**StudentHive** Dokumentáció

A diákmunka jövője🚀

Tartalomjegyzék

[Bevezetés 3](#_Toc193457859)

[Honnan jött az ötlet? 3](#_Toc193457860)

[A projekt bemutatása 3](#_Toc193457861)

[Az oldal funkciói 3](#_Toc193457862)

[Fejlesztői dokumentáció 3](#_Toc193457863)

[A fejlesztői környezetről 3](#_Toc193457864)

[Kialakított adatszerkezet bemutatása 3](#_Toc193457865)

[Az adatbázis modell 3](#_Toc193457866)

[Főbb jellemzők 3](#_Toc193457867)

[Kapcsolatok és idegenkulcsok 3](#_Toc193457868)

[Program főbb algoritmusai 3](#_Toc193457869)

[Autentikáció 3](#_Toc193457870)

[Iskolaszövetkezet létrehozása 3](#_Toc193457871)

[Műszak hozzáadása 3](#_Toc193457872)

[Műszak törlése és email értesítés 3](#_Toc193457873)

[Tesztdokumentáció 3](#_Toc193457874)

[Továbbfejlesztési lehetőségek 3](#_Toc193457875)

[Felhasználói dokumentáció 3](#_Toc193457876)

[A program célja és funkciói 3](#_Toc193457877)

[Szükséges hardver- és szoftverkövetelmények 3](#_Toc193457878)

[Telepítési és indítási útmutató 3](#_Toc193457879)

[A programról részletesen 3](#_Toc193457880)

[Hibakezelés és hibaüzenetek magyarázata 3](#_Toc193457881)

[Összefoglalás 3](#_Toc193457882)

[Fejlesztési tapasztalatok 3](#_Toc193457883)

[Hogyan tovább? 3](#_Toc193457884)

[Eredmények értékelése 3](#_Toc193457885)

# 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.

Az **asztali alkalmazás** fejlesztése **C# és WPF .NET 8** környezetben történik, amely modern és reszponzív 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.

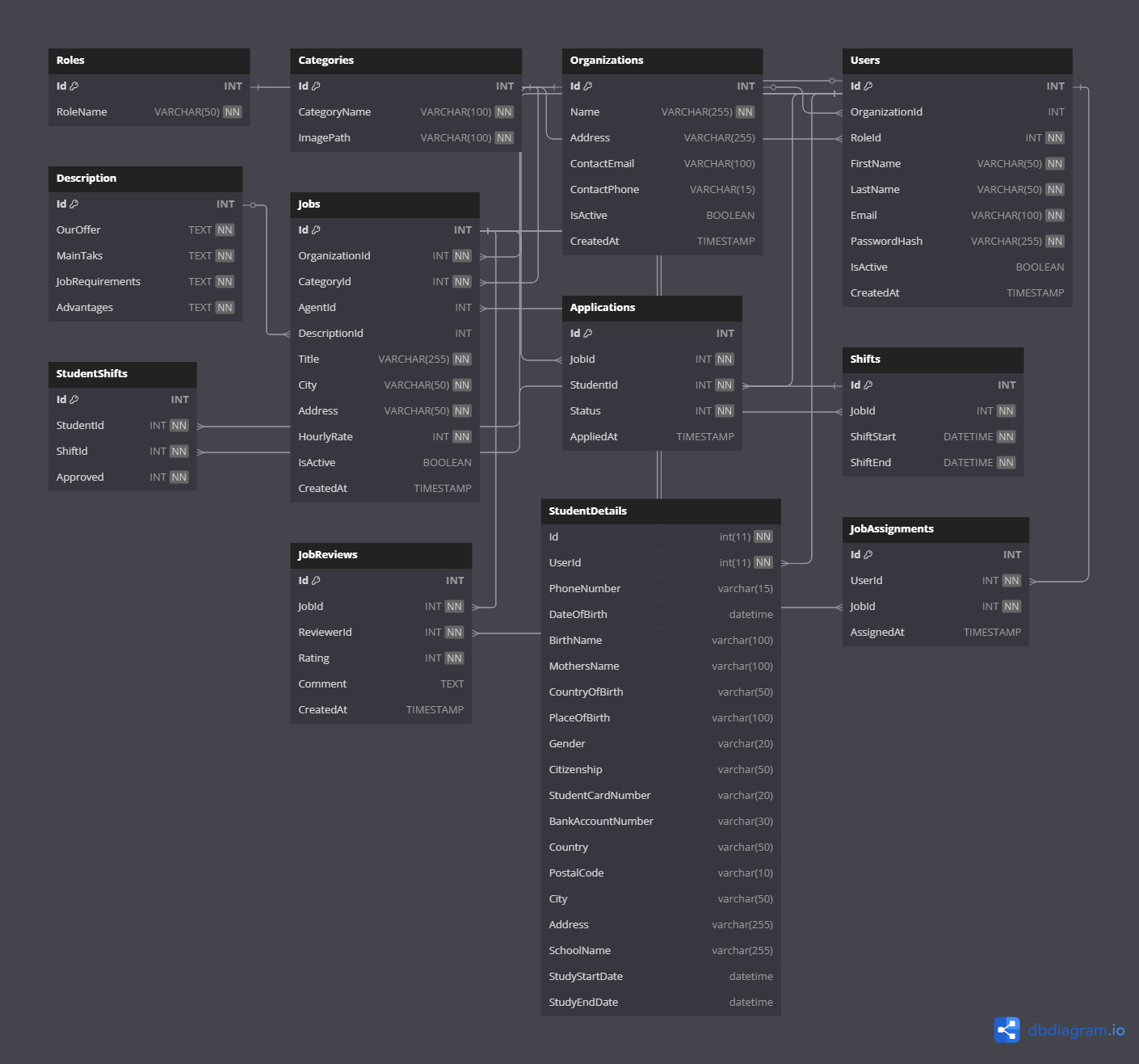
A fejlesztői eszközök között a **Visual Studio 2022** és a **Visual Studio Code** biztosítják az optimális munkakörnyezetet. Az alkalmazásstruktúra és a csomagválasztás célja egy jól optimalizált, biztonságos és könnyen bővíthető rendszer kialakítása mind webes, mind asztali környezetben.

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.

## Kialakított adatszerkezet bemutatása

### Az adatbázis modell



### Főbb jellemzők

#### Felhasználói szerepkörök

A **Roles** táblában négy különböző szerepkör található: Admin, Organization, Agent, User. A **Users** tábla tárolja a felhasználókat, mindenki kap egy szerepkört (RoleId), és opcionálisan egy szervezethez (OrganizationId) is kapcsolódhat.

#### 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 (pl. "Informatikai", "Vendéglátás").

Az **Description** tábla az állás részletes leírását tárolja (OurOffer, MainTasks, JobRequirements, Advantages).

#### Jelentkezések és értékelések

A **Applications** tábla tartja nyilván, hogy egy User (diák) mely állásokra jelentkezett. A **JobReviews** tábla lehetőséget biztosít a diákok számára, hogy értékeljék az állásokat.

#### 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 álláskategóriák kapcsolata

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

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

## 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 tokent** 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 azonosítókat 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** metódussal.

Ez a metódus:

* 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 algoritmusa (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

## Továbbfejlesztési lehetőségek

# Felhasználói dokumentáció

## A program célja és funkciói

## Szükséges hardver- és szoftverkövetelmények

## Telepítési és indítási útmutató

## A programról részletesen

## Hibakezelés és hibaüzenetek magyarázata

# Összefoglalás

## Fejlesztési tapasztalatok

## Hogyan tovább?

## Eredmények értékelése