

StudentHive Dokumentáció

A diákmunka jövője 

Tartalomjegyzék

Bevezetés	5
Honnan jött az ötlet?	5
A projekt bemutatása	5
Az oldal funkciói	5
Fejlesztői dokumentáció	6
A fejlesztői környezetről	6
Miért React a frontendhez?	7
Miért ASP.NET Core a backendhez?	8
Frontend architektúra	9
Általános frontend felépítés	9
Backend architektúra és API végpontok	12
Általános backend felépítés	12
API végpontok listája	13
Példák API-hívásokra	16
Kialakított adatszerkezet bemutatása	18
Az adatbázis modell	18
Főbb jellemzők	19
Kapcsolatok és idegenkulcsok	20
Biztonsági szempontok	22
Program főbb algoritmusai	22
Autentikáció	22

Iskolaszövetkezet létrehozása	23
Műszak hozzáadása	24
Műszak törlése és email értesítés	25
Teszt dokumentáció.....	26
Tesztkörnyezet	26
Funkcionális Tesztelés	27
Hibakezelési teszt	31
Tesztfelhasználók	32
Továbbfejlesztési lehetőségek.....	32
Felhasználói dokumentáció.....	34
A program célja és funkciói.....	34
A program célja	34
A rendszer funkciói.....	34
Be nem jelentkezett felhasználók lehetőségei.....	35
Technikai háttér.....	35
Biztonsági megoldások	35
Felhasználói élmény és támogatás.....	35
Jövőbeli fejlesztési tervek.....	35
Szükséges hardver- és szoftverkövetelmények	36
Felhasználói követelmények (webalkalmazás használatához)	36
Fejlesztői és üzemeltetői követelmények (backend, adatbázis).....	36
Telepítési és indítási útmutató	38
Alapvető szoftverek telepítése.....	38
A projekt letöltése	41
Frontend megnyitása	41
Frontend futtatása	42

Adatbázis indítása és tesztelése	42
A backend projekt megnyitása	43
Backend futtatása.....	43
A program használata	45
Bejelentkezés nélkül elérhető funkciók	45
Bejelentkezés diák profillal.....	50
Bejelentkezés közvetítői profillal	59
Bejelentkezés iskolaszövetkezet profillal	69
Bejelentkezés adminisztrátori profillal.....	79
Javasolt tesztelési sorrend	84
Asztali alkalmazás fejlesztői dokumentáció	85
A fejlesztői környezetről	85
Miért C# WPF az asztali alkalmazáshoz?	85
WPF architektúra	87
Általános felépítés	87
Program főbb algoritmusai	87
Regisztrációs algoritmus.....	87
Bejelentkezési algoritmus	88
Felhasználó státuszának változtatása	88
Felhasználói adatok módosítása	89
Program fájlok ismertetése	90
Models mappa.....	90
User osztály	90
Services mappa.....	91
ApiService osztály.....	91
Styles mappa	93

MainWindow	94
LoginPage	97
RegisterPage	101
UserModifyPage	105
Teszt dokumentáció	108
Teszt dokumentáció - ApiService Tesztek	108
Teszt felépítése	109
Tesztelési scenáriók és tesztek	109
Továbbfejlesztési lehetőségek	115
Többfelhasználós támogatás	Hiba! A könyvjelző nem létezik.
Dinamikus jogosultságkezelés	Hiba! A könyvjelző nem létezik.
Jobb keresési és szűrési lehetőségek	115
Beépített értesítési rendszer	115
Fejlettebb statisztikai és analitikai eszközök	115
Automatizált munkaszerződés-generálás	116
Mobilalkalmazás fejlesztése	116
Felhasználói profilok és személyre szabás	116
Szinkronizálás más rendszerekkel	116
Jobb hibaüzenetek és felhasználói visszajelzések	117
Funkcionális tesztelés	117
Regisztráció tesztelése	117
Bejelentkezés tesztelése	118
Adatok módosításának tesztelése	118

Bevezetés

Honnan jött az ötlet?

Az ötlet éppen munka közben született meg, mellyel saját magunk, illetve rengeteg diáktársunk életét könnyíthetnénk meg a diákmunkaszerzés területén. Az egész koncepciót arra építettük fel, hogy mi diákok minek tudnánk a legnagyobb hasznát venni, illetve mivel lehetne forradalmasítani ezt az egyre növekvő piacot.

A projekt bemutatása

Egy olyan platformot szerettünk volna létrehozni, amely összeköti az ország valamennyi iskolaszövetkezetét a dolgozni vágyó diákokkal. A tanuló a regisztrációit követően láthatja az összes csatlakozott munkaadó álláshirdetéseit, jelentkezhet azokra és vállalhat műszakokat, mindezt egy helyen. A rendszer különböző megkötéseket tesz a szövetkezetek számára az állások közzétételénél, ezzel is növelve az átláthatóságot. Nem egy egyszerű álláskereső portált akartunk, hanem annál jóval többet. Miután felvételt nyert a diák az adott munkára hozzáfér egy foglalórendszerhez, ahol kedvére válogathat az elérhető műszakok között. Mindezek mellett a diákok értékelhetik a meghirdetett munkákat ezzel is növelve a tökéletes állás megtalálását.

Az oldal funkciói

Négy szerepkört különböztetünk meg, illetve, ha a felhasználó nincs bejelentkezve, ilyenkor tud böngészni a munkák között, megtekinteni azok értékeléseit. Szerepköreink a következők: diák, közvetítő, iskolaszövetkezet, rendszergazda. A diák tud jelentkezni munkákra, műszakokra, adott esetben (12 órás határon kívül) visszamondani azokat. Ezzel egy átlátható környezet teremtünk egyszerűsítve a beosztások kezelését. A közvetítőt a szövetkezet jelöli egy munkára ezt követően a beérkező állásjelentkezéseket tudja kezelni, hozzáadni új műszakokat, illetve elfogadni, elutasítani a rá érkező jelentkezéseket. Szövetkezetként új munkákat lehet hozzáadni, közvetítőket felvenni, kezelni azokat. Rendszergazda fő feladat a szövetkezetek felvétele, asztali alkalmazásunkban a felhasználók kezelése.

Fejlesztői dokumentáció

A fejlesztői környezetről

A fejlesztői környezet több technológiát ötvöz annak érdekében, hogy hatékony, modern és skálázható alkalmazásokat lehessen létrehozni mind webes, mind asztali környezetben.

A **weboldal** fejlesztése **React** és **Vite v6.0.6** alapokon történik, biztosítva a gyors fejlesztési folyamatokat és optimalizált teljesítményt. A frontend több népszerű csomagot használ, például az **ag-grid** táblázatokhoz, az **axios** API-hívásokhoz, a **react-router-dom** az útvonalkezeléshez, és a **react-toastify** értesítésekhez. Az alkalmazás a **TypeScript** segítségével típusbiztos és könnyen karbantartható kódbázist biztosít.

A **backend** ASP.NET alapú, a **.NET 6** verziót használva. Az alkalmazás **JWT-alapú hitelesítést** valósít meg az **AspNetCore.Authentication.JwtBearer** csomaggal, az adatok tárolásához **MySQL** adatbázist használ, amelyhez az **EntityFrameworkCore.Tools** és a **MySql.Data** csomagok biztosítanak ORM támogatást. A dokumentáció generálására a **Swashbuckle.AspNetCore** (Swagger) kerül alkalmazásra.

A fejlesztői eszközök között a **Visual Studio 2022** és a **Visual Studio Code** biztosítják az optimális munkakörnyezetet. Az alkalmazásstruktúra és a csomagválasztás célja egy jól optimalizált, biztonságos és könnyen bővíthető rendszer kialakítása mind webes, mind asztali környezetben.

A fejlesztési folyamat során **Jira** és **GitHub** eszközök segítik a projektmenedzsmentet és a verziókezelést. A **Jira** feladata a feladatok nyomon követése, sprinttervezés és a fejlesztési folyamat strukturált menedzselése. A backlog kezelésére, a feature fejlesztések nyomon követésére és a hibajavítások priorizálására használjuk, biztosítva ezzel az átlátható és hatékony munkavégzést.

A **GitHub** a verziókezelést biztosítja, lehetővé téve a csapat számára a hatékony együttműködést, a kódváltozások követését és a különböző fejlesztési ágak kezelését.

Miért React a frontendhez?

Komponens alapú architektúra

A React lehetővé teszi az újrahasználató UI-komponensek kialakítását, amelyek csökkentik a kódbázis redundanciáját és megkönnyítik a karbantartást. Ezáltal a felhasználói felület könnyen bővíthető és testre szabható.

Gyors és responszív felület

A React a Virtual DOM segítségével optimalizálja a felhasználói felület frissítését, így az alkalmazás gördülékenyen és hatékonyan kezeli a dinamikus tartalmat. Ez különösen fontos egy olyan platformon, ahol a felhasználók folyamatosan böngészik az álláshirdetéseket és interakcióba lépnek az elemekkel.

TypeScript támogatás

A TypeScript használata növeli a típusbiztonságot, csökkenti a hibalehetőségeket és segít a fejlesztőknek jobban átlátni a kódbázist. Ez hosszú távon stabilabb és megbízhatóbb fejlesztési környezetet biztosít.

Gazdag ökoszisztéma

A React széleskörű könyvtár- és eszköztámogatása révén könnyen integrálható külső komponensekkel (pl. React Router, Recharts, React-Toastify, Axios, AG-Grid), amelyek segítenek az alkalmazás gyors fejlesztésében és skálázhatóságában.

Gyors fejlesztés Vite segítségével

A Vite fejlesztői környezetként való használata rendkívül gyors hot reloadot és fejlesztési élményt biztosít, amely különösen hasznos nagyobb alkalmazások esetén.

Miért ASP.NET Core a backendhez?

Magas teljesítmény és skálázhatóság

Az ASP.NET Core kiváló teljesítményt nyújt az aszinkronous I/O és a lightweight, modularis architektúrája révén. Az optimalizált Kestrel webkiszolgáló lehetővé teszi, hogy az alkalmazás magas terhelés mellett is gyorsan és megbízhatóan működjön.

Többrétegű architektúra és tiszta kódszerkezet

Az ASP.NET Core támogatja az MVC (Model-View-Controller) architektúrát, ami segíti a kód szervezését és könnyebbé teszi az üzleti logika, az adatkezelés és a megjelenítés szétválasztását.

Beépített autentikáció és jogosultságkezelés

A rendszer JWT (JSON Web Token) alapú hitelesítést használ, amely biztonságos és skálázható megoldást kínál a felhasználói jogosultságok kezelésére. Az ASP.NET Core beépített Identity és Authentication Middleware támogatása egyszerűsíti az autentikáció és az engedélyezési folyamatokat.

Kiváló adatbázis támogatás és ORM

Az Entity Framework Core segítségével hatékonyan lehet kezelni az adatbázist, miközben lehetőség van MySQL, PostgreSQL vagy akár Microsoft SQL Server használatára is. Az ORM (Object-Relational Mapping) támogatás megkönnyíti a fejlesztést és csökkenti az SQL hibák esélyét.

REST API és könnyű integráció

Az ASP.NET Core Web API támogatásával RESTful szolgáltatásokat lehet létrehozni, amelyek gyorsan és biztonságosan kommunikálnak a frontenddel. A Swagger integráció lehetővé teszi az API dokumentáció automatikus generálását, megkönnyítve ezzel a fejlesztői munkát.

Frontend architektúra

Általános frontend felépítés

A frontend egy **React + TypeScript** alapú webalkalmazás, amely az **ASP.NET Core** backend API-val kommunikál.

Főbb technológiák

- React → Modern komponensalapú fejlesztés
- TypeScript → Típusbiztos fejlesztés
- Axios → Backend kommunikáció
- JWT alapú hitelesítés → Biztonságos bejelentkezés

Csomagok

- AOS → Görgetésre aktiválódó animációk
- ag-grid → Táblázatok generálásért felelős csomag
- jwt-decode → JWT token dekódolására szolgál
- react-calendar → Műszakkezelésre használt naptár kiegészítő
- react-confirm-alert → Megerősítő ablak (pl. törlés)
- react-router-dom → Navigációhoz szükséges csomag
- react-toastify → Információs üzenetek megjelenítése (pl. hibaüzenetek)
- recharts → Kezelőfelületen statisztikák megjelenítésére használjuk

Fő mappák és szerepük

- **categories/** → A különböző kategóriákhoz kapcsolódó képeket tartalmazza.
- **public/** → A nyilvánosan elérhető fájlokat (képek, ikonok, logók) tartalmazza.
- **src/** → A teljes alkalmazás fő forráskódját tartalmazza.

src/ alatti fő mappák

- **assets/** → Az alkalmazásban használt képek, ikonok tárolására szolgál.
- **components/** → Újrafelhasználható React komponenseket tartalmaz.

Komponensmappák:

- **Navbar/** → Navigációs sáv.
- **Sidebar/** → Oldalsó navigációs sáv a kezelőfelületekhez.

- **Modals/** → Felugró ablakok alapja.
 - **ProtectedRoute/** → A szerepkör szerinti oldalelérést biztosítja.
 - **WorkPage/** → Konkrét munka adatait tölti be.
 - **AgentCard/** → Közvetítő munkáit tölti be kártyaként.
 - **ApplicationCard/** → Jelentkezések megjelenítése kártyaként.
 - **DashboardTitle/** → Kezelőpultokon az oldal címének megjelenítéséért felelős.
 - **Dialog/** → Felugró ablakok vázát adja.
 - **Footer/** → Az oldal láblécéért felelős.
 - **Modals/** → Felugró ablakok tartalmát szolgáltatja.
 - **ScrollToTop/** → Oldalváltáskor automatikusan az oldal tetejére navigál.
 - **ShiftApplications/** → Diákoknak a leadott műszakjelentkezéseket jeleníti meg kártyaként.
 - **ShiftCard/** → Létrehozott műszakokat jeleníti meg kártyaként.
 - **ShiftPage/** → Egy adott munkának a műszak hozzáadásáért és elérhető műszakok kilistázásáért, illetve egyéb munkához tartozó információk megjelenítéséért felelős.
 - **Title/** → Fő- és alcím hozható vele létre.
 - **UserCard/** → Diákoknak az elérhető műszakok kilistázásáért felelős kártyaként.
 - **UserNavbar/** → Diák szerepkörrel belépve a kezelőfelület navigációs sávja.
 - **UserShiftCard/** → Leadott műszakjelentkezések kilistázása, törlése.
 - **UserShiftPage/** → Ezen az oldalon tekintheti meg a diák az adott napra milyen műszakok elérhetőek.
 - **WorkCard/** → Elérhető munkák megjelenítése kártyás formában.
 - **WorkPage/** → Egy adott munka oldala, minden releváns információval, illetve értékelési lehetőséggel.
- **features/** → Az egyes funkciókhoz kapcsolódó oldalak és nézetek tárolására szolgál.
- Funkció mappák:

- **AdminDashboard/** – Az adminisztrátor felületéhez tartozó komponensek és oldalak.
 - AdminDashboardView/ – Az admin főoldala, statisztikák megjelenítésére szolgál.
 - AdminSettings/ – Admin profil beállítások kezelése (e-mail, jelszó).
 - ExistingOrg/ – Létező szervezetek listázása és kezelése.
 - NewOrg/ – Új szervezet létrehozása különböző adatok megadásával.
- **AgentDashboard/** – A közvetítők felületéhez tartozó funkciók.
 - AgentJobs/ – Közvetítőhöz rendelt munkák kezelése.
 - AgentStudentList/ – Az közvetítőhöz tartozó diákok listázása.
 - StudentApplications/ – Állásokhoz beérkező jelentkezések elfogadására/elutasítására szolgál.
 - AgentSettings/ – Közvetítői profil beállításai (név, e-mail, jelszó).
- **OrganizationDashboard/** – Az iskolaszövetkezetek számára készült felület.
 - CurrentAgents/ – A szervezethez tartozó ügynökök listázása és kezelése.
 - CurrentJobs/ – A szervezet által meghirdetett munkák megjelenítése, módosítása, ügynökhöz rendelése.
 - OrganizationSettings/ – A szervezet beállításai (profil adatok, szövetkezeti adatok).
 - OrganizationDashboard/ – Köszöntő oldal, szövetkezethez tartozó statisztikák megjelenítése.
 - NewJob/ – Alapadatok megadásával új munka hozható létre.
 - AddAgents/ – Név, e-mail megadásával új közvetítőt vehet fel a szövetkezet.
- **UserDashboard/** – A felhasználókhöz kapcsolódó funkciók.
 - UserApplications/ – A felhasználó által leadott jelentkezések listázása és visszavonása.
 - UserManageShifts/ – Felhasználóhoz rendelt műszakok kezelése.

- UserJobs/ – A diákhoz tartozó munkák megtekintése.
 - UserDashboardView/ – Köszöntő navigációs felület.
 - UserProfile/ – A felhasználói profil megtekintése és szerkesztése.
- **Contact/** – Kapcsolatfelvételi űrlap, StudentHive-val kapcsolatos információkkal.
- **Index/** – Szekciókból álló főoldal, a felhasználó ezzel találkozik az oldal meglátogatásakor.
- **Login/** – Bejelentkező felület.
- **Register/** – Regisztrációs felület.
- **Works/** – Munkák megjelenítéséért felelős oldal szűrési opciókkal ellátva.
- **OrganizationRequest/** – Információkat tartalmazó oldal egy új szervezet felvételével kapcsolatban.
- **pages/** → Speciális oldalak
 - **NotFound/** – 404-es hibaoldal.
 - **Unauthorized/** – Jogosulatlan hozzáférés esetén megjelenő oldal.
- **utils/** → Különböző segédfüggvényeket és konfigurációs fájlokat tartalmaz.
 - **authUtils.ts** – Authentikációval kapcsolatos segédfüggvények.
 - **routes.ts** – Navigációs szükséges oldal elérési útvonalakat tartalmazó fájl.
- **App.tsx** → A teljes oldal megjelenítésért felel, illetve az elérési útvonalak oldalon történő implementálását végzi.

Backend architektúra és API végpontok

Általános backend felépítés

A backend egy **ASP.NET Core** alapú **REST API**, amely **MySQL** adatbázist használ az adatok tárolására és kezelésére.

Az API több rétegből épül fel

- **Controllers:** A vezérlők felelősek az API végpontok kezeléséért és a kérések feldolgozásáért.
- **DatabaseHelper:** Egy segédosztály, amely az adatbázis-műveletek végrehajtásáért felelős.

- **Schema.sql:** Első indításkor egy üres induló adatbázisért létrehozásáért felelős.
- **Hitelesítés és jogosultságkezelés:** Az API **JWT tokeneket** használ a felhasználók hitelesítésére és jogosultságok ellenőrzésére.
- **Felhasználói szerepkörök:** A rendszerben négy szerepkör található (Admin, Organization, Agent, User), amelyek különböző hozzáférési jogosultságokkal rendelkeznek.

A backend struktúrája

- **AuthController:** Felelős a regisztrációért, bejelentkezésért, és token generálásért.
- **UserController:** A felhasználói adatok és jelentkezések kezelését végzi.
- **AgentController:** A közvetítők által kezelt munkák, műszakok és jelentkezések kezeléséért felel.
- **OrganizationController:** Az iskolaszövetkezetek általános műveleteit kezeli, ezek a munka létrehozás, módosítás, illetve a közvetítők felvétele és azok munkához rendelése.
- **AdminController:** Rendszergazdai feladat elvégzésére szolgál, főként a szövetkezetek létrehozására szolgál.
- **GeneralController:** Általános folyamatokat tartalmaz, nem feltétlenül szerepkörhöz kötött.

Általános jellemzők

- Token alapú hitelesítés (JWT)
- Felhasználói azonosítás és jogosultságellenőrzés
- Adatbázis-lekérdezések paraméterezett SQL utasításokkal
- Hibakezelés és megfelelő státuszkódok visszaadása

API végpontok listája

Admin végpontok (/api/admin)

- **GET /users-by-month** – Visszaadja a regisztrált felhasználók számát havonta.
- **GET /total-organizations-and-users** – Lekéri az összes regisztrált szervezet és felhasználó számát.

- **GET /organizations** – Visszaadja az összes szervezet listáját.
- **PUT/organization/{organizationId}/password** – Egy szervezet adminisztrátorának jelszavát módosítja.
- **POST /new-organization** – Új szervezetet hoz létre.
- **PUT /settings** – Adminisztrációs beállításokat frissít.
- **PUT /organization/{organizationId}** – Frissíti egy szervezet adatait.
- **GET /admin-details** – Lekéri az adminisztrátor részleteit.

Agent végpontok (/api/agent)

- **GET /work-titles** – Visszaadja az elérhető munkák címét.
- **GET /agent-work-cards** – Lekéri az ügynökhöz rendelt munkák listáját.
- **GET /work-details/{id}** – Lekéri egy adott munka részletes adatait.
- **GET /manage-shifts/{jobId}** – Lekéri az adott munkához tartozó műszakokat.
- **GET /job-title/{jobId}** – Lekéri egy adott munka címét.
- **GET /shift-starts/{jobId}** – Lekéri egy adott munkához tartozó műszakkezdéseket.
- **GET /shift-applications/{jobId}** – Lekéri a munkára érkezett jelentkezéseket.
- **PATCH /shift-applications/{id}/accept** – Elfogad egy műszakra való jelentkezést.
- **PATCH /shift-applications/{id}/decline** – Elutasít egy műszakra való jelentkezést.
- **GET /applications** – Visszaadja az összes jelentkezést.
- **PATCH /applications/{id}/accept** – Elfogad egy jelentkezést.
- **GET /student-list** – Lekéri az ügynökhöz tartozó diákokat.
- **POST /add-shift** – Új műszakot ad hozzá egy munkához.
- **DELETE /delete-shift/{id}** – Töröl egy műszakot.
- **PATCH /applications/{id}/decline** – Elutasít egy jelentkezést.
- **PUT /profilesettings** – Frissíti a közvetítő profilbeállításait.

Auth végpontok (/api/auth)

- **POST /register** – Regisztrál egy új felhasználót.
- **POST /register-admin** – Regisztrál egy új adminisztrátort (csak hitelesített kliens használhatja).
- **POST /login** – Bejelentkezteti a felhasználót és visszaad egy JWT tokenet.

- **POST /logout** – Kijelentkezteti a felhasználót.
-

General végpontok (/api/general)

- **GET /workcards** – Visszaadja az elérhető munkák listáját.
 - **GET /workcards/{id}** – Lekéri egy adott munka részletes adatait.
 - **GET /cities** – Lekéri az elérhető városokat.
 - **GET /alluser** – Visszaadja az összes felhasználót.
 - **PATCH /update-user-status/{id}** – Frissíti egy felhasználó státuszát.
 - **PATCH /update-user-password/{id}** – Frissíti egy felhasználó jelszavát.
 - **PATCH /update-user-profile/{id}** – Frissíti egy felhasználó profilját.
 - **GET /jobreviews/{jobId}** – Lekéri az adott munkához tartozó értékeléseket.
 - **POST /jobreviews** – Új értékelést ad hozzá egy munkához.
-

Organization végpontok (/api/organization)

- **GET /total-students-and-jobs/{orgId}** – Lekéri az adott szervezethez tartozó diákok és munkák számát.
- **GET /jobs-created-by-month/{orgId}** – Visszaadja az adott szervezet által havonta létrehozott munkákat.
- **POST /new-agent** – Új ügynököt hoz létre.
- **POST /new-job** – Új munkát hoz létre.
- **GET /jobs** – Visszaadja a szervezet által létrehozott munkák listáját.
- **GET /job/{id}** – Lekéri egy adott munka részleteit.
- **GET /categories** – Visszaadja az elérhető munkakategóriákat.
- **PATCH /assign-agent/{agentId}/{JobId}** – Hozzárendel egy ügynököt egy munkához.
- **GET /agents** – Lekéri a szervezethez tartozó közvetítők listáját.
- **PATCH /toggle-job-status/{jobId}** – Módosítja egy munka állapotát (aktív/inaktív).
- **PATCH /toggle-agent-status/{id}** – Módosítja egy ügynök állapotát (aktív/inaktív).
- **PUT /update-job/{jobId}** – Frissíti egy adott munka adatait.
- **PUT /orgsettings** – Frissíti a szervezet beállításait.
- **GET /organization-details** – Lekéri a szervezet részletes adatait.

User végpontok (/api/user)

- **GET /user-jobs** – Lekéri a felhasználóhoz rendelt munkákat.
- **GET /user-applications** – Lekéri a felhasználó munkajelentkezéseit.
- **DELETE /delete-application/{applicationId}** – Töröl egy jelentkezést.
- **GET /profile** – Lekéri a felhasználó profilját.
- **POST /apply** – Jelentkezik egy munkára.
- **GET /list-user-shifts/date/{date}** – Lekéri a felhasználó műszakjait egy adott napon.
- **DELETE /delete-shift/{shiftId}** – Töröl egy műszakot.
- **GET /list-shifts/{jobId}/date/{date}** – Lekéri az adott munkához tartozó műszakokat egy adott napon.
- **POST /apply-shift** – Jelentkezik egy műszakra.
- **GET /student-details-datas** – Lekéri a felhasználó személyes adatait.
- **PUT /student-details** – Frissíti a felhasználó személyes adatait.

Példák API-hívásokra

PUT /api/admin/organization/{organizationId}/password

- **Leírás:** Egy szervezet adminisztrátorának jelszavát módosítja.
- **Bemenet:** PasswordUpdateRequest

```
{
  "newPassword": "NewStrongPass123"
}
```
- **Válaszok:**
 - **Sikeres jelszóváltoztatás (200 OK)**

```
{"message": "Jelszó sikeresen módosítva!"}
```
 - **Felhasználó nincs hitelesítve (401 Unauthorized)**


```
{ "message": "Felhasználó azonosítása sikertelen." }
```

- **Érvénytelen bemenet - üres jelszó (400 Bad Request)**

```
{ "message": "A jelszó nem lehet üres!" }
```

- **Érvénytelen bemenet - nem megfelelő jelszó (400 Bad Request)**

```
{ "message": "Legalább egy mezőnek (email vagy jelszó) meg kell változnia." }
```

- **Nem létező szervezet (404 Not Found)**

```
{ "message": "Szervezet nem található!" }
```

- **Adatbázis hiba (500 Internal Server Error)**

```
{ "message": "Hiba történt a jelszó módosítása közben.", "details": "SQL error details here..." }
```

POST /api/agent/add-shift

- **Leírás:** Új műszakot ad hozzá egy munkához.

- **Bemenet:** ShiftRequest

```
{
  "jobId": 12,
  "shiftStart": "2024-04-15T08:00:00",
  "shiftEnd": "2024-04-15T16:00:00"
}
```

- **Válaszok:**

- **Sikeres műszak hozzáadás (200 OK)**

```
{ "message": "Műszak sikeresen hozzáadva!" }
```

- **Felhasználó nincs hitelesítve (401 Unauthorized)**

```
{ "message": "Felhasználó azonosítása sikertelen." }
```

- **Érvénytelen bemenet (400 Bad Request)**

```
{ "message": "Érvénytelen adatok!" }
```

- **A műszak időpontja a múltban van (400 Bad Request)**

```
{ "message": "A műszak kezdő- vagy befejező időpontja nem lehet a jelenlegi dátum előtt!" }
```

- **A műszak túl későn kezdődik (400 Bad Request)**

```
{ "message": "A műszak kezdő időpontja nem lehet több, mint egy héttel a jelenlegi dátum után!" }
```

- **A műszak vége korábban van, mint a kezdete (400 Bad Request)**

```
{ "message": "A műszak befejező időpontja nem lehet korábbi, mint a kezdő időpont!" }
```

- **A műszak időtartama túl hosszú (400 Bad Request)**

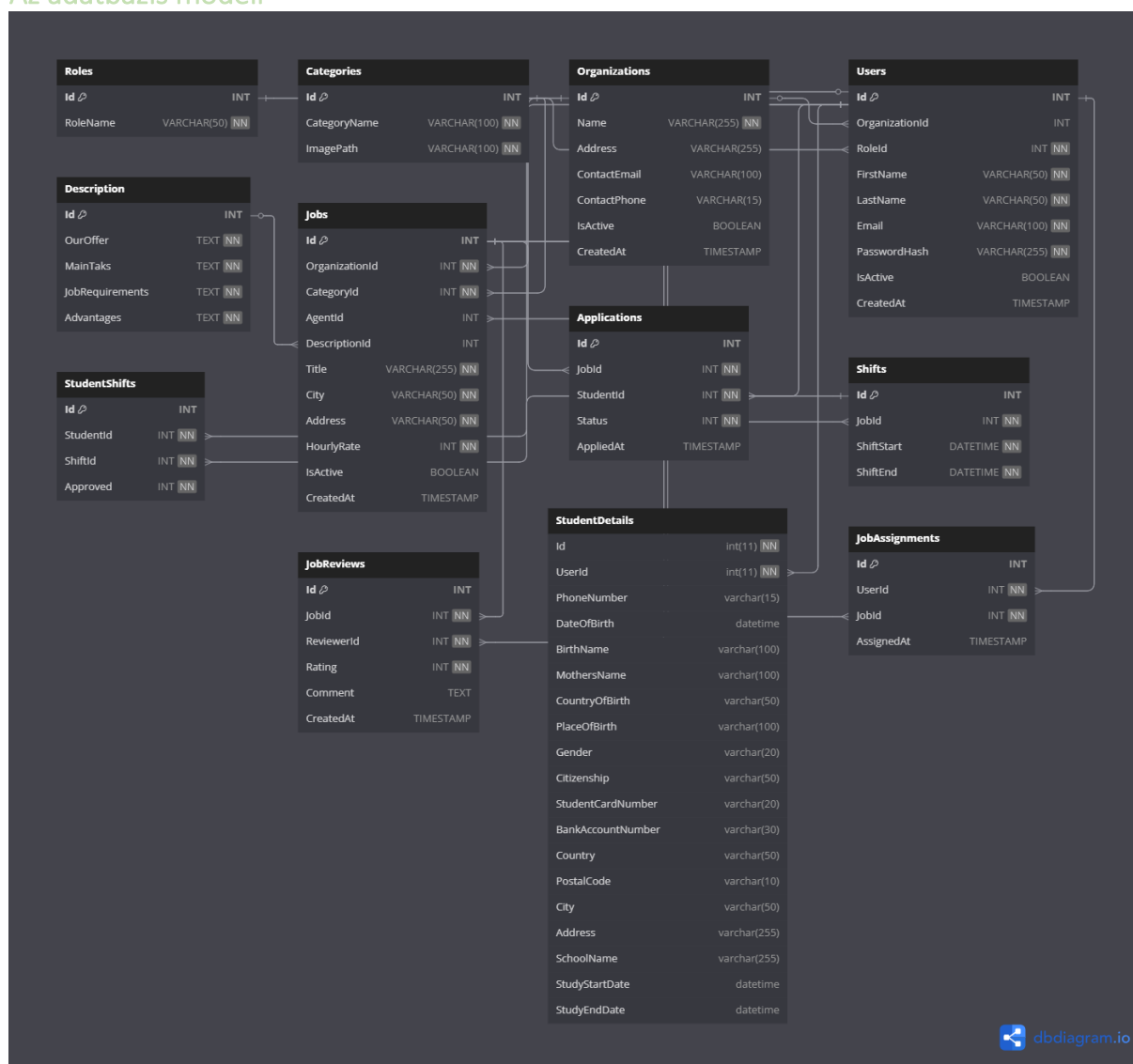
```
{ "message": "A műszak hossza nem haladhatja meg a 24 órát!" }
```

- **Adatbázis hiba (500 Internal Server Error)**

```
{ "message": "Hiba a műszak hozzáadása során!", "details": "SQL error details here..." }
```

Kialakított adatszerkezet bemutatása

Az adatbázis modell



Főbb jellemzők

Felhasználói szerepkörök

A felhasználók különböző szerepkörökben működhetnek a rendszerben, amelyet a **Roles** tábla határoz meg.

- **Admin** (Adminisztrátor) – Teljes körű jogosultságokkal rendelkezik, menedzselheti a rendszert.
- **Organization** (Munkáltató, Iskolaszövetkezet) – Állásokat hirdethet és diákokat alkalmazhat.
- **Agent** (Közvetítő) – Az iskolaszövetkezet nevében koordinálja a jelentkezéseket és a diákokat.
- **User** (Diák) – Állásokra jelentkezhet, munkákat vállalhat.

A **Users** tábla tárolja a felhasználók személyes adatait, e-mail címüket, jelszóhash-üket és szervezetük azonosítóját (ha van ilyen).

Szervezetek és állások kezelése

A **Organizations** tábla tartalmazza a munkáltató szervezeteket, amelyek állásokat hirdethetnek meg. Az **Jobs** tábla az állásokat tárolja, kategóriákba sorolva (CategoryId), és szervezetekhez kapcsolódva (OrganizationId). Az AgentId mező egy adott User-re hivatkozik, aki az állásért felelős.

Kategóriák és leírások

Az **Categories** tábla az állások kategorizálására szolgál. Az egyes állások mindig egy adott kategóriába sorolhatók, például:

- Informatikai, mérnöki munkák
- Vendéglátás, gyorsétterem
- Áruházi, bolti, eladói

Az állások leírása külön táblában (**Description**) kerül tárolásra, amely tartalmazza:

- Feladatokat (MainTasks)
- Követelményeket (JobRequirements)

- Előnyöket (Advantages)
- Juttatások, mit ajánl a cég (OurOffer)

Jelentkezések és értékelések

A diákok jelentkezéseit az **Applications** tábla tárolja, amely rögzíti:

- A jelentkező diák azonosítóját (StudentId)
- A kiválasztott állás azonosítóját (JobId)
- A jelentkezés státuszát (Status)

A **JobReviews** tábla biztosítja, hogy a diákok értékelhessék a munkáltatókat:

- **Értékelés** (1-5 csillag)
- **Vélemény szövegesen**
- **Dátum**

Ez segíti a jövőbeli jelentkezőket a megfelelő munkahely kiválasztásában.

Munkarend és műszakok

A **Shifts** tábla az állásokhoz kapcsolódó műszakokat tárolja (ShiftStart, ShiftEnd).

A **StudentShifts** táblában tárolódik, hogy egy diák (StudentId) milyen műszakokat vállalt el.

Diákok adatai

A **StudentDetails** tábla a diákok részletes személyes adatait tartalmazza (pl. születési hely, anyja neve, bankszámlaszám, iskola).

Állásmegbízások

A **JobAssignments** tábla tartalmazza, hogy egy adott felhasználó (UserId) melyik álláshoz lett hozzárendelve.

Kapcsolatok és idegenkulcsok

Felhasználók és szerepkörök kapcsolata

A **Users** (felhasználók) tábla egy idegen kulccsal (RoleId) kapcsolódik a **Roles** (szerepkörök) táblához. Egy felhasználónak mindig van egy szerepe, de egy szerepkörhöz több felhasználó is tartozhat.

- Kapcsolat típusa: Egy-sok (1:N)
- Idegen kulcs: **Users.RoleId** → **Roles.Id**
- Megkötések: **Users.RoleId** értéke csak olyan Id lehet, amely létezik a **Roles** táblában.

Felhasználók és szervezetek kapcsolata

A **Users** tábla kapcsolódik az **Organizations** (szervezetek) táblához, mivel egy felhasználó egy szervezethez tartozhat.

- Kapcsolat típusa: Egy-sok (1:N)
- Idegen kulcs: **Users.OrganizationId** → **Organizations.Id**
- Megkötések: Egy felhasználónak lehet szervezete, de nem kötelező (NULL megengedett).

Szervezetek és álláshirdetések kapcsolata

A **Jobs** (állások) tábla egy szervezethez kapcsolódik, mivel egy szervezet több állást is közzétehet.

- Kapcsolat típusa: Egy-sok (1:N)
- Idegen kulcs: **Jobs.OrganizationId** → **Organizations.Id**
- Megkötések: Egy álláshirdetés mindig egy szervezethez tartozik.

Diákok és állások kapcsolata (jelentkezések)

A **Users** tábla és a **Jobs** tábla közötti kapcsolatot a **Applications** tábla valósítja meg, mivel egy diák több állásra is jelentkezhet, és egy állásra több diák is jelentkezhet.

- Kapcsolat típusa: Sok-sok (M:N) köztes táblával
- Idegen kulcsok: **Applications.StudentId** → **Users.Id**
Applications.JobId → **Jobs.Id**
- Megkötések: Egy jelentkezés csak létező diákkal és állással történhet.

Állások és állaskategóriák kapcsolata

A **Jobs** tábla egy kategóriához kapcsolódik a **Categories** táblán keresztül, amely az elérhető állaskategóriákat tartalmazza.

- Kapcsolat típusa: Egy-sok (1:N)
- Idegen kulcs: Jobs.CategoryId → **JobCategories.Id**
- Megkötések: Egy álláshirdetés mindig egy kategóriába tartozik.

Biztonsági szempontok

Jelszavak titkosítása

A felhasználók jelszavai bcrypt hash-eléssel kerülnek tárolásra.

Adatvédelem

Az érzékeny adatokat (pl. diák személyes adatai) külön táblában (StudentDetails) tároljuk.

Tranzakciókezelés

A jelentkezések és állások módosításai tranzakciókba foglalhatók, így elkerülhetők az inkonzisztens állapotok.

Program főbb algoritmusai

Autentikáció

Regisztrációs algoritmus

- Az alkalmazás ellenőrzi az e-mail cím formátumát (**IsValidEmail** függvény, regex validációval).
- Az alkalmazás biztosítja, hogy a jelszó megfeleljen a minimális biztonsági követelményeknek (**IsValidPassword** függvény).
- Az adatbázisban lekérdezi, hogy az adott e-mail cím már létezik-e.
- Ha az e-mail cím egyedi, a jelszó egy **bcrypt hash**-el titkosítva kerül tárolásra.
- Az új felhasználó bekerül az adatbázisba alapértelmezett szerepkörrel (**User – RoleId = 4**).
- A felhasználóhoz egy üres **StudentDetails** rekord is létrejön.

Bejelentkezési algoritmus

- Az alkalmazás lekéri az adott e-mail címhez tartozó felhasználói adatokat (ID, jelszó hash, szerepkör, aktív státusz).

- Ellenőrzi, hogy a fiók aktív-e.
- Az alkalmazás **bcrypt** segítségével összehasonlítja a megadott jelszót az adatbázisban tárolt hash értékkel.
- Sikeres bejelentkezés esetén az alkalmazás **JWT token** generál, amely a felhasználó azonosítóját és szerepkörét tartalmazza.

JWT token generálása

- A rendszer egy **JSON Web Token** (JWT) alapú hitelesítési mechanizmust használ.
- A **token** egyedi azonosítót tartalmaz (userId, role), és egy titkos kulccsal van aláírva (GenerateJwtToken függvény).
- Ha a felhasználó a "Maradjak bejelentkezve" opciót választja, akkor a token érvényessége **2 év**, egyébként **1 nap**.

Iskolaszövetkezet létrehozása

Adatok validálása

Az API először ellenőrzi a bejövő kérést (**NewOrganizationRequest** objektumot), hogy minden kötelező mező ki legyen töltve:

- **OrgName** – az iskolaszövetkezet neve
- **Email** – az adminisztrátor e-mail címe
- **PhoneNumber** – az iskolaszövetkezet telefonszáma
- **Address** – az iskolaszövetkezet címe

Ha bármelyik mező üres, a rendszer hibát dob és megszakítja a feldolgozást.

Jelszó generálása

Mivel az új iskolaszövetkezet adminisztrátora kezdetben nem állít be saját jelszót, a rendszer egy véletlenszerű jelszót generál a **GenerateRandomPassword** módszerrel.

Ez a módszer:

- Egy 10 karakter hosszú jelszót hoz létre, amely tartalmaz nagybetűket, kisbetűket és számokat.
- A jelszót bcrypt algoritmussal titkosítja, mielőtt adatbázisba kerülne.

E-mail küldése a felhasználónak

A rendszer automatikusan küld egy e-mailt az új adminisztrátornak a belépési adatokkal (**SendEmail** metódus).

Ez az e-mail tartalmazza:

- A regisztrált e-mail címet
- A generált jelszót
- Figyelmeztetést, hogy az első bejelentkezés után cserélje le a jelszót

Műszak hozzáadása

Felhasználó azonosítás

Először ellenőrzi, hogy a felhasználó be van-e jelentkezve, `ClaimTypes.NameIdentifier` használatával.

Bemeneti adatok validálása

Ellenőrzi, hogy a `JobId`, `ShiftStart` és `ShiftEnd` érvényes-e
Budapesti időzónára konvertálja az időpontokat

Validálja, hogy:

- A műszak nem kezdődhet a múltban
- A műszak nem kezdődhet egy hétnél később
- A befejezés nem lehet a kezdés előtt
- A műszak hossza nem haladhatja meg a 12 órát

Adatbázis művelet

SQL INSERT parancsot hajt végre a Shifts táblába

Paraméterezetten adja át az értékeket az SQL injekcióval szembeni védelem érdekében

Visszatérési érték kezelése

Siker esetén 200 OK státuszkóddal tér vissza

Hiba esetén 500-as hibakóddal és részletes hibaüzenettel

Műszak törlése és email értesítés

Műszak adatok lekérése

- Ellenőrzi, hogy a törlendő műszak létezik-e a **Shifts** táblában
- Lekéri a műszak részleteit (jobId, kezdési és befejezési idő)

Érintett diákok azonosítása

- Lekérdezi az összes diákot, aki jelentkezett a műszakra a **StudentShifts** táblából

Adatbázis műveletek

- Először törli a jelentkezéseket a **StudentShifts** táblából
- Ezután törli magát a műszakot a **Shifts** táblából

Email értesítések küldése

Minden érintett diáknak egyenként:

- Lekéri a diák elérhetőségét és nevét a **Users** táblából
- Személyre szabott üzenetszöveget generál, amely tartalmazza:
 - A diák teljes nevét
 - A törölt műszak időpontját (kezdés és befejezés)
 - Általános tájékoztatást és iránymutatást
- Külső **SMTP** szolgáltatást használ (Gmail) a levelek kiküldésére
- Az email küldés hibakezelését a metóduson belül kezeli le

Az email küldés algoritmus (SendEmail privát metódus)

- **SMTP** kliens létrehozása megfelelő konfigurációval (host, port, SSL, hitelesítés)
- **MailMessage** objektum összeállítása (feladó, címzett, tárgy, szövegtörzs)
- Az email elküldése és erőforrások felszabadítása
- Hibakezelés és naplózás kivételek esetén

Tesztdokumentáció

Tesztkörnyezet

Hardver

Az alkalmazás az alábbi konfigurációkon lett kipróbálva:

Hardver	Processzor	RAM	GPU
Asztali PC	AMD Ryzen 5 2600	16GB DDR4	RX580
Laptop	Intel I7 8750H	24GB DDR4	RTX 2070
Virtualizált környezet	4 vCPU	8GB RAM	Nincs dedikált GPU

Operációs rendszerek

A tesztelés során az alábbi operációs rendszereken futtattuk az alkalmazást:

Operációs rendszer	Verzió
Windows 11 Pro	24H2
Windows 10 Pro	22H2

Böngészők

Az alkalmazás **reszponzív működését** és kompatibilitását különböző böngészőkben teszteltük.

Böngésző	Verzió
Google Chrome	132.0.6834.214
Opera GX	117.0.5408.100

Funkcionális Tesztelés

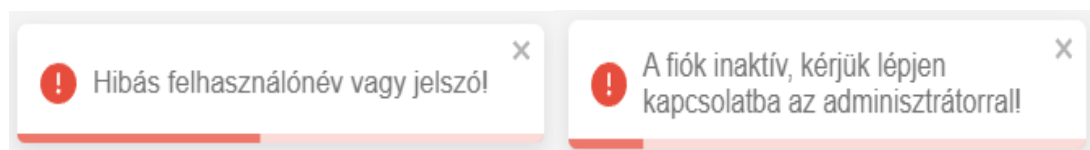
A tesztelés során a következő főbb funkciókat ellenőriztük.

Felhasználói hitelesítés

- **Teszt:** Bejelentkezés helyes és helytelen adatokkal
- **Elvárt eredmény:**
 - Helyes adatok esetén a rendszer bejelentkeztet
 - Helytelen adatoknál megfelelő hibaüzenetet ad vissza

Tesztadatok:

E-mail	Jelszó	Elvárt eredmény
user@example.com	Jelszó123	Sikeresen bejelentkezés
user@example.com	RosszJelszó	Hibás felhasználónév vagy jelszó!
user@example.com	Jelszó123	A fiók inaktív, kérjük lépjen kapcsolatba az adminisztrátorral.

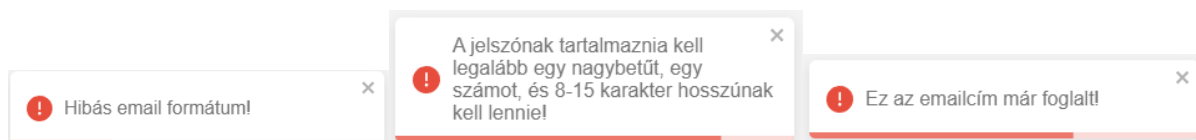


Új felhasználó regisztráció

- **Teszt:** Regisztráció különböző típusú adatokkal
- **Elvárt eredmény:**
 - Hibás formátum esetén megfelelő hibaüzenet
 - Sikeres regisztráció esetén belépési lehetőség

Tesztadatok:

Név	E-mail	Jelszó	Elvárt eredmény
Bánkuti László	helytelen@gmail	Jelszó123	Hibás email formátum!
Bánkuti László	user@example.com	Jelszó123	Ez az emailcím már foglalt!
Bánkuti László	user2@example.com	Jelszó	A jelszónak tartalmaznia kell legalább egy nagybetűt, egy számot, és 8-15 karakter hosszúnak kell lennie!
Bánkuti László	user2@example.com	Jelszó123	Sikeres regisztráció!



Munkák megtekintése és jelentkezés

Elvárt eredmény:

- A munka adatai helyesen jelennek meg
- Jelentkezés sikerességéről üzenet tájékoztasson

Tesztadatok:

Munka neve	Jelentkezés állapota	Elvárt eredmény
Raktári munka	Sikeres jelentkezés	Sikeres jelentkezés!
Pultos álláslehetőség	Már jelentkezett	Már jelentkezettél erre a munkára.

Új szövetkezet létrehozása

Elvárt eredmény:

- Helyes adatok esetén a szervezet létrejön és e-mailt kap
- Hibás adatok esetén hibaüzenet

Tesztadatok:

Teszt	Bemenet	Elvárt eredmény
Üres szervezetrév	""	A szövetkezet neve megadása kötelező!
Hosszú szervezetrév	"Nagyon hosszú szövetkezet neve, amely meghaladja az 50 karaktert"	A szövetkezet neve legfeljebb 50 karakter lehet!
Speciális karakterek a névben	"Cég#123"	A szövetkezet neve csak betűket és szóközöket tartalmazhat!
Érvénytelen e-mail	"invalid-email"	Érvénytelen email cím!
Üres telefonszám	""	Érvénytelen telefonszám! 10-15 számjegy kell, opcionális '+' előtaggal.

Rövid telefonszám	"12345"	Érvénytelen telefonszám! 10-15 számjegy kell, opcionális '+' előtaggal.
Üres cím	""	A cím megadása kötelező!
Hosszú cím	"Nagyon hosszú cím, amely meghaladja a 100 karakteres limitet és emiatt hibaüzenetet kell kapjunk"	A cím legfeljebb 100 karakter lehet!

▪ **Sikeres eset:**

OrgName	Email	PhoneNumber	Address	Elvárt eredmény
StudentHive	info@studenthive.com	+36123456789	Budapest, Fő utca 10.	Szövetkezet sikeresen hozzáadva!

Új munka létrehozása

▪ **Elvárt eredmény:**

- Helyes adatok esetén a szervezet létrejön és e-mailt kap
- Hibás adatok esetén hibaüzenet

▪ **Tesztadatok:**

Teszt	Bemenet	Elvárt eredmény
Üres munka neve	""	A munka neve megadása kötelező.
Hosszú munka neve	"Nagyon hosszú munka neve, amely meghaladja a 84 karakteres limitet, és emiatt hibaüzenetet kell kapjunk"	A munka neve nem lehet hosszabb 84 karakternél.
Speciális karakterek a munka nevében	"Cég#123"	A munka neve csak betűket és szóközöket tartalmazhat!
Érvénytelen órabér	"0"	Az órabér nem lehet 0 vagy negatív szám.
Hiányzó kategória	""	Kategória kiválasztása kötelező.
Üres városnév	""	A város megadása kötelező.

Speciális karakterek a városnévben	"Debrecen!@#"	A város neve csak betűket és szóközöket tartalmazhat!
Üres cím	""	A cím megadása kötelező!
Hosszú cím	"Nagyon hosszú cím, amely meghaladja a 100 karakteres limitet és emiatt hibaüzenetet kell kapjunk"	A cím legfeljebb 100 karakter lehet!
Üres leírásmezők	"" az OurOffer, MainTaks, JobRequirements, Advantages mezőknél	Minden mező kitöltése kötelező!

▪ **Sikeres eset:**

Title	HourlyRate	CategoryId	City	Address	OurOffer	MainTaks	JobRequirements	Advantages	Elvárt eredmény
Pincér	2000	3	Budapest	Kossuth tér 1.	Jó fizetés	Vendégek kiszolgálása	Megbízhatóság	Előny az angol nyelvtudás	A munka sikeresen létrehozva!

Műszak hozzáadása

▪ **Elvárt eredmény:**

- Helyes adatok esetén a műszak létrejön és megjelenik a listában
- Hibás adatok esetén hibaüzenet

▪ **Tesztadatok:**

Teszt	Bemenet	Elvárt eredmény
Üres műszakkezdés	shiftStart = ""	Érvénytelen adatok!
Üres műszakvége	shiftEnd = ""	Érvénytelen adatok!
Kezdő időpont a múltban	shiftStart = "2024-03-20T10:00", shiftEnd = "2024-03-20T18:00" (mai dátum: 2024-03-22)	A műszak kezdő- vagy befejező időpontja nem lehet a jelenlegi dátum előtt!
Kezdő időpont túl későn	shiftStart = "2024-04-10T10:00" (mai dátum: 2024-04-01)	A műszak kezdő időpontja nem lehet több, mint egy héttel a jelenlegi dátum után!

Műszak vége korábban van, mint a kezdete	shiftStart = "2024-04-05T14:00", shiftEnd = "2024-04-05T10:00"	A műszak befejező időpontja nem lehet korábbi, mint a kezdő időpont!
Műszak túl hosszú	shiftStart="2024-04-05T08:00", shiftEnd = "2024-04-06T08:30"	A műszak hossza nem haladhatja meg a 24 órát!

▪ Sikeres eset:

JobId	ShiftStart	ShiftEnd	Elvárt eredmény
12	"2024-04-05T08:00"	"2024-04-05T16:00"	Műszak sikeresen hozzáadva!

▪ Edge Case tesztek:

Teszt	Bemenet	Elvárt eredmény
Műszak kezdő időpontja 6 nappal előre	"shiftStart": "2024-04-10T08:00" (mai dátum: 2024-04-04)	Műszak sikeresen hozzáadva!
Műszak kezdő időpontja 8 nappal előre	"shiftStart": "2024-04-12T08:00" (mai dátum: 2024-04-04)	A műszak kezdő időpontja nem lehet több, mint egy héttel a jelenlegi dátum után!
Műszak 12 órás hosszú	"shiftStart": "2024-04-05T08:00", "shiftEnd": "2024-04-05T20:00"	Műszak sikeresen hozzáadva!
Műszak 24 óránál hosszabb	"shiftStart": "2024-04-05T08:00", "shiftEnd": "2024-04-06T10:00"	A műszak hossza nem haladhatja meg a 24 órát!

Hibakezelési teszt

A rendszer tesztelve lett különböző rossz adatokkal és hibás felhasználói interakciókkal.

Tesztforgatókönyv	Elvárt eredmény	Valós eredmény
Adatok nélkül küldött API hívás	400 Bad Request	Hibás adatok
Jogosultság nélkül API hívása	401 Unauthorized	Felhasználói azonosítás sikertelen

SQL Injection próbálkozás	Sikertelen támadás	Input validáció blokkolta
Jogosultság nélkül oldal elérése	401 Unauthorized	Átirányítás Unauthorized oldalra
Nem létező oldal elérése	404 Not Found	Átirányítás NotFound oldalra

Tesztfelhasználók

A teszteléshez létrehoztunk előre beállított tesztfelhasználókat. Minden funkció teszteléséhez szükséges további fiókokat létrehozni (email küldés valós e-mail cím megadásával [tempmail]).

Szerepkör	Felhasználónév	Jelszó
Admin	admin@ex.com	Asd12345
Organization MelóDiák	melodiak@ex.com	Asd12345
Organization Jóker	joker@ex.com	Asd12345
Agent MelóDiák	fogarasi@ex.com	Asd12345
Agent Jóker	toth@ex.com	Asd12345
User	user@ex.com	Asd12345
User	user2@ex.com	Asd12345

Továbbfejlesztési lehetőségek

Jogosultságkezelés és szerepkörök bővítése

Jelenleg az adminisztrátorok manuálisan hozzák létre az iskolaszövetkezeteket. Bevezethető lenne egy önregisztrációs folyamat az iskolaszövetkezetek számára, amelyet az admin jóváhagyhat.

A jelenlegi szerepkörök mellé új jogosultsági szintek is definiálhatók, például moderátorok vagy részleges adminisztrátorok, akik bizonyos műveleteket elvégezhetnek.

Fejlettebb keresési és szűrési lehetőségek

Az állások keresése jelenleg alapvető paraméterek szerint történik. Bővíteni lehetne többdimenziós szűrési lehetőségekkel, például órabérsáv, elérhetőség vagy munkáltatói értékelések alapján.

Értesítési rendszer bevezetése

Jelenleg a felhasználók e-mailben kapnak visszajelzéseket bizonyos eseményekről (pl. regisztráció, jelszómódosítás).

Hasznos lenne egy push értesítési rendszer vagy beépített üzenetközpont, amely valós időben tájékoztatná a felhasználókat az álláslehetőségekről, elfogadott jelentkezésekről vagy új üzenetekről.

Automatizált munkaszerződés-generálás

Jelenleg a diákok jelentkezés után adminisztratív módon kerülnek be az iskolaszövetkezetek adatbázisába.

Egy automatikus munkaszerződés-generáló rendszer bevezetése lehetővé tenné a digitális szerződéskötést a platformon belül, amely jogilag hiteles dokumentumokat generál PDF formátumban.

Fejlettebb statisztikai modul

Az adminisztrátorok számára egy részletesebb analitika beépítése hasznos lenne, például a munkák sikerességi arányáról, a diákok aktivitásáról vagy az egyes állásajánlatok népszerűségéről.

Interaktív grafikonok segítségével az adminisztrátorok jobban átláthatnák a rendszer működését.

Mobilalkalmazás fejlesztése

A jelenlegi rendszer webalkalmazásként működik, de egy natív mobilalkalmazás (Android/iOS) fejlesztése jelentősen növelné a felhasználói élményt.

A mobilalkalmazás push értesítésekkel, gyors regisztrációval és térképes keresővel egészíthetné ki a jelenlegi funkciókat.

Összegzés

A rendszer számos módon továbbfejleszthető, beleértve az automatizálást, az élő értesítéseket, a statisztikai elemzéseket és a mobilalkalmazás integrálását. A jövőbeli fejlesztések célja, hogy a platform még felhasználóbarátabb, hatékonyabb és skálázhatóbb legyen a diákmunka piacán.

Felhasználói dokumentáció

A program célja és funkciói

A program célja

A fejlesztett platform célja, hogy megkönnyítse a diákok számára a munkavállalást, miközben biztosítja az iskolaszövetkezetek és munkáltatók számára az átlátható álláshirdetést és jelentkezéskezelést. A rendszer lehetővé teszi a diákok számára, hogy egyszerűen böngézhessenek az elérhető munkák között, jelentkezhessenek a számukra megfelelő pozíciókra, és könnyedén kezeljék műszakjaikat.

Az alkalmazás nem csupán egy álláskereső platform, hanem egy teljes körű foglalórendszer is, amely segíti a diákokat a munkavállalás optimalizálásában. Az értékelési funkció révén a munkavállalók és a munkáltatók is visszajelzést adhatnak, ezzel elősegítve a transzparens és megbízható munkavégzést.

A rendszer funkciói

A platform négy fő szerepkört különböztet meg:

1. Diák

- Böngézhethet a meghirdetett munkák között
- Jelentkezhethet munkákra és műszakokra
- A szabályoknak megfelelően visszamondhatja a foglalt műszakokat
- Értékelheti az elvégzett munkákat

2. Közvetítő

- A szövetkezet által kijelölt személyként kezelheti az adott állásokra érkező jelentkezéseket
- Új műszakokat adhat hozzá

- Elfogadhatja vagy elutasíthatja a diákok műszak jelentkezéseit

3. Iskolaszövetkezet

- Új állásokat hozhat létre
- Közvetítőket jelölhet ki és kezelheti őket

4. Rendszergazda

- Jóváhagyhatja az iskolaszövetkezetek regisztrációját
- Felhasználókat és jogosultságokat kezelhet az asztali alkalmazásban

Be nem jelentkezett felhasználók lehetőségei

Azok a látogatók, akik még nem regisztráltak vagy nem jelentkeztek be, böngészhetik a munkalehetőségeket, valamint megtekinthetik a diákok által adott értékeléseket.

A platform célja, hogy minden résztvevő számára egy átlátható, hatékony és könnyen kezelhető rendszert biztosítson, amely elősegíti a diákok és a munkaadók sikeres együttműködését.

Technikai háttér

- A rendszer egy webalkalmazásként és asztali alkalmazásként is működik, így a különböző szerepkörök igényeihez igazodik.
- A háttérrendszer **MySQL adatbázist** használ az adatok tárolására.
- A **React** biztosítja a dinamikus és reszponzív felhasználói felületet.
- Az alkalmazás **ASP.NET** alapú backenddel rendelkezik, amely biztosítja a gyors és biztonságos adatkezelést.

Biztonsági megoldások

- A felhasználók adatait biztonságos, titkosított formában tároljuk.
- Jogosultsági szintek szabályozzák, hogy ki milyen adatokat érhet el és módosíthat.

Felhasználói élmény és támogatás

- Az oldal letisztult, könnyen használható felülettel rendelkezik, amely mobilra és asztali nézetre egyaránt optimalizált.
- Probléma esetén **ügyfélszolgálati támogatás** is elérhető e-mailen keresztül.

Jövőbeli fejlesztési tervek

- További automatizált értesítések (pl. munkaértesítők, elérhető új műszakok).
- Egyéni ajánlások a diákok számára a korábbi jelentkezések és értékelések alapján.
- Mobilalkalmazás fejlesztése az egyszerűbb hozzáférés érdekében.

Szükséges hardver- és szoftverkövetelmények

A rendszer megfelelő működéséhez az alábbi hardver- és szoftverfeltételek teljesítése szükséges.

Felhasználói követelmények (webalkalmazás használatához)

Hardverkövetelmények

- Processzor: **Intel Core i3 (vagy AMD ekvivalens) vagy jobb**
- Memória (RAM): **4 GB vagy több**
- Tárhely: **Legalább 500 MB szabad tárhely** (a böngésző gyorsítótárának és adatainak tárolásához)
- Kijelző: **Legalább 1366×768 felbontású monitor**
- Internetkapcsolat: **Stabil szélessávú internet** (ajánlott: minimum 5 Mbps)

Szoftverkövetelmények

- Operációs rendszer: **Windows 10+, macOS 11+ vagy Linux (Ubuntu 20.04+)**
- Böngésző: **Google Chrome (ajánlott), Mozilla Firefox, Microsoft Edge, Opera GX vagy Safari** (az alkalmazás támogatja az újabb verziókat)
- JavaScript és sütik engedélyezése szükséges a böngészőben

Fejlesztői és üzemeltetői követelmények (backend, adatbázis)

Frontend, backend futtatási követelmények:

- Processzor: **Intel Core i3 (vagy AMD ekvivalens) vagy jobb**
- Memória (RAM): **8 GB**
- Tárhely: **Legalább 5 GB szabad hely az adatbázis és naplófájlok számára**
- Internetkapcsolat: **Megbízható, minimum 50 Mbps feltöltési sebességű kapcsolat**
- OS: **Windows 10 vagy Windows 11**

Szoftverkörnyezet

Node.js és csomagkezelő

- **Node.js:** 18.x vagy újabb (ajánlott: 18.17.0 vagy frissebb)
- **npm:** 9.x vagy újabb (npm a Node.js telepítésével automatikusan elérhető)

A **Vite 6.0.6** és a **React 18.3.1** miatt legalább **Node.js 18.x** verzióra van szükség.

Alkalmazás és keretrendszerek

- **React:** 18.3.1
- **React DOM:** 18.3.1
- **React Router DOM:** 7.1.1
- **Recharts (grafikonokhoz):** 2.15.0
- **React Calendar (naptárkezeléshez):** 5.1.0
- **React Confirm Alert:** 3.0.6

React 18.3.1 és React Router DOM 7.1.1 biztosítják az alkalmazás megfelelő működését.

Stílus és UI

- **Tailwind CSS:** 3.4.17
- **AOS (animációkhoz):** 3.0.0-beta.6
- **Autoprefixer (CSS támogatás):** 10.4.20
- **PostCSS:** 8.4.49

Tailwind CSS 3.4.17-hez szükséges a megfelelő PostCSS verzió.

Adatkezelés és biztonság

- **Axios (HTTP-kérésekhez):** 1.7.9
- **JWT-decode (felhasználói hitelesítéshez):** 4.0.0

Axios 1.7.9 és JWT-decode 4.0.0 biztosítja az API-kommunikációt és a hitelesítést.

Adatrács és táblázatok

- **ag-Grid Community:** 33.0.4
- **ag-Grid Enterprise:** 33.1.1
- **ag-Grid React:** 33.0.4

Mivel az **ag-Grid 33.x** sorozatot használod, mindhárom csomagnak ezt a verzióját kell telepíteni.

Fejlesztői eszközök

- **Vite:** 6.0.6 (React alkalmazás gyors indításához)
- **ESLint:** 9.17.0 (kóddellenőrzéshez)
- **@vitejs/plugin-react:** 4.3.4 (React támogatás Vite-hez)
- **TypeScript:** 5.6.3 (ha TypeScript-et használasz)
- **@types/react:** 18.3.18 (React típusdefiníciók TypeScript-hez)

Vite 6.0.6 és TypeScript 5.6.3 szükséges a fejlesztési környezethez.

Összegzés: Minimum szükséges verziók

- **Node.js:** 18.17.0 vagy újabb
- **React:** 18.3.1
- **Vite:** 6.0.6
- **TypeScript:** 5.6.3
- **Tailwind CSS:** 3.4.17
- **Axios:** 1.7.9
- **ag-Grid:** 33.0.4 vagy újabb

Telepítési és indítási útmutató

Alapvető szoftverek telepítése

Node.js telepítése

A **Node.js** egy futtatókörnyezet, amelyre a StudentHive alkalmazás működéséhez szükség van.

1. Nyisd meg a böngésződet, és menj a következő oldalra:
<https://nodejs.org/>
2. Töltsd le a **Latest (LTS) verziót** (ajánlott stabil verzió).
3. Futtasd a letöltött **telepítőt** (.msi Windowsra, .pkg Macre, vagy a megfelelő Linux csomagot).
4. Kövesd a telepítő utasításait, és engedélyezd a **PATH beállítások módosítását**, ha kéri.
5. Telepítés után ellenőrizd a verziókat:

```
node -v  
npm -v
```

Ha a két parancs sikeresen kiírja a verziószámokat (pl. v18.17.0), akkor a Node.js és az npm sikeresen telepítve van.

Git telepítése (ha szükséges)

A **Git** nem kötelező, de segít a projekt letöltésében és kezelésében.

1. Nyisd meg: <https://git-scm.com/>
2. Töltsd le és telepítsd a megfelelő verziót az operációs rendszeredhez.
3. A telepítés után ellenőrizd, hogy sikeresen települt:

```
git --version
```

Ha verziószámot ad vissza (pl. `git version 2.39.1`), akkor minden rendben.

Kódszerkesztő (ajánlott: Visual Studio Code)

Ha nincs telepítve fejlesztői környezeted, ajánlott a **VS Code** használata:

1. Nyisd meg: <https://code.visualstudio.com/>
2. Töltsd le és telepítsd a legfrissebb verziót.

Visual Studio 2022 telepítése

Az ASP.NET és WPF futtatásához szükség van a **Visual Studio 2022** fejlesztői környezetre.

1. Nyisd meg a következő oldalt:
<https://visualstudio.microsoft.com/downloads/>
2. Töltsd le a **Visual Studio 2022 Community** verziót (ingyenes).
3. Indítsd el a letöltött **telepítőt**.

Telepítés közben válaszd ki a szükséges fejlesztői csomagot:

- **ASP.NET and web development**
 - **.NET desktop development**
 - **Data storage and processing**
4. Kattints a **Telepítés** gombra, és várd meg, amíg a folyamat befejeződik.

Amennyiben már telepítve van:

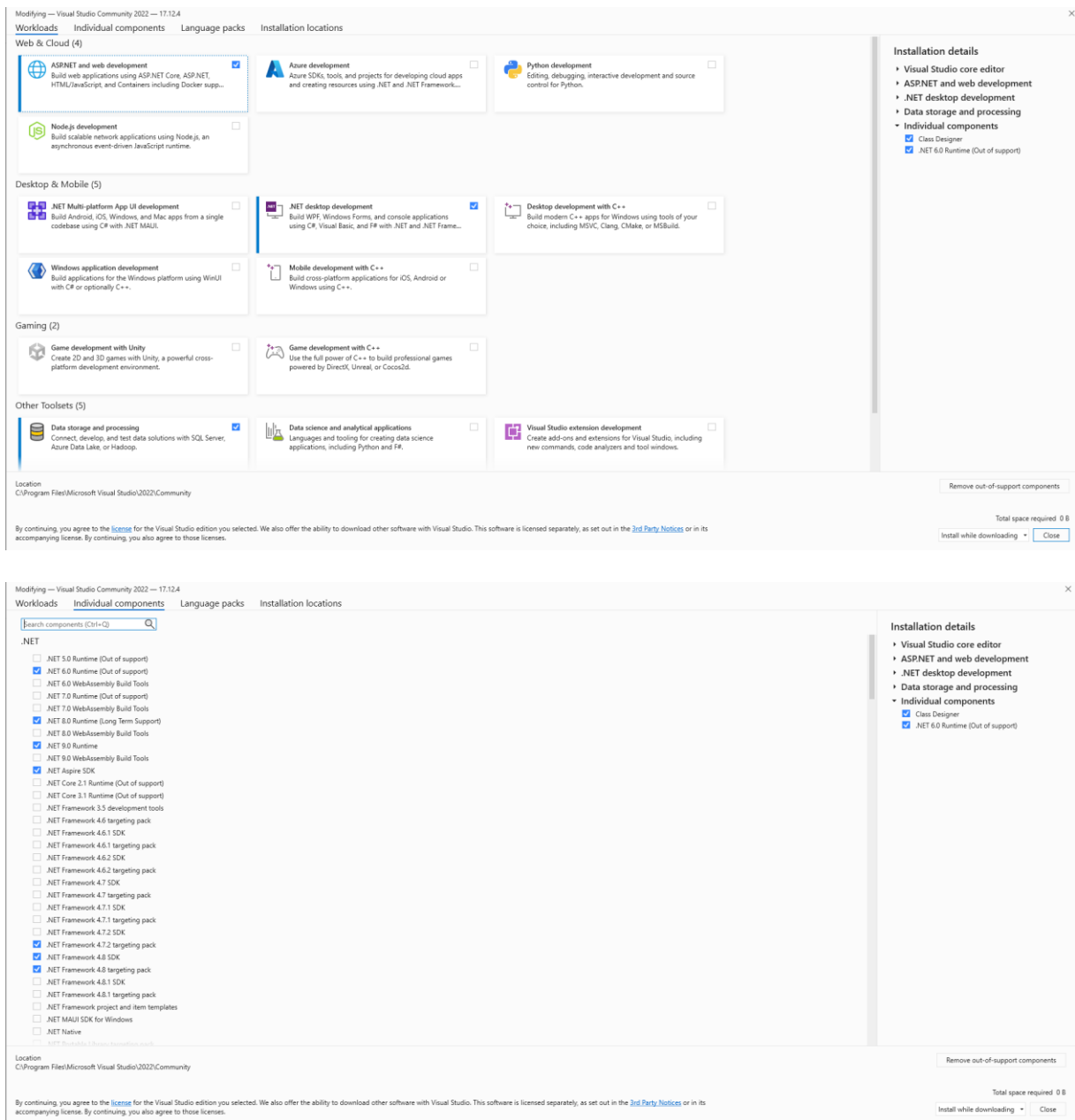
1. Nyisd meg a **Visual Studio Installert**
2. A Visual Studio 2022 Community-nél Modify gombra kattinva lehet módosítani a fejlesztői csomagokon, illetve a .NET verziókon.

Fontos! A backend .NET6 verzió fut, tehát telepítve kell lennie. Ha nincs így telepíthető:

1. Nyisd meg:
<https://dotnet.microsoft.com/en-us/download/dotnet/6.0>
2. Töltsd le a **.NET 6 SDK (SDK 6.0.428)** verziót az operációs rendszeredhez.
3. Futtasd a telepítőt, és kövesd az utasításokat.
4. Telepítés után ellenőrizd, hogy sikeresen feltelepült:

```
dotnet --version
```

Ha a parancs kiír egy verziószámot (pl. `6.0.412`), akkor a .NET SDK telepítve van.



XAMPP letöltése és telepítése

1. Nyisd meg a következő oldalt:
<https://www.apachefriends.org/download.html>
2. Válaszd ki a **Windows** verziót, és töltsd le az **XAMPP 8.x.x** telepítőt.
3. Futtasd a letöltött `xampp-windows-x64-x.x.x.exe` fájlt.
4. A telepítés során válaszd ki a szükséges komponenseket (**Apache, MySQL** legyen kiválasztva, a többi opcionális).

5. Telepítés után kattints a **Finish** gombra.

A projekt letöltése

A kód letöltése Git segítségével

Ha **Git** telepítve van, akkor klónozhatsz a projektet:

```
git clone https://github.com/lacydeth/StudentHive.git
```

Ha nincs Git, töltsd le manuálisan:

1. Nyisd meg a GitHub oldalt.
2. Kattints a **Code** gombra, majd válaszd a **Download ZIP** opciót.
3. Csomagold ki egy tetszőleges mappába.

Ezután lépj be a projekt könyvtárába:

```
cd StudentHive
```

Frontend megnyitása

A React projekt megnyitása CMD-vel

Ha a **VS Code** telepítve van (belépés a projektmappába, majd kódszerkesztő megnyitása):

```
cd frontend  
code .
```

A React projekt megnyitása kódszerkesztőből

Ha a VS Code telepítve van:

1. **VS Code** futtatása új ablakban
2. **File** → **Open folder**
3. Leklónozott **StudentHive** mappában a **frontend** mappa kiválasztása
4. **Mappaválasztás** gomb megnyomása

Szükséges csomagok telepítése

A projekt működéséhez különböző **npm csomagok** szükségesek. Ezeket a **package.json** fájl tartalmazza, és egyszerűen telepíthetők:

```
npm install
```

Ez letölti az összes szükséges függőséget.

Ha a telepítés sikeres, egy **node_modules/** mappa jön létre, amely az összes szükséges könyvtárat tartalmazza.

Frontend futtatása

Fejlesztői mód indítása

A fejlesztői szerver indításához futtasd az alábbi parancsot:

```
npm run dev
```

Ez elindítja a **Vite** fejlesztői szerveret, és meg kell jelennie egy üzenetnek:

```
VITE v6.0.6 ready in 1380 ms

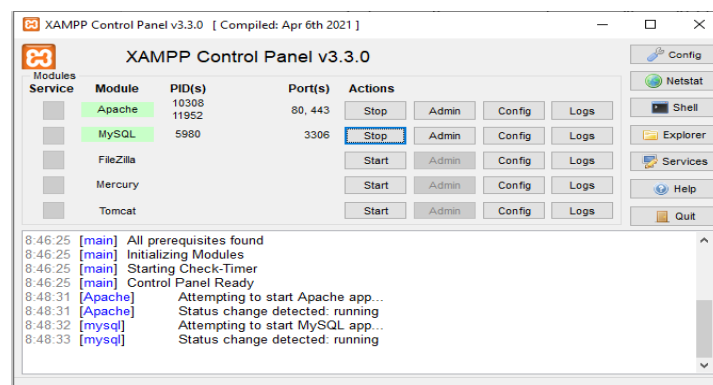
→ Local:   http://localhost:3000/
→ Network: http://192.168.6.5:3000/
→ Network: http://172.28.16.1:3000/
→ Network: http://172.27.0.1:3000/
→ press h + enter to show help
```

Nyisd meg a böngészőt, és látogasd meg a **http://localhost:3000/** címet.

Adatbázis indítása és tesztelése

XAMPP Control Panel megnyitása

1. Keresd meg az **XAMPP Control Panel**-t a Start menüben, és indítsd el.
2. A vezérlőpanelen két fő szolgáltatást kell elindítani:
 - **Apache** (webszerver)
 - **MySQL** (adatbázis)
3. Kattints a **Start** gombra az **Apache** és **MySQL** mellett.
4. Ha minden rendben van, a modul zöld háttérrel jelenik meg.



Apache tesztelése (webszerver)

1. Nyiss meg egy böngészőt, és írd be:

```
http://localhost
```

2. Ha az XAMPP alapértelmezett oldala betöltődik, az Apache sikeresen fut.

MySQL tesztelése (adatbázis)

1. Nyisd meg a **phpMyAdmin** felületet:

```
http://localhost/phpmyadmin
```

2. Ha az oldal megnyílik, a MySQL szerver működik.

A backend projekt megnyitása

Amennyiben a **XAMPP** sikeresen elindult következhet a backend indítása. A projekt a **backend/** mappában található.

Ha Visual Studio 2022-t használsz:

1. Nyisd meg a Visual Studio-t.
2. Kattints a "**Open a project or solution**" opcióra.
3. Navigálj a backend mappába, és válaszd ki a **.sln** fájlt.
4. Kattints a **Megnyitás** gombra.

Másik opció:

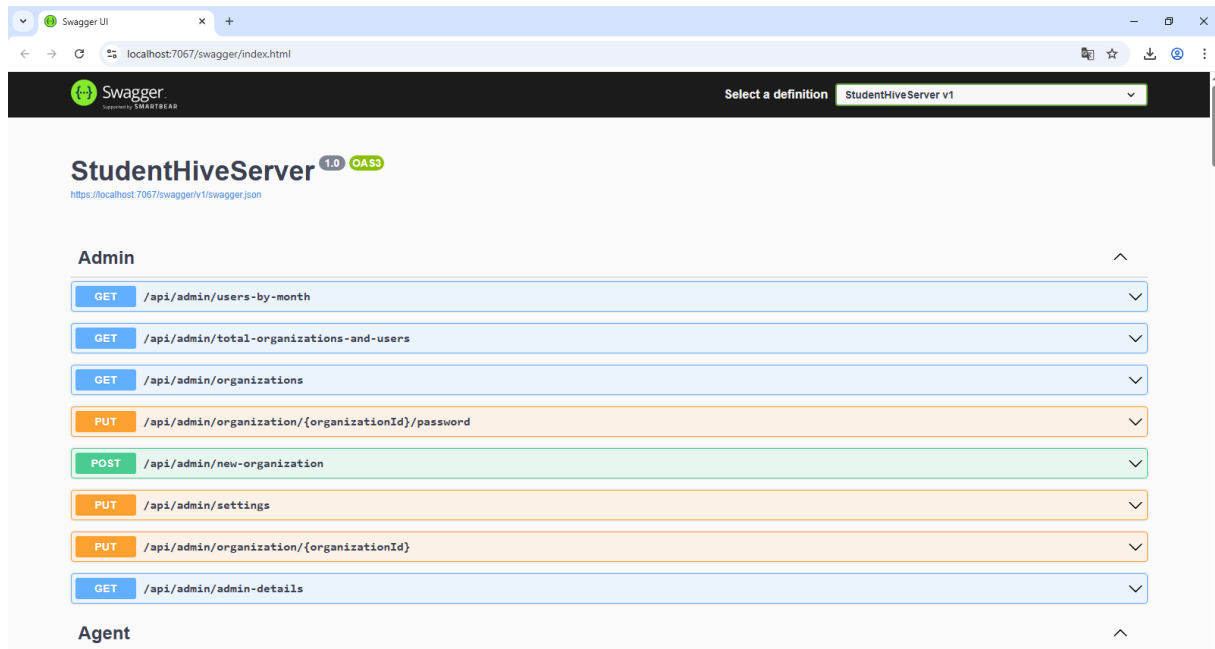
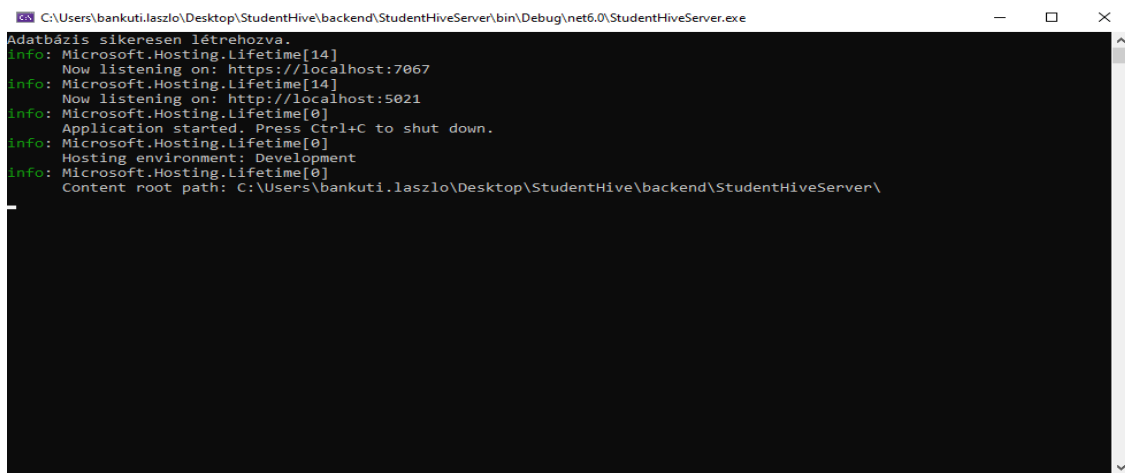
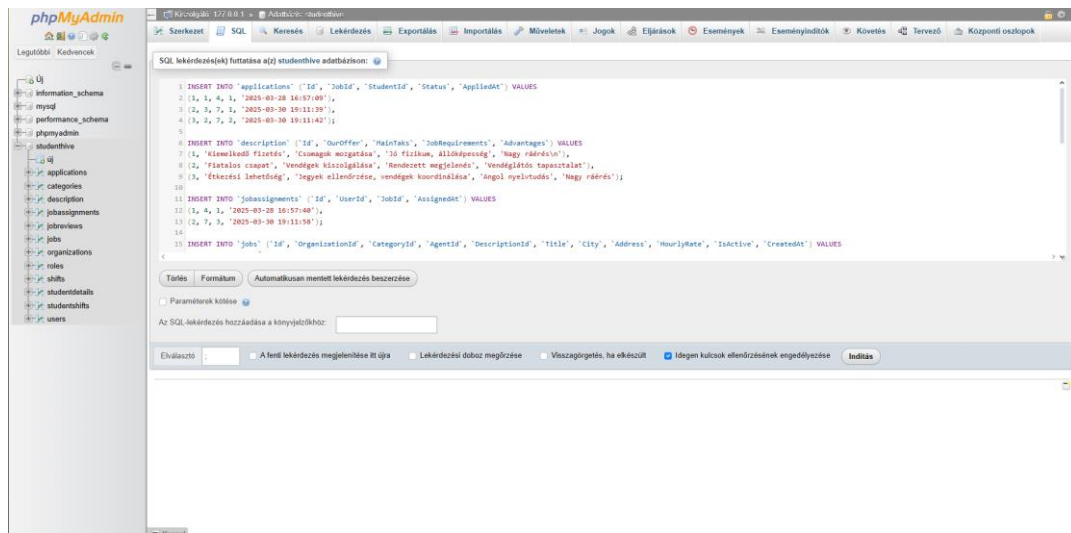
1. A lehúzott repositoryban a backend mappa megnyitása.
2. **StudentHiveServer.sln** fájl megnyitásával elindul a projekt.

Backend futtatása

Ha Visual Studio-t használsz:

1. Kattints a **Zöld Indítás** gombra (**F5**), vagy válaszd ki a **Debug → Start Debugging** opciót.
2. A szerver elindul a <https://localhost:7067> címen, és megjelenik egy terminál.

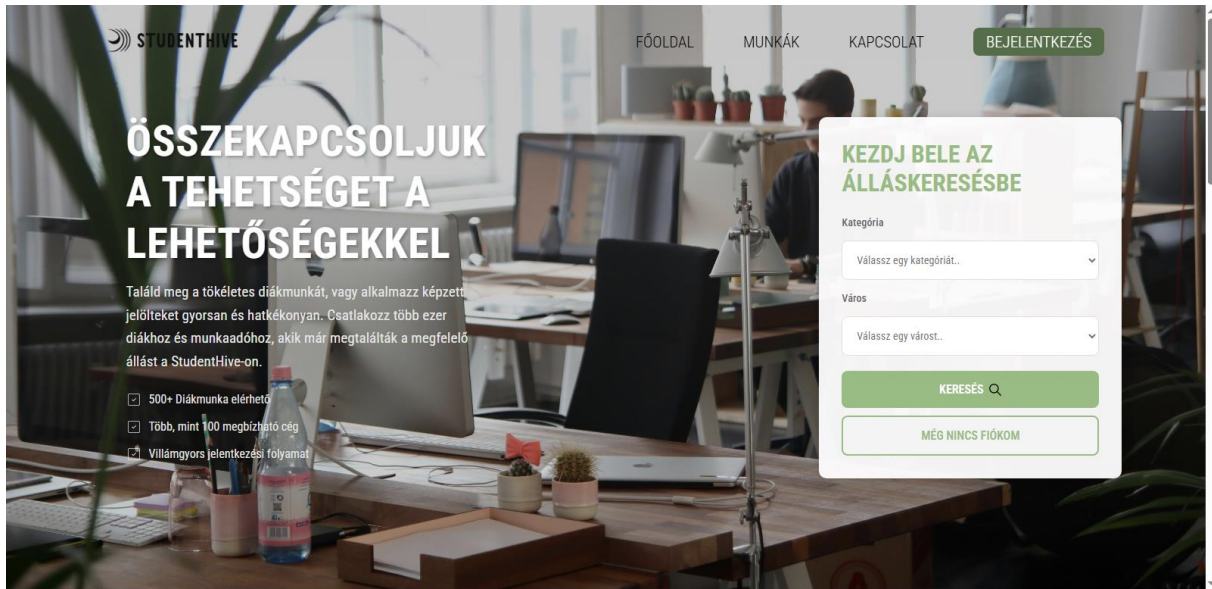
Amennyiben a **XAMPP** fut a backend automatikusan létrehozza az üres adatbázist, ennek a sikerességéről üzenetben tájékoztat (Az adatbázis sikeresen létrehozva). Továbbá elindul a **Swagger** is, megjelenítve az összes API route-ot. A teszteléshez szükséges adatok a **studenthive-imports** fájlban találhatóak. A tartalmát a **phpmyadmin** panelban a studenthive adatbázist kiválasztva az SQL fül alatt lehet beilleszteni. Ha ez sikerült a futtatás gombra kell kattintani és az adatok fel is töltődnek. Emellett megtalálható a **studenthive-complete** fájl is, amiben a táblákkal együtt a tesztadatok is megtalálhatóak, ha nem hozná létre a backend az adatbázist! Az adatbázisnak a nevének **studenthive** kell legyen!



A program használata

Bejelentkezés nélkül elérhető funkciók

Főoldal




A felhasználó a **StudentHive** meglátogatását követően egy köszöntő oldal fogadja. A felső navigációs sávból elérheti a következőket:


- Főoldal
- Munkák
- Kapcsolat
- Bejelentkezés

A pontosabb álláskeresés érdekében egy szűrővel ellátott panel található, ahol a felhasználó beállíthatja a keresett **kategóriát** (11 kategóriából választhat) és **várost**. Ezt követően a **keresés** gombra kattintva a beállított szűrőkkel megjelennek az elérhető munkák. A panelen található még egy **még nincs fiókom** gomb, amely a regisztrációs felületre irányítja a felhasználót.


FUNKCIÓINK
Modern álláskeresős




Egyszerű adminisztráció
Minden folyamat egy helyen:
könnyen kezelhető rendszer a
munkaügyi adminisztrációhoz.




Rugalmas időbeosztás
Időpontok és műszakok egyszerű
tervezése és módosítása valós
időben.



Átlátható pénzügyek
Részletes statisztikák és
kimutatások a bérek és kiadások
nyomon követésére.



Szerződéskötés
Gyors és egyszerű: automatikus
szerződéskötés és módosítás
néhány kattintással.



StudentHive története


Miért alapult meg a StudentHive diákmunka fórum?

The **StudentHive** diákmunka fórum azzal a céllal jött létre, hogy összekösse a fiatalokat a legjobb munkalehetőségekkel egy könnyen átlátható, modern platformon.

Az alapítók felismerték, hogy sok diák számára kihívást jelent a megfelelő munka megtalálása, miközben a cégeknek is nehéz elérni a megbízható fiatal munkaerőt. A StudentHive egy olyan közösségi tér, ahol a diákok egyszerűen böngészhetnek az állásajánlatok között, megoszthatják tapasztalataikat, és közvetlenül kapcsolatba léphetnek a munkaadókkal.

A platform célja, hogy átlátható, gyors és biztonságos lehetőséget nyújtson a diákoknak a munkavállalásra.


A főoldalon található még pár információs szekció, itt a StudentHive történetét és különböző előnyeiről olvashat az érdeklődő.



Diák vagyok

Csatlakozz a StudentHive közösséghez diákként, és találd meg álmaid diákmunkáját! Böngéssz több száz állásajánlat között, jelentkezz egyszerűen, és építsd karrieredet már az iskolapadból.

Regisztráció diákként »



Iskolaszövetkezet vagyok

Hirdess állásokat egyszerűen és érj el motivált diákokat azonnal! Platformunk segítségével gyorsan megtalálhatod a megfelelő jelölteket, és hatékonyan kezelheted az adminisztrációt.

Regisztráció iskolaszövetkezetként »

Két kártya segítségével a látogató eldöntheti milyen szerepkörrel akar csatlakozni az oldalhoz. **Regisztráció diákként** gomb átirányítja a regisztrációs oldalra, a **regisztráció iskolaszövetkezetként** pedig egy információs oldalra, ahol tájékozódhat a csatlakozáshoz szükséges lépésekről.

Rólunk

Elősegítjük a diákokat, hogy egyszerű lehetőségeket találjanak és építsenek karrierjüket.

Oldalak

Szövetkezeteknek
Munkák
Kapcsolat

Elérhetőségeink

✉ info.studenthive@gmail.com

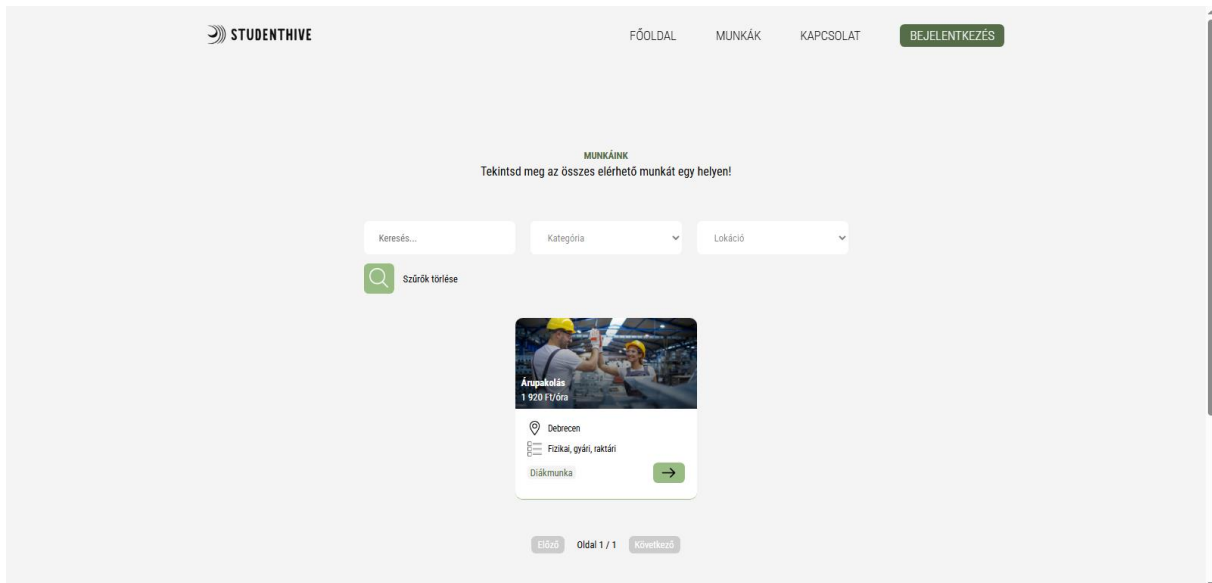
☎ +36123456789

📍 Debrecen, Széchenyi u. 58, 4025

© 2025 StudentHive. Minden jog fenntartva.

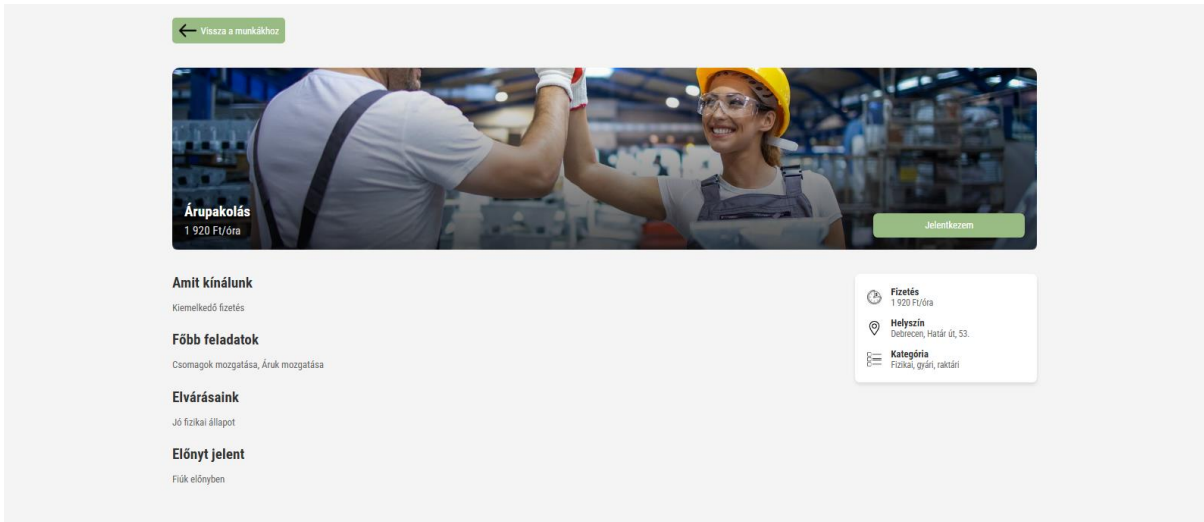
Emellett az oldal alján lábléc található. Bal oldalon egy szlogen, középen a főbb oldalakat érheti el, csak úgy, mint a navigációs sávból. Jobb oldalt a cégünk elérhetőségeivel találkozhat a felhasználó.

Munkák



A munkák oldal elérhető bejelentkezés nélkül is, itt böngészhet kedvére a látogató álláslehetőségeink között. A felső sávban a **pozíció nevének** beírásával lehet keresni, ezek mellett a főoldal szintű megtekinthető **kategória** és **lokáció** szűrés elérhető. A **zöld keresés** gombra kattintva lehet alkalmazni a beállított szűrőket, a **szűrők törlése** gombbal visszavonhatja azokat.

Az elérhető pozíciók kártyaként jelennek meg itt a főbb információk láthatóak, illetve egy **tovább** gomb, amely átirányít a munka saját oldalára.



[← Vissza a munkákhoz](#)

Árupalás
1 920 Ft/óra

Jelentkezem

Amit kínálunk
Kiemelkedő fizetés

Főbb feladatok
Csomagok mozgatása, Áruk mozgatása

Elvárásaink
Jó fizikai állapot

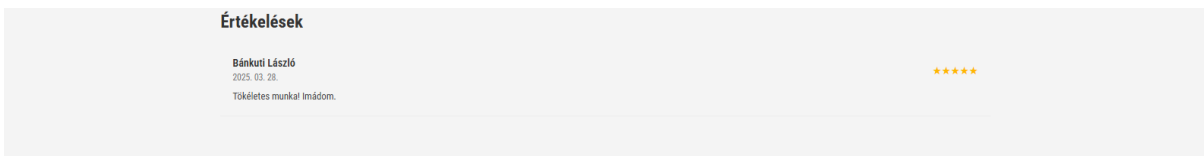
Előnyt jelent
Fiúk előnyben

Fizetés
1 920 Ft/óra

Helyszín
Debrecen, Határ út, 53.

Kategória
Fizikai, gyári, raktári

A munka saját oldalán leírás látható, fizetés, helyszín, kategória, illetve órabér. A jelentkezés gomb látható, azonban bejelentkezés nélkül a rendszer nem enged jelentkezni.



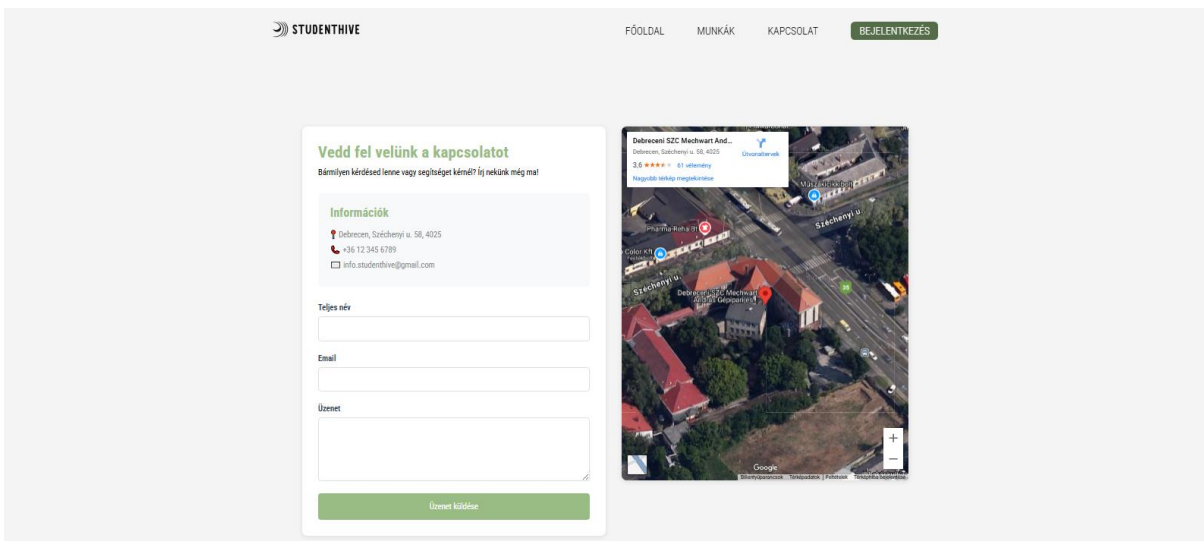
Értékelések

Bánkuti László
2025. 03. 28.
Tökéletes munkát láttam.

★★★★★

Az oldal alján a munkára beérkezett értékelések olvashatóak. Értékelni csak bejelentkezés után van lehetőség.

Kapcsolat



STUDENTHIVE

FŐOLDAL MUNKÁK KAPCSOLAT **BEJELENTKEZÉS**

Vedd fel velünk a kapcsolatot
Bármilyen kérdésed lenne vagy segítséget kérsz? Írj nekünk még ma!

Információk
Debrecen, Széchenyi u. 58, 4025
+36 12 345 6789
info.studenthive@gmail.com

Teljes név

Email

Üzenet

Üzenet küldése

Debreceni SZC Medwart And...
Debrecen, Széchenyi u. 58, 4025
3.6 ★★★★★ 61 vélemény
Nagyszerű helyszíni munkakörülmények

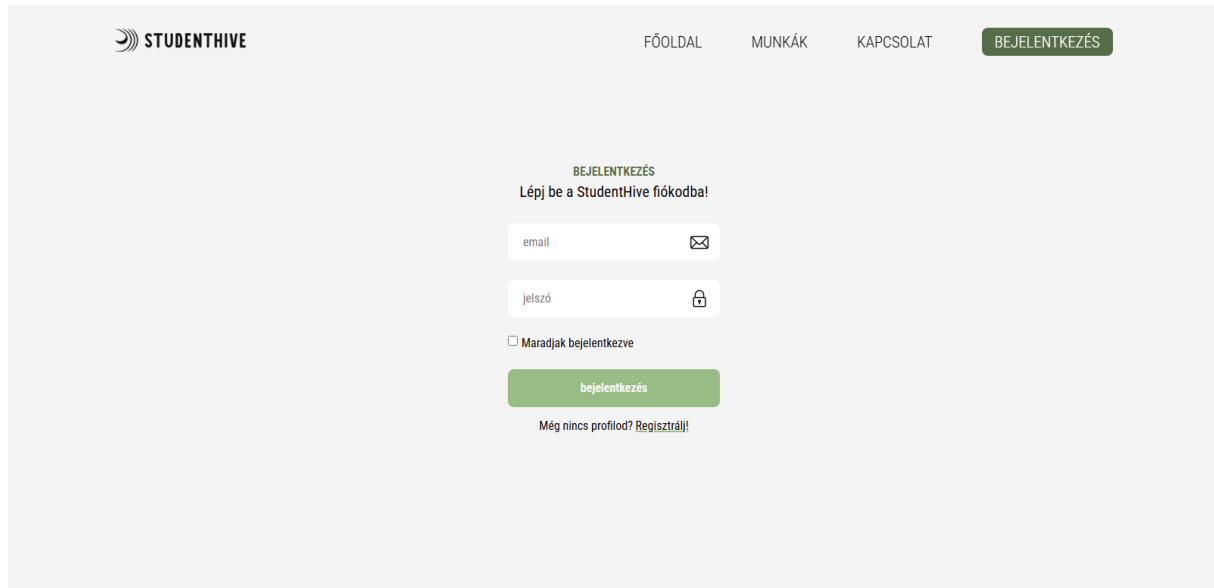
Google

A kapcsolat oldalon a láblécben szintúgy elhelyezett információk láthatóak, illetve egy kapcsolati űrlap. Üzenet küldéséhez szükséges (minden mezőt ki kell tölteni):

- Név (legalább két karakter hosszú)
- Email (helyes email formátum)
- Üzenet (legalább 10 karakter hosszú)

Jobb oldali panelben a **StudentHive** székhely lokációját láthatja a felhasználó.

Bejelentkezés



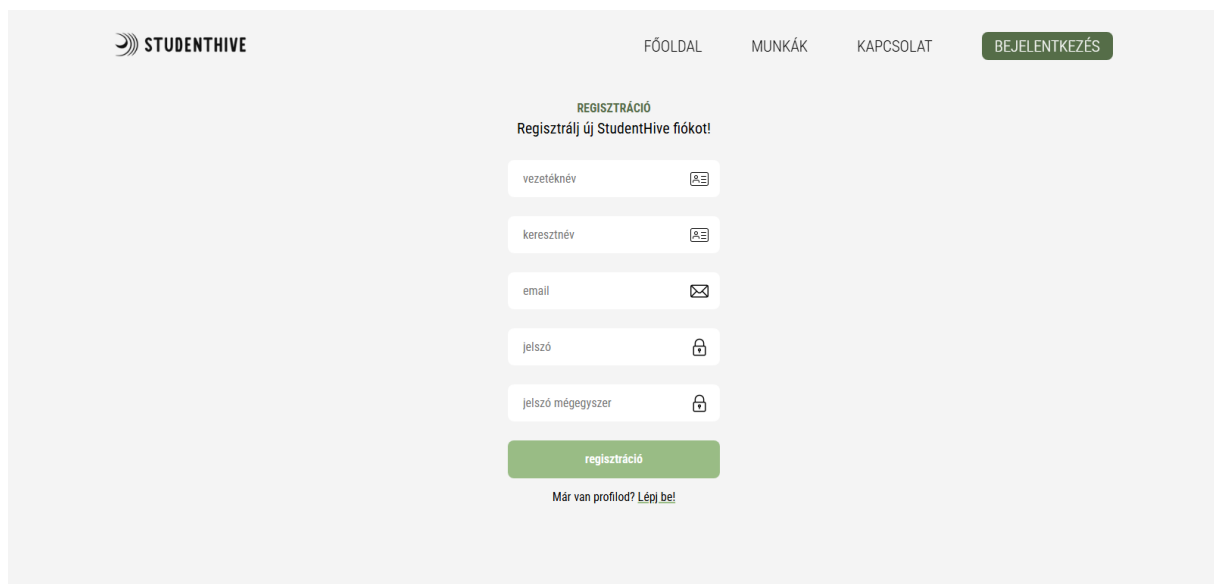
The screenshot shows the StudentHive login interface. At the top, there is a navigation bar with the StudentHive logo on the left and links for 'FŐOLDAL', 'MUNKÁK', 'KAPCSOLAT', and a green 'BEJELENTKEZÉS' button on the right. The main content area is titled 'BEJELENTKEZÉS' and 'Lépj be a StudentHive fiókodba!'. It contains two input fields: 'email' and 'jelszó', each with a corresponding icon (envelope and lock). Below these fields is a checkbox labeled 'Maradjak bejelentkezve'. A green 'bejelentkezés' button is positioned below the checkbox. At the bottom, there is a link that says 'Még nincs profilod? [Regisztrálj!](#)'.

Bejelentkezéshez szükséges adatok:

- Email
- Jelszó

A **maradjak bejelentkezve** checkbox bepipálásával a token **2 évig** fog élni, enélkül **1 nap** után lejár és újra be kell lépni. **Regisztrálj** gomb átirányít a regisztrációs felületre.

Regisztráció



The screenshot shows the StudentHive registration interface. At the top, there is a navigation bar with the StudentHive logo on the left and links for 'FŐOLDAL', 'MUNKÁK', 'KAPCSOLAT', and a green 'BEJELENTKEZÉS' button on the right. The main content area is titled 'REGISZTRÁCIÓ' and 'Regisztrálj új StudentHive fiókot!'. It contains five input fields: 'vezetéknév', 'keresztnev', 'email', 'jelszó', and 'jelszó meggyeszer', each with a corresponding icon (A-Z, a-z, envelope, and lock). A green 'regisztráció' button is positioned below the last two fields. At the bottom, there is a link that says 'Már van profilod? [Lépj be!](#)'.

Regisztrációhoz szükséges adatok:

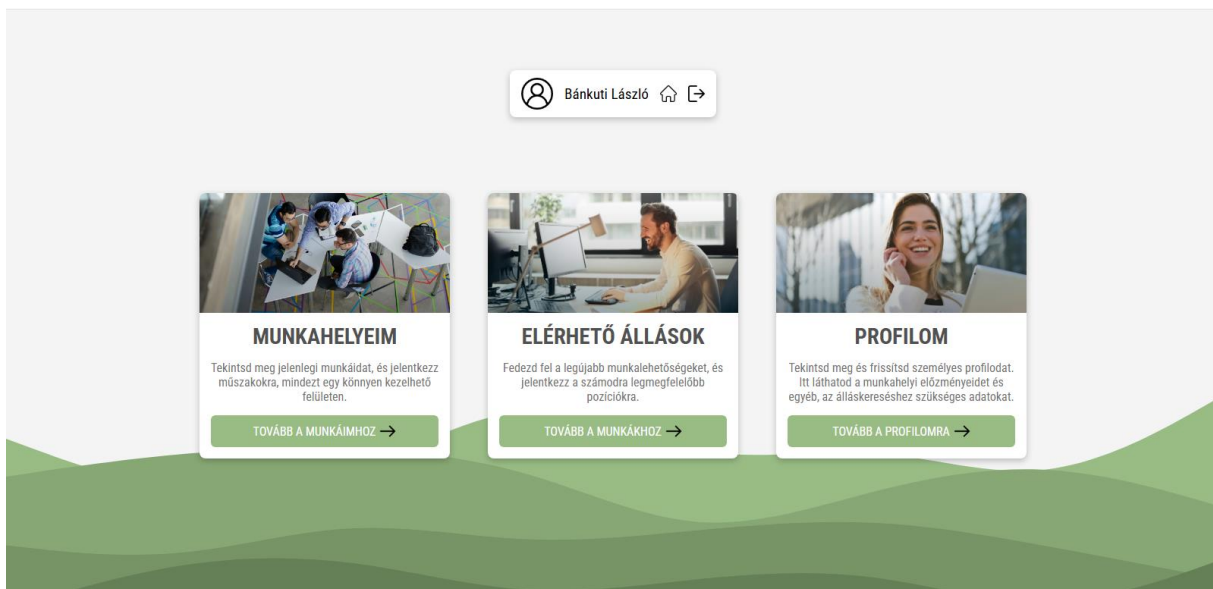
- Vezetéknév (csak betűket tartalmazhat és maximum 64 karakter hosszú)
- Keresztnev (csak betűket tartalmazhat és maximum 64 karakter hosszú)

- Email (nem lehet használatban, helyes email formátum)
- Jelszó (nagybetűt, számot kell tartalmazni, illetve 8-15 karakter között kell lennie)

Sikeres regisztrációt követően az oldal felugró üzenetben tájékoztat, ellenkező esetben kiírja a probléma okát. A **lépj be** gomb átirányít a bejelentkező felületre.

Bejelentkezés diák profillal

Köszöntőoldal

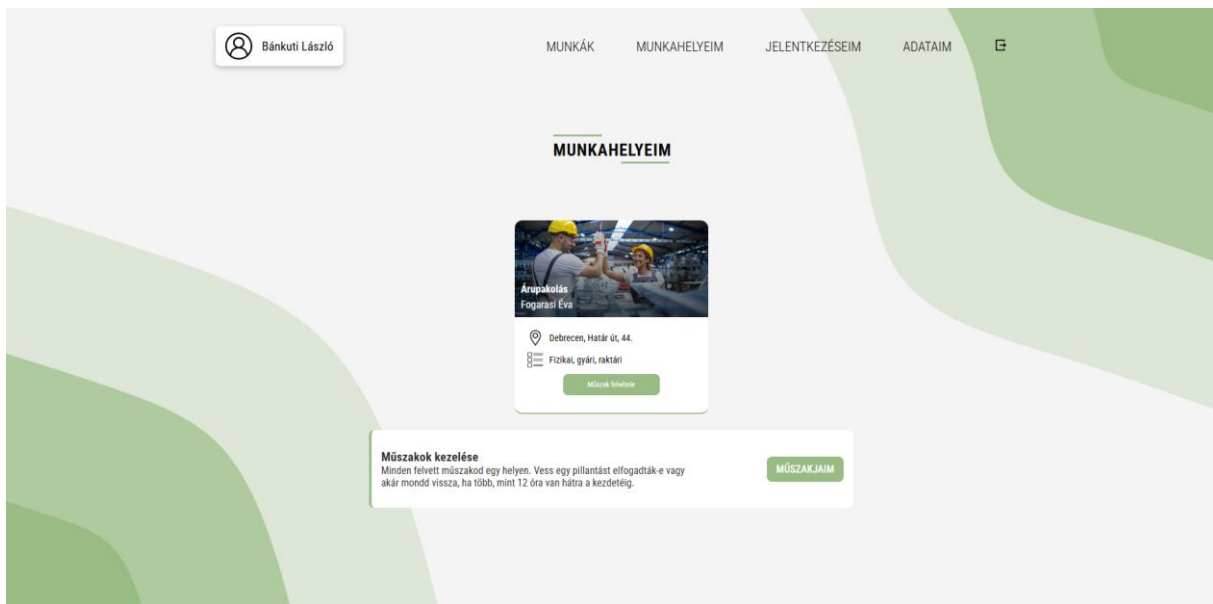


A bejelentkezett diák ezzel az oldallal találkozik először. A felső sávban látható a **név**, illetve **vissza a főoldalra** és **kijelentkezés** gomb.

A három kártyával navigálhat a felhasználó:

- Munkahelyeim: állások, ahová már felvették a diákot
- Elérhető állások: munkák oldal az összes elérhető állással
- Profilom: az álláskeresőkhöz szükséges profiladatok megadása

Munkahelyeim



A "**Munkahelyeim**" oldal lehetőséget biztosít a felhasználóknak, hogy megtekinthessék az általuk elfogadott munkákat, valamint kezelhessék műszakjaikat. Az oldalon az elfogadott munkák listája és egy műszakkezelő szekció található.

Funkciók és használat

1. Oldal felépítése

Az oldal három fő részből áll:

- **Navigációs sáv:** Az oldal tetején található, és segíti a többi funkció elérését.
- **Munkahelyek listája:** Az elfogadott munkák kártyák formájában jelennek meg, lapozható listában.
- **Műszakok kezelése szekció:** Lehetőséget biztosít a műszakok megtekintésére és visszamondására.

2. Munkák megtekintése

1. Az oldal betöltésekor automatikusan lekérdezi az elfogadott munkákat.
2. Ha van elérhető munka, azok kártyák formájában jelennek meg.
3. Ha nincs elfogadott munka, egy üzenet jelenik meg, amely javasolja a munkakereső oldal meglátogatását.

3. Lapozás a munkahelyek között

- Ha több mint három munkahely tartozik a felhasználóhoz, akkor az oldal lapozható.
- Az "Előző" és "Következő" gombokkal lehet lépkedni az oldalak között.
- A jelenlegi oldal és az összes oldal száma a gombok között látható.

4. Műszakok kezelése

1. A "Műszakjaim" gombra kattintva a felhasználó a műszakkezelő oldalra navigálhat.
2. Az oldalon megtekintheti a műszakjai állapotát (elfogadott vagy visszamondható).
3. A műszak visszamondására lehetőség van, ha annak kezdete előtt legalább **12 óra** van hátra.

Hibaelhárítás

1. Nem jelenik meg egyetlen munka sem:

- Ellenőrizze, hogy bejelentkezett-e.
- Ha nincs elfogadott munka, használja az álláskereső linket.

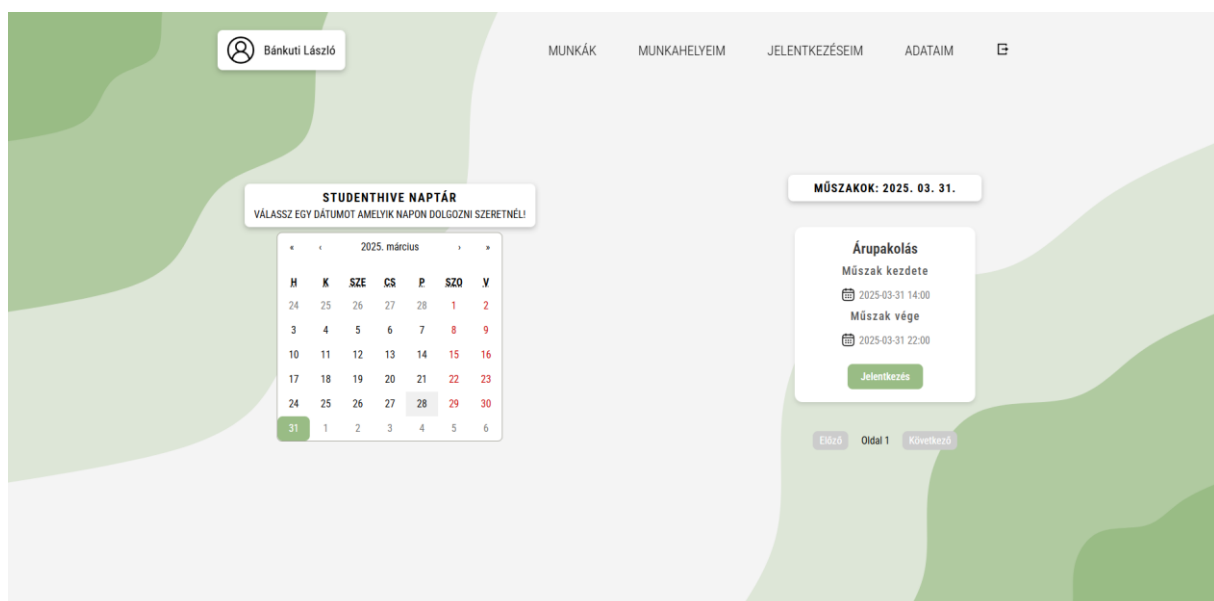
2. Nem működik a lapozás:

- Ha csak 3 vagy kevesebb munkahelye van, nincs szükség lapozásra.
- Próbálja meg frissíteni az oldalt, és ellenőrizze az internetkapcsolatot.

3. A műszakkezelő gomb nem működik:

- Próbálja meg frissíteni az oldalt.
- Ellenőrizze, hogy megfelelő jogosultságokkal rendelkeznek-e.

Műszak felvétele



Egy felhasználói felület, amely lehetővé teszi a felhasználók számára, hogy elérhető műszakokat tekintsenek meg és jelentkezzenek rájuk egy adott napon. Az oldal tartalmaz egy naptárat, amely segítségével a felhasználók kiválaszthatják a kívánt dátumot, valamint egy műszaklistát, amely megmutatja az adott napra vonatkozó műszakokat.

Az oldal használata

1. Navigáció az oldalra

Az oldalra belépve a felhasználó a következőket láthatja:

- **Navigációs sáv** – Az oldal tetején található, amely lehetőséget biztosít a további oldalakra való átlépéshez.
- **Naptár**– A bal oldalon elhelyezkedő modul, ahol kiválasztható az a nap, amelyen a felhasználó dolgozni szeretne.
- **Műszakok listája** – Az aktuálisan kiválasztott dátumhoz tartozó műszakok listája jelenik meg.

2. Dátum kiválasztása

1. A felhasználó kattintson a naptárban egy adott dátumra.
2. A rendszer ellenőrzi, hogy a kiválasztott dátum a jövőben van-e. Ha múltbeli dátumot választ, hibaüzenetet kap:
„Nem választhatsz múltbeli dátumot!”
3. Ha érvényes dátumot választ, a műszakok listája automatikusan frissül.

3. Műszakok megtekintése

- A kiválasztott dátumhoz tartozó műszakok egy kártyaformátumban jelennek meg.
- Minden műszak tartalmazza a következő információkat:
 - **Műszak neve**
 - **Kezdési időpont**
 - **Befejezési időpont**
 - **Jelentkezési lehetőség**

4. Oldalak közötti navigáció (Lapozás)

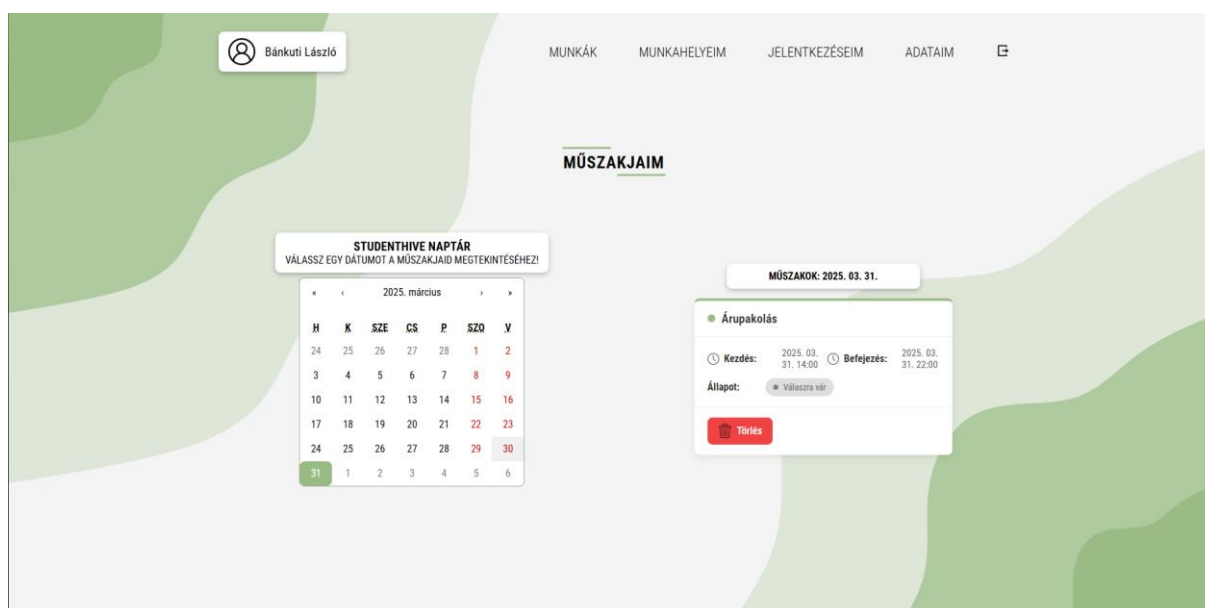
Ha több műszak is elérhető egy napon, akkor a listázás oldalakra bontva jelenik meg.

- **„Előző” gomb** – Az előző oldalra léphet vissza.
- **„Következő” gomb** – Ha több műszak van, ezzel lehet tovább lépni a következő oldalra.

5. Műszakra jelentkezés

1. A felhasználó a kívánt műszaknál rákattinthat a „Jelentkezés” gombra.
2. A rendszer ellenőrzi, hogy a műszak már elkezdődött-e. Ha igen, hibaüzenetet kap: „Nem jelentkezhetsz múltbeli műszakra!”
3. Ha a jelentkezés sikeres, egy megerősítő üzenet jelenik meg: „Sikeres jelentkezés!”

Műszakjaim



A "Műszakjaim" oldal lehetőséget biztosít a felhasználóknak, hogy megtekinthessék és kezelhessék a műszakjaikat. Az oldal tartalmaz egy naptárt a műszakok dátum szerinti megjelenítéséhez, valamint egy listát az adott napra vonatkozó műszakokról, amelyek törölhetők is.

Funkciók és használat

1. Oldal felépítése

Az oldal három fő részből áll:

- **Navigációs sáv:** Az oldal tetején található navigációs sáv lehetőséget biztosít az egyéb felhasználói funkciók elérésére.
- **Naptár:** Egy interaktív naptár, amelyben kiválasztható egy adott dátum, és megjeleníthetők a hozzá tartozó műszakok.
- **Műszaklista:** Az adott naphoz tartozó műszakokat jeleníti meg, lehetőséget biztosítva azok törlésére.

2. Műszakok megtekintése

1. Az oldal betöltésekor automatikusan a mai dátumhoz tartozó műszakok jelennek meg.
2. Más dátumhoz tartozó műszakok megtekintéséhez kattintson a naptárban egy kívánt dátumra.
3. A dátum kiválasztása után az oldal automatikusan frissül, és megjeleníti az adott napra beosztott műszakokat.

3. Műszak törlése

1. A műszaklistában minden műszak külön kártyán jelenik meg.
2. A törléshez kattintson a műszakhoz tartozó "Törlés" gombra.
3. A rendszer egy biztonsági ellenőrzést végez (több, mint 12 óra van hátra a műszakig), és a műszak sikeres törléséről értesítést küld.
4. A törlés után az oldal automatikusan frissül, és a műszaklista frissített állapotban jelenik meg.

4. Lapozás a műszakok között

- Ha egy adott nap több műszakkal rendelkezik, a lista több oldalra oszlik.
- Az "Előző" és "Következő" gombok segítségével lehet váltani az oldalak között.
- A jelenlegi oldal és az összes oldal száma a gombok között látható.

Hibaelhárítás

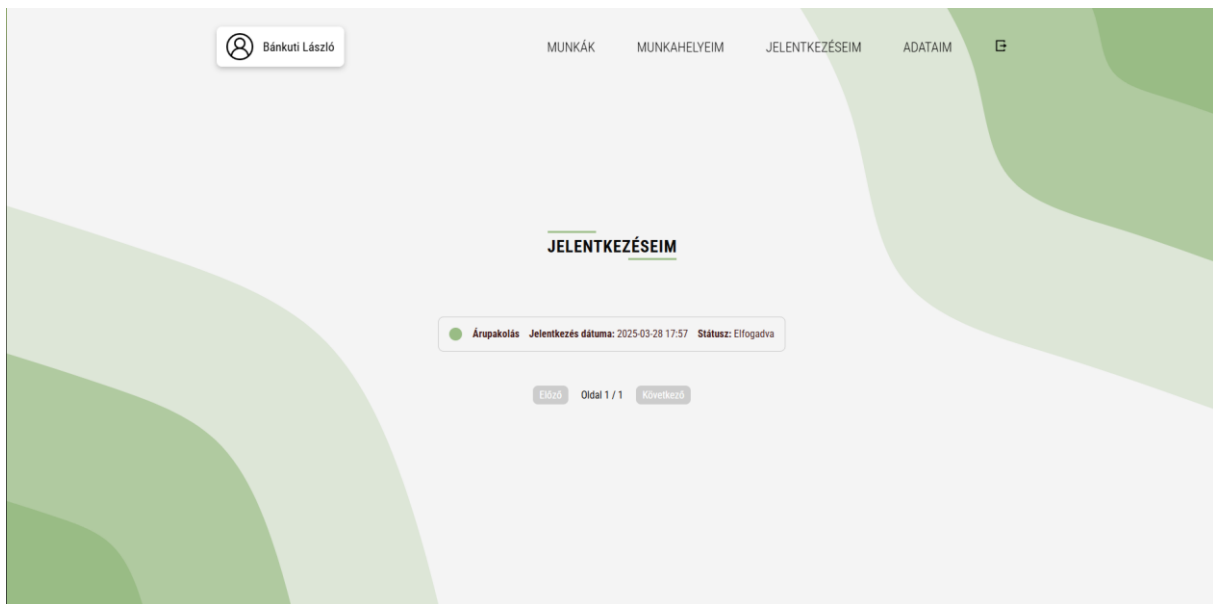
1. Nincs műszak az adott napon:

- Ha nincs műszak a kiválasztott napon, egy "Nincs műszak a kiválasztott napon" üzenet jelenik meg.

2. Sikertelen műszak törlés:

- Ha a törlés sikertelen, egy hibaüzenet jelenik meg.
- Ellenőrizze, hogy rendelkezik-e megfelelő jogosultságokkal.
- Próbálja meg újratölteni az oldalt, és ismételje meg a műveletet.

Jelentkezéseim



A "Jelentkezéseim" oldal lehetőséget biztosít a felhasználóknak, hogy megtekinthessék és kezelhessék az állásjelentkezéseiket. Az oldal tartalmaz egy listát az összes leadott jelentkezésről, amelyeket törölhetnek is.

Funkciók és használat

1. Oldal felépítése

Az oldal három fő részből áll:

- **Navigációs sáv:** Az oldal tetején található navigációs sáv lehetőséget biztosít az egyéb felhasználói funkciók elérésére.
- **Jelentkezési lista:** Az összes aktív jelentkezést megjeleníti.
- **Lapozás:** Ha sok jelentkezés van, több oldalra osztva lehet őket böngészni.

2. Jelentkezések megtekintése

1. Az oldal betöltésekor automatikusan megjelennek a felhasználó korábbi jelentkezései.
2. A jelentkezések kártyák formájában jelennek meg, amelyek tartalmazzák az állás nevét és egyéb releváns információkat.
3. Ha nincs leadott jelentkezés, egy üzenet jelenik meg, amely javasolja az álláskeresés elkezdését.

3. Jelentkezés törlése

1. A jelentkezési listában minden jelentkezés külön kártyán jelenik meg.

2. A törléshez kattintson a jelentkezéshez tartozó "Törlés" gombra.
3. A rendszer egy biztonsági ellenőrzést végez, és a sikeres törlésről értesítést küld.
4. A törlés után az oldal automatikusan frissül, és a jelentkezési lista frissített állapotban jelenik meg.

4. Lapozás a jelentkezések között

- Ha egy felhasználónak sok jelentkezése van, a lista több oldalra oszlik.
- Az "Előző" és "Következő" gombok segítségével lehet váltani az oldalak között.
- A jelenlegi oldal és az összes oldal száma a gombok között látható.

Hibaelhárítás

1. Nincs leadott jelentkezés:

- Ha nincs jelentkezés, egy "Nincs leadott jelentkezésed" üzenet jelenik meg.

2. Sikertelen jelentkezés törlés:

- Ha a törlés sikertelen, egy hibaüzenet jelenik meg.
- Ellenőrizze, hogy rendelkezik-e megfelelő jogosultságokkal.
- Próbálja meg újratölteni az oldalt, és ismételje meg a műveletet.

Adataim

The screenshot shows the 'ADATAIM' user profile page. The navigation bar includes 'MUNKÁK', 'MUNKAHELYEIM', 'JELENTKEZÉSEIM', and 'ADATAIM'. The 'ADATAIM' section is active, displaying a form for personal data. The form is divided into three tabs: 'Személyes adatok', 'Lakcím', and 'Dokumentumok'. The 'Személyes adatok' tab is selected, showing fields for 'Általános információk' (Vezetéknév, Keresztnév, Email, Telefonszám) and 'Születési adatok' (Születési dátum, Születési név, Anyja neve, Születési ország, Születési hely, Nem, Állampolgárság). A 'Adatok mentése' button is at the bottom right.

A felhasználók itt tekinthetik meg és módosíthatják személyes adataikat, például nevüket, elérhetőségüket, lakcímüket és iskolai adataikat.

Az oldal három fő adatcsoportot tartalmaz:

- **Személyes adatok**
- **Lakcím**
- **Dokumentumok**

Ha az adatok nem töltődnek be megfelelően, frissítse az oldalt vagy ellenőrizze, hogy be van-e jelentkezve.

Adatok Módosítása és Mentése

Adatmódosítás lépései:

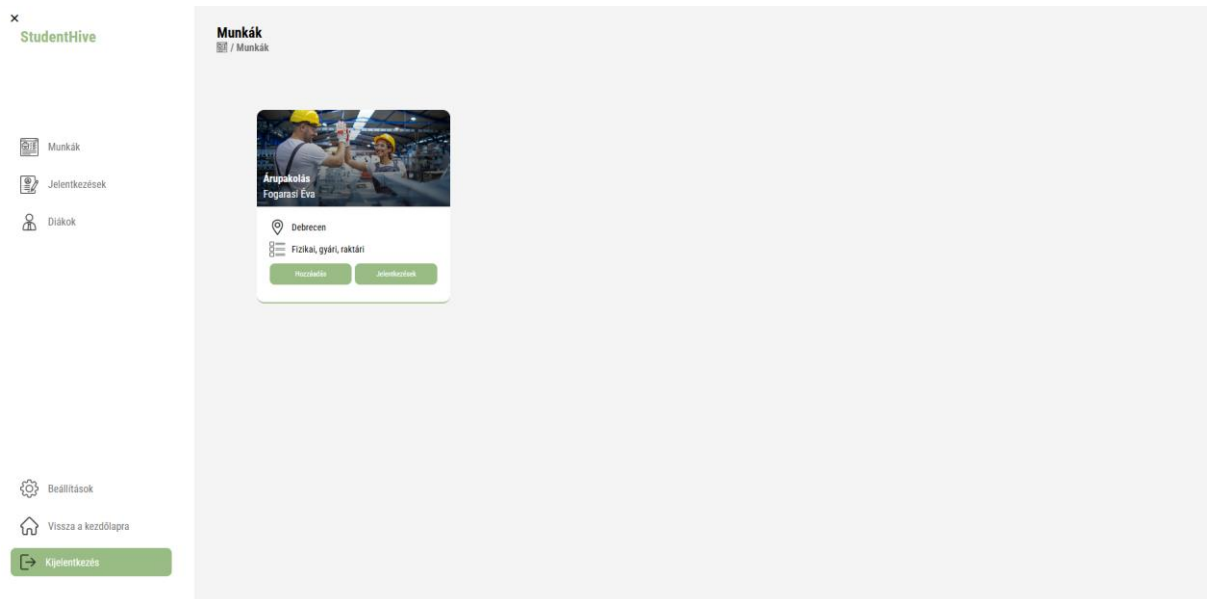
1. Válassza ki a módosítani kívánt adatot, majd írja be az új értéket a mezőbe.
2. Ügyeljen arra, hogy az adatok a következő követelményeknek megfeleljenek:
 - **Név, születési hely, állampolgárság stb.:** Csak betűket tartalmazhat és legfeljebb 64 karakter hosszú lehet.
 - **Email cím:** Érvényes email formátum (pl.: user@example.com).
 - **Telefonszám:** Pontosan 11 számjegyből kell állnia.
 - **Diákigazolvány szám:** Pontosan 11 számjegyből kell állnia.
 - **Bankszámlaszám:** Pontosan 8 számjegyből kell állnia.
 - **Irányítószám:** Pontosan 4 számjegyből kell állnia.
 - **Cím:** Legfeljebb 64 karakter hosszú lehet.
3. Ha minden módosítást elvégezt, kattintson a **Mentés** gombra.

Adatmentés és Visszajelzések

- **Sikeres mentés:** Ha az adatok megfelelőek, a rendszer elmenti azokat, és egy zöld értesítést jelenít meg:
"Az adatok sikeresen frissítve."
- **Hibás adatbevitel:** Ha egy adat nem felel meg a megadott követelményeknek, a rendszer figyelmeztetést küld és piros színnel jelzi a hibát. Példa:
"A telefonszámnak pontosan 11 számjegyből kell állnia."
- **Kapcsolódási hiba:** Ha az adatok mentése nem sikerült serverhiba miatt, próbálja újra később vagy ellenőrizze az internetkapcsolatát.

Bejelentkezés közvetítői profilal

Munkák



Az oldalon a közvetítők megtekinthetik az általuk kezelt munkalehetőségeket.

Az oldal elérése és funkciói

A **Közvetítői Munkák** oldal megtekintéséhez kövesse az alábbi lépéseket:

1. Bejelentkezés

- Győződjön meg róla, hogy be van jelentkezve a fiókjába.
- Ha nincs bejelentkezve, adja meg felhasználónevét és jelszavát.

2. Az oldal megnyitása

- A bal oldali **Sidebar (Navigációs menü)** segítségével kattintson a *Munkák* menüpontra.
- Az oldal betöltése után a közvetítőhöz rendelt munkák listája jelenik meg.

Felhasználói Felület és Navigáció

Munkák megjelenítése

- Az oldal betöltése után a közvetítő saját munkái automatikusan lekérdezésre kerülnek.
- Az egyes munkákat **kártyák** formájában jeleníti meg a rendszer, amelyek a következő adatokat tartalmazzák:
 - **Munka címe**

- **Közvetítő neve**
- **Helyszín**
- **Kategória**
- **Kép a munkáról**

Az adott kártyán két gomb lesz elérhető:

- **Hozzáadás:** Átirányít a műszak hozzáadása oldalra.
- **Jelentkezések:** Átirányít a jelentkezések oldalra, ahol a közvetítő el tudja fogni a műszak jelentkezéseket.

Adatok frissítése és betöltése

- Az adatok az oldal betöltésekor automatikusan frissülnek.
- A rendszer a közvetítő egyedi azonosítóját használja a releváns munkák lekérdezésére.
- Ha az adatok nem jelennek meg:
 1. Ellenőrizze, hogy be van-e jelentkezve.
 2. Próbálja meg frissíteni az oldalt.
 3. Ha továbbra is probléma van, ellenőrizze az internetkapcsolatát.

Műszak hozzáadása

A **Műszakok kezelése** oldalon az ügynök hozzáadhat új műszakokat egy adott munkához, valamint megtekintheti a meglévő műszakokat.

Funkciók:

- Munka adatainak megtekintése (név, helyszín, fizetés, kategória)

- Műszakok listázása
- Új műszak hozzáadása dátum/idő megadásával
- Lapozás a műszakok között

Használat:

Munka részleteinek megtekintése

1. A kiválasztott munka részletei megjelennek, beleértve:
 - A munka címét
 - Az ügynök nevét
 - A fizetést
 - A várost és a pontos címet
 - A kategóriát
2. Az oldal tetején **Vissza a munkákhoz** gombbal visszatérhet a munkák listájához.

Műszakok listázása

1. A már hozzáadott műszakok listája megjelenik az oldalon.
2. A műszakok egy **ShiftCard** komponensben láthatók, tartalmazzák:
 - A műszak címét
 - A kezdési és befejezési időpontokat
 - Törlés gomb (e-mailt küld minden érintett diáknak törlés esetén)

Új műszak hozzáadása

1. Az **Új műszak hozzáadása** űrlapon meg kell adni:
 - **Műszak kezdete:** dátum és idő
 - **Műszak vége:** dátum és idő
2. Nyomja meg a „**Műszak hozzáadása**” gombot.
3. Sikeres művelet esetén megjelenik egy visszajelzés.
4. Az új műszak automatikusan hozzáadódik a listához.

Műszakok lapozása

- Egy oldalon **4 műszak** jelenik meg.
- A „**Előző**” és „**Következő**” gombokkal lehet navigálni az oldalak között.

Hibakezelés és visszajelzések

Adatbetöltési problémák

- Ha a munkák vagy műszakok betöltése sikertelen, a konzolon egy hibaüzenet jelenik meg.
- Ellenőrizze az internetkapcsolatot és a szerver elérhetőségét.

Műszak hozzáadás sikertelen

- Ha a műszak időpontjai nem megfelelőek, egy hibaüzenet jelenik meg.
- Ellenőrizze, hogy a kezdési idő **korábbi**, mint a befejezési idő.
- A műszak hossza maximum 24 óra lehet.
- Nem lehet egy hétnél távolabbi időpontot magadni.
- Győződjön meg róla, hogy van jogosultsága műszak hozzáadására.

Jogosultsági problémák

- Ha nem rendelkezik megfelelő jogosultsággal, az oldal nem engedélyezi a műveleteket.
- Ellenőrizze, hogy be van-e jelentkezve, és a megfelelő token van-e elmentve a böngészőben.

Műszak jelentkezők

Műszak jelentkezők
 Műszak JELENTKEZÉSEK
 Tekintsd meg az elfogadásra váró jelentkezéseket!

Műszak kezdete: Összes:

Azonosító	Diák neve	Pozíció	Műszak kezdete	Műszak vége	Státusz	Műveletek
1	László Bánkúti	Árupalós	2025-03-31 14:00	2025-03-31 22:00	Válaszra vár	✓ ✕

1 to 1 of 1 Page 1 of 1

Beállítások
 Vissza a kezdőlapra
 Kijelentkezés

Ez a felület lehetővé teszi az ügynökök számára, hogy kezeljék a diákok műszak jelentkezéseit. A rendszerben megtekinthetik a beérkezett jelentkezéseket, azok állapotát, valamint elfogadhatják vagy elutasíthatják azokat.

Navigáció a műszak jelentkezők oldalra:

- A bal oldali **oldalsávból (Sidebar)** elérhető a „Műszak jelentkezők” menüpont.

- A felület betöltését követően a rendszer automatikusan megjeleníti az adott ügynökhöz tartozó jelentkezéseket.

Szűrési lehetőségek

A jobb oldali szűrőmezők segítségével lehet szűrni a jelentkezéseket:

- **Műszak kezdete:**
 - Egy legördülő menüből választható ki a releváns műszakkezdési időpont.
- **Státusz szerinti szűrés:**
 - „Összes” – Minden jelentkezés megjelenítése
 - „Válaszra vár” (0) – A még nem elbírált jelentkezések
 - „Elfogadva” (1) – Az elfogadott jelentkezések
 - „Elutasítva” (2) – Az elutasított jelentkezések

Ha egy szűrő kiválasztásra kerül, az adatok automatikusan frissülnek.

3. Műszak jelentkezések táblázat

A jelentkezések egy **Ag-Grid táblázatban** jelennek meg.

A táblázat oszlopai:

- **Azonosító** – A jelentkezés egyedi azonosítója
- **Diák neve** – A jelentkező diák neve
- **Pozíció** – Az adott állás megnevezése
- **Műszak kezdete** – A műszak indulási időpontja
- **Műszak vége** – A műszak befejezési időpontja
- **Státusz** – A jelentkezés aktuális állapota („Válaszra vár”, „Elfogadva”, „Elutasítva”)
- **Műveletek** – Gombok az elfogadáshoz vagy elutasításhoz

Jelentkezés elfogadása/elutasítása

Az egyes jelentkezések jobb oldalán található **„Műveletek” oszlopban** két gomb érhető el:

Elfogadás gomb:

- A zöld gombra kattintva egy megerősítő ablak jelenik meg:
„Biztosan elfogadod ezt a jelentkezést?”
- Ha az „Igen” opciót választjuk, a státusz **„Elfogadva”** lesz, és a diák értesítést kap.

Elutasítás gomb:

- A piros gombra kattintva egy megerősítő ablak jelenik meg: „Biztosan elutasítod ezt a jelentkezést?”
- Ha az „Igen” opciót választjuk, a státusz **„Elutasítva”** lesz.

Megjegyzés: Azok a jelentkezések, amelyek már el lettek fogadva vagy elutasítva, nem módosíthatók újra.

Munka jelentkezések

Jelentkezések
MUNKA JELENTKEZÉSEK
Tekintsd meg az elfogadásra váró jelentkezéseket!

Pozíció: Összes:

Azonosító	Diák neve	Pozíció	Iskolaszövetkezet	Státusz	Jelentkezés dátuma	Műveletek
1	László Bányai	Árupalás	MeloDiák Iskolaszövetkezet	Elfogadva	2025-03-28 17:57	

1 to 1 of 1 Page 1 of 1

Beállítások
Vissza a kezdőlapra
Kijelentkezés

Ez az alkalmazás lehetővé teszi a diákok jelentkezéseinek kezelését, amely az iskolaszövetkezetek és pozíciók elfogadását vagy elutasítását teszi lehetővé. Az alábbiakban bemutatjuk, hogyan használhatod az alkalmazás különböző funkcióit.

Kezdő képernyő

Miután belépsz az alkalmazásba, a fő képernyőn megjelenik az összes jelentkezés, amely az általad hozzáférhető pozíciókhoz és státuszokhoz van rendelve. Az alkalmazás a következő fő elemekből áll:

- **Oldalsáv:** Bal oldalon található, amely segít navigálni az alkalmazás különböző részei között.
- **Szűrők:** A szűrők segítségével szűkítheted a jelentkezéseket pozíció vagy státusz alapján.
- **Jelentkezési lista:** Az elfogadásra váró jelentkezések megjelenítése, melyek közül egyeseket elfogadhatsz vagy elutasíthatsz.

Szűrők használata

A jelentkezések listáját szűrheted két szempont alapján:

- **Pozíció:** A legördülő menüből választhatsz egy adott pozíciót, hogy csak azokat a jelentkezéseket jelenítse meg, amelyek az adott pozícióhoz tartoznak.
- **Státusz:** Választhatsz a következő státuszok közül:
 - **Válaszra vár:** Azok a jelentkezések, amelyeket még nem fogadtak el vagy utasítottak el.
 - **Elfogadva:** A már elfogadott jelentkezések.
 - **Elutasítva:** Azok a jelentkezések, amelyek elutasításra kerültek.

A szűrők alkalmazásával könnyedén áttekintheted az összes várakozó, elfogadott vagy elutasított jelentkezést.

Jelentkezések kezelése

A jelentkezéseket kétféleképpen kezelheted: **Elfogadás** és **Elutasítás**.

- **Elfogadás:** Ha egy jelentkezést elfogadsz, kattints az „Elfogadás” gombra, amely a jelentkezést elfogadja, és az állapotot "Elfogadva" státuszra állítja.
- **Elutasítás:** Ha egy jelentkezést elutasítasz, kattints az „Elutasítás” gombra, amely a jelentkezést elutasítja, és az állapotot "Elutasítva" státuszra változtatja.

A megerősítő ablak biztosítja, hogy véletlenül se történjenek nem kívánt műveletek. A művelet előtt mindig meg kell erősíteni, hogy valóban elfogadod vagy elutasítod a jelentkezést.

Táblázatos nézet

A jelentkezéseket egy táblázatban láthatod, amely az alábbi adatokat tartalmazza:

- **Azonosító:** Az egyes jelentkezések egyedi azonosítója.
- **Diák neve:** A diák neve, aki a jelentkezést tette.
- **Pozíció:** Az adott pozíció, amire jelentkeztek.
- **Iskolaszövetség:** Az iskola vagy szövetség neve.
- **Státusz:** A jelentkezés aktuális státusza (Válaszra vár, Elfogadva, Elutasítva).
- **Jelentkezés dátuma:** A jelentkezés dátuma.

Az "Műveletek" oszlopban található az elfogadásra és elutasításra szolgáló gombok.

Diákok

Diákok
Tekintsd meg és módosítsd a diákok adatait!

Azonosító	Vezetéknév	Keresztnév	Email
4	László	Bányai	user@ex.com

1 to 1 of 1 Page 1 of 1

Az **AgentStudentList** komponens egy dinamikus diáklistát jelenít meg, amelyet az ügynökök számára a rendszer biztosít. A diákok adatai táblázatos formában jelennek meg az **AgGridReact** komponens segítségével. Az ügynökök képesek az adatokat böngészni és szükség esetén módosítani.

2. Funkciók

- **Oldalsó menü:** Az oldal bal oldalán található menü segít a navigálásban, és a kijelölt oldal tartalmát frissíti.
- **Diákok adatainak megjelenítése:** A rendszer az ügynök bejelentkezését követően egy API hívást végez a diákok adatainak lekérésére.
- **Paginalás:** A diákok adatainak megjelenítése oldalonként történik, az adatokat a **pagination** modul kezeli.
- **Dinamikus adatfrissítés:** Az adatok automatikusan frissülnek az oldal betöltésekor.

3. Felhasználói Felület Elemei

- **Oldalsó Menü (Sidebar):** A bal oldalon található menü, amely az ügynöki navigációt segíti. Az oldalsó menü az ablak szélességétől függően elrejtőzhet vagy kinyílhat.
- **Diákok Táblázata:** Az **AgGridReact** táblázatot használ a diákok adatainak megjelenítésére. Az oszlopok a következőket tartalmazzák:
 - **Azonosító:** A diák azonosítója.

- **Vezetéknév:** A diák vezetéknév.
- **Keresztnév:** A diák keresztnév.
- **Email:** A diák e-mail címe.

Közvetítői beállítások

Az **AgentSettings** komponens célja, hogy lehetővé tegye az ügynökök számára a profiljuk adatainak módosítását. Az oldalon található űrlapon keresztül a felhasználó változtathatja meg a nevét, e-mail címét és jelszavát.

Felhasználói Felület Elemei

- **Oldalsó Menü (Sidebar):** A bal oldali menü segíti a navigációt az ügynökök számára, biztosítva az egyes oldalak közötti váltást.
- **Beállítások Űrlap:** Az ügynökök a következő mezőkben módosíthatják profiljukat:
 - **Vezetéknév:** A felhasználó vezetéckneve.
 - **Keresztnév:** A felhasználó keresztnéve.
 - **Email:** A felhasználó e-mail címe.
 - **Új jelszó:** Az új jelszó, amelyet a felhasználó beállít.
 - **Új jelszó megegyeszer:** A jelszó megerősítése, amelynek egyeznie kell az előző mezővel.
- **Mentés gomb:** Az űrlap végén található gomb, amely lehetővé teszi az adatok mentését.

Használat

Profiladatok Módosítása

Miután belépett a rendszerbe, az ügynök a **Profil beállítások** oldalra navigálhat. Itt a következő adatokat módosíthatja:

- **Vezetéknév:** A vezetéknévet írja be az első mezőbe.
- **Keresztnév:** A keresztnévet a második mezőbe kell megadni.
- **Email:** Az új e-mail címét adhatja meg az email mezőben.
- **Új jelszó:** Az új jelszó megadásához töltsse ki az "Új jelszó" mezőt.
- **Új jelszó megegyeszer:** A jelszó megerősítéséhez írja be ugyanazt az új jelszót, amit az előző mezőbe írt.

A Jelszavak Ellenőrzése

Miután kitöltötte az összes mezőt, a rendszer ellenőrzi, hogy az új jelszavak megegyeznek-e. Ha a két jelszó nem egyezik, a felhasználó hibaüzenetet kap, és a módosításokat nem lehet elmenteni.

Adatok Mentése

A **Mentés** gombra kattintva a rendszer elküldi az adatokat a szervernek, amely frissíti a felhasználói adatokat. A mentés sikeres befejezéséről értesítést kap, ha minden rendben volt, vagy hibaüzenetet, ha valami nem stimmel.

Hibakezelés

- **Jelszavak nem egyeznek:** Ha az új jelszó és a jelszó megerősítése nem egyezik, a rendszer hibaüzenetet küld, és nem engedi a mentést.
- **Token hiányzik:** Ha a rendszer nem talál érvényes token, hibát jelez a felhasználó számára.
- **API hiba:** Ha a backend API nem elérhető vagy hibát észlel, a felhasználó hibát kap.

Bejelentkezés iskolaszövetkezet profillal

Közvetítő felvétele

Ez az oldal lehetővé teszi, hogy új közvetítőket adjunk hozzá az alapadatok megadásával. Az adatok kitöltése után a rendszer automatikusan regisztrálja az új közvetítőt.

Űrlap Kitöltése

1. **Vezetéknév:** Add meg az új közvetítő vezetéknévét. A mező legalább két karakter hosszú, és csak betűket tartalmazhat. Ha a vezetéknévet nem megfelelően adod meg, a rendszer hibaüzenetet jelenít meg.
2. **Keresztnév:** Add meg az új közvetítő keresztnévét. A mező legalább két karakter hosszú, és csak betűket tartalmazhat. Ha a keresztnévet nem megfelelően adod meg, a rendszer hibaüzenetet jelenít meg.
3. **Közvetítő email címe:** Add meg az új közvetítő email címét. A mező érvényes email formátumot vár. Ha a megadott email cím nem helyes, a rendszer hibaüzenetet jelenít meg. Erre az email címre fog érkezni a fiókhoz generált jelszó.

Minden mező kitöltése kötelező.

Hibakezelés

- Ha bármelyik mezőt nem megfelelően töltöd ki, a rendszer az adott mező alatt hibaüzenetet jelenít meg.

- Az alábbi hibák esetén jelenhetnek meg üzenetek:
 - A keresztnév és vezetéknév nem tartalmazhatnak speciális karaktereket és legalább két karakter hosszúak kell legyenek.
 - Az email cím formátuma nem megfelelő.

Közvetítő Felvétele

A **Közvetítő felvétele** gombra kattintva az új közvetítő regisztrálásra kerül. A rendszer az alábbiakat ellenőrzi a beküldött adatok előtt:

1. A mezőkben található értékek validálása, hogy megfelelnek-e a követelményeknek.
2. Amennyiben a validálás sikeres, az adatokat a backend API felé küldi, és egy visszajelzést ad a felhasználónak a sikeres vagy sikertelen regisztrációról.

Visszajelzések

- Sikeres regisztráció után a felhasználó egy zöld siker üzenetet kap a "Sikeres regisztráció!" üzenettel.
- Ha bármilyen hiba történik a regisztráció során (pl. email cím már létezik, vagy egyéb rendszerhiba), akkor egy piros hibaüzenet jelenik meg a megfelelő hibaüzenettel.

Meglévő közvetítők

Meglévő közvetítők
Tekintsd meg és kezeld az aktuális közvetítőket!

Azonosító	Vezetéknév	Keresztnév	Email	Aktiv	Művelet
3	Fogarasi	Éva	fogarasi@ex.com	Igen	

1 to 1 of 1 Page 1 of 1

Ez a komponens lehetővé teszi a közvetítők kezelését a rendszerben, például a közvetítők státuszának módosítását és a hozzájuk tartozó műveletek elvégzését. A közvetítők listáját egy táblázatban jeleníti meg, amely a következő adatokat tartalmazza:

- Azonosító
- Vezetéknév
- Keresztnév
- Email cím
- Aktivitás státusza (Aktív/Nem aktív)

Funkcionalitás:

1. Adatok betöltése:

- A `fetchAgents` függvény az API-n keresztül lekéri az összes közvetítőt a rendszerből, és megjeleníti azokat a táblázatban.

2. Közvetítők státuszának módosítása:

- A közvetítők státuszának változtatása egy megerősítő párbeszédpanel segítségével történik, amelyet a `confirmStatusChange` függvény indít el.
- A státuszt a `toggleAgentStatus` függvény frissíti az API hívásával.

3. Műveletek (Pl. Jelszó módosítása):

- A "Műveletek" oszlopban található gombok segítségével az adminisztrátorok módosíthatják a közvetítő státuszát vagy elérhetik a jelszó módosító modált.
- A jelszó módosító modal a `OrgPasswordModal` komponenssel történik.

4. Táblázat működése:

- A közvetítők adatai az `ag-grid-react` komponens segítségével jelennek meg. A táblázat támogatja a lapozást és a reszponzív viselkedést.
- Az oszlopok egyedileg konfigurálhatók, mint például az "Aktív" státusz, amely szöveges formátumban jeleníti meg az értéket (Igen/Nem).

Használati útmutató:

1. Közvetítők listájának megtekintése:

Miután beléptél a rendszerbe, az admin felületén keresd meg a "Meglévő közvetítők" oldalt, ahol a közvetítők listáját láthatod.

2. Közvetítő státuszának módosítása:

- A státusz változtatásához kattints a megfelelő "Aktiválás" vagy "Deaktiválás" gombra, amely az "Műveletek" oszlopban található.
- A rendszer kérni fog egy megerősítést, hogy biztosan szeretnéd módosítani a közvetítő státuszát.
- Miután megerősítetted, az új státusz azonnal frissül, és a táblázatban is tükröződik.

3. Jelszó módosítása:

- Kattints a "Kulcs" ikonnal jelölt gombra a kívánt közvetítő mellett, hogy megnyisd a jelszó módosító modalt.
- A modalban lehetőség lesz a közvetítő jelszavának módosítására.

Munka létrehozása

Ez a funkció lehetővé teszi, hogy a felhasználók új munkákat hozzanak létre az adminisztrációs felületen. Az új munka adatainak kitöltése után a felhasználó könnyedén regisztrálhatja a munkahelyet a rendszerben.

Funkciók:

- Munka neve
- Órabér
- Kategória választás
- Város és cím
- Munkaköri leírások
- Különleges elvárások és előnyök megadása

Lépésről lépésre használat:

1. Munka neve (Title):

- Írja be a munka nevét, amely legfeljebb 84 karakter hosszú lehet.
- A munka neve csak betűket és szóközöket tartalmazhat.
- Hibaüzenet jelenik meg, ha nem érvényes karaktereket ad meg.

2. Kategória kiválasztása:

- Válassza ki a megfelelő kategóriát a legördülő menüből.
- Ha nincs kategória kiválasztva, hibaüzenet jelenik meg, amely figyelmezteti a felhasználót.

3. Órabér megadása:

- Adja meg az órabért egy számjegyben (pozitív szám).
- Ha az órabér 0 vagy negatív érték, hibaüzenet jelenik meg.

4. Város és cím megadása:

- Adja meg a munkahely városát és címét.
- A cím mező kitöltése nem kötelező, de a város megadása szükséges.

5. Leírás és Elvárások:

- A rendszer két szakaszban kér be adatokat a munkával kapcsolatban:
 - **Amit kínálunk:** Írja le a munkavégzésről szóló ajánlatot.
 - **Főbb feladatok:** Írja le a munka főbb feladatait.
 - **Álláshoz tartozó elvárások:** Az elvárások, amelyeket az új munkavállalóval szemben támasztanak.
 - **Előnyt jelent:** Olyan előnyöket soroljon fel, amelyek a jelölt számára előnyösek, de nem kötelezőek.

6. Formátum ellenőrzés és hibák:

- A rendszer ellenőrzi, hogy a kitöltött mezők megfelelnek-e az előírt feltételeknek.
- Ha egy mező érvénytelen adatot tartalmaz (pl. nem megfelelő karakterek), akkor egy hibaüzenet jelenik meg.

7. Munka regisztrálása:

- Miután minden mezőt kitöltött, kattintson a **Munka létrehozása** gombra.
- A rendszer ellenőrzi a formát és a szükséges mezők kitöltését.
- Sikeres regisztráció után értesítés jelenik meg, és a mezők automatikusan visszaállnak alapértelmezettre.

Hasznos tippek:

- **Visszajelzések:** A rendszer azonnal visszajelzést ad, ha valamelyik mező nincs megfelelően kitöltve.
- **Többszöri regisztráció:** Az űrlap elküldése után ismét elérhetőek a mezők, és új adatokat is hozzáadhat.

Hibaüzenetek:

- "A munka neve csak betűket és szóközöket tartalmazhat!"
- "A város neve csak betűket és szóközöket tartalmazhat!"
- "A munka neve megadása kötelező."
- "Az órabér nem lehet 0 vagy negatív szám."
- "Kategória kiválasztása kötelező."
- "Felhasználó nincs hitelesítve." (Ha a felhasználó nincs bejelentkezve)

Rendszerkövetelmények:

- A munka létrehozásához szükséges, hogy a felhasználó rendelkezzen érvényes hitelesítési tokennel, amely a felhasználói fiók bejelentkezése után elérhető.

Létrehozott munkák

Létrehozott munkák
LÉTREHOZOTT MUNKÁKAT
Tekintsd meg és kezeld a létrehozott munkákat!

Azonosító	Pozíció	Kategória	Helyszín	Cím	Órabér	Aktív	Műveletek
1	Árúpakolás	Fizikai, gyári, raktári	Debrecen	Határ út, 44.	1960	Igen	

1 to 1 of 1 Page 1 of 1

A CurrentJobs komponens a "Létrehozott munkák" kezelésére szolgál, ahol a felhasználók megtekinthetik, módosíthatják és kezelhetik az aktuális munkákat. A komponens lehetővé teszi a munkák állapotának megváltoztatását, a munkák részletes megtekintését, valamint az adott munka módosítását.

Funkciók

- **Munkák megjelenítése:** A rendszer lekéri és megjeleníti az összes aktív munkát.
- **Munka státuszának módosítása:** A felhasználók módosíthatják egy munka státuszát (aktív/inaktív).
- **Munka részletek megtekintése:** A felhasználók megtekinthetik egy munka részletes információit.

- **Munka módosítása:** A felhasználók módosíthatják a munka adatait, például cím, kategória stb.
- **Közvetítő kijelölése:** Be lehet osztani egy közvetítőt a munkára, ezután elérhető lesz, illetve a közvetítő tudja kezelni a műszakokat és jelentkezéseket.

Komponens részletes ismertetése

1. useEffect és adatlekérés

- A komponens betöltődésekor az useEffect hook segítségével meghívja a fetchJobs függvényt, amely lekéri az aktuális munkákat a backend API-ból.
- A munkák JSON formátumban érkeznek, és a komponens tárolja azokat az activeJobs állapotban.

2. Adatok megjelenítése AG Grid táblázatban

- A munkák listája az AgGridReact komponenst használva kerül megjelenítésre.
- Az AG Grid táblázat oszlopai dinamikusan vannak meghatározva a columnDefs változóban.
- Az oszlopok a következő adatokat jelenítik meg:
 - Azonosító (id)
 - Pozíció (title)
 - Kategória (categoryName)
 - Helyszín (city)
 - Cím (address)
 - Órabér (hourlyRate)
 - Aktív (isActive)

3. Műveletek oszlop

- Az utolsó oszlop a különböző műveletekhez tartozik, mint például:
 - **Munka megtekintése:** Az adott munka részletes információinak megjelenítése egy modális ablakban.
 - **Munka módosítása:** A munka adatait módosító felület megjelenítése.
 - **Státusz módosítása:** A munka aktív státuszának változtatása (aktív/inaktív).
- A műveletekhez tartozó gombok képek formájában jelennek meg (pl. view.png, briefcase.png, onbutton.png, offbutton.png).

4. Dialogusok és modális ablakok

- A komponens tartalmaz egy Dialog komponenst, amely egy modális ablakot kezel. Ez az ablak akkor jelenik meg, amikor a felhasználó valamelyik műveletet kiválasztja (pl. munka részletes megtekintése vagy módosítása).
- A JobViewModal és JobPatchViewModal modális ablakok jelennek meg a munka részletes információinak megtekintése vagy módosítása céljából.

5. Státusz módosításának megerősítése

- A munka státuszának módosítása előtt a komponens egy megerősítő párbeszédablakot (confirm dialog) jelenít meg a felhasználónak. Ez megerősíti a státuszváltás szándékát.
- A megerősítés a react-confirm-alert csomagot használja a párbeszédablak megjelenítésére.

6. Adatok frissítése

- Miután a felhasználó módosította a munka státuszát vagy más adatokat, a fetchJobs függvény újra lekéri az összes munkát és frissíti a táblázatot.

Kezelési műveletek

- **Munka megtekintése:**

- Kattints a "View" gombra a műveletek oszlopban.
- Ez megnyitja a munka részletes információit egy modális ablakban, ahol tudod azokat módosítani.

- **Munka módosítása:**

- Kattints a "Briefcase" gombra.
- Ekkor megjelenik egy új felület, amelyen közvetítőt lehet kijelölni egy adott munkához. A munka csak ez után fog megjelenni a diákok számára.

- **Munka státuszának módosítása:**

- Kattints a státuszt mutató gombra.
- A rendszer megerősíti, hogy biztosan szeretnéd módosítani a munka státuszát.

Iskolaszövetkezeti beállítások

A "Szervezet Beállítások" oldal lehetővé teszi a felhasználók számára, hogy módosítsák a szervezetük profiladatokat, például az e-mail címet, jelszót, cégnév, cím, és kapcsolatfelvételi információkat. Ezen az oldalon történik az adatváltoztatás és azok mentése.

Funkciók

A beállítások módosításához a felhasználónak lehetősége van:

- **E-mail cím frissítése**
- **Új jelszó beállítása**
- **Cég megnevezésének módosítása**
- **Cég címének módosítása**
- **Kapcsolattartó e-mail címének és telefonszámának frissítése**

Oldal felépítése

- **Sidebar:** Az oldal bal oldali sávja, amely a menüpontokat és navigációt tartalmazza. A menü linkek a szervezeti beállításokhoz vezetnek.
- **Beállítási űrlap:** A központi részben található űrlap a különböző beállítások megadására szolgál:
 - **E-mail cím:** A felhasználó e-mail címének módosítása.
 - **Új jelszó:** Az új jelszó megadása.
 - **Jelszó megerősítése:** Az új jelszó megerősítése a biztonság érdekében.
 - **Cég megnevezése:** A szervezet neve, amelyet frissíthet a felhasználó.
 - **Cím:** A cég hivatalos címe.

- **Kapcsolattartó e-mail cím:** A cég hivatalos e-mail címe.
- **Kapcsolattartó telefonszám:** A cég hivatalos telefonszáma.
- **Mentés gomb:** Az adatok mentésére szolgál. **Adatbevitel és validálás**
- **E-mail címek validálása:** Az e-mail címek formátuma érvényes kell, hogy legyen (valami@domain.com).
- **Jelszó validálása:** Az új jelszónak legalább egy nagybetűt és egy számot kell tartalmaznia, és minimum 8 karakter hosszúnak kell lennie.
- **Cégnév validálása:** A cégnév csak betűket és szóközöket tartalmazhat, és maximum 64 karakter hosszú lehet.
- **Telefonszám validálása:** A telefonszámnak pontosan 11 számjegyből kell állnia.

Hogyan használjuk?

1. **Nyissa meg az oldal beállításait:** Navigáljon a "Szervezet beállítások" oldalra a navigációs menüből.
2. **Módosítsa a szükséges adatokat:**
 - **E-mail cím:** Adja meg az új e-mail címet.
 - **Jelszó:** Ha új jelszót szeretne beállítani, írja be az új jelszót, és erősítse meg.
 - **Cégnév:** Adja meg a cég nevét.
 - **Cím:** Frissítse a cég hivatalos címét.
 - **Kapcsolattartó e-mail és telefonszám:** Adja meg a kapcsolattartó e-mail címet és telefonszámot.
3. **Mentse el a módosításokat:** A módosítások elmentéséhez kattintson a "Mentés" gombra. Ha minden adat érvényes, a rendszer frissíti a szervezet profilját.

Hibakezelés és figyelmeztetések

- Ha a jelszavak nem egyeznek, vagy a jelszó nem felel meg az előírt kritériumoknak, a rendszer hibaüzenetet jelenít meg.
- Ha az e-mail cím vagy a telefonszám nem felel meg a formátumnak, hibaüzenet jelenik meg.
- Ha bármelyik mező érvénytelen adatot tartalmaz, a felhasználót figyelmezteti a rendszer.

Visszajelzés és értesítések

- A módosítások sikeres mentése után egy értesítés jelenik meg, amely megerősíti a frissítéseket.

- Hibás adatbevitel esetén hibaüzenet jelenik meg a felhasználó számára, amely tájékoztatja őt a problémáról.

Bejelentkezés adminisztrátori profillal

Szövetkezet felvétele

A **Szövetkezet Felvétele** oldal egy adminisztrátori felületet biztosít új szövetkezetek létrehozására. Az oldal lehetővé teszi az adminisztrátorok számára, hogy szövetkezetek alapvető adatait rögzítsék.

Űrlap Beviteli Mezők

Az alábbi adatokat kell megadni az új szövetkezet hozzáadásához:

- **Szövetkezet Neve:** A szövetkezet hivatalos neve (max. 50 karakter). Csak betűk és szóközők megengedettek.
- **Email Cím:** A szövetkezet hivatalos email címe. Érvényes email formátumot kell megadni.
- **Telefonszám:** A szövetkezet hivatalos telefonszáma. Az elfogadott formátum 10-15 számjegy, opcionálisan '+' előtaggal.
- **Szövetkezet Címe:** A szövetkezet hivatalos címe (max. 100 karakter). A mező nem lehet üres.

Form Validálás

Mielőtt az űrlapot el lehet küldeni, a rendszer validálja a megadott adatokat. Az alábbiak szerint történik a validálás:

- **Szövetkezet neve:** Meg kell adni a szövetkezet nevét, amely maximum 50 karakter hosszú lehet, és csak betűket és szóközőket tartalmazhat.
- **Email:** A megadott email címnek érvényes formátumban kell lennie (pl. user@domain.com).
- **Telefonszám:** A telefonszámnak 10-15 számjegy hosszúnak kell lennie, és opcionálisan tartalmazhat '+' előtagot.
- **Cím:** A cím nem lehet üres, és maximum 100 karakter hosszú lehet.

Ha bármelyik mező nem felel meg a követelményeknek, a rendszer hibaüzenetet jelenít meg.

Új Szövetkezet Hozzáadása

Miután minden adatot helyesen kitöltöttek, az "**Szövetkezet felvétele**" gomb segítségével lehet véglegesíteni az új szövetkezet hozzáadását. A sikeres felvétel esetén egy visszajelző üzenet jelenik meg a felhasználó számára. A megadott email címre megerősítő üzenetet küldünk a profilhoz tartozó jelszóval.

Hibák Kezelése

Ha valamilyen hiba történik (például érvénytelen adatbevitel vagy hálózati hiba), akkor a rendszer hibaüzenetet jelenít meg a felhasználó számára a képernyőn. A hibaüzenetek segítenek a felhasználónak a problémák azonosításában és azok kijavításában.

Visszajelzések és Értesítések

A rendszer a **react-toastify** könyvtárat használja értesítések megjelenítésére, amelyek tájékoztatják a felhasználót az adatbevitel állapotáról. A következő értesítések jelenhetnek meg:

- **Sikeres felvétel:** Amikor a szövetkezet sikeresen hozzáadódik.
- **Hibás adatok:** Ha bármelyik adatmező nem felel meg az elvárásoknak (például érvénytelen email cím vagy telefonszám).

Meglévő szervezetek

Meglévő szervezetek
MEGLÉVŐ SZÖVETKEZETEK
Tekintsd meg és módosítsd a szervezetek adatait!

Azonosító	Név	Cím	Email	Telefonszám	Létrehozás	Műveletek
2	Melődiák Iskolaszövetke...	Debrecen, Piac utca 22.	melodiak@ex.com	+36705357145	2025-03-28	

Page Size: 1 to 1 of 1 Page 1 of 1

Az ExistingOrg komponens lehetővé teszi az adminisztrátorok számára, hogy megtekinthessék és módosíthassák a rendszerben szereplő meglévő szervezeteket. Az alkalmazás lekéri és megjeleníti a szervezetek adatait, mint például a nevüket, kapcsolattartóikat és a létrehozás dátumát. Emellett lehetőséget biztosít a szervezetek adatainak megtekintésére és a jelszavak módosítására.

Funkciók

1. Szervezetek listája:

- Az alkalmazás betölti és megjeleníti az összes szervezetet az adatbázisból, beleértve az alábbi adatokat:
 - Azonosító
 - Név
 - Cím
 - Email
 - Telefonszám
 - Létrehozás dátuma

2. Műveletek:

- Az adminisztrátorok minden egyes szervezethez két műveletet végezhetnek el:

- **Szövetkezet megtekintése és módosítása:** Részletes információk megtekintésére szolgáló ablak jelenik meg, ahol módosítani is tudja azokat.
- **Jelszó módosítása:** Az adminisztrátorok módosíthatják a szövetkezetek jelszavát.

3. Részletes nézet:

- A "Szövetkezet megtekintése" gombra kattintva egy új ablak nyílik, amelyben az adott szövetkezet részletes adatai láthatók.
- A szövetkezet adatai közé tartozik a neve, címe, kapcsolattartó emailje, telefonszáma és egyéb releváns információk.

4. Jelszó módosítás:

- A "Jelszó módosítása" gombra kattintva egy új ablak jelenik meg, amely lehetővé teszi a szövetkezet jelszavának megváltoztatását.

Használat

1. Szövetkezetek megtekintése:

- Az adminisztrátorok belépnek az alkalmazásba és a "Meglévő szövetkezetek" oldalra navigálnak.
- Itt egy táblázatban megjelennek az összes szövetkezet adatai.
- A táblázat oszlopai tartalmazzák az azonosítót, a nevet, a címet, a kapcsolattartó emailjét és telefonszámát, valamint a szövetkezet létrehozásának dátumát.
- Az adminisztrátorok bármelyik szövetkezetről választhatják a "Szövetkezet megtekintése" vagy a "Jelszó módosítása" műveleteket.

2. Szövetkezet adatainak megtekintése és módosítása:

- A "Szövetkezet megtekintése" gomb lenyomásával egy modális ablak jelenik meg, amely tartalmazza a szövetkezet részletes adatait.
- Itt az adminisztrátorok áttekinthetik és módosíthatják a szövetkezet minden fontos adatát.

3. Szövetkezet jelszavának módosítása:

- A "Jelszó módosítása" gomb lenyomásával egy másik modális ablak jelenik meg, amely lehetőséget biztosít a szövetkezet jelszavának megváltoztatására.
- Az új jelszó megadása után az adminisztrátorok megerősíthetik a módosítást.

Hibakezelés

- Ha a szövetkezetek adatainak lekérése során hiba lép fel, a rendszer automatikusan megjeleníti a hibaüzenetet.
- Ha a jelszó módosítása során probléma merül fel, a rendszer figyelmeztetést ad, és nem hajtja végre a műveletet.

Adminisztrátori beállítások

Az AdminSettings oldal lehetőséget biztosít az adminisztrátorok számára, hogy módosítsák saját profiljuk adatait, beleértve az email címet és a jelszót. Az oldal tartalmazza a beállítások frissítésére szolgáló űrlapot, ahol az adminisztrátorok megadhatják az új adatokat.

Funkciók

1. Profil beállítások módosítása:

- Az adminisztrátorok módosíthatják az email címüket és a jelszavukat.
- Az űrlap három mezőt tartalmaz:
 - **Email:** Az adminisztrátor új email címét adhatja meg.
 - **Új jelszó:** Az adminisztrátor új jelszót állíthat be.
 - **Új jelszó megegyezően:** A jelszó megerősítése szükséges a biztonság érdekében.

2. Validálás és Hibakezelés:

- Ha az email cím és a jelszavak nem felelnek meg a formátumnak vagy a két jelszó nem egyezik, a rendszer hibát jelez.
- Ha a felhasználói azonosítás sikertelen, az adminisztrátor egy hibaüzenetet kap.

3. Adatok mentése:

- A módosított adatok mentése az axios.put HTTP kérés segítségével történik.
- A kérés az adminisztrátor jelszavát és email címét küldi el a backend szervernek frissítés céljából.

Használat

1. Profil adatainak módosítása:

- Az adminisztrátorok az oldalon található **email**, **új jelszó** és **új jelszó megegyeszer** mezőkben tudják megadni az új adatokat.
- A mezők kitöltése után a "Mentés" gombra kattintva az új adatokat elmenthetik.

2. Jelszó megerősítése:

- A jelszó módosításakor a felhasználónak meg kell adnia kétszer az új jelszót: egyszer az **Új jelszó** mezőben, majd újra a **Új jelszó megegyeszer** mezőben.
- Ha a két jelszó nem egyezik, hibaüzenet jelenik meg, és a mentés nem történik meg.

3. Sikeres adatmentés:

- Amennyiben az új adatok sikeresen elmentésre kerülnek, a felhasználó egy sikeres mentésről értesítő üzenetet kap.
- Ha hiba történik (pl. jelszóval kapcsolatos problémák vagy szerver hiba), a rendszer hibaüzenetet küld.

Hibakezelés

- Ha az adminisztrátor nem tudja azonosítani a felhasználót a token alapján, vagy ha a jelszavak nem egyeznek, a rendszer hibát jelez, és nem hajtja végre a frissítést.
- Az alkalmazás a hibákat a toast.error függvény segítségével jeleníti meg.

Javasolt tesztelési sorrend

1. Adminisztrátori profil létrehozása a WPF alkalmazásban.
2. Bejelentkezés a weboldalon a profilba, új szervezet létrehozása valós email címmel a jelszó kiküldése érdekében (megváltoztatható a Meglévő szervezetek oldalon, tempmail használata javasolt).
3. Adminisztrátori funkciók tesztelése.
4. Belépés a létrehozott iskolaszövetkezeti profilba.

5. Új közvetítő felvétele, szintén valós email címmel a jelszó kiküldéséhez (megváltoztatható a Meglévő közvetítők oldalon, tempmail használata javasolt).
6. Új munka létrehozása.
7. Létrehozott munkák oldalon hozzárendelni egy közvetítőt a hozzáadott munkához.
8. Bejelentkezés a létrehozott közvetítői profilba.
9. Új műszak felvétele valamelyik munkához.
10. Belépés egy diák profilba, jelentkezés az elérhető munkák közül valamelyikre.
11. A közvetítői profillal elfogadni a jelentkezését a diáknak a munkára.
12. A diák profillal jelentkezni egy műszakra, majd a közvetítői profillal elfogadni/elutasítani azt.

Ezekkel a lépésekkel az oldal funkcióinak a többségét könnyedén tesztelheted.

Asztali alkalmazás fejlesztői dokumentáció

A fejlesztői környezetről

A fejlesztői környezet több technológiát ötvöz annak érdekében, hogy hatékony, modern és skálázható alkalmazásokat lehessen létrehozni.

Az asztali alkalmazás fejlesztése C# és WPF .NET 8 környezetben történik, amely modern és rezponzív felhasználói élményt biztosít. A felület kialakításához Material Design alapú csomagok kerülnek felhasználásra, mint például a MaterialDesignColors, MaterialDesignThemes és MaterialDesignThemes.MahApps. Az adatok kezeléséhez a Newtonsoft.Json csomag nyújt támogatást.

A fejlesztői eszközök között a Visual Studio 2022 biztosítja az optimális munkakörnyezetet. Az alkalmazásstruktúra és a csomagválasztás célja egy jól optimalizált, biztonságos és könnyen bővíthető rendszer kialakítása.

Miért C# WPF az asztali alkalmazáshoz?

Komponens alapú architektúra

A WPF támogatja az újrahasználatos UI-elemek és vezérlők létrehozását, amelyek csökkentik a kódbázis redundanciáját és megkönnyítik a karbantartást. Az MVVM (Model-View-

ViewModel) tervezési minta segítségével a felhasználói felület és az üzleti logika tisztán elválasztható, így az alkalmazás könnyen bővíthető és testre szabható.

Gyors és reszponzív felület

A WPF kihasználja a hardveres gyorsítást és az XAML-alapú UI-t, amely gördülékeny és hatékony felhasználói élményt biztosít. Az adatközpontú vezérlők és az optimalizált binding mechanizmus lehetővé teszi a dinamikus tartalmak gyors megjelenítését és frissítését, ami különösen fontos egy adminisztrációs alkalmazásnál, ahol nagymennyiségű adat kezelésére van szükség.

Erős típusbiztonság és stabilitás

A C# erős típusellenőrzéssel és kifinomult objektumorientált megközelítéssel rendelkezik, amely csökkenti a futásidejű hibákat és növeli az alkalmazás stabilitását. Az aszinkron programozási lehetőségek (async/await) pedig hatékony adatkezelést és felhasználói interakciót biztosítanak.

Gazdag ökoszisztéma és testreszabhatóság

A WPF számos beépített és külső vezérlőt kínál, amelyekkel modern és esztétikus felhasználói felületek alakíthatók ki. A Material Design témák és komponensek (MaterialDesignColors, MaterialDesignThemes, MahApps) segítségével az alkalmazás modern és intuitív kinézetet kap.

Optimalizált teljesítmény és natív Windows integráció

A WPF közvetlenül a Windows környezethez illeszkedik, így kihasználja az operációs rendszer funkcióit, például a natív API-khoz való hozzáférést, a beépített biztonsági mechanizmusokat és a Windows-specifikus interakciókat.

Fejlett fejlesztői eszközök és hatékony fejlesztési folyamat

A Visual Studio 2022 erőteljes fejlesztői eszközkészletet biztosít, amely támogatja az élő XAML előnézetet, a hatékony hibakeresést és a beépített teljesítményoptimalizálási eszközöket. Az

optimalizált fejlesztői élmény lehetővé teszi a gyors iterációt és a hatékony fejlesztési folyamatokat.

WPF architektúra

Általános felépítés

Az asztali alkalmazás egy C# és WPF alapú alkalmazás, amely az ASP.NET Core backend API-val kommunikál. A fejlesztés során az MVVM (Model-View-ViewModel) architektúra biztosítja a logikai rétegek tiszta elválasztását, elősegítve a karbantarthatóságot és a skálázhatóságot.

Főbb technológiák

- **WPF** → Modern, XAML-alapú UI-fejlesztés és responszív asztali alkalmazás
- **C#** → Erős típusbiztonság és objektumorientált fejlesztés
- **HttpClient** → Backend API-val való kommunikáció

Program főbb algoritmusai

Regisztrációs algoritmus

A regisztrációs folyamat az asztali alkalmazásban a következőképpen történik:

1. E-mail cím és jelszó ellenőrzése:

- Az alkalmazás először ellenőrzi, hogy a felhasználó által megadott e-mail cím nem üres és a helyes formátumban van-e.
- A jelszót is ellenőrzi, hogy nem üres, mielőtt a további lépések megtörténnének.

2. Új felhasználói adat létrehozása:

- Ha mindkét mező érvényes, a felhasználó adatainak (keresztnév, vezetéknév, e-mail cím, jelszó) elküldése a backend API felé történik.
- Az API végpont POST /api/auth/register-admin-et használja, amelyhez JSON formátumban küldjük el az adatokat.

3. Backend API válasza:

- A backend a regisztrációs kérés feldolgozása után válaszol. Ha a regisztráció sikeres, a felhasználó értesítést kap arról, hogy a regisztráció sikeres volt, és az ablak bezárul.
- Ha valamilyen hiba történik (pl. már létező e-mail cím vagy más validációs hiba), akkor egy hibaüzenetet kap a felhasználó.

Bejelentkezési algoritmus

A bejelentkezési folyamat az asztali alkalmazásban az alábbi lépésekből áll:

1. Felhasználói adatok lekérése és validálása:

- Az alkalmazás bekéri a felhasználó e-mail címét és jelszavát a felhasználói felületről.
- Ezeket az adatokat JSON formátumban küldi el a backend API számára.

2. Hitelesítés az API-n keresztül:

- A backend az e-mail cím és jelszó alapján ellenőrzi a felhasználó adatait.
- A jelszó bcrypt hash-elte értéke kerül összehasonlításra a megadott jelszóval.

3. JWT token generálása:

- Sikeres bejelentkezés esetén a backend válaszában visszaküldi a JWT token, amely tartalmazza a felhasználó azonosítóját (userId) és szerepkörét (role).
- Az alkalmazás a választ dinamikusan deszerializálja, és a JWT token alapján beállítja a felhasználó szerepkörét.

4. Szerepkör ellenőrzése:

- Az alkalmazás ellenőrzi, hogy a felhasználó szerepköre "admin"-e.
- Ha igen, a felhasználó átirányításra kerül az admin felületre, és a bejelentkezési ablak bezárul.
- Ha nem, a rendszer egy figyelmeztető üzenetet jelenít meg, miszerint nincs jogosultság a belépéshez.

Felhasználó státuszának változtatása

A felhasználói státusz módosítása az alkalmazásban a következő lépésekből áll:

1. Jogosultság ellenőrzése (isAdmin):

- Az adminisztrátor jogosultságát az **isAdmin()** metódus ellenőrzi.
- Ha a felhasználó szerepköre nem "admin", akkor a rendszer egy figyelmeztető üzenetet jelenít meg, és a státusz módosítása nem történik meg.
- Ha a felhasználó jogosultsága megfelelő, akkor a rendszer folytatja a státusz módosítását.

2. Felhasználó kijelölése:

- A felhasználó státuszának módosításához először ki kell választani egy felhasználót a **UserListBox** listából.
- Ha nincs kijelölve felhasználó, akkor a rendszer hibaüzenetet jelenít meg.

3. A státusz módosításának végrehajtása:

- Az alkalmazás a **ToggleUserStatusAsync()** aszinkron metódust hívja meg, amely az API szolgáltatás segítségével végrehajtja a státusz módosítását.
- A módosítást követően egy üzenet jelenik meg a felhasználó számára, amely tájékoztatja őt a sikeres státusz frissítésről.

4. Hiba kezelés:

- Ha bármilyen hiba történik a státusz módosítása közben (pl. API hiba), akkor a rendszer a **ShowError()** metódus segítségével megjeleníti a hibaüzenetet a felhasználó számára.

Felhasználói adatok módosítása

A felhasználói adat módosítása az alkalmazásban a következő lépésekből áll:

1. Felhasználói adatok betöltése:

- A UserModifyPage oldal betöltésekor a kiválasztott felhasználó adatainak betöltése történik.
- Az adatokat a konstruktor fogadja el, amely az adott felhasználó objektumát kapja paraméterül. Az adatok betöltése a UI elemekbe történik, például a **FirstNameInput**, **LastNameInput**, **EmailInput** mezőkbe.

2. Jogosultság ellenőrzése:

- A **UpdateProfile()** metódus hívása előtt az alkalmazás ellenőrzi, hogy a bejelentkezett felhasználó rendelkezik-e adminisztrátori jogosultsággal.
- Ha a felhasználó nem rendelkezik megfelelő jogosultsággal (nem admin), akkor a rendszer figyelmeztetést jelenít meg, és nem folytatja a műveletet.

3. Felhasználói adatok frissítése:

- Ha az adminisztrátor jogosultságot kapott, akkor az alkalmazás az **ApiService.UpdateUserProfileAsync()** metódust hívja meg, amely az API-n keresztül frissíti a felhasználó adatait.
- Az új adatokat az UI elemekből (pl. név, email cím) gyűjti össze, majd a szerverre küldi.

4. API válaszána kezelésé:

- Ha az API válasza sikeres, akkor a rendszer értesíti a felhasználót, hogy a profil sikeresen frissítve lett.
- Ha a válasz hiba, akkor a rendszer hibaüzenetet jelenít meg.

5. Hiba kezelés:

- Ha bármilyen hiba történik a hálózati kérés során, akkor a rendszer egy hálózati hibát jelez a felhasználónak a **MessageBox** segítségével.

6. Módosítások visszavonása:

- A felhasználó a **Cancel_Click()** metódus segítségével visszavonhatja a módosításokat. A rendszer megjelenít egy megerősítő ablakot, hogy biztosan be akarja-e zárni az oldalt. Ha igen, az oldal bezárul, és a módosítások nem kerülnek végrehajtásra.

Program fájlok ismertetése

Models mappa

A **Models** mappa tartalmazza az alkalmazás adatmodelljeit, amelyek az API-ból érkező adatok reprezentációjára és az alkalmazáson belüli adatkezelésre szolgálnak.

SessionManager osztály

Elérési út: StudentHiveWpf.Services.SessionManager

Leírás:

Ez egy statikus osztály, amely az aktuális bejelentkezett felhasználó szerepkörét (Role) tárolja a munkamenet során.

Mivel a WPF alkalmazás nem tárolja el a JWT token, ezért a felhasználó jogosultságát a Role változó segítségével kezeljük.

Tulajdonságok:

- Role (*string*) – Az aktuális felhasználó szerepköre, amely alapértelmezés szerint üres (*string.Empty*).

Megjegyzés:

- Az alkalmazás bezárásával a munkamenet véget ér, és a Role értéke elveszik.
- Csak az adott munkamenet során tárolja az adatokat, nem marad meg az alkalmazás újraindítása után.

User osztály

Elérési út: StudentHiveWpf.Models.User

Leírás:

Ez az osztály a rendszerben lévő felhasználókat reprezentálja. Az API-ból érkező adatokat ebbe az osztályba deszerializáljuk, és ezzel dolgozunk az alkalmazáson belül.

Tulajdonságok:

- `Id (int)` – Az egyedi felhasználói azonosító.
- `OrganizationId (int?)` – A felhasználóhoz tartozó szervezet azonosítója (ha van).
- `RoleId (int?)` – A felhasználóhoz rendelt szerepkör azonosítója (pl. admin, diák, ügynök).
- `FirstName (string)` – A felhasználó keresztnéve.
- `LastName (string)` – A felhasználó vezetéknéve.
- `Email (string)` – A felhasználó email címe.
- `IsActive (bool)` – Azt jelzi, hogy a felhasználó aktív-e a rendszerben.
- `CreatedAt (DateTime)` – A felhasználó létrehozásának dátuma és időpontja.

Megjegyzés:

- Az `OrganizationId` és `RoleId` nullable típusok, tehát lehetnek null értékek is, ha a felhasználó nincs szervezethez rendelve vagy nincs szerepköre.
- Az `IsActive` tulajdonság azt jelzi, hogy a felhasználó fiókja aktív-e (pl. ha törlik, akkor `false` lesz).

Services mappa

A **Services** mappa tartalmazza az alkalmazásban használt szolgáltatásokat, amelyek az API-val való kommunikációt és egyéb üzleti logikát valósítanak meg.

ApiService osztály

Elérési út: `StudentHiveWpf.Services.ApiService`

Leírás:

Az `ApiService` osztály felelős az API-val való kapcsolattartásért. Az osztály metódusai lehetővé teszik a felhasználók adatainak lekérdezését, módosítását és egyéb műveletek végrehajtását az API segítségével.

Konstruktorok:

- `ApiService(IHttpClientFactory httpClientFactory):`
 - A függőségi injektálás segítségével létrehoz egy `HttpClient` példányt.
 - Az API végpont alapcíme: `https://localhost:7067/api/general/`.
- `ApiService():`

- Ha nincs `IHttpClientFactory`, akkor manuálisan hoz létre egy `HttpClient` példányt.

Metódusok:

Felhasználók kezelése

- `GetAllUsersAsync()`:
 - Lekéri az összes felhasználót az API-ból.
 - Visszatérési érték: `List<User>`.
- `ToggleUserStatusAsync(int userId)`:
 - Frissíti egy felhasználó aktív/inaktív státuszát az API-n keresztül.
- `UpdateUserPasswordAsync(int userId, string newPassword)`:
 - Módosítja a felhasználó jelszavát.
 - Visszatérési érték: `HttpResponseMessage`, amely tartalmazza az API választ.
- `UpdateUserProfileAsync(int userId, string firstName, string lastName, string email)`:
 - Frissíti a felhasználó profilját (keresztnev, vezetéknév, email).
 - Ellenőrzi, hogy a név és az email formátuma megfelelő-e.
 - Visszatérési érték: `HttpResponseMessage`.

Jelszó visszaállítás

- `ResetUserPasswordAsync(int userId, string newPassword, string userEmail)`:
 - Új jelszót állít be egy felhasználónak és emailben értesíti.
 - Ha az API sikeresen frissíti a jelszót, akkor elküldi az új jelszót a felhasználó email címére.

Segédfüggvények

- `SendEmail(string toEmail, string emailBody, string name)`:
 - Gmail SMTP szerveren keresztül küld egy emailt a felhasználónak.
 - **Megjegyzés:** Az SMTP hitelesítő adatok nem biztonságos módon vannak tárolva a kódban. Érdemes lenne egy **biztonságos konfigurációs fájlban vagy környezeti változóban** tárolni ezeket!
- `GenerateRandomPassword(int length = 12)`:
 - Létrehoz egy véletlenszerű erős jelszót.
 - A jelszó betűket, számokat és speciális karaktereket tartalmaz.
- `IsValidEmail(string email)`:
 - Reguláris kifejezéssel ellenőrzi, hogy egy email cím érvényes-e.

- `IsValidName(string name)`:
 - Ellenőrzi, hogy egy név csak betűket és szóközőket tartalmaz-e, és nem hosszabb 64 karakternél.

Styles mappa

A **Styles** mappában található két **ResourceDictionary** fájl, amelyek a WPF alkalmazás stílusait és megjelenését kezelik. Ezek a fájlok az alkalmazás különböző elemeinek, mint például gombok, szövegdobozok és táblázatok, megjelenését szabályozzák.

MainWindowStyle resource dictionary

Ez a fájl több különböző stílust tartalmaz, amelyek a gombok és táblázatok megjelenését befolyásolják.

ToggleButton stílus:

- A **ToggleButton** elemre alkalmazott stílus lehetővé teszi egyedi háttérszín és animációk beállítását, amelyek pirosról zöldre váltanak, amikor a gomb állapota megváltozik (be van kapcsolva, ki van kapcsolva).
- A gomb hátterének színe a zöld (#FF4CD661) és piros (Red) között váltakozik, és a gomb belső ellipszisének helyzete is változik.

DataGrid stílus:

- A **DataGrid** egyedi megjelenése definiált: a sorok háttérszíne, a fejléc színei, a cellák viselkedése, és az alternáló sorok színezése.
- Az egyes **DataGridCell**-ek háttere áttetsző, és különböző színek alkalmazásra kerülnek, amikor egy cella kiválasztásra kerül, vagy a felhasználó az egérrel rámutat.
- A **DataGridColumnHeader** stílusban a fejléc háttere zöld színű (#556D48), és az egérrel való rámutatásra változik a szín.

Button stílusok:

- A **CustomPasswordButton** és **CustomEditButton** stílusok két gomb egyedi megjelenését határozzák meg. A gombok háttere zöld (#99BC85), és ha az egér rámutat, a háttér sötétebb zöldre vált (#4e6b4e).
- Ezek a gombok egy egyszerű, kerekített sarkú dizájnt alkalmaznak, és színt váltanak, ha a felhasználó fölé viszi az egeret.

UserModifyPage resource dictionary

Ez a fájl az alkalmazásban használt **TextBox** és **TextBlock** elemek alapvető stílusait tartalmazza.

TextBox stílus:

- A **TextBox** stílus meghatározza a háttér színét (#99BC85), valamint a méretet (30px magasság és 120px szélesség). Ez biztosítja, hogy minden szövegdoboz egységesen jelenjen meg az alkalmazásban.

TextBlock stílus:

- A **TextBlock** stílus szintén a háttér színét (#99BC85) és a betűszínt (#000000) szabályozza. Ezen kívül beállítja a szöveg méretét (16px), hogy az egységesen jelenjen meg az alkalmazásban.

MainWindow

MainWindow.xaml fájl leírása

Elérési út: StudentHiveWpf/MainWindow.xaml

Leírás: A MainWindow.xaml a WPF alkalmazás fő ablakát (Window) tartalmazza. Ez az ablak a felhasználók kezelésére szolgál, és tartalmazza az adminisztrátori felületet, ahol a felhasználók listája megjeleníthető, valamint lehetőség van a felhasználók adatainak módosítására és kezelésére.

A fájl főbb elemei:

Window beállítások:

- Az ablak **magassága** 700 pixel, **szélessége** 1200 pixel.
- Az ablak **pozicionálása** középre történik (`WindowStartupLocation="CenterScreen"`).
- **Átlátszóság** engedélyezve van, és az ablak stílusa testreszabott (`WindowStyle="None"`).
- A háttér az "AliceBlue" szín adja, és a szövegek színét a dinamikus `MaterialDesignBody` erőforrás szabályozza.
- A **stílusok** egy külön fájlból (`MainWindowStyle.xaml`) töltődnek be.

Konténerek:

- A fő **Border** elem körül veszi az ablak tartalmát, lekerekített sarkokkal (`CornerRadius="12"`) és egy vizuálisan vonzó **linear gradient háttérrel**, amely a színek átmenetét használja (#556D48 és #99BC85).

- A második **Border** szegélye 4 pixeles vastagságú és a háttér egy másik színátmenetet használ a színek fokozatos változásához.

Fejléc:

- A fejlécben található egy **TextBlock**, amely a "Felhasználók Kezelése" szöveget jeleníti meg.
- Ezen kívül két **Button** található:
 - Az **X gomb**, amely bezárja az ablakot.
 - A **Kijelentkezés gomb**, amely egy logout ikont jelenít meg.

Felhasználói lista és műveletek:

- Az alábbi **DataGrid** jeleníti meg a felhasználók listáját:
 - Az oszlopok a felhasználók adatait tartalmazzák, például ID, vezetéknév, keresztnév, aktív státusz, email és a létrehozás dátuma.
 - Az utolsó oszlop tartalmazza a felhasználóval kapcsolatos műveletek gombjait, mint a **Módosítás**, **Státusz váltás** (aktív/inaktív), és a **Jelszó változtatás** gombok.

Műveletek:

- A gombok és a toggle button (státusz váltás) mindegyike rendelkezik egy-egy **Click** vagy **Checked** eseménnyel, amelyek a kód mögötti logikát aktiválják.
 - A **Módosítás** gombot a felhasználó adatainak szerkesztésére használják.
 - A **Státusz váltás** lehetővé teszi a felhasználó aktív vagy inaktív státuszának módosítását.
 - A **Jelszó változtatás** gomb a felhasználó jelszavának módosításához vezet.

Fontosabb beállítások és megjegyzések:

- A **gombok** stílusai és az egyes elemek **ControlTemplate**-jei is testreszabásra kerültek az ablak dizájnjának megfelelően.
- A fájlban használt **LinearGradientBrush** és **ImageBrush** elemek a vizuális megjelenés szebbé tételét szolgálják, például a szegélyek és hátterek színátmenetes megjelenítésével.
- A **DataGrid** segítségével a felhasználói adatokat táblázatos formában jeleníti meg az alkalmazás, és a felhasználók kezelésére szolgáló interakciókat biztosít.

MainWindow.xaml.cs fájl leírása

Elérési út: StudentHiveWpf/MainWindow.xaml.cs

Leírás: A MainWindow.xaml.cs a MainWindow osztály mögötti kódot tartalmazza, amely a felhasználók kezelésére szolgál. A fájl biztosítja a felhasználói adatok betöltését, módosítását, státuszának változtatását, és jelszó visszaállítását. Az adminisztrátori jogosultságokkal rendelkező felhasználók számára készült, és az alkalmazás főablakában fut.

A fájl főbb elemei és funkciói:

ApiService inicializálása:

- A fájl az ApiService osztályt használja, amely felelős az adatok lekéréséért és frissítéséért az API-n keresztül. A szolgáltatás aszinkron módon kommunikál a backenddel, hogy biztosítsa az adatok dinamikus frissítését.

Admin jogosultságok ellenőrzése (IsAdmin metódus):

- A IsAdmin() metódus ellenőrzi, hogy az aktuális felhasználó admin jogosultsággal rendelkezik-e. Ha nem, akkor egy hibaüzenet jelenik meg, és a művelet nem hajtható végre.

Adatok betöltése (LoadDataAsync metódus):

- Az LoadDataAsync() metódus aszinkron módon lekéri az összes felhasználót az API-ból, és azokat megjeleníti a UserListBox-ban. Ha a felhasználó nem admin, akkor a művelet megszakad.

Felhasználó státuszának módosítása (ToggleUserStatusAsync metódus):

- A ToggleUserStatusAsync() metódus lehetővé teszi a felhasználó aktív/inaktív státuszának módosítását. Ha a felhasználó admin, akkor az alkalmazás kommunikál az API-val a státusz frissítésére, és a lista újratöltésre kerül.

Felhasználó szerkesztése (EditUserAsync metódus):

- A EditUserAsync() metódus lehetővé teszi egy felhasználó adatainak szerkesztését. Ehhez egy új UserModifyPage ablak nyílik meg, amelyen keresztül a felhasználói adatokat módosíthatják.

Jelszó visszaállítása (ResetUserPasswordAsync metódus):

- A ResetUserPasswordAsync() metódus lehetővé teszi egy felhasználó jelszavának visszaállítását. Az új jelszó generálásra kerül, és emailben elküldésre. A felhasználót figyelmezteti, hogy megerősítse a műveletet.

Hibaüzenet kezelés (ShowError metódus):

- A `ShowError()` metódus egy egységes módot biztosít a hibák megjelenítésére. Ez a metódus használható minden hibás művelet esetén.

Eseménykezelők:

- Az alábbi események kezelése biztosítja a felhasználói interakciókat:
 - **ToggleButton_Checked** és **ToggleButton_Unchecked**: A felhasználó státuszának módosítása (aktív/inaktív).
 - **btnedit_Click**: A felhasználó adatainak szerkesztése.
 - **btnPassword_Click**: A felhasználó jelszavának módosítása.
 - **btnLogout_Click**: Kijelentkezés a rendszerből, és visszatérés a bejelentkezési oldalra.
 - **btnClose_Click**: Az ablak bezárása.

Fontosabb megjegyzések:

- A fájlban használt `MessageBox` segítségével különböző hibaüzenetek és információk jeleníthetők meg a felhasználók számára.
- Az alkalmazás aszinkron módon működik a műveletek végrehajtása közben (`async` és `await`), hogy elkerülje a felhasználói felület blokkolását.
- Az admin jogosultságok ellenőrzése minden művelet előtt biztosítja, hogy a rendszer csak azoknak a felhasználóknak adjon hozzáférést, akik megfelelő jogosultságokkal rendelkeznek.

Összegzés: Ez a fájl a `MainWindow` osztály működését irányítja, amely az adminisztrátorok számára biztosítja a felhasználók kezelését. A fájl lehetővé teszi a felhasználók adatainak módosítását, státuszának változtatását, és jelszó visszaállítását az alkalmazásban.

LoginPage

LoginPage.xaml fájl leírása

Elérési út: `StudentHiveWpf/Views/LoginPage.xaml`

Leírás: A `LoginPage.xaml` a WPF alkalmazás bejelentkezési oldalának megjelenését és dizájnját határozza meg. A fájl az alkalmazás felhasználóinak bejelentkezési formját tartalmazza, amely lehetővé teszi az email cím és jelszó megadását, majd a felhasználók hitelesítését. Az oldal a Material Design XAML könyvtárat használja, hogy modern és felhasználóbarát felületet biztosítson.

A fájl főbb elemei és funkciói:

Alapértelmezett beállítások:

- Az ablak a CenterScreen pozícióra van állítva, és AllowsTransparency="True" beállítással rendelkezik, így az ablak átlátszó hátteret kap. A WindowStyle None, így az ablaknak nincs hagyományos kerete.
- A háttér színe AliceBlue, míg a szövegek színe a MaterialDesignBody dinamikus erőforrással van beállítva.

Bezárás gomb:

- Az ablak jobb felső sarkában található egy X gomb, amely lehetővé teszi az alkalmazás bezárását. A gomb stílusa egy dinamikus színváltozást tartalmaz, amely a gomb fölé vitt egér hatására módosul.
- A gomb egy Click eseményt vált ki, amely a btnClose_Click metódust hívja meg, bezárva az ablakot.

Felhasználói felület (UI):

- Az ablak központjában található egy StackPanel, amely vertikálisan rendezi el az elemeket.
 - **Logo:** Az Image elem egy 100x100 pixeles logót jelenít meg, amely az "StudentHive" alkalmazás vizuális azonosítóját képviseli.
 - **Cím:** A "StudentHive" felirat egy TextBlock elemben jelenik meg, fehér színnel és középre igazítva.

Bejelentkezési űrlap:

- **Email mező:** A TextBox az email cím megadására szolgál. A mező háttérképét egy felhasználói ikon képviseli, és a felhasználói interakciók során a szöveg színe világosszürkévé válik.
- **Jelszó mező:** A PasswordBox a jelszó megadására szolgál, szintén háttérképpel (kulcs ikon) és világosszürke színnel.

Gombok:

- **Bejelentkezés gomb:** A Button, amely lehetővé teszi a felhasználó számára a bejelentkezést, fehér háttérrel és kék színnel a hover állapotban. A gomb egy Click eseményt vált ki, amely a LoginButton_Click metódust hívja meg.
- **Regisztráció gomb:** Ugyanolyan stílust kapott, mint a bejelentkezés gomb, de a háttér színe narancssárgára változik a hover hatására. A gomb egy Click

eseményt vált ki, amely a RegisterButton_Click metódust hívja meg, és a regisztrációs oldalra navigál.

Események és interakciók:

- A felhasználói interakciók gombokkal történnek, amelyek mindegyike a megfelelő háttérszín változtatását is tartalmazza, amikor az egér fölé kerül.
- A bejelentkezés és regisztráció gombok eseményei az alkalmazás logikáját irányítják, lehetővé téve a felhasználók bejelentkezését vagy a regisztrációs oldalra való navigálást.

Stílusok és dizájn:

- A gombok és a szövegek modern, letisztult dizájnnal rendelkeznek, amely a Material Design elveire épít. A gombok színátmenetes háttérrel rendelkeznek, amely fokozza az alkalmazás vizuális vonzerejét.
- A szövegek és elemek elhelyezése a felhasználói élményre összpontosít, könnyen érthető és navigálható.

Összegzés:

Ez a fájl biztosítja a LoginPage nézetet, amely a felhasználó bejelentkezési oldalát jeleníti meg. Az oldal lehetővé teszi az email és jelszó megadását, miközben a felhasználói interakciók segítségével biztosítja a bejelentkezést és regisztrációt. A dizájn a Material Design elveit követi, és az alkalmazás egyedi vizuális élményét biztosítja.

LoginPage.xaml.cs fájl leírása

Elérési út: StudentHiveWpf/Views/LoginPage.xaml.cs

Leírás: A LoginPage.xaml.cs fájl a bejelentkezési oldal logikai működését kezeli a WPF alkalmazásban. Ez a fájl felelős a felhasználói interakciók kezeléséért, például a bejelentkezési folyamatért, az API-val való kommunikációért és az esetleges hibák kezeléséért. A bejelentkezés sikeres végrehajtása után a felhasználót a megfelelő oldalra navigálja.

A fájl főbb elemei és funkciói:

HttpClient beállítása:

- Az _httpClient objektum egy HttpClient, amely az API-val való kommunikációt biztosít. Az alapértelmezett BaseAddress a `https://localhost:7067/api/auth/`, ami az autentikációs végpontot jelöli.

LoginButton_Click metódus:

- **Funkció:** A metódus a bejelentkezési gomb kattintására aktiválódik. A felhasználó által megadott email cím és jelszó értékei a TextBox és a PasswordBox elemekből származnak.
- **Műveletek:**
 - A bejelentkezési adatokat egy anonim típusú objektumba (request) csomagolja, majd JSON formátumba szerializálja a JsonConvert.SerializeObject segítségével.
 - A JSON adatokat StringContent típusúvá alakítja, majd a PostAsync metódussal elküldi az API-nak az autentikációs végponton (login).
 - Ha a válasz sikeres (HTTP 200 OK), a válasz JSON tartalmát deszerializálja és kinyeri a token és role értékeket.
 - Ha a szerep admin, megnyitja a MainWindow fő ablakot, és bezárja a bejelentkezési oldalt. Ha a szerep nem admin, figyelmeztetést jelenít meg, hogy a felhasználónak nincs megfelelő jogosultsága.
 - Hibás adatbevitel esetén hibaüzenetet jelenít meg ("Hibás email vagy jelszó!").
 - Ha bármilyen egyéb hiba történik a kérés küldése vagy válasz feldolgozása közben, azt is kezelni tudja, és megfelelő hibaüzenetet ad.

RegisterButton_Click metódus:

- **Funkció:** A metódus a regisztrációs gomb kattintására aktiválódik, és megnyitja a RegisterPage oldalt, ahol a felhasználó regisztrálhat.
- **Műveletek:**
 - A RegisterPage ablakot új példányban megnyitja (registerPage.Show()).
 - A bejelentkezési oldal elrejtésre kerül (this.Hide()), így a felhasználó nem látja azt a regisztrációs folyamat során.
 - Az Closed eseményen keresztül biztosítja, hogy amikor a regisztrációs ablak bezárul, a bejelentkezési oldal ismét megjelenjen.

btnClose_Click metódus:

- **Funkció:** A metódus akkor hívódik meg, amikor a felhasználó rákattint a bezárás gombra (X gomb).
- **Műveletek:**

- Bezárja a LoginPage ablakot (`this.Close()`), lehetővé téve a felhasználó számára, hogy kilépjen az alkalmazásból.

Összegzés:

Ez a fájl a bejelentkezési oldal logikai működését biztosítja. A felhasználó adatainak ellenőrzését és a bejelentkezési folyamatot az API segítségével valósítja meg. Az alkalmazás lehetővé teszi, hogy a felhasználók bejelentkezzenek, és ha az admin szerepkörbe tartoznak, hozzáférést kapjanak az adminisztrációs felülethez. A regisztrációra is van lehetőség, és az ablak megfelelően kezel minden hibát, amely a bejelentkezés vagy regisztráció során felmerülhet.

RegisterPage

A RegisterPage fájl a regisztrációs oldalt definiálja a WPF alkalmazásban. Ez az oldal lehetővé teszi a felhasználók számára, hogy megadják adataikat (vezetéknév, keresztnév, email, jelszó) és regisztráljanak a rendszerbe. A felhasználói felület modern, szép színekkel és egyéni vezérlőelemekkel lett kialakítva. Az oldal tartalmazza a szükséges űrlapmezőket és gombokat a regisztrációs folyamat lebonyolításához.

RegisterPage.xaml fájl leírása

Elérési út: StudentHiveWpf/Views/RegisterPage.xaml

A fájl főbb elemei és funkciói:

Window beállítások:

- A Window elem az ablakot definiálja, amely a regisztrációs felületet tartalmazza.
- Az ablak neve "Regisztráció", és méretezése 750x600 pixeles, valamint az ablak középre van igazítva a képernyőn.
- Az ablak háttérszíne AliceBlue, és az ablak stílusa átlátszó, lehetővé téve a dekoráció nélküli megjelenést (`WindowStyle="None"`).

Bejelentkezés címke és bezárás gomb:

- Az ablak tetején található egy "Bejelentkezés" szöveges címke, amely a felhasználó számára jelzi, hogy a regisztrációs oldalra navigálhat.

- A bezárás gomb (X) biztosítja, hogy a felhasználó bezárhassa az ablakot, ha nem kíván regisztrálni. A gomb Click eseménykezelője a `btnClose_Click` metódust hívja.

Űrlap mezők:

- **Vezetéknév (LastNameTextBox):** A felhasználó itt adhatja meg a vezetéknévét. A mező egyéni háttérképpel rendelkezik, amely egy felhasználó ikont jelenít meg balra.
- **Keresztnév (FirstNameTextBox):** Hasonlóan a vezetéknévhez, a keresztnévet kell megadni.
- **Email (EmailTextBox):** A felhasználó email címét kell beírni. A mező is tartalmaz egy ikon hátteret.
- **Jelszó (PasswordBox):** A jelszó beírására szolgáló mező, szintén egyedi ikon háttérrel, amely egy kulcsot ábrázol.

Stílusok és design:

- Az oldal színvilága zöld árnyalatú, ami egy vonzó és barátságos megjelenést biztosít.
- Minden szöveges mező (pl. vezetéknév, keresztnév, email, jelszó) aláhúzott stílusú, világos színű szegéllyel rendelkezik, amely kiemeli a mezőket.
- A mezők feletti szövegek a Montserrat betűtípust használják, és minden egyes mezőhöz tartozik egy magyarázó szöveg (TextBlock), amely az adott mező célját ismerteti.
- Az ikonok (user.png és key.png) segítenek a felhasználóknak könnyebben felismerni a mezők funkcióját.

Regisztrációs gomb:

- A regisztrációs gomb a felhasználói űrlap alján található, és a felhasználó adatai elküldésére szolgál.
- A gomb stílusa dinamikusan változik, amikor az egérmutató fölé kerül: az alap háttérszíne fehér, míg amikor az egér fölötté van, a háttér kék árnyalatúra változik.
- A gomb tartalmazza a "Regisztráció" szöveget, és a Click eseménykezelője a `LoginButton_Click` metódust hívja, amely a regisztrációt kezdeményezi.

Elrendezés:

- Az oldal elrendezését egy Grid és egy StackPanel segítségével alakították ki.
- A Grid két sorra van felosztva: az első sor a címke és a bezárás gomb, a második sor tartalmazza az űrlapot.
- Az űrlap mezőit egy StackPanel helyezkedik el, amely biztosítja a vertikális elrendezést, hogy a felhasználók könnyen kitölthessék az adatokat.

Összegzés:

Ez a fájl a regisztrációs oldal megjelenését és funkcionalitását határozza meg. A felhasználó a regisztrációs űrlap kitöltésével regisztrálhat a rendszerben. A felület felhasználóbarát, könnyen navigálható, és vizuálisan vonzó, a zöld színvilág és az ikonok segítenek a felhasználóknak a megfelelő mezők felismerésében. A regisztráció sikeres végrehajtásához szükséges adatokat a rendszer a háttérben dolgozza fel.

RegisterPage.xaml.cs fájl leírása

Elérési út: StudentHiveWpf/Views/RegisterPage.xaml.cs

Leírás: A RegisterPage.xaml.cs fájl a regisztrációs oldal funkcionalitását biztosítja a WPF alkalmazásban. Ez a fájl kezeli a felhasználói interakciókat, például a regisztrációs gomb megnyomását, és végrehajtja az adatküldést az API felé, hogy egy admin felhasználót regisztráljon. A regisztrációs űrlap kitöltése után a rendszer az adatokat egy JSON kéréssel elküldi az API-nak, és megfelelő visszajelzést ad a felhasználónak a regisztráció sikerességéről vagy hibájáról.

A fájl főbb elemei és funkciói:

HttpClient beállítások:

- Az osztályban használt HttpClient példány inicializálása történik a konstruktorban. Az HttpClient a háttérben végzi a HTTP kérések kezelését.
- Két alapértelmezett fejléc is hozzáadásra kerül:
 - X-Client-Id: Egyedi azonosító, amely jelzi, hogy a kérés a StudentHiveWpfClient alkalmazásból származik.
 - X-API-Key: Az API kulcs, amely a kommunikációt biztosítja az API-val.

btnClose_Click metódus:

- Ez a metódus a "Bezárás" gomb (X) kattintására aktiválódik.
- A gomb lenyomásakor az ablak bezárul (this.Close()).

LoginButton_Click metódus:

- A regisztrációs gomb (Regisztráció) kattintására hívódik meg, és elindítja a regisztrációs folyamatot.
- A metódus aszinkron módon hívja meg a RegisterAdmin() metódust, amely elvégzi a regisztrációs kérést az API felé.

RegisterAdmin metódus:

- **Input validáció:** A metódus először ellenőrzi, hogy a felhasználó megadta-e az email címet és a jelszót. Ha valamelyik mező üres, akkor hibaüzenetet jelenít meg, és megakadályozza a regisztrációt.
- **Adatok küldése:** Ha a validáció sikeres, a felhasználó adatai (vezetéknév, keresztnév, email és jelszó) egy RegisterRequest objektumban kerülnek elküldésre.
- Az adatokat JSON formátumban szerializálja a rendszer, és HTTP POST kérést küld az API register-admin végpontjára.
- **Válasz kezelése:** Ha a kérés sikeres (2xx státuszkód), akkor egy sikeres regisztrációs üzenetet jelenít meg. Ha nem sikerül, akkor a hibaüzenetet jeleníti meg, amelyet az API válaszából olvas ki.
- **Hibakezelés:** Ha a kérés során bármilyen kivétel lép fel, a rendszer figyelmeztetést ad a felhasználónak a hiba részleteivel.

Osztályok:

- **RegisterRequest osztály:** A regisztrációs adatokat tároló osztály, amely tartalmazza a következő tulajdonságokat:
 - FirstName: A felhasználó keresztnéve.
 - LastName: A felhasználó vezetéknéve.
 - Email: A felhasználó email címe.
 - Password: A felhasználó jelszava.
- **ErrorResponse osztály:** Az API válasza a hibák kezelésére. Ez az osztály tartalmaz egy Message tulajdonságot, amelyben a hibaüzenet található.

Összegzés: A RegisterPage.xaml.cs fájl a regisztrációs funkcióért felelős kódot tartalmazza a WPF alkalmazásban. A felhasználó által megadott adatokat az alkalmazás az API felé küldi, és visszajelzést ad a sikeres vagy sikertelen regisztrációról. A fájl biztosítja a felhasználói interakciók kezelését és az aszinkron kommunikációt az API-val a regisztrációs adatkezelés érdekében.

UserModifyPage

A UserModifyPage egy WPF ablakot definiál, amely a felhasználói adatok módosítására szolgál. A felhasználó adatainak megjelenítésére és szerkesztésére alkalmas felületet biztosít, amelyben a felhasználói információk, például a vezetéknev, keresztnév, email cím, azonosítók és egyéb adatok láthatóak és szerkeszthetők. A felhasználó a módosított adatokat mentheti, vagy az ablakot bezárhatja.

UserModifyPage.xaml fájl leírása

Elérési út: StudentHiveWpf/Views/UserModifyPage.xaml

A fájl főbb elemei és funkciói:

Ablak beállítások:

- Az ablak (Window) beállításai között szerepel a cím (Title), a magasság és szélesség (Height, Width), valamint az ablak középre pozicionálása (WindowStartupLocation="CenterScreen").
- Az ablak háttérszínének (Background="AliceBlue") és átlátszóságának (AllowsTransparency="True") beállítása is megtörténik.
- Az ablak stílusát is testreszabja (WindowStyle="None"), hogy az ablaknak nincsenek keretei vagy gombjai (minimális stílust használ).

Stílusok és erőforrások:

- A fájl tartalmazza a színek és ecsetek definiálását, például a fekete (BlackColor), fehér (WhiteColor), zöld árnyalatok (SmoothGreen, DarkGreen, HoverGreen), és egyéb háttérszínek.
- Ezeket a színeket különböző SolidColorBrush formájában deklarálja, amelyeket a gombok, szövegdozok és egyéb UI elemek színezésére használ.

Stílusok:

- A ButtonStyle és TextBoxStyle stílusok testreszabják a gombok és szövegdozok megjelenését:
 - **ButtonStyle:** A gombok háttérszínét és előtérszínét, betűméretét, párnázását, szegélyét és egér fölötti hatást szabályozza. Az egér fölé helyezve a háttérszín változik.

- **TextBoxStyle:** A szövegdobozok háttérszínét, előtér színét, szegélyét és magasságát szabályozza, valamint a szöveg kitöltési területét (párnázást).

UI elemek:

- **TextBlock:** A szövegelemek (pl. "Felhasználó Adatai", "Vezetéknév", "Keresztnév") a felhasználói adatokat címkézik.
- **TextBox:** A felhasználói adatokat (pl. vezetéknév, keresztnév, email cím) tartalmazó szövegdobozok, amelyek lehetővé teszik a felhasználók számára a módosításokat. Néhány mező, mint az IdInput, RoleIdInput, OrganizationIdInput, CreatedAtInput, és IsActiveInput nem szerkeszthető, mivel ezek az adatok a rendszer által generáltak, és a felhasználó nem módosíthatja őket.

Buttons (Gombok):

- **Adatok módosítása gomb:** A gomb rákattintva elindítja a felhasználói adatokat módosító műveletet (ezt a UpdateProfile metódus végzi el).
- **Bezárás gomb:** A gomb lenyomásával az ablak bezárul (Cancel_Click).

Layout és elrendezés:

- Az ablak egy Border elembe van ágyazva, amely körülveszi a tartalmat, és lekerekített sarkokkal rendelkezik (CornerRadius="10").
- Az elrendezés Grid elemet és StackPanel-t használ, hogy az összes űrlapmező függőleges irányban egymás alá rendezve jelenjen meg.
- A StackPanel a címkék (TextBlock) és a beviteli mezők (TextBox) elemeket tartalmazza, és 280px szélességű, 10px margóval van elhelyezve.

Összegzés:

A UserModifyPage.xaml fájl egy felhasználói adatokat módosító űrlapot biztosít a WPF alkalmazásban. Az ablakon belül a felhasználó adatainak megjelenítésére és módosítására van lehetőség. A fájl színbeállításokat, stílusokat, valamint a felhasználói interakciókat kezeli, amelyek az adatok szerkesztését és a műveletek (pl. adatmentés, ablak bezárása) kezelését biztosítják.

UserModifyPage.xaml.cs fájl leírása

Leírás: A UserModifyPage.xaml.cs fájl a UserModifyPage.xaml fájlhoz tartozó kód-behúzó fájl, amely az adott felhasználó adatainak módosítását és kezelését végzi el. A felhasználói adatokat tartalmazó űrlapot biztosít, amelyben a felhasználó személyes adatait, például nevét, email címét és azonosítóját lehet módosítani. A módosításokat a rendszer a megfelelő API-n keresztül frissíti.

A fájl főbb elemei és funkciói:

Konstruktor:

- A konstruktorban (UserModifyPage(User user)) egy HttpClient példányt hoz létre, amely az API alapcímére van beállítva (<https://localhost:7067/api/general/>).
- Az ablak betöltődésekor a felhasználó adatai átkerülnek az űrlap mezőibe. A felhasználó adatait (FirstName, LastName, Email, Id, RoleId, OrganizationId, CreatedAt, IsActive) a konstruktorban kapott User objektumból olvassa ki, és a megfelelő szövegdobozokba (FirstNameInput, LastNameInput, EmailInput, stb.) helyezi.

Profil frissítése:

- A UpdateProfile metódus felelős a felhasználói profil frissítéséért. A metódus először ellenőrzi, hogy a felhasználó szerepköre adminisztrátor-e (SessionManager.Role), és ha nem, akkor egy hibaüzenetet jelenít meg, hogy nincs jogosultsága a művelethez.
- Ha az admin jogosultságok megfelelőek, akkor egy ApiService példányt használva a módosított adatokat (vezetéknév, keresztnév, email) elküldi a szervernek az UpdateUserProfileAsync aszinkron API hívással.
- Ha a válasz sikeres (response.IsSuccessStatusCode), akkor értesítést kapunk arról, hogy a profil sikeresen frissítve lett, és az ablak bezárul.
- Ha a válasz nem sikeres, akkor egy hibaüzenetet jelenít meg a rendszer.

Ablak bezárása:

- A Cancel_Click metódus kezeli az ablak bezárásának logikáját. A felhasználót megerősítésre kéri, hogy biztosan be akarja-e zárni az oldalt.
- Ha a felhasználó a "Yes" gombra kattint, akkor az ablak bezárul, különben a műveletet megszakítja.

Hibakezelés:

- Az API hívások és adatfrissítések során hibakezelést alkalmaz. Ha hiba lép fel (pl. hálózati hiba), akkor a rendszer egy hibaüzenetet jelenít meg, amely tájékoztatja a felhasználót a problémáról.

Összegzés:

A `UserModifyPage.xaml.cs` fájl biztosítja a felhasználói profil módosítását a WPF alkalmazásban. Az ablak betöltésekor a felhasználó adatai megjelennek az űrlapon, és a felhasználó módosíthatja ezeket az adatokat. A módosításokat az alkalmazás a backend API-n keresztül frissíti, és a felhasználó visszajelzést kap a sikeres vagy sikertelen frissítésről. Az ablak bezárásakor a rendszer megerősítést kér a felhasználótól, hogy biztosan be akarja-e zárni az oldalt.

Teszt dokumentáció

Teszt dokumentáció - ApiService Tesztek

A teszt dokumentáció célja a **ApiService** osztály által végrehajtott műveletek automatizált tesztelésének leírása. A tesztek célja annak biztosítása, hogy az API-k megfelelően működjenek, és minden elvárt funkcionalitás jól implementálva legyen. Az alábbi tesztek az API végpontjait vizsgálják, például felhasználói adatokat lekérni, frissíteni, illetve az egyes műveletek hibás bemenetek kezelését.

Tesztkörnyezet és Tesztelési Stratégiák

Fejlesztői környezet

- **Operációs rendszer:** Windows
- **Böngésző:** Nem alkalmazott böngésző tesztelés
- **Adatbázis szerver:** A tesztek nem érintik közvetlenül az adatbázist, mivel az API végpontok a backend szolgáltatásokat tesztelik, amelyek a háttérben az adatbázisra építenek.

Automata tesztelés

- **Unit tesztek:** A tesztek célja a különböző API végpontok válaszainak validálása, az egyes műveletek eredményeinek ellenőrzése.

- **Integrációs tesztek:** Az API válaszainak és a backend együttműködésének tesztelése, az adatbázissal való interakciók, pl. felhasználói adat frissítése vagy a felhasználói státusz változtatása.

Teszt felépítése

A **StudentHive WPF alkalmazás** teszteléséhez **Mock** objektumokat használtunk, amelyek lehetővé teszik a valós függőségek helyettesítését és az API-hívások szimulálását.

Mi az a Mock tesztelés?

A mockolás (mocking) egy tesztelési technika, amely során egy adott osztály vagy komponens függőségeit szimuláljuk, hogy ellenőrizhessük annak működését különböző szituációkban anélkül, hogy valódi külső rendszereket kellene használnunk.

Az alkalmazásunk esetében a **Moq** könyvtár segítségével hoztunk létre **Mock HttpResponseMessage** objektumokat, amelyek szimulálják az **HttpClient** válaszait az API-hívások során. Ennek köszönhetően pontosan szabályozhatjuk, hogy egy adott kérés milyen választ adjon vissza (például sikeres, sikertelen vagy hibás válasz), így biztosítva a kód robusztus és alapos tesztelését.

Hogyan működik a mockolás a tesztekben?

- **Mock HttpResponseMessage** használata: Az **HttpClient** osztályt egy mockolt verzióval helyettesítettük, amely előre definiált válaszokat ad az API-hívásokra.
- **Ellenőrzés és validálás:** A tesztekben az elvárt válaszokat összehasonlítjuk a tényleges eredményekkel, például azt vizsgáljuk, hogy egy API-hívás valóban visszaad-e egy adott felhasználói listát vagy megfelelő státuszkódot.
- **Izolált tesztkörnyezet:** Nem kell valós szervert vagy adatbázist futtatnunk, a tesztek gyorsan, kiszámítható módon futnak le.

Tesztelési scenáriók és tesztek

1. GetAllUsersAsync_ShouldReturnUsers

- **Mit tesztel?** Az összes felhasználó lekérhető-e az API-ból.

- **Elvárt eredmény:** Az API sikeresen visszaad egy nem üres felhasználói listát.
- **Sikertelen teszt esetén:**
 - Ha az API 404-es vagy más hibakódot ad vissza.
 - Ha az API üres listát ad vissza annak ellenére, hogy léteznek felhasználók.

```
[Fact]
0 references
public async Task GetAllUsersAsync_ShouldReturnUsers()
{
    var users = new List<User>
    {
        new User { Id = 1, FirstName = "John", LastName = "Doe", Email = "john.doe@example.com" },
        new User { Id = 2, FirstName = "Jane", LastName = "Smith", Email = "jane.smith@example.com" }
    };

    var jsonResponse = JsonConvert.SerializeObject(users);
    var responseMessage = new HttpResponseMessage
    {
        StatusCode = HttpStatusCode.OK,
        Content = new StringContent(jsonResponse, Encoding.UTF8, "application/json")
    };

    _httpMessageHandlerMock
        .Protected()
        .Setup<Task<HttpResponseMessage>>(
            "SendAsync",
            ItExpr.Is<HttpRequestMessage>(req => req.Method == HttpMethod.Get && req.RequestUri.ToString() == $"{_baseApiUrl}alluser"),
            ItExpr.IsAny<CancellationToken>()
        )
        .ReturnsAsync(responseMessage);

    var result = await _apiService.GetAllUsersAsync();

    Assert.NotNull(result);
    Assert.Equal(2, result.Count);
    Assert.Equal("John", result[0].FirstName);
    Assert.Equal("Jane", result[1].FirstName);
}
```

2. GetAllUsersAsync_ShouldReturnNonEmptyList_WhenUsersExist

- **Mit tesztel?** Legalább egy felhasználót visszaad-e az API, ha már regisztráltunk egyet.
- **Elvárt eredmény:** A lekérdezés sikeres, és az API legalább egy felhasználót tartalmaz.
- **Sikertelen teszt esetén:**
 - Ha az API üres listát ad vissza.
 - Ha az API nem megfelelő státuszkóddal válaszol (pl. 500-as hiba).

```
[Fact]
0 references
public async Task GetAllUsersAsync_ShouldReturnNonEmptyList_WhenUsersExist()
{
    var createUserRequest = new
    {
        FirstName = "Test",
        LastName = "User",
        Email = "test.user@example.com",
        Password = "TestPassword123!"
    };

    var createdUserResponse = new HttpResponseMessage
    {
        StatusCode = HttpStatusCode.Created
    };

    _httpMessageHandlerMock
        .Protected()
        .Setup

```

3. ToggleUserStatusAsync_ShouldReturnSuccess_WhenUserExists

- **Mit tesztl?** Egy létező felhasználó státuszának módosítása sikeresen végrehajtható-e.
- **Elvárt eredmény:** Az API válasza **200 OK** és a státusz módosul.
- **Sikertelen teszt esetén:**
 - Ha az API 400 vagy 500-as hibával tér vissza.

```
[Fact]
0 references
public async Task ToggleUserStatusAsync_ShouldReturnSuccess_WhenUserExists()
{
    int userId = 1;
    var responseMessage = new HttpResponseMessage
    {
        StatusCode = HttpStatusCode.OK
    };

    _httpMessageHandlerMock
        .Protected()
        .Setup

```

4. ToggleUserStatusAsync_ShouldReturnNotFound_WhenUserDoesNotExist

- **Mit tesztel?** Egy nem létező felhasználó státuszának módosítása megfelelő hibát ad-e vissza.
- **Elvárt eredmény:** Az API válasza **404 Not Found**.
- **Sikertelen teszt esetén:**
 - Ha az API nem 404-et ad vissza.
 - Ha a hívás mégis végrehajtódik egy nem létező felhasználóra.

```
[Fact]
0 references
public async Task ToggleUserStatusAsync_ShouldReturnNotFound_WhenUserDoesNotExist()
{
    int userId = 9999;
    var responseMessage = new HttpResponseMessage
    {
        StatusCode = HttpStatusCode.NotFound
    };

    _httpMessageHandlerMock
        .Protected()
        .Setup

```

5. UpdateUserPasswordAsync_ShouldReturnSuccess_WhenPasswordUpdated

- **Mit tesztel?** Egy létező felhasználó jelszava sikeresen frissíthető-e.
- **Elvárt eredmény:** Az API válasza **200 OK**.
- **Sikertelen teszt esetén:**
 - Ha az API nem 200-at ad vissza.
 - Ha a frissített jelszó nem megfelelően kerül tárolásra.

```
[Fact]
0 references
public async Task UpdateUserPasswordAsync_ShouldReturnSuccess_WhenPasswordUpdated()
{
    int userId = 1;
    var newPassword = "newSecurePassword123";
    var requestBody = new { NewPassword = newPassword };
    var content = new StringContent(JsonConvert.SerializeObject(requestBody), Encoding.UTF8, "application/json");

    var responseMessage = new HttpResponseMessage
    {
        StatusCode = HttpStatusCode.OK
    };

    _httpMessageHandlerMock
        .Protected()
        .Setup

```

6. UpdateUserPasswordAsync_ShouldReturnNotFound_WhenUserDoesNotExist

- **Mit tesztel?** Egy nem létező felhasználó jelszavának módosítása megfelelő hibát ad-e vissza.
- **Elvárt eredmény:** Az API válasza **404 Not Found**.
- **Sikertelen teszt esetén:**
 - Ha az API nem 404-et ad vissza.

- Ha a művelet mégis sikeres egy nem létező felhasználóra.

```
[Fact]
public async Task UpdateUserPasswordAsync_ShouldReturnNotFound_WhenUserDoesNotExist()
{
    int nonExistentUserId = 99999;
    var newPassword = "NewSecurePassword123!";
    var requestBody = new { NewPassword = newPassword };
    var content = new StringContent(JsonConvert.SerializeObject(requestBody), Encoding.UTF8, "application/json");

    var responseMessage = new HttpResponseMessage
    {
        StatusCode = HttpStatusCode.NotFound
    };

    _httpMessageHandlerMock
        .Protected()
        .Setup

```

7. ToggleUserStatusAsync_ShouldReturnResponseWithinExpectedTime

- **Mit tesztel?** A felhasználói státusz módosítására adott válaszdő megfelelő-e.
- **Elvárt eredmény:** A válaszdő **kevesebb mint 1 másodperc**.
- **Sikertelen teszt esetén:**
 - Ha a válaszdő meghaladja az 1 másodpercet.
 - Ha az API késlekedést okoz a válaszdásban.

```
[Fact]
public async Task ToggleUserStatusAsync_ShouldReturnResponseWithinExpectedTime()
{
    int existingUserId = 1;
    var responseMessage = new HttpResponseMessage
    {
        StatusCode = HttpStatusCode.OK
    };

    _httpMessageHandlerMock
        .Protected()
        .Setup

```

8. UpdateUserProfileAsync_ShouldReturnSuccess_WhenProfileUpdated

- **Mit tesztel?** Egy felhasználó profiladatainak módosítása sikeres-e.
- **Elvárt eredmény:** Az API válasza **200 OK**, és a módosított adatok mentésre kerülnek.
- **Sikertelen teszt esetén:**
 - Ha az API nem 200-at ad vissza.
 - Ha az új adatok nem kerülnek elmentésre.

```
[Fact]
[References]
public async Task UpdateUserProfileAsync_ShouldReturnSuccess_WhenProfileUpdated()
{
    int userId = 1;
    string firstName = "John";
    string lastName = "Doe";
    string email = "john.doe@example.com";
    var requestBody = new { FirstName = firstName, LastName = lastName, Email = email };
    var content = new StringContent(JsonConvert.SerializeObject(requestBody), System.Text.Encoding.UTF8, "application/json");

    var responseMessage = new HttpResponseMessage
    {
        StatusCode = HttpStatusCode.OK
    };

    _httpClientMock
        .Protected()
        .Setup(
            "SendAsync",
            ItExpr.Is<HttpRequestMessage>(req =>
                req.Method == HttpMethod.Patch &&
                req.RequestUri.ToString() == $"{_baseApiUrl}update-user-profile/{userId}"),
            ItExpr.IsAny<CancellationToken>())
        .ReturnsAsync(responseMessage);

    var response = await _httpClient.PatchAsync($"{_baseApiUrl}update-user-profile/{userId}", content);
    Assert.Equal(HttpStatusCode.OK, response.StatusCode);
}
```

9. UpdateUserProfileAsync_ShouldReturnSuccess_ForDifferentUsers

- **Mit tesztel?** Több különböző felhasználó profiladatainak módosítása sikeres-e.
- **Elvárt eredmény:** Minden egyes tesztelt felhasználónál **200 OK** válasz érkezik.
- **Sikertelen teszt esetén:**
 - Ha bármelyik felhasználónál az API nem 200-at ad vissza.
 - Ha az adatok nem frissülnek.

```
[Theory]
[InlineData(1, "John", "Doe", "john.doe@example.com")]
[InlineData(2, "Jane", "Smith", "jane.smith@example.com")]
[References]
public async Task UpdateUserProfileAsync_ShouldReturnSuccess_ForDifferentUsers(int userId, string firstName, string lastName, string email)
{
    var requestBody = new { FirstName = firstName, LastName = lastName, Email = email };
    var content = new StringContent(JsonConvert.SerializeObject(requestBody), System.Text.Encoding.UTF8, "application/json");

    var responseMessage = new HttpResponseMessage
    {
        StatusCode = HttpStatusCode.OK
    };

    _httpClientMock
        .Protected()
        .Setup(
            "SendAsync",
            ItExpr.Is<HttpRequestMessage>(req =>
                req.Method == HttpMethod.Patch &&
                req.RequestUri.ToString() == $"{_baseApiUrl}update-user-profile/{userId}"),
            ItExpr.IsAny<CancellationToken>())
        .ReturnsAsync(responseMessage);

    var response = await _httpClient.PatchAsync($"{_baseApiUrl}update-user-profile/{userId}", content);
    Assert.Equal(HttpStatusCode.OK, response.StatusCode);
}
```

10. UpdateUserProfileAsync_ShouldReturnNotFound_WhenUserDoesNotExist

- **Mit tesztel?** Egy nem létező felhasználó profiljának módosítása megfelelő hibát ad-e vissza.
- **Elvárt eredmény:** Az API válasza **404 Not Found**.
- **Sikertelen teszt esetén:**
 - Ha az API nem 404-et ad vissza.
 - Ha az API sikeres választ küld egy nem létező felhasználóra.

```
[Fact]
0 references
public async Task UpdateUserProfileAsync_ShouldReturnNotFound_WhenUserDoesNotExist()
{
    int nonExistentUserId = 99999;
    string firstName = "Ghost";
    string lastName = "User";
    string email = "ghost.user@example.com";

    var requestBody = new { FirstName = firstName, LastName = lastName, Email = email };
    var content = new StringContent(JsonConvert.SerializeObject(requestBody), System.Text.Encoding.UTF8, "application/json");

    var responseMessage = new HttpResponseMessage
    {
        StatusCode = HttpStatusCode.NotFound
    };

    _httpMessageHandlerMock
        .Protected()
        .Setup

```

Továbbfejlesztési lehetőségek

A WPF alapú asztali alkalmazásban számos továbbfejlesztési lehetőség van, amelyek segíthetnek javítani a felhasználói élményt, növelhetik a funkcionalitást és fokozhatják az adminisztrációs hatékonyságot. Íme néhány fejlesztési javaslat:

Jobb keresési és szűrési lehetőségek

A felhasználók számára egy fejlettebb keresési és szűrési mechanizmus segítené a munkák gyorsabb és hatékonyabb megtalálásában. Javasolt szűrési lehetőségek:

- **Bérszűrés:** A felhasználók sávos bérintervallumok alapján kereshetnének.
- **Szűrés elérhetőség szerint:** A munkák elérhetősége (pl. heti munkaidő) alapján történő szűrés.
- **Munkáltatói értékelések szerinti szűrés:** A diákok véleménye és értékelése alapján történő keresés.

Beépített értesítési rendszer

A felhasználók naprakész értesítéseket kaphatnának a legfrissebb aktivitásokról a rendszerben:

- **Push értesítések:** Az alkalmazás értesíthetne a fontos eseményekről, például amikor új álláshirdetés jelenik meg, vagy egy alkalmazás státusza frissül.
- **Üzenetközpont:** Az alkalmazásba beépíthető lenne egy központ, ahol a felhasználók minden fontos információt és értesítést megtalálhatnak.

Fejlettebb statisztikai és analitikai eszközök

A rendszerbe beépíthetők lennének olyan analitikai eszközök, amelyek segítenek az adminisztrátorok számára a platform teljesítményének és hatékonyságának mérésében:

- **Grafikonok és diagramok:** Munkák sikerességi arányának, felhasználói aktivitásnak és alkalmazási trendeknek a vizualizálása.
- **Jelentések generálása:** Részletes statisztikák generálása, amelyek segítenek az adminisztrátoroknak a jövőbeli döntések meghozatalában.

Automatizált munkaszerződés-generálás

A felhasználók számára lehetőség lenne digitális szerződéskötésre az alkalmazáson belül. Ez magában foglalhatná:

- **Automatikus munkaszerződések generálása:** A platform automatikusan létrehozhatna PDF formátumban szerződéseket a felhasználók számára, miután elvégezték a szükséges lépéseket (pl. munkavállalói információk megadása).
- **Aláíráskezelés:** A diákok aláírhatnák a szerződést digitálisan.

Mobilalkalmazás fejlesztése

A WPF alkalmazás kiegészíthető egy mobil alkalmazással, amely Android és iOS platformokon is működhet:

- **Push értesítések és regisztráció:** A mobil alkalmazás támogatná a gyors regisztrációt, belépést és az állásajánlatok értesítéseit.
- **Térképes keresés:** A felhasználók kereshetnek állásokat hely alapján, térképen megjelenítve a legközelebbi munkalehetőségeket.

Felhasználói profilok és személyre szabás

A felhasználók számára lehetőséget biztosítani arra, hogy testreszabják profiljaikat:

- **Profilfotók hozzáadása:** A felhasználók képesek lehetnek profilfotót feltölteni.
- **Munkakeresési preferenciák:** A felhasználók előre beállíthatják, milyen típusú munkákat keresnek (pl. ideiglenes munkák, részmunkaidős állások).

Szinkronizálás más rendszerekkel

Az alkalmazás bővíthető lenne, hogy szinkronizáljon harmadik féltől származó alkalmazásokkal és szolgáltatásokkal:

- **Harmadik féltől származó API-k:** Az alkalmazás képes lehetne más munkaerő-piaci platformok adatainak integrálására, pl. LinkedIn vagy más munkaügyi alkalmazások.
- **Google Calendar integráció:** Az állásinterjúk és egyéb események szinkronizálása a felhasználók Google Naptárjával.

Jobb hibaüzenetek és felhasználói visszajelzések

A rendszer felhasználói élményét javíthatnánk, ha pontosabb és érthetőbb hibaüzeneteket adna a felhasználóknak:

- **Részletesebb hibaüzenetek:** A felhasználókat érthető üzenetek tájékoztatnák arról, mi történt hibásan, és hogyan orvosolhatják a problémát.
- **Interaktív visszajelzési rendszer:** A felhasználók értékelhetnék a szolgáltatásokat és javaslatokat tehetnének a fejlesztők számára.

Funkcionális tesztelés

Regisztráció tesztelése

Sikeres regisztráció (pozitív tesztesetek)

admin@example.com	SecurePass123!	Sikeres regisztráció
test.admin@domain.hu	P@ssw0rd!	Sikeres regisztráció
studenthive.admin@gmail.com	M0stSecure_999	Sikeres regisztráció
valid.email@company.com	LongPass_2024#	Sikeres regisztráció

Sikertelen regisztráció (negatív tesztesetek)

invalidemail.com	SecurePass123!	"Az email cím formátuma érvénytelen!"
admin@.com	SecurePass123!	"Az email cím formátuma érvénytelen!"
(üres mező)	SecurePass123!	"Az email cím megadása kötelező!"
admin@example.com	12345	"A jelszónak legalább 8 karakter hosszúnak kell lennie!"
admin@example.com	password	"A jelszónak tartalmaznia kell legalább egy számot és egy speciális karaktert!"
admin@example.com	SecurePass123! <i>(már létező email)</i>	"Ez az email cím már foglalt!"

admin@example.com	(üres mező)	"A jelszó megadása kötelező!"
-------------------	-------------	-------------------------------

Bejelentkezés tesztelése

Sikeres regisztráció (pozitív tesztesetek)

Email	Jelszó	Várt Eredmény
Valid@email.com	ValidPassword123	Sikeres bejelentkezés

Sikertelen bejelentkezés (negatív tesztesetek)

Email	Jelszó	Várt eredmény
wrong@email.com	SecurePass123!	"hibás email cím vagy jelszó"
admin@example.com	wrongpassword	"hibás email cím vagy jelszó"
(üres mező)	SecurePass123!	"hibás email cím vagy jelszó"
admin@example.com	(üres mező)	"hibás email cím vagy jelszó"

Adatok módosításának tesztelése

Sikeres adatmódosítás (pozitív tesztesetek)

Tesztesetek	Bemeneti Adatok	Várt eredmény
Admin sikeres módosítás	Admin bejelentkezve, helyes adatok megadva	"Profil sikeresen frissítve!"

Sikertelen adatmódosítás (negatív tesztesetek)

Tesztesetek	Bemeneti Adatok	Várt eredmény
-------------	-----------------	---------------

Nem admin próbál módosítani	Nem admin bejelentkezve, helyes adatok megadva	"Nincs jogosultságod ehhez a művelethez!" hibaüzenet
Hiányzó vezetéknév	Vezetéknév mező üres	"Hiba történt a frissítés során." hibaüzenet
Hiányzó keresztnév	Keresztnév mező üres	"Hiba történt a frissítés során." hibaüzenet
Hiányzó email cím	Email mező üres	"Hiba történt a frissítés során." hibaüzenet
Érvénytelen email formátum	tesztemail.com (érvénytelen email cím)	"Hiba történt a frissítés során." hibaüzenet
Sikertelen kapcsolat a szerverhez	Internetkapcsolat nincs	"Hiba történt a frissítés során." hibaüzenet
Szerver hibát küld	Szerver válasza 500-as HTTP státuszkód	"Hiba történt a frissítés során." hibaüzenet
Bezárás megerősítés után	"Bezárás" gomb megnyomva, "Igen" választva	Ablak bezárul
Bezárás megszakítása	"Bezárás" gomb megnyomva, "Nem" választva	Ablak nyitva marad

Asztali alkalmazás felhasználói dokumentáció

A program célja és funkciói

A **StudentHive Adminisztrációs Alkalmazás** egy asztali alkalmazás, amelyet az adminisztrátorok számára fejlesztettek ki a diákmunka-szervezetek kezelésére. Az alkalmazás

lehetővé teszi a felhasználói fiókok módosítását, a szerepkörök beállítását és az adminisztrációs folyamatok egyszerűsítését.

Az alkalmazás célja

Az alkalmazás fő célja, hogy egy könnyen használható felületet biztosítson az adminisztrátoroknak a rendszerben regisztrált felhasználók kezelésére. Az adminisztrátorok az alkalmazás segítségével:

- Megtekinthetik és szerkeszthetik a felhasználók adatait
- Szerkeszthetik a felhasználó státuszát
- Nyomon követhetik a fiókok létrehozásának dátumát

Szükséges hardver- és szoftverkövetelmények

Minimális rendszerkövetelmények (alapvető működéshez)

Ez a konfiguráció lehetővé teszi, hogy egy **Visual Studio 2022 projekt** fusson, valamint egy **XAMPP** szerver is elindítható legyen rajta.

- **Processzor:** 64-bites, legalább 4 magos Intel vagy AMD processzor (pl. Intel Core i3 8. generáció vagy AMD Ryzen 3)
- **Memória (RAM):** 8 GB
- **Tárhely:** Legalább **40 GB szabad hely** (Visual Studio, XAMPP és az adatbázis számára)
- **Grafika:** DirectX 11 kompatibilis grafikus kártya
- **Operációs rendszer:** Windows 10 (vagy újabb, 64-bites)

◆ **Megjegyzés:** Ez a konfiguráció lehetővé teszi a fejlesztést, de **nagyobb projektek esetén lassú lehet a rendszer.**

Ajánlott rendszerkövetelmények (stabil és gyors működéshez)

Ha egyszerre két **Visual Studio 2022** projektet kell futtatni, miközben a **XAMPP** is működik, akkor erősebb hardver szükséges.

- **Processzor:** Legalább egy **Intel Core i5 10. generációs** vagy **AMD Ryzen 5 5600X**

- **Memória (RAM): 16 GB vagy több**
- **Tárhely: SSD ajánlott (legalább 100 GB szabad hely)**
- **Grafika:** NVIDIA GTX 1050 / AMD Radeon RX 560 vagy erősebb
- **Operációs rendszer:** Windows 10 vagy Windows 11 (64-bites)

◆ **Megjegyzés:** Ez a konfiguráció lehetővé teszi, hogy zökkenőmentesen fusson egyszerre több projekt a Visual Studio-ban és a háttérben egy **MySQL adatbázis** is működjön XAMPP alatt.

Fejlesztői környezet – A fejlesztéshez használt hardver (a projekt fejlesztése és tesztelése ezen a konfiguráción történt)

Fejlesztői környezet – A fejlesztéshez használt hardver (a projekt fejlesztése és tesztelése ezen a konfiguráción történt)

A fejlesztés során az alábbi eszközt használtuk, amely biztosította a gyors és stabil működést:

- **Processzor:** Intel Core i7-8750H
- **Memória:** 24 GB RAM
- **Videókártya:** NVIDIA RTX 2070
- **Tárhely:** SSD ajánlott, minimum 50 GB szabad tárhely

Indítási útmutató

WPF projekt megnyitása

Amennyiben a **XAMPP** sikeresen elindult következhet a desktop indítása. A projekt a **desktop/** mappában található.

Ha Visual Studio 2022-t használsz:

1. Nyisd meg a Visual Studio-t.
2. Kattints a "**Open a project or solution**" opcióra.
3. Navigálj a backend mappába, és válaszd ki a **.sln** fájlt.

4. Kattints a **Megnyitás** gombra.

Másik opció:

1. A lehúzott repositoryban a backend mappa megnyitása.
2. **StudentHiveWpf.sln** fájl megnyitásával elindul a projekt.

WPF Alkalmazás Futtatási Útmutató

A WPF alkalmazás megfelelő működéséhez szükséges, hogy a backend (StudentHiveServer) és az adatbázis-kezelő (XAMPP) is elinduljon. Az alábbi lépések segítségével futtathatod az alkalmazást.

Előkészületek

1. XAMPP indítása:

- a. Nyisd meg a XAMPP Control Panel alkalmazást.
- b. Indítsd el az **Apache** és **MySQL** szolgáltatásokat.

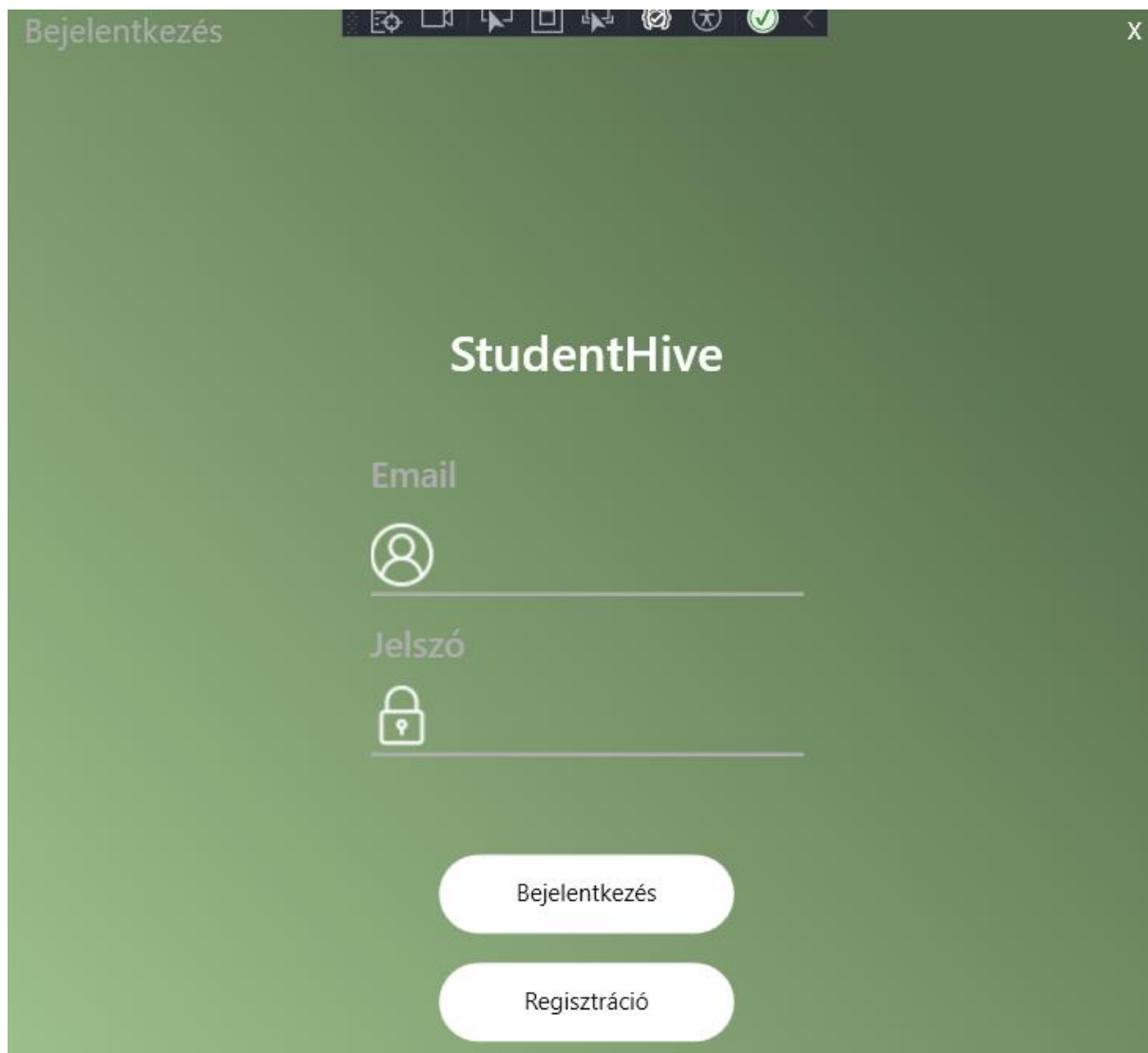
2. Backend (StudentHiveServer) indítása:

- a. Nyisd meg a **Visual Studio** alkalmazást.
- b. Nyisd meg a **StudentHiveServer** projektet.
- c. A menüsorban válaszd ki a **Debug → Start Debugging** (F5) opciót.
- d. A szerver elindul, létrehozza az adatbázist (ha még nem létezik), és elérhetővé teszi az API végpontokat a Swagger felületen.

WPF alkalmazás indítása

1. Visual Studio-ban történő futtatás:

- a. Nyisd meg a **WPF alkalmazás** projektet.
- b. Kattints a zöld **Indítás** gombra (F5), vagy válaszd ki a **Debug → Start Debugging** opciót.
- c. A program elindul, és megjelenik a bejelentkezési / regisztrációs felület.



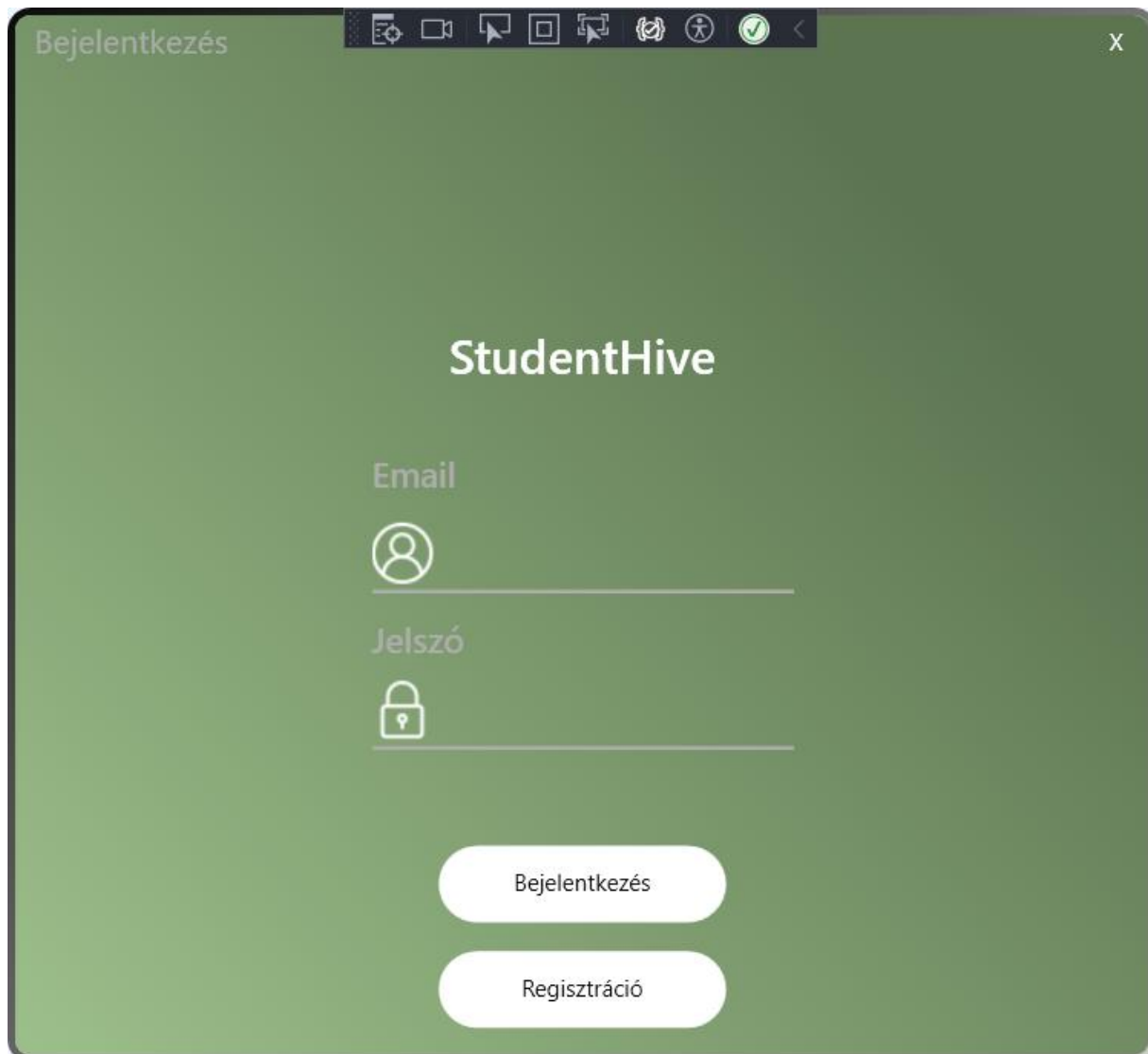
The screenshot shows a web application interface for 'StudentHive'. At the top left, there is a tab labeled 'Bejelentkezés' (Login). The main heading 'StudentHive' is centered. Below it, there are two input fields: 'Email' with a person icon and 'Jelszó' (Password) with a lock icon. At the bottom, there are two buttons: 'Bejelentkezés' and 'Regisztráció' (Registration).

Fontos Megjegyzések

- Ha az adatbázis még nem létezik, a backend automatikusan létrehozza azt. Sikeres létrehozás esetén egy üzenet jelenik meg: **„Az adatbázis sikeresen létrehozva.”**
- A backend indítása után a Swagger dokumentáció elérhetővé válik, ahol az API végpontokat megtekintheted és tesztelheted.
- A WPF alkalmazás nem fog megfelelően működni, ha a **StudentHiveServer** vagy a **XAMPP MySQL szolgáltatása** nincs elindítva.

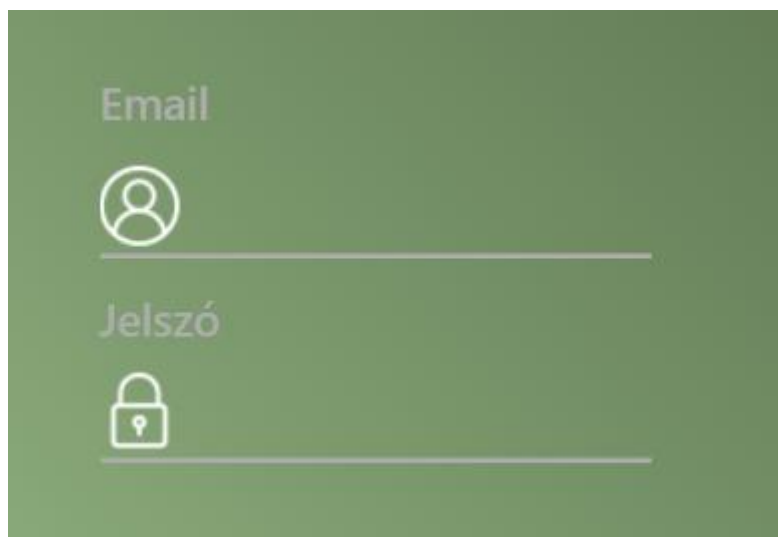
A programról részletesen

Bejelentkező felület



a felhasználót a program indítását követően egy bejelentkezés felület jön szembe itt 5 dolgot tud tenni a felhasználó:

1. A jobb felső sarokban az x megnyomásával betudja zárni a programot a felhasználó

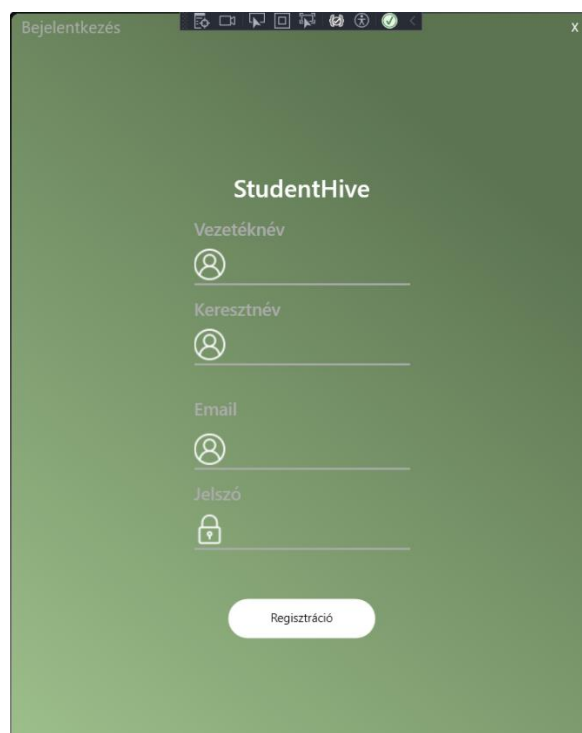


Email

Jelszó

2. Az első vonalra tudja megadni a felhasználó az email címét
3. A második vonalra a jelszót kell megadni
4. A bejelentkezés gomb segítségével ha helyes email cím és jelszó lett megadva átirányítja a felhasználót a fő felületre
5. a regisztráció gomb megnyomásával a felhasználó egy fiókot tud létrehozni

Regisztrációs felület



Bejelentkezés

StudentHive

Vezetéknév

Keresztnév

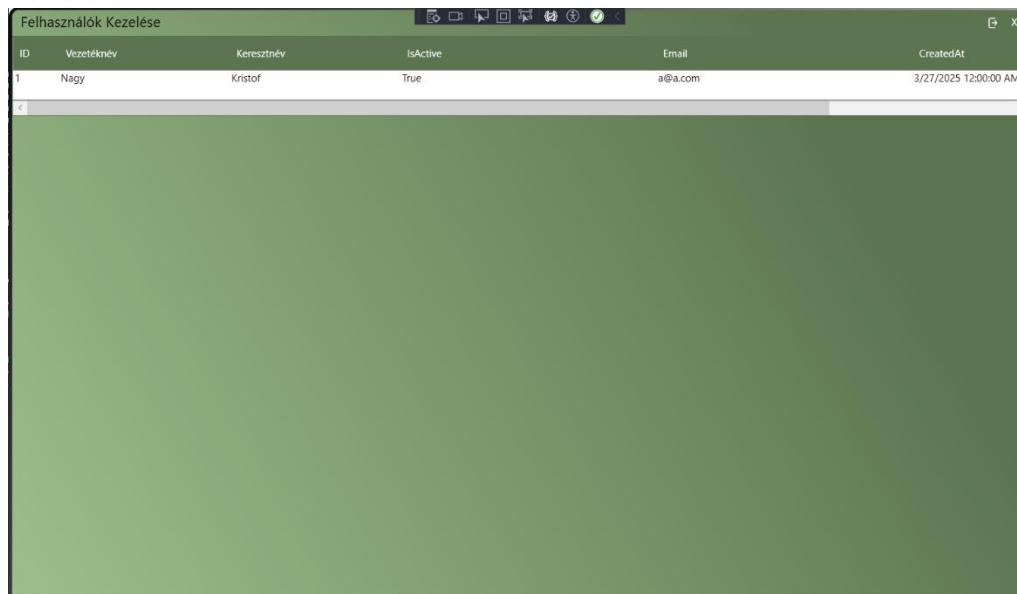
Email

Jelszó

Regisztráció

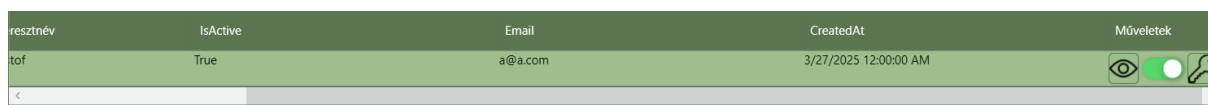
Itt a felhasználó az adatokat megadva minek a sorrendje vezetéknév, keresztnév, email és jelszó, ha mindent megadott a felhasználó akkor a regisztráció gomb megnyomásával létre tud hozni egy fiókot

Fő felület



Felsorolás szerint a felhasználó itt is 5 dolgot tud csinálni:

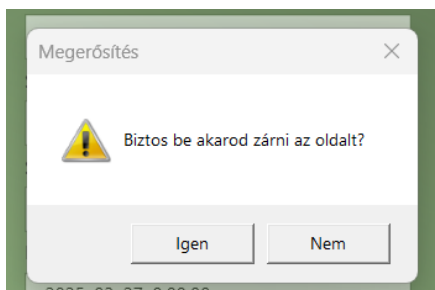
1. jobb felső sarokban található az x gomb, ami a program bezárását jelenti
2. A tőle balra található gomb a kijelentkezés gomb, aminek a segítségével fiókot lehet váltani



Emellett 3 gomb található még minden felhasználóhoz:

1. A szem gomb megnyomásával megnyílik a felhasználó adatai, amit felsorolás szerint láthatunk és tudjuk a vezetéknévét keresztnévét és email címét változtatni a felhasználónak

A. Ha megnyomjuk a bezárás gombot akkor feljön egy megerősítő ablak, ami után eldöntjük, hogy amennyiben véletlen nyomtuk meg akkor a nem gombot kell megnyomni és semmi sem történik és pedig, ha tényleg be akartuk zárni a felhasználó adatai oldalt akkor az igennel megerősítjük



B. adatok módosítása gomb megnyomása esetén, ha változtattunk a felhasználó adatain pld. az Email-t megváltoztatom admin@admin.com-ról [test@test.com-ra](mailto:test@test.com)
Lépései az email szöveges beviteli mezőbe beírom a kívánt email címet ezután megnyomom az adatok módosítása gombot és megnyomom a felugró ablakon az ok gombot, ami értesített, hogy a profilt sikeresen frissítettük

Felhasználó Adatai

Vezetéknév

Keresztnév

Email

Azonosító

Vezetéknév

Keresztnév

Email

Azonosító

Felhasználó Adatai

Vezetéknév

Keresztnév

Email

Azonosító

1

Szer

1

Szo

Fiók

2025. 03. 27. 0:00:00

Fiók Aktivitása

True

Adatok módosítása

Bezárás

Siker

Profil sikeresen frissítve!

OK

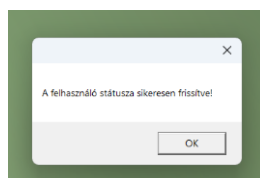
Keresztnév	IsActive	Email	CreatedAt
Kristof	True	test@test.com	3/27/2025 12:00:00 AM

2. a világos zöld gombbal, ami a kulcstól balra található azzal tudjuk állítani a felhasználónak a státuszát, ha zöld akkor aktív és tudja használni a fiókot a felhasználó és ha piros akkor viszont a fiók nem használható

A. A zöld gomb megnyomása, azaz a felhasználó státuszát változtatjuk



B. Kapunk egy üzenetet, hogy a felhasználó státuszát sikeresen frissítettük

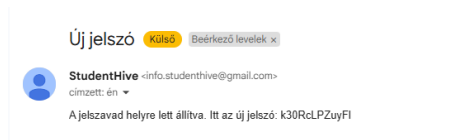


C. És megváltozik a színe a gombnak és most már piros lesz, ami jelzi, hogy inaktív



3. A kulcs gombbal lehet új jelszót generálni a felhasználónak, amit kiküldünk emailben

A. Előtte feljön egy ablak, ami megkérdezi, hogy biztosan szeretnénk e jelszót változtatni amennyiben igen nyomja meg az igen gombot ezzel kiküldi a felhasználónak az emailt és az adatbázisba is bekerül az új jelszó



B. Ha mégsem szeretné megváltoztatni a felhasználónak a jelszavát a nem gombbal minden folyamatot le tud állítani

