos fontos információt tartalmaz a felhasználóról. Az oldal központi része a profilkép, amely lehetőséget ad a felhasználóknak, hogy személyre szabják megjelenésüket, valamint az alapvető személyes adatok, mint név, életkor, súly és edzési célok, amelyek segítenek a felhasználóknak saját fejlődésük nyomon követésében. A rangsorban való helyezés is megjelenik, így a felhasználók mindig láthatják, hol tartanak másokhoz képest, és folyamatosan motiválhatják magukat a fejlődésre. Emellett az edzésekkel kapcsolatos statisztikák is kiemelt szerepet kapnak, amelyek bemutatják a felhasználó által elért eredményeket, például az edzések számát, az elvégzett gyakorlatokat és az előrehaladást. Az oldal interaktív jellege a Material-UI és a Framer Motion könyvtárak használatának köszönhetően dinamikus elemekkel és animációkkal van fűszerezve, így a felhasználók nemcsak statikus információkat kapnak, hanem élményt is, miközben navigálnak az oldalon. Az animációk segítenek a tartalom kiemelésében, miközben a felhasználói élmény fokozódik, így az oldal nemcsak funkcionális, hanem vizuálisan is vonzó.

*Forrás fájl:* \TopForm\reactapp1.client\src\components/profilePage.jsx

Beállítások oldal**:** Ez a komponens a felhasználói beállítások kezelésére szolgál, lehetőséget biztosítva a felhasználó adatainak, profilképének és jelszavának frissítésére. Az oldalon a felhasználó könnyedén módosíthatja az alapvető információkat, miközben az adatokat API hívások segítségével töltjük be és mentjük el. A profilkép részben az alapértelmezett kép vagy a felhasználó által feltöltött új kép jelenik meg, biztosítva a személyre szabott megjelenést. A felhasználói adatok szekcióban a felhasználónév és az email cím szerkesztése lehetséges, míg a jelszó változtatásához az aktuális és az új jelszó megadása szükséges, mindkét jelszót meg kell erősíteni, hogy elkerüljük a hibás módosítást. A frissítések a POST metódussal történnek, token alapú hitelesítéssel, így biztosítva a biztonságos adatkezelést. A felhasználói élmény javítása érdekében a felület MUI komponensek és Framer Motion animációk segítségével dinamikusan jeleníti meg a változtatásokat, például a betöltési állapotot, hibák és sikerüzenetek kezelését. Az oldal különböző állapotokat kezel, mint például a betöltési időszak, hibaüzenetek és a sikeres módosításokat követő értesítések, így biztosítva, hogy a felhasználó mindig tudja, mi történik az adataival.

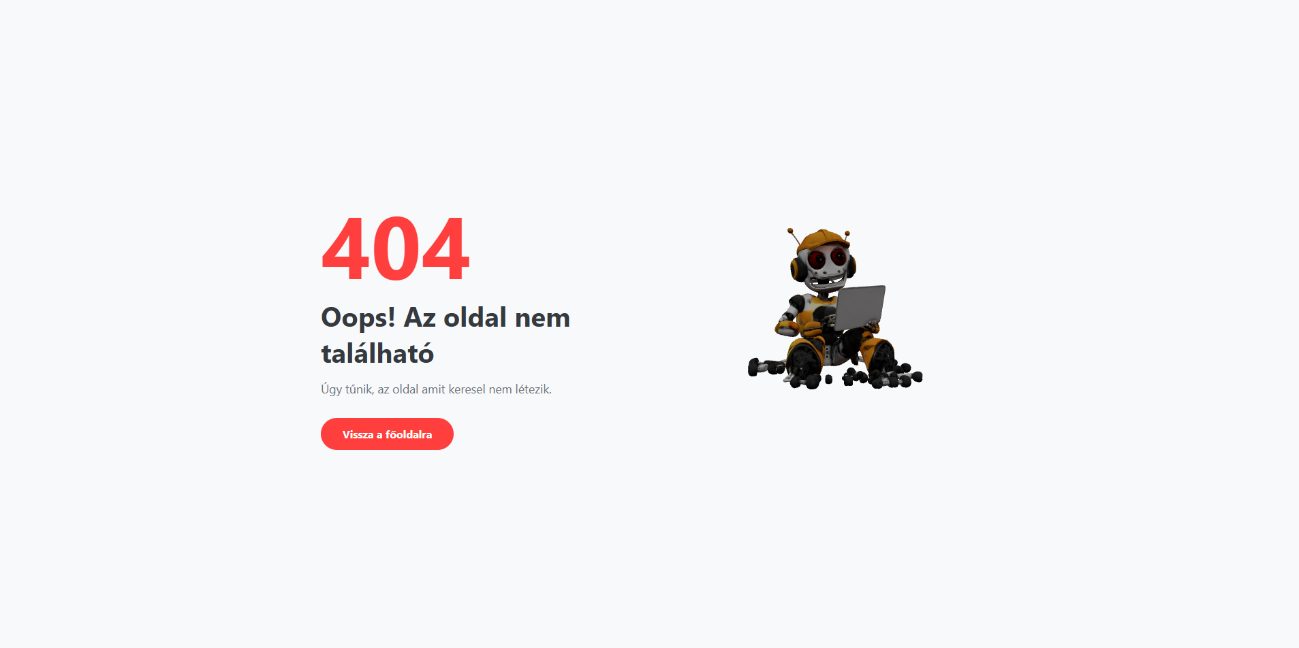
*Forrás fájl:* \TopForm\reactapp1.client\src\components/settings.jsx

A felhasználó nincs bejelentkezve oldal: Ez a komponens az "Access Denied" oldalt jeleníti meg, amikor a felhasználó jogosulatlanul próbál hozzáférni egy védett területhez, például edzéstervekhez vagy egyéb személyre szabott tartalmakhoz. A React Three Fiber és a Drei könyvtár segítségével egy látványos 3D-s modelt jelenítünk meg, amely fokozza az élményt és figyelmezteti a felhasználót, hogy bejelentkezés szükséges. A 3D-s modell a BodyguardModel, amely egy GLTF fájlból töltődik be, és a vászonra renderelődik, így dinamikus és interaktív vizuális hatást biztosít. Az oldal stílusa animációkkal és vizuális elemekkel van kiegészítve: pulzáló körök, animált szövegek és az oldal alsó része, amely mind hozzájárul a dinamikus és figyelemfelkeltő megjelenéshez. A navigációs lehetőségek között a felhasználó két opcióval rendelkezik:

* Bejelentkezés (SIGN IN): Ez a gomb átirányítja a felhasználót a bejelentkezési oldalra, hogy elvégezhesse a szükséges hitelesítést.
* Vissza a főoldalra (RETURN HOME): Ez az opció visszavezet a főoldalra, ahol a felhasználó egyéb információkat találhat.

*Forrás fájl:* \TopForm\reactapp1.client\src\ErrorPages/AccessDeniedPage.jsx

Hibás URL oldal: Ez a komponens egy 404-es hibaoldalt jelenít meg, amikor a felhasználó egy nem létező oldalra próbál navigálni. A dizájn tiszta és modern, a háttérben pedig egy 3D-s modell jelenik meg, amely interaktívan forog.

* 3D Modell: A useGLTF hook segítségével betöltjük az Error\_modell.glb fájlt. A modell optimalizált világítással jelenik meg, és a OrbitControls lehetővé teszi a modell körbeforgatását. A useEffect gondoskodik a memória kezelésről, hogy a modell eltávolításkor ne hagyjon felesleges erőforrásokat.
* Stílusok: A styled-components segítségével valósul meg a responszív dizájn, ami két oszlopra osztja a tartalmat kisebb és nagyobb képernyőkön. A színek és a szövegek a hibás oldal hangulatát erősítik, a fő gomb pedig a felhasználót visszavezeti a főoldalra.
* Interakciók: Az oldal biztosít egy hívogató "Vissza a főoldalra" gombot, és az animált 3D-s modell dinamikus megjelenítése egyedi élményt ad a felhasználóknak.

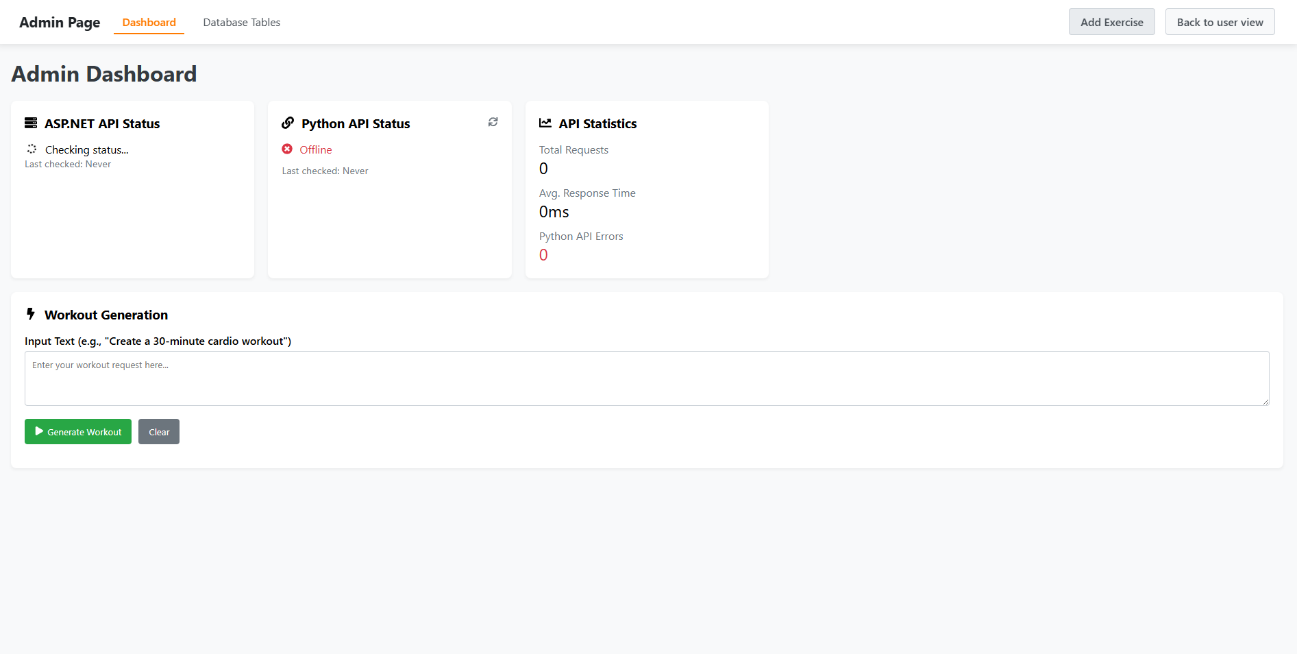
*Forrás fájl:* \TopForm\reactapp1.client\src\ErrorPages/PageNotFound.jsx

Admin Dashboard**:** Ez a kód egy adminisztrációs irányítópultot (Dashboard) valósít meg, amely az API-k státuszát és különböző statisztikákat jelenít meg. Két fő API-t figyel: egy ASP.NET API-t és egy Python API-t, amelyek státuszát és válaszidőit ellenőrzi, valamint a generált edzéstervekhez kapcsolódó adatokat is nyújt.

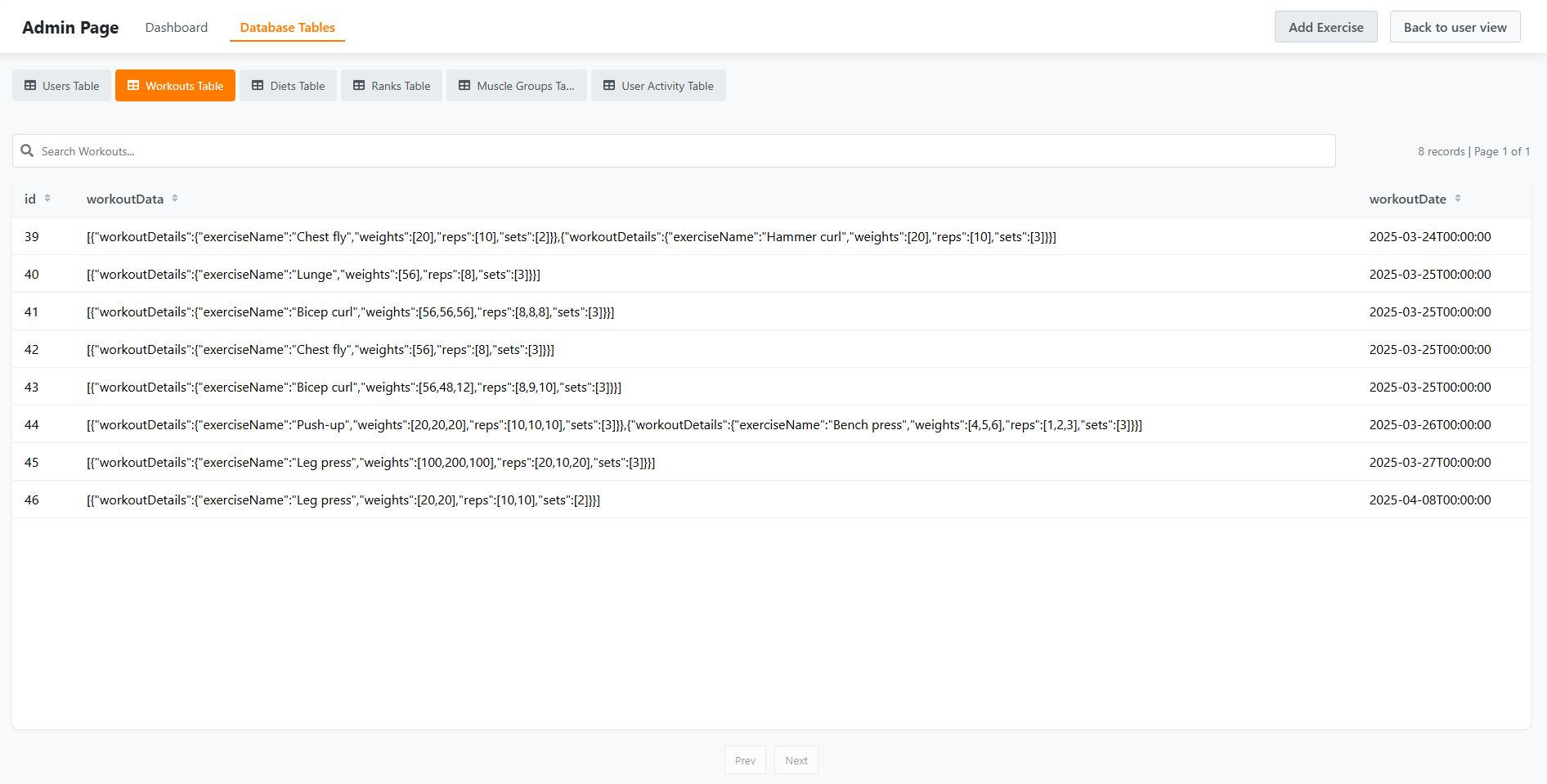
Főbb funkciók:

* API státuszok: Az ASP.NET és Python API állapotának folyamatos ellenőrzése és frissítése.
* API statisztikák: Összes kérések, válaszidők és hibák statisztikai megjelenítése.
* Szöveg generálás: Az admin felület lehetővé teszi szöveg generálását edzésterv alapján, és figyelemmel kíséri a folyamatot, jelezve, ha hibák lépnek fel.

A kód a useEffect hookot használja, hogy folyamatosan frissítse az API státuszt, és az API-k hibáira figyelmeztető üzeneteket ad. Az edzéstervek generálása közben egy input mezőn keresztül lehet megadni a kívánt szöveget, és az alkalmazás visszaadja a generált választ.

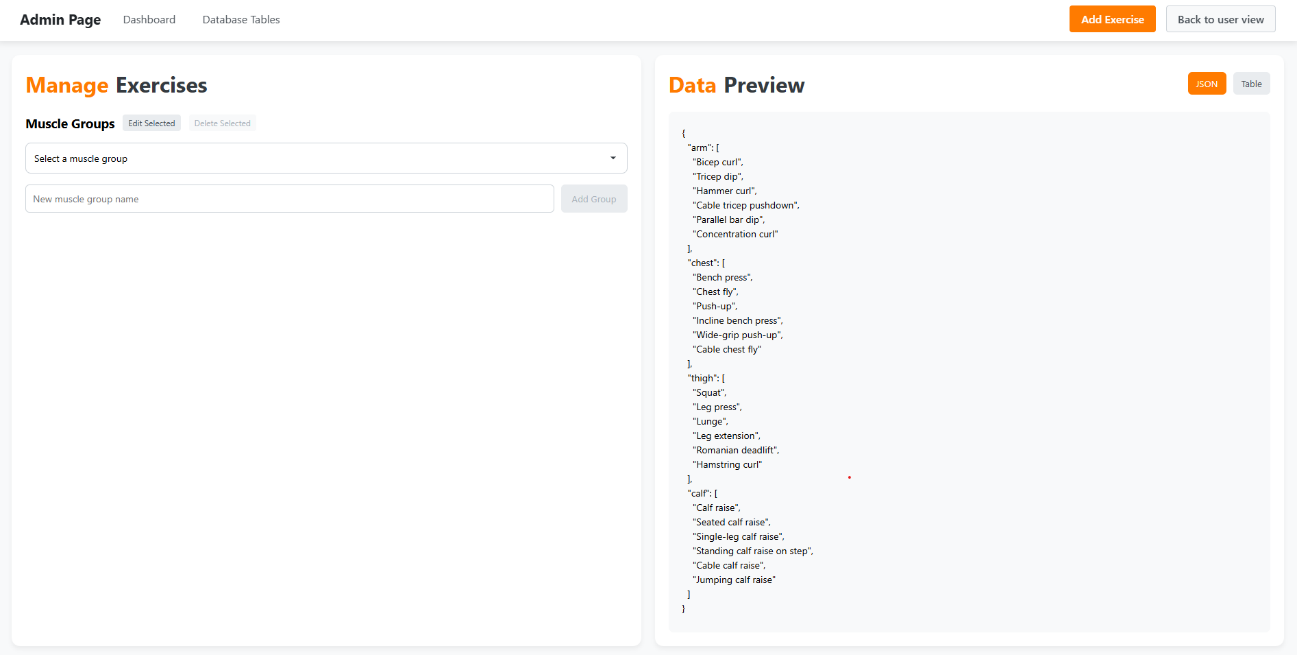
A UI React-ikonokat használ az állapotok vizuális ábrázolására, mint a "zöld" (működő), "piros" (offline) és "sárga" (instabil) státuszok.

*Forrás fájl:* \TopForm\reactapp1.client\src\Admin\AdminComponents/DashboardContent.jsx

Admin adatbázis oldal: Ez a komponens egy admin felületet biztosít adatbázis kezeléshez, ahol hat különböző tábla (felhasználók, edzések, étrendek, rangok, izomcsoportok és felhasználói tevékenységek) adatait lehet megtekinteni és kezelni. A felület lehetővé teszi az adatok rendezését oszlopok szerint, keresést a táblákban, valamint oldaltöréses lapozást a nagyobb adathalmazok könnyebb kezelése érdekében. A felhasználói táblánál különleges funkcióként inline szerkesztés és törlés is elérhető, mindezt egy letisztult, sötét téma alapú felületen. A komponens intelligens tooltip funkcióval rendelkezik, ami egérmutatóval megjeleníti a kapcsolódó felhasználói adatokat, miközben a Material-UI komponensek és a Framer Motion animációk professzionális megjelenést biztosítanak. A rendszer hibakezelése részletes hibaüzeneteket jelenít meg, és lehetőséget biztosít az újrapróbálkozásra kapcsolódási problémák esetén.

*Forrás fájl:* \TopForm\reactapp1.client\src\Admin\AdminComponents/DatabaseTables.jsx

Admin edzés hozzá adás oldal: Ez a komponens lehetővé teszi a gyakorlatok és izomcsoportok kezelését egy lokálisan betöltött JSON fájl alapján, a viewMode tárolja a nézet módot, a exercises a gyakorlatokat, a selectedMuscleGroup az aktuális izomcsoportot, a newExercise az új gyakorlat nevét, a newMuscleGroup az új izomcsoport nevét, az editMode a szerkesztett elem típusát, az editValue pedig az új értéket, miközben a isLoading a betöltési állapotot kezeli. A funkciók közé tartozik a gyakorlat és izomcsoport hozzáadása, törlése, szerkesztése, a reszponzív dizájn és a betöltési animáció, valamint a komponens API-k vagy adatbázisok integrálásával bővíthető.

*Forrás fájl:* \TopForm\reactapp1.client\src\Admin\AdminComponents/ExercisesEditor.jsx

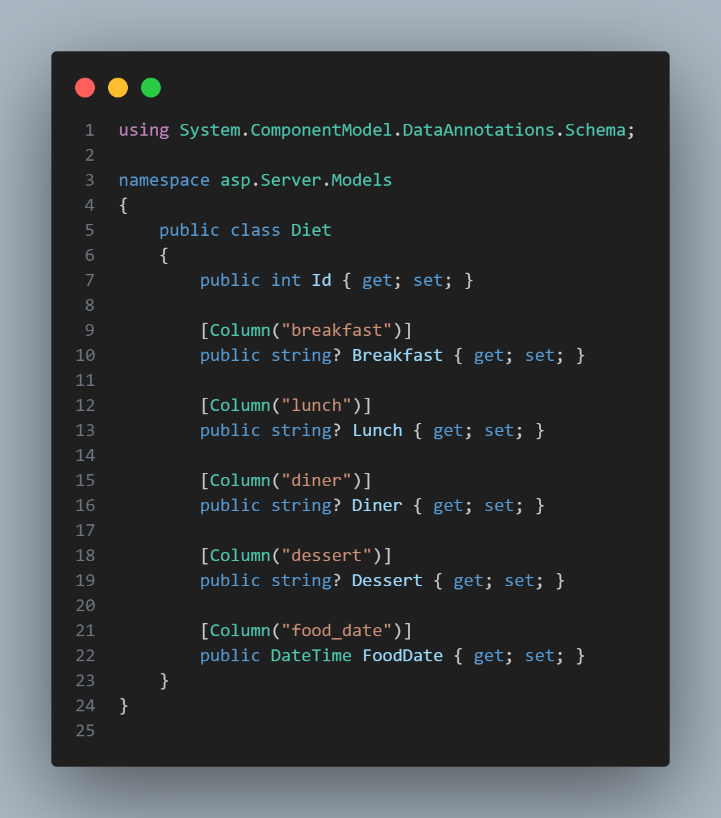
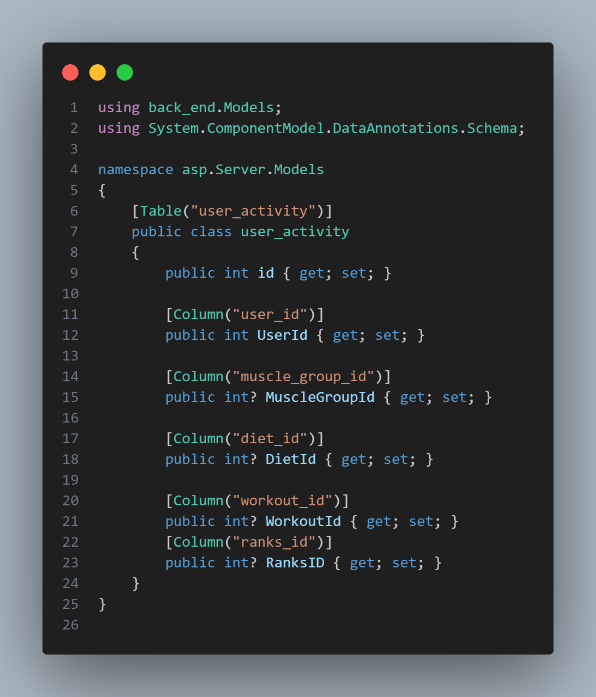
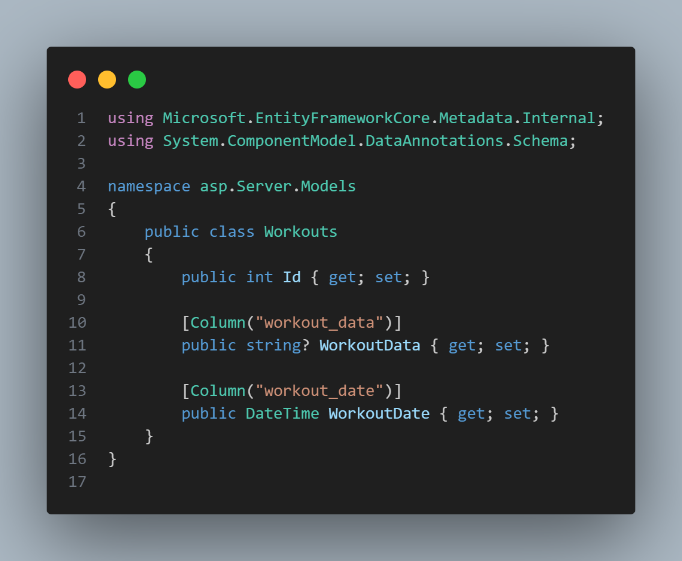
# Back End

**(ASP .NET 8.0)**

ApplicationDbContext:Az *ApplicationDbContext* osztály az *Entity Framework Core* segítségével kezeli az adatbázist, és biztosítja az adatmodellek közötti kapcsolatokat. A *DbSet*-ek segítségével az alkalmazás különböző táblák (pl. *Users, Workouts, UserActivity*) adatain dolgozik, lehetővé téve az új rekordok hozzáadását, meglévő rekordok frissítését vagy törlését. Az *OnModelCreating* metódusban az oszlopneveket és a táblák közötti kapcsolatokat testre szabhatjuk, míg az *OnConfiguring* metódusban figyelmeztetéseket és konfigurációs beállításokat alkalmazhatunk. Az alkalmazás így könnyedén kommunikál az adatbázissal, és végrehajthatóak a szükséges műveletek.

*Forrás fájl:* \TopForm\ReactApp1.Server\Data/ApplicationDbContext.cs

Modellek:

A következő táblázatban láthatóak azok a modellek, amelyeket az ASP.NET alapú back-endben használtunk. Ezek segítségével az API műveleteket tud végrehajtani az adatbázissal való kommunikáció során. Minden modell egy-egy adatbázistáblának felel meg vagy a tábla töredékének a könnyebb kezelhetőség érdekében (*ODT* modellek), lehetővé téve az adatok lekérdezését, frissítését és mentését. A kódban összes 7 modell kap külön fájlt, ebből 6 rendes modell, 1 *ODT* modell, de a *Controllerekben* is találhatóak *ODT* modellek.

*Forrás mappa:* \TopForm\ReactApp1.Server\Models/

## Felhasználói felület kontrollerei:

DietController (*POST*): Egy *API* végpontot biztosít a felhasználók étrendjének rögzítésére. Csak hitelesített felhasználók érhetik el (*[Authorize]* *attribútum*).

**Bemenet**: A kérés tartalmazhat étkezéseket: Reggeli, Ebéd, Vacsora, Desszert. Mindegyik lista *FoodItem* objektumokat tartalmaz (név, adag, kalória).

**Feldolgozás lépései:**

1. **Validáció:** Ellenőrzi, hogy legalább egy étkezés meg van-e adva.
2. **Felhasználó azonosítása:** A *JWT* *tokenből* kinyeri a *UserId*-t.
3. **Diet rekord létrehozása:** Sorokba rendezi a listákat JSON formátumba, majd menti.
4. ***user\_activity* frissítés:**
   * Ha van korábbi *user\_activity* rekord *DietId* nélkül frissíti.
   * Ha nincs új *user\_activity* rekordot hoz létre.
5. **Tranzakciókezelés:** Minden adatbázis művelet tranzakcióban történik, hibakezeléssel.

*Forrás fájl:* \TopForm\ReactApp1.Server\Controllers/DietController.cs

GenerateWorkoutControler: Három API végpontot biztosít, amely lehetővé teszi edzéstervek generálását egy Python alapú AI szolgáltatás segítségével. A backend közvetítőként működik a frontend és a Python API között.

**Fő végpont** (*POST*): /api/generateworkout/generate

**Bemenet:**

A kérés tartalmaz egy InputText mezőt, ami leírja a kívánt edzéstervet (pl. szint, napok száma, célzott izomcsoportok).

**Feldolgozás lépései:**

1. **Validáció:** Ellenőrzi, hogy az InputText mező nem üres-e.
2. **Kérés küldése a Python API-nak:** A megadott szöveget POST kéréssel továbbítja a http://localhost:5000/generate címre.
3. **Válasz feldolgozása:**
   * Ha a Python válasza JSON, akkor közvetlenül visszaküldi.
   * Ha nem JSON, akkor "generatedText" kulcs alatt küldi vissza az eredményt.
4. **Hibakezelés:**
   * Sikertelen HTTP kérés → 503-as válasz.
   * Egyéb kivétel → 500-as válasz, naplózással.

**Második végpont** (GET): /api/generateworkout/health

**Cél:** Ellenőrzi az ASP.NET API működését.

**Harmadik végpont** (GET): /api/generateworkout/python-status

**Cél:** Lekérdezi a Python API elérhetőségét és működését.

**Feldolgozás lépései:**

* Tesztkérés küldése az AI API-hoz.
* Válasz alapján visszaadja az elérhetőséget és állapotkódot.

*Forrás fájl:* \TopForm\ReactApp1.Server\Controllers/GenerateWorkoutController.cs

GetDietController (*GET*): Lehetővé teszi az adott dátumhoz tartozó diéta lekérdezését hitelesített felhasználók számára. Az *API* a *UserActivity* tábla alapján szűri a felhasználóhoz tartozó diétákat.

**Bemenet:** Dátum szöveg (*string*) formátumban (pl. "2025-04-10")

**Feldolgozás lépései:**

1. **Felhasználó azonosítása:** A *JWT* *tokenből* kinyeri a *UserId*-t.

* Ha hiányzik vagy érvénytelen: 401-es hiba kóddal tér vissza

1. **Dátum validálása:** Ellenőrzi a dátum formátumát.

* Ha nem átalakítható: 400-as hiba kóddal tér vissza

1. **UserActivity ellenőrzés:** Lekéri az összes olyan *DietId*-t, amely a felhasználóhoz tartozik.
2. **Diéta lekérdezés:** A *Diet* táblából kiszűri azokat a rekordokat, amelyek:

* A *FoodDate* egyezik a megadott dátummal.
* A *DietId* szerepel a felhasználó UserActivity rekordjai között.

1. **Deszerializálás:** A diéta mezőket (*Breakfast*, *Lunch*, *Diner*, *Dessert*) visszaalakítja lista (*List<FoodItem>*) formára, ha nem üres.

**Válaszok:**

* 200 (*OK*)**:** Sikeres válasz esetén visszaadja az adott naphoz tartozó diétá(ka)t.
* 400 (*Bad* *Request*): Érvénytelen dátum formátum.
* 401 (*Unauthorized*): Hiányzó vagy hibás UserId.
* 404 (*Not* *Found*): Nincs a felhasználónak étrendje a megadott dátumra.

*Forrás fájl:* \TopForm\ReactApp1.Server\Controllers/GetDietController.cs

GetWorkoutController (GET): Célja, hogy visszaadja egy felhasználó adott napi edzéseit. A rendszer csak hitelesített felhasználókat szolgál ki, és a *UserActivity* táblán keresztül szűri ki a kapcsolódó edzéseket.

**Bemenet:** A kívánt dátum szöveg (*string*) formában (pl. "2025-04-10")

**Feldolgozás lépései:**

1. **Felhasználó azonosítása:** A *JWT tokenből* kiolvassa a *UserId*-t.
   * Ha hiányzik vagy hibás: 401-es hiba kóddal tér vissza.
2. **Dátum validálása:** Ellenőrzi, hogy a dátum paraméter helyesen formázott dátum-e.
   * Ha nem: 400-as hiba kóddal tér vissza.
3. ***UserActivity* ellenőrzés:** Lekérdezi a *UserActivity* táblából a felhasználóhoz tartozó *WorkoutId* értékeket.
4. **Edzések lekérdezése:** A *Workouts* táblából lekérdezi a megadott naphoz tartozó edzéseket a lekért *ID*-k alapján.
5. ***WorkoutData* feldolgozása:** Az edzések *WorkoutData* mezőjét *JSON*-ról objektummá alakítja a *ParseWorkoutData* metódus segítségével.

**Válaszok:**

* 200(*OK*): Sikeres válasz esetén visszatér az edzések listájával.
* 400(*Bad Request*): Hibás dátum formátum.
* 401(*Unauthorized*): A *JWT* *tokenből* nem olvasható ki érvényes *UserId*.
* 404(*Not* *found*): Nincs edzés a megadott napra.

*Forrás fájl:* \TopForm\ReactApp1.Server\Controllers/GetWorkoutController.cs

LeaderboardController (*GET*): egy végpontot biztosít, amely visszaadja az összes felhasználóhoz tartozó ranglista információt: profil adatokat, edzéseket, rangokat és izomcsoportokat.

**Autentikáció:** Ez a végpont nemigényel hitelesítést.

**Feldolgozás lépései:**

1. **Felhasználók lekérdezése**: Lekéri az összes felhasználót, beleértve:
   * *Id*
   * *Username*
   * *ProfilePicture* (*base64-re konvertálva*)
2. **Rangok és izomcsoportok lekérdezése**: Lekéri az összes rekordot a *Ranks* és *MuscleGroups* táblákból.
3. ***UserActivity* lekérdezése**: Lekéri minden felhasználó összes aktivitását (*UserActivity*).
4. **Leaderboard összeállítása**:

* Minden felhasználóhoz:
  + - Lekéri a hozzá tartozó *WorkoutId*-kat.
    - Betölti a megfelelő *Workouts* rekordokat.
    - Az első aktivitás alapján meghatározza a *Rank*-ot és *MuscleGroup*-ot.
    - Az eredményhez csatolja a felhasználó profil képét is *base64* formában.

**Válaszok: 200 (*OK*):** Sikeres válasz esetén visszatér az ranglista listájával.

*Forrás fájl:* \TopForm\ReactApp1.Server\Controllers/LeaderboardController.cs

LoginController (*POST*): célja, hogy a felhasználók bejelentkezését kezelje. A rendszer *JWT* alapú hitelesítést használ, és visszaad egy hozzáférési *tokent*, ha a felhasználó adatai érvényesek.

**Bemenet**:

* A felhasználó neve.
* A felhasználó jelszava.

**Feldolgozás lépései:**

1. **Bemenet ellenőrzése:**
   * Ha a bemenet nem tartalmazza a Felhasználónév és Jelszó mezőket, akkor 400 (*Bad* *Request*) válasz érkezik, amely figyelmeztet arra, hogy a felhasználónév és jelszó szükséges.
2. **Felhasználó keresése:**
   * A rendszer a megadott Felhasználónév alapján keres egy felhasználót az adatbázisban.
   * Ha a felhasználó nem található, akkor 404 (*Not* *Found*) válasz érkezik.
3. **Jelszó ellenőrzése:**
   * A rendszer ellenőrzi, hogy a megadott jelszó megegyezik-e a tárolt jelszóval, amelyet *bcrypt*-tel *hashelünk*.
   * Ha a jelszó nem egyezik, akkor 401 (*Unauthorized*) válasz érkezik.
4. ***JWT* *token* generálása:**
   * Ha a felhasználó és a jelszó helyes, a rendszer generál egy *JWT* *token*-t.
   * A *token* tartalmazza a felhasználó azonosítóját (*UserId*), nevét, email címét és születési dátumát.
   * A válasz tartalmazza a generált *tokent*, amelyet a felhasználó később hitelesítésre használhat.

*Forrás fájl:* \TopForm\ReactApp1.Server\Controllers/LoginController.cs

ProfileController (GET): Célja, hogy visszaadja a felhasználó profilját, beleértve az alap adatokat, izomcsoportot, aktivitásokat és edzéseket. A rendszer *JWT* *token* alapú hitelesítést alkalmaz, amelyet az *Authorization* *header* tartalmaz.

**Bemenet:** *JWT* *token* az *Authorization* *header*-ben.

**Feldolgozás lépései:**

1. **Hitelesítés ellenőrzése**: Ha a *token* érvénytelen vagy hiányzik, 401 (*Unauthorized*) válasz érkezik.
2. **Felhasználói adatok lekérdezése**: A felhasználó nevét, felhasználónevét, e-mailjét és profilképét kérdezi le.
3. **Felhasználói aktivitások és rang lekérdezése**: A rendszer lekéri a felhasználó aktivitásait és rangját.
4. **Izomcsoport adatainak lekérdezése**: Lekérdezi a felhasználó izomcsoportjait.
5. **Edzés adatok lekérdezése**: A felhasználó edzésadatait lekéri.
6. **Profil adatainak összeállítása és visszaadása**: A rendszer összeállítja a profil adatokat és JSON formátumban visszaadja.

**Válasz:**

A válasz egy JSON objektumot tartalmaz, amely az alábbi adatokat tartalmazza:

* Felhasználó neve, felhasználóneve, e-mailje, profilképe.
* Izomcsoportok nevei és súlyai.
* Edzések és gyakorlatok.
* Rang és pontszám.

*Forrás fájl:* \TopForm\ReactApp1.Server\Controllers/ProfileController.cs

Registration2Controller (*POST*): célja, hogy a felhasználó adatait és izomcsoportjait frissítse az API-n keresztül. Ezt a kontrollert alkalmazzuk a regisztráció második szakaszánál.

**Bemenet:**

* *JWT**token* az *Authorization* *header*-ben, amely biztosítja a felhasználó hitelesítését.

**Feldolgozás lépései:**

1. **Hitelesítés ellenőrzése:**
   * A *JWT* *tokenből* lekérdezi a felhasználó azonosítóját (*UserId*).
   * Ha a *token* hiányzik vagy érvénytelen, 401 (*Unauthorized*) válasz érkezik.
2. **Felhasználó ellenőrzése:**
   * A *user\_id* alapján lekérdezi az adatbázisból a felhasználót.
   * Ha a felhasználó nem található, 404 (*Not* *Found*) válasz érkezik.
3. **Izomcsoportok frissítése:**
   * A megadott izomcsoportok listáját hozzáadja a *MuscleGroups* táblához.
4. **Felhasználói aktivitások frissítése:**
   * Az aktivitásokat frissíti, hogy a hozzáadott izomcsoportok adatainak *ID*-ja hozzá legyen rendelve a felhasználó aktivitásaihoz.
5. ***Transaction* kezelés:**
   * Az adatbázis műveletek tranzakcióban történnek így, ha bármelyik lépés sikertelen, a tranzakció visszavonásra kerül, hogy az adatbázis ne maradjon inkonzisztens.
6. **Válasz:**
   * Ha minden sikeresen végrehajtódig, a rendszer 200 (*OK*) választ ad, amely tartalmazza a frissítést.
   * Ha hiba történik, 500(*Internal* *Server* *Error*) választ ad a részletes hibaüzenettel.

*Forrás fájl:* \TopForm\ReactApp1.Server\Controllers/Register2Controller.cs

RegistrationController (*POST*): Ez a felelős a felhasználói regisztrációs folyamat kezeléséért. A regisztráció során a felhasználó megadhatja a szükséges adatokat (felhasználónév, email, jelszó, név, születési dátum), és ha minden adat érvényes, a rendszer létrehozza a felhasználót az adatbázisban, titkosítja a jelszót, valamint egy *JWT* *token*-t generál a felhasználó számára a későbbi hitelesítéshez.

**Regisztráció folyamat:**

1. **Validálás:**
   * Ellenőrzi, hogy a *request* nem null és hogy a Felhasználónév és Jelszó nem üresek.
   * Ha a felhasználónév már létezik az adatbázisban, 400 (*Bad* *Request*) válasz érkezik.
2. **Jelszó titkosítása:**
   * A jelszót a *BCrypt* segítségével titkosítja, hogy biztonságosan tárolhassa az adatbázisban.
3. **Felhasználó létrehozása:**
   * Létrehozza az új felhasználót a *Users* táblában, beleértve a felhasználói adatokat (*username*, *email*, *name*, *birthdate*, *password*).
4. **Felhasználói aktivitás rögzítése:**
   * Az új felhasználóhoz létrehoz egy *user\_activity* rekordot, amely a felhasználó *ID*-ját tartalmazza.
5. ***JWT* *token* generálása:**
   * A sikeres regisztráció után egy *JWT* *token*-t generál, amelyet a válaszban visszaküld. A *token* tartalmazza a felhasználó *ID*-ját és egyéb fontos adatokat (*username*, *email*, *name*, *birthdate*).
6. ***Transaction* kezelés:**
   * A regisztrációs folyamat tranzakcióval történik, hogy biztosítva legyen, hogy ha bármelyik lépés hiba történik, a változtatások visszagörgetésre kerüljenek.

*Forrás fájl:* \TopForm\ReactApp1.Server\Controllers/RegistrationController.cs

UserController: Különböző felhasználói műveletekhez biztosít végpontokat, beleértve a profilkép feltöltést, profil frissítést, a felhasználói adatok lekérését és a *JWT* *token* alapú hitelesítést.

**1. Profilkép feltöltése (*POST*)**

* **Végpont**: /upload-profile-picture
* **Leírás**: Lehetővé teszi a felhasználók számára, hogy feltöltsenek egy profilképet, amelyet *base64* formátumban adnak meg.

**2. Profilkép lekérése (*GET*)**

* **Végpont**: /get-profile-picture
* **Leírás**: Lehetővé teszi a felhasználók számára, hogy letöltsék a profilképüket.

**3. Felhasználói adatok frissítése (*POST*)**

* **Végpont**: /update
* **Leírás**: Lehetővé teszi a felhasználók számára, hogy frissítsék felhasználói nevüket és/vagy jelszavukat. A jelszó frissítése esetén az aktuális (régi) jelszót is meg kell adni.

**4. Felhasználói adatok lekérése (*GET*)**

* **Végpont**: /details
* **Leírás**: Lehetővé teszi a felhasználók számára, hogy lekérjék a nevüket, felhasználónevüket és email címüket.

*Forrás fájl:* \TopForm\ReactApp1.Server\Controllers/UserController.cs

WorkoutController (POST): Ez a végpont lehetővé teszi a felhasználók számára, hogy edzéseket rögzítsenek, beleértve az edzés neveit, súlyokat, ismétléseket és szetteket. Az edzés teljesítményének kiszámítása is megtörténik, és a felhasználó rangja ennek megfelelően frissül.

**Folyamat**

* **Validálás**: Minden mezőnek érvényes adatokat kell tartalmaznia, és minden szükséges mezőnek jelen kell lennie.
* **Felhasználói autentikáció**: A UserId kinyerésre kerül a JWT tokenből, és a kérés ehhez a felhasználóhoz lesz rendelve.
* **Edzés részletek tárolása**: Az edzés adatokat JSON formátumban tároljuk az adatbázisban.
* **Pontszám számítása**: Az edzés pontszámát a súlyok, ismétlések és szettek alapján számítjuk ki.
* **Rang frissítése**: Ha a felhasználónak már van rangja, azt frissítjük az új pontokkal. Ha nincs, új rangot hozunk létre és rendeljük hozzá.
* **Tranzakció kezelés**: Az edzés és a felhasználói aktivitás adatainak mentése tranzakcióval történik, hogy biztosítsuk az adatkonzisztenciát.

**Segédmetódusok**

* **CalculatePointsAndName**: Ez a metódus kiszámítja a teljes edzés pontszámát az adott súlyok, ismétlések és szettek alapján. Emellett meghatározza a rang nevét a teljes pontszám alapján.
* **GetNameBasedOnPoints**: Ez a metódus rangot rendel a pontszám alapján.

*Forrás fájl:* \TopForm\ReactApp1.Server\Controllers/WorkoutController.cs

Admin Kontrollerek:

AdminTableController (*GET*): Célja, hogy a rendszergazda számára hozzáférést biztosítson az adatbázisban tárolt különböző táblákhoz (például felhasználók, edzések, diéták, izomcsoportok, rangok, felhasználói aktivitások).

A controller *API* végpontokat biztosít, amelyek a következő adatokat szolgáltatják:

* **Felhasználók (*Users*)**
* **Edzések (*Workouts*)**
* **Diéták (*Diet*)**
* **Izomcsoportok (*MuscleGroups*)**
* **Rangok (*Ranks*)**
* **Felhasználói aktivitások (*UserActivity*)**

Minden végpont a megfelelő táblát lekérdezi és visszaadja az adatokat *JSON* formátumban.

*Forrás fájl:* \TopForm\ReactApp1.Server\Controllers/AdminTableController.cs

AdminUserTableController (*PUT*, *DELETE*):

***DeleteUser*** végpont egy adott felhasználót és a hozzá tartozó adatokat (aktivitások, edzések, diéták, rangok, izomcsoportok) törli az adatbázisból.

***UpdateUserByID*** végpont a felhasználó adatait frissíti az adatbázisban.

**Működési logika (*UpdateUserByID*):**

* A frissítés előtt ellenőrzi, hogy létezik-e a felhasználó az adatbázisban.
* Ha a felhasználó nem található, 404 (*Not* *Found*) hibát küld vissza.
* Ha létezik, frissíti a felhasználó nevét, e-mail címét, felhasználónevét és nemét.
* Végül menti a változtatásokat az adatbázisba, és egy sikeres válasz küldése után visszatér.

*Forrás fájl:* \TopForm\ReactApp1.Server\Controllers/AdminUserTableController.cs

Mesterséges Intelligencia: Ez a *Flask* alkalmazás egy edzésgeneráló *API*-t biztosít, amely a *DistilGPT-2* nyelvi modellt használja a *transformers* könyvtár segítségével. Az *API* célja, hogy személyre szabott edzésprogramokat generáljon felhasználói bemenetek alapján. Az edzésprogramok a felhasználó által megadott izomcsoportokra és nehézségi szintre vonatkoznak, és a generált edzés tartalmazza a gyakorlatokat, az ismétléseket, a sorozatokat, valamint a használandó súlyokat.

Rendszerkövetelmények

* *Flask*: A webes alkalmazás kiszolgálására
* *transformers*: A *GPT-2* modell használatához
* *Flask-CORS*: A *CORS* engedélyezéséhez a böngésző-kliens kommunikáció során
* *PyTorch*: A modell futtatásához szükséges könyvtár

**API végpont**: *POST* *(/generate)*

* Leírás: Az *API* ezen az végponton fogadja a felhasználói kéréseket és generál egy edzésprogramot. A bemeneti *JSON* tartalmazza az *InputText* (vagy *inputText*) mezőt, amely tartalmazza a kívánt edzés típusát.

**Válasz Struktúra**:

* *generatedText*: Az *API* által generált edzésprogram, amely gyakorlatokat, ismétléseket, sorozatokat és súlyokat tartalmaz.
* *timestamp*: Az edzésprogram generálásának időpontja *ISO* *8601* formátumban.

**Hibák Kezelése:**

* 400 - *Bad Request*: Ha a kérés nem *JSON* formátumban érkezik, vagy hiányzik az *InputText* mező.
* 500 - *Internal Server Error*: Belső hiba, ha a modell generálása vagy az *API* működése során probléma lép fel.

**Alkalmazott Algoritmusok:**

1. **Edzésgenerálás**:
   * Az *InputText* alapján a rendszer megpróbálja azonosítani a kívánt izomcsoportokat és nehézségi szintet. Ha az inputban szereplő nehézségi szint nem található meg, akkor a rendszer alapértelmezés szerint "közepes" nehézségi szintet használ.
   * A *DistilGPT-2* modell generálja a szöveget, amelyből az edzésprogramot kiolvassuk. Ha a generált szöveg érvényes, akkor azt visszaküldjük. Ha nem, akkor a rendszer egy helyettesítő edzéstervet generál.
2. ***Fallback* Edzésgenerálás**:
   * Ha a modell nem ad vissza érvényes válaszokat, akkor az alkalmazás egy egyszerűsített szabályrendszert alkalmazva generál egy edzésprogramot. Ebben az esetben a rendszer random módon választ gyakorlatokat a különböző izomcsoportokból és a felhasználó által megadott nehézségi szint szerint határozza meg a sorozatok és ismétlések számát.

**Főbb Funkciók és Részletek:**

1. **Kezdés**: Az *API* *POST* kéréseket fogad, amelyek tartalmazzák az edzésprogram igényeit (pl. izomcsoportok és nehézségi szint).
2. **Szövegfeldolgozás**: Az *InputText* alapján a rendszer a *GPT-2* modellt használja, hogy szöveget generáljon az edzésprogramról.
3. **Validálás**: A generált szöveg megfelelőségét ellenőrizzük egy reguláris kifejezés segítségével. Ha nem találunk megfelelő gyakorlatokat, akkor a *fallback* funkció lép működésbe.
4. **Visszajelzés**: A válaszban szereplő edzésprogram tartalmazza az összes szükséges információt (gyakorlat, ismétlés, sorozat, súly), és az időbélyeg is szerepel benne.

*Forrás fájl:* \TopForm\ReactApp1.Server\AI\ai.py.cs

# A projekt tesztelése

A projekt fejlesztése során különböző tesztelési megközelítéseket alkalmaztunk annak érdekében, hogy biztosítsuk a rendszer stabilitását, a kód helyességét és a felhasználói élmény minőségét. Az alábbi négy módszert használtuk a tesztelés során:

**1. Manuális Tesztelés**

A fejlesztés során az egyik elsődleges tesztelési módszer a manuális tesztelés volt. Ez lehetővé tette számunkra, hogy közvetlenül ellenőrizzük a rendszer működését. A manuális tesztelés során a fejlesztők és tesztelők lépésről lépésre vizsgálták meg az alkalmazás különböző funkcióit, beleértve a felhasználói felületet, a funkcionális műveleteket és az integrációkat, biztosítva, hogy minden komponens megfelelően működjön.

**2. Swagger Tesztelés**

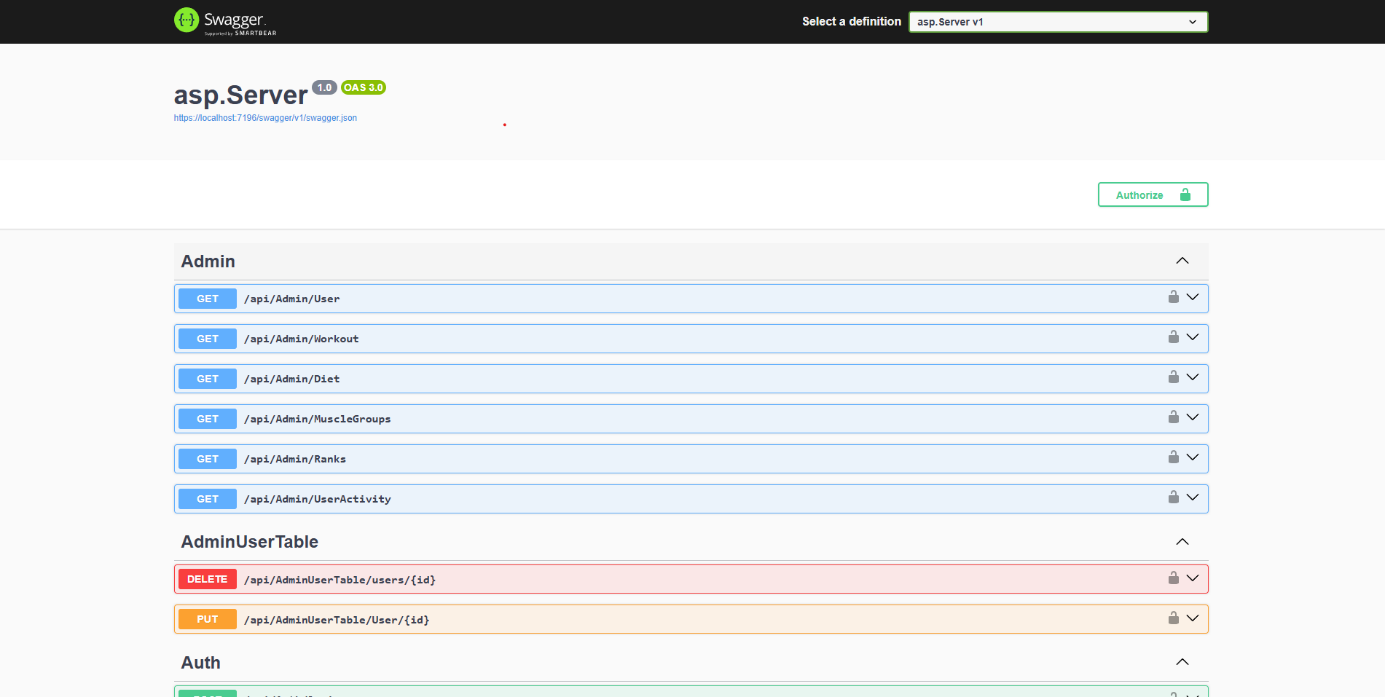
A **Swagger UI** segítségével dokumentáltuk és teszteltük az API végpontokat. A Swagger UI lehetővé tette számunkra, hogy közvetlenül teszteljük az API hívásokat és lássuk az eredményeket. Ez különösen fontos volt az API végpontok helyes működésének ellenőrzésére, mivel automatikusan generált dokumentációval és tesztelési lehetőséggel rendelkezett, így gyorsan és hatékonyan végezhettük el az API teszteléseket.

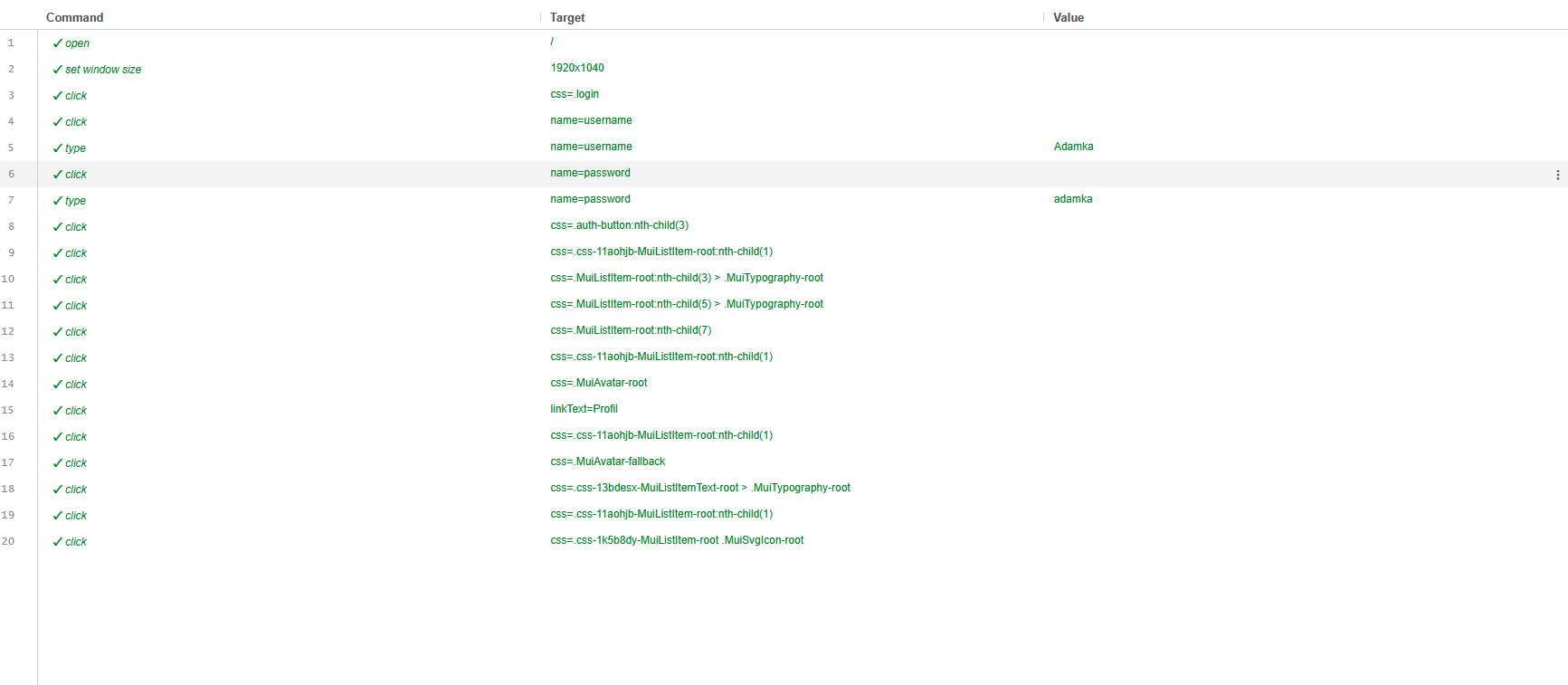
**3. NUnit Tesztek**

A **NUnit** keretrendszert használtuk automatizált egységtesztek írására. Az egységtesztelés biztosította, hogy a kód minden egyes komponense és funkciója megfelelően működik, és segített a hibák korai felismerésében. A tesztek során figyeltünk az alkalmazás logikai rétegeire, adatbázis műveleteire és felhasználói funkcióira, biztosítva a rendszer megbízhatóságát.

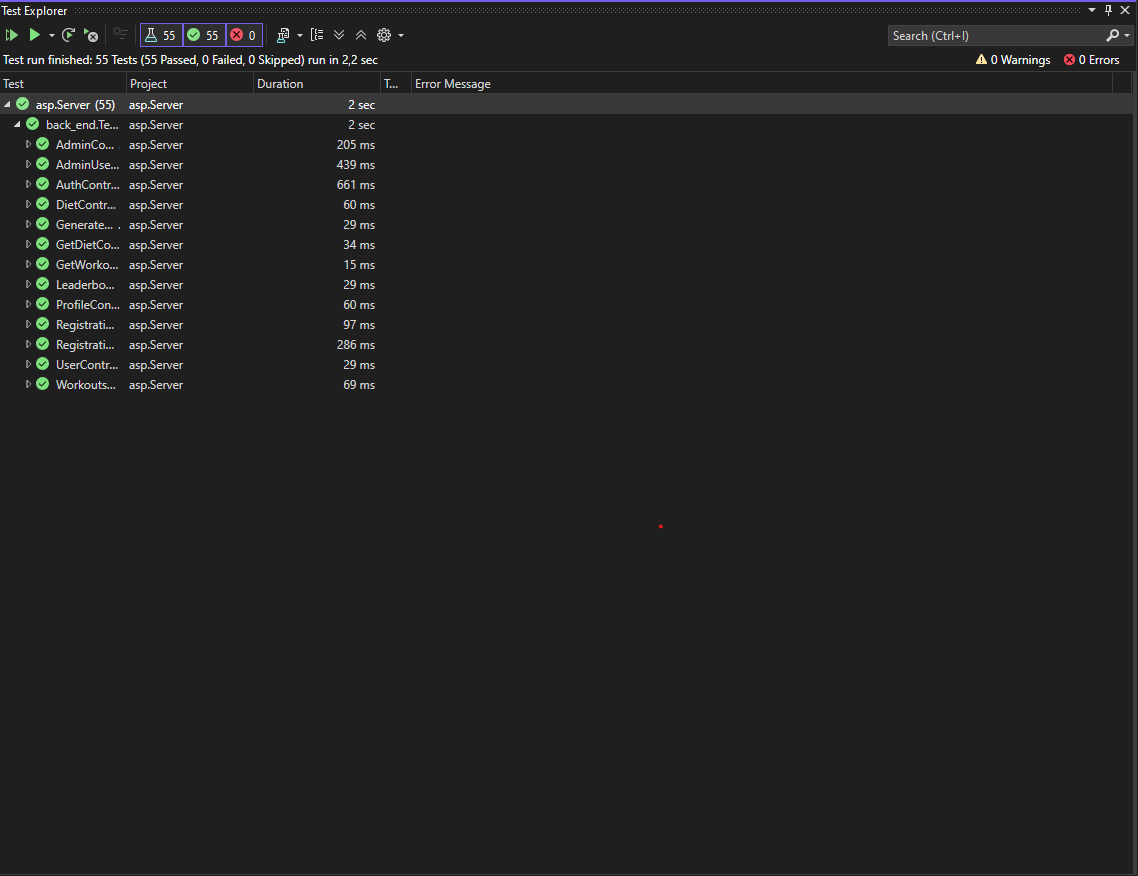
**4. Selenium Tesztelés**

A **Selenium** automatizált tesztelő eszközt használtunk a felhasználói felület (UI) tesztelésére. A Selenium lehetővé tette számunkra, hogy end-to-end teszteket futtassunk, és ellenőrizzük az alkalmazás viselkedését a felhasználói interakciók során. A böngészőkompatibilitás és a felhasználói élmény tesztelése mellett a Selenium segített az alkalmazás működésének automatizált ellenőrzésében, beleértve az űrlapok kitöltését, gombok kattintását és más UI elemeket.

Swagger tesztelés:

Selenium tesztelés**:**

NUnit tesztek:

Összes 55db tesztet írtunk, minden API végpontnak legalább hármat, 

**Például:**  
GetWorkoutControllerTests:

A GetWorkoutControllerTests osztály célja a GetWorkoutController nevű ASP.NET Core vezérlő GetWorkoutByDate metódusának egységszintű tesztelése. A vezérlő felelős egy adott dátumhoz tartozó edzésadatok lekérdezéséért, amelyhez a felhasználói jogosultságot is ellenőrzi.

A tesztek biztosítják, hogy az API helyesen reagál különböző bemenetekre és állapotokra – például érvénytelen felhasználói azonosítóra, hibás dátumformátumra, vagy ha nem található adat.

**Tesztkörnyezet felépítése**

1.**InMemory adatbázis használata**

* A tesztek függetlenek a valódi adatbázistól, mivel egy InMemoryDatabase-t használnak az ApplicationDbContext konfigurálásával.
* Ez lehetővé teszi, hogy minden teszt egy „tiszta lappal” induljon.

2. **Tesztadatok inicializálása a** SetUp() **metódusban**

* Létrehoz egy minta edzésnaplót (Workouts), valamint egy user\_activity rekordot, amely egy adott felhasználót (UserId = 1) hozzárendel az edzéshez.

3. **Tisztítás a** TearDown() **metódusban**

* A tesztek után törli az adatbázist, hogy ne maradjanak visszaállított adatok.

**Tesztek részletes leírása**

GetWorkoutByDate\_ReturnsUnauthorized\_WhenUserIdIsInvalid

* **Cél:** Ellenőrizni, hogy az API visszautasítja az azonosítatlan felhasználót.
* **Elvárt eredmény:** 401 Unauthorized státusz és megfelelő hibaüzenet.

GetWorkoutByDate\_ReturnsBadRequest\_WhenDateIsInvalid

* **Cél:** Ellenőrizni, hogy a rosszul formázott dátum bemenet ("invalid-date") hibát okoz.
* **Elvárt eredmény:** 400 BadRequest és hibaüzenet a dátumformátumról.

GetWorkoutByDate\_ReturnsNotFound\_WhenNoWorkoutsFoundForDate

* **Cél:** Megnézni, hogyan viselkedik az API, ha nincs edzés az adott dátumra.
* **Elvárt eredmény:** 404 NotFound státusz és hibaüzenet: „Nincs edzés erre a napra.”

GetWorkoutByDate\_ReturnsOk\_WhenWorkoutsFoundForDate

* **Cél:** Tesztelni, hogy a rendszer megfelelően visszaadja az edzésadatokat érvényes felhasználó és dátum esetén.
* **Elvárt eredmény:** 200 OK státusz, a válaszban az edzésadatok.

**Kiegészítő segédfüggvény**

SetupControllerContext(string userId)

* Ez a segédfüggvény állítja be a ControllerContext-et úgy, hogy az adott userId szerepeljen benne mint ClaimsPrincipal.
* A UserId claim segítségével a vezérlő tudja, melyik felhasználóhoz tartoznak az adatok.

**Fontos megjegyzések**

* A tesztek biztosítják a vezérlő robusztusságát különféle peremfeltételek mentén.
* A Random().Next() használata Id generálásra nem garantál egyediséget – célszerű lehet fix értékkel helyettesíteni teszteléskor a kiszámíthatóság érdekében.
* A WorkoutData JSON formátumban tárolt struktúra, de a tesztek nem validálják annak tartalmát – csak a lekérdezés sikerességét

Adatbázis  
A képen szöveg, képernyőkép, Betűtípus, szám látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

**Ez az ER (Entity-Relationship) modell a TopForm nevű fitness- és egészségügyi alkalmazás adatbázis-struktúráját mutatja be. A rendszer célja a felhasználók edzésének, étrendjének, izomcsoportjainak és fejlődésének nyomon követése, valamint gamifikált rangrendszer biztosítása.**

**users – Felhasználók**

**A users tábla a rendszerben regisztrált felhasználók adatait tárolja.**

| **Oszlop** | **Típus** | **Leírás** |
| --- | --- | --- |
| **id** | **int** | **Egyedi azonosító** |
| **username** | **varchar(255)** | **Felhasználónév** |
| **email** | **varchar(255)** | **E-mail cím** |
| **password** | **varchar(512)** | **Titkosított jelszó** |
| **ProfilePicture** | **blob** | **Profilkép** |
| **name** | **varchar(255)** | **Teljes név** |
| **BirthDate** | **date** | **Születési dátum** |
| **men** | **tinyint(1)** | **Nem (1 = férfi, 0 = nő)** |

**diet – Étrend**

**Az diet tábla a napi étkezéseket tárolja.**

| **Oszlop** | **Típus** | **Leírás** |
| --- | --- | --- |
| **id** | **int** | **Egyedi étrend azonosító** |
| **breakfast, lunch, diner, dessert** | **longtext** | **Étkezési adatok** |
| **food\_date** | **date** | **Az étrend dátuma** |

**workouts – Edzések**

**A workouts tábla a felhasználók edzéseit írja le.**

| **Oszlop** | **Típus** | **Leírás** |
| --- | --- | --- |
| **id** | **int** | **Egyedi edzésazonosító** |
| **workout\_data** | **longtext** | **Edzés részletei** |
| **workout\_date** | **date** | **Edzés dátuma** |

**muscle\_groups – Izomcsoportok**

**Az muscle\_groups tábla különféle izomcsoportokat és azokhoz kapcsolódó súlyadatokat tárol.**

| **Oszlop** | **Típus** | **Leírás** |
| --- | --- | --- |
| **id** | **int** | **Izomcsoport-azonosító** |
| **name1, name2, name3, name4** | **varchar(255)** | **Izomcsoportok nevei** |
| **kg1, kg2, kg3, kg4** | **int(255)** | **Megmozgatott súly (kg)** |
| **date** | **date** | **Dátum** |

**ranks – Rangok**

**A ranks tábla a gamifikációs rendszer rangjait tárolja.**

| **Oszlop** | **Típus** | **Leírás** |
| --- | --- | --- |
| **id** | **int** | **Rangazonosító** |
| **rank\_name** | **varchar(255)** | **Rang neve** |
| **points** | **int(255)** | **Elért pontszám** |

**user\_activity – Felhasználói tevékenységek**

**A user\_activity a központi kapcsolótábla, amely összeköti a többi entitást, és minden egyes felhasználói naphoz tartozó összesített adatokat tárolja.**

| **Oszlop** | **Típus** | **Leírás** |
| --- | --- | --- |
| **id** | **int** | **Tevékenység azonosító** |
| **user\_id** | **int** | **Kapcsolat a users táblához** |
| **diet\_id** | **int** | **Kapcsolat a diet táblához** |
| **workout\_id** | **int** | **Kapcsolat a workouts táblához** |
| **muscle\_group\_id** | **int** | **Kapcsolat a muscle\_groups táblához** |
| **ranks\_id** | **int** | **Kapcsolat a ranks táblához** |

**Kapcsolatok**

* **Egy felhasználóhoz (users) több user\_activity tartozhat (1:N kapcsolat).**
* **Minden user\_activity egy adott nap étrendjére, edzésére, izomcsoport adataira, és rangjára hivatkozik (N:1 kapcsolatok).**
* **Az adatmodell biztosítja a teljes naplózást: mit ettél, mit edzettél, hogyan fejlődtél, milyen szinten vagy épp.**

Reflexió

A projekt során rengeteget tanultunk, nemcsak a technikai megvalósításról, hanem a tervezés, csapatmunka és problémamegoldás fontosságáról is. A projekt célja az volt, hogy egy webalkalmazást készítsünk el, amely egy edzéskövetést és egészséges életmódot támogató alkalmazás.

A munka elején kihívást jelentett a megfelelő struktúra kialakítása, valamint az, hogy a projektet modulárisan építsük fel. Ezen a ponton megtanultuk, mennyire fontos a jó tervezés és az előzetes dokumentáció. A kód írása során igyekeztünk betartani a tiszta kód elveit, és figyeltünk arra, hogy jól olvasható és karbantartható legyen.

A projekt során számos új technológiát ismerhettünk meg, például az ASP.Net + React és egyéb Frontend és Backend fejlesztői platformokat. Ezek használata eleinte kihívást jelentett, de ahogy egyre többet gyakoroltunk, úgy váltak egyre természetesebbé.

A legnagyobb kihívást a mesterséges intelligencia futtatása jelentette, de végül sikerült megoldani tanáraink segítségével és rengeteg kutatással. Ez sokat segített abban, hogy fejlődjünk, és magabiztosabban álljunk hozzá a jövőbeli feladatokhoz.

Összességében a projekt nagyban hozzájárult a programozási tudásunk elmélyítéséhez, és segített abban is, hogy jobban átlássuk a szoftverfejlesztési folyamat egészét. A jövőben szeretnénk továbbfejleszteni a projektet, és még több funkciót hozzáadni, valamint jobban optimalizálni a kódot.

Források

[Deepseek](https://www.deepseek.com/): A Python alapú mesterséges intelligencia integráció során használtuk, különösen az AI válaszainak konfigurálásakor. Ez az eszköz segítette az AI viselkedésének finomhangolását és a specifikus válaszstruktúra kialakítását.

[ChatGPT 4.0](https://chat.openai.com/): A kezdőoldalon megjelenő nyitóképet ezzel a generatív AI eszközzel készítettük el, kihasználva annak vizuális képgeneráló képességeit.

[W3Schools](https://www.w3schools.com/): A projekt során számos technikai dokumentációt és példát merítettünk ebből az oktatóanyag-gyűjteményből, különösen a HTML, CSS, JavaScript és React szekciók hasznosították.

[Stack Overflow](https://stackoverflow.com/): A fejlesztési folyamat során felmerült technikai hibák és problémák megoldásában nyújtott nagy segítséget. A közösségi válaszok alapján több hibát is sikeresen elhárítottunk.

[Hugging Face](https://huggingface.co/): A mesterséges intelligenciához szükséges modelleket, köztük a DistilGPT2 modellt is erről a platformról szereztük be. A modell integrálása lehetővé tette az intelligens válaszok generálását a backend oldalon.

[Meshy AI](https://www.meshy.ai/): A 3D-s modellek generálásához használtuk. Ez az eszköz biztosította a szükséges vizuális elemeket, amelyek gazdagították az alkalmazás megjelenését.

[Logo.com](https://logo.com/): A projekt hivatalos logóját ezen a felületen terveztük meg, kihasználva az egyszerű és intuitív logókészítő lehetőségeket.

[FreePik](https://www.freepik.com/): Néhány vizuális elem és kép, amelyet a projektben felhasználtunk, erről a weboldalról származik. A képek hozzájárultak a felhasználói felület vizuális élményének növeléséhez.

[Figma](https://www.figma.com/): Az oldal kezdeti dizájnjainak tervezése ebben az online UI/UX tervező eszközben történt. A közös munkafelület lehetővé tette a csapat tagjai számára a gyors és hatékony vizuális együttműködést.

[CodeViz](https://www.codeviz.io/): A projekt struktúrájának és moduljainak vizualizálásához ezt az eszközt használtuk, amely grafikus ábrát készített a kódbázisunkról.

[Material UI (MUI)](https://mui.com/): A front-end fejlesztés során ebből a komponenskönyvtárból importáltuk azokat az elemeket, amelyek a modern, reszponzív felhasználói felületet biztosították.

[ReChart](https://recharts.org/): Az adatok vizualizálására szolgáló diagramokat ezzel az eszközzel készítettük el. A projekt során például statisztikák, ranglisták és edzési adatok megjelenítésére használtuk.