

Vas Megyei Szakképzési Centrum
Nádasdy Tamás Technikum és Kollégium

Pályaválasztó

Herczeg Marcell, Hegedüs Tamás, Molnár Dominik

Konzulens:

Varga Gábor

2024

Nyilatkozat

Alulírott, Herczeg Marcell, Hegedüs Tamás, Molnár Dominik szoftverfejlesztő és tesztelő szakos hallgatók kijelentjük, hogy a Pályaválasztó weboldal kidolgozása a saját munkánk, abban csak a megjelölt forrásokat, és a megjelölt mértékben használtuk fel, az idézés szabályainak megfelelően, a hivatkozások pontos megjelölésével. Eredményeink saját munkán, számításokon, kutatáson, valós méréseken alapulnak, és a legjobb tudásunk szerint hitelesek.

Csepreg, 2024.04.09

hallgatók

Kivonat

Pályaválasztó

A záródolgozatunkban egy olyan weboldalt mutatunk be, ami megkönnyíti a munka vállalókat és a munkáltatók munkakeresését. A munkavállalóknak megkönnyebbíti a munkakeresést, a munkáltatók pedig könnyebben tudják kiszűrni a jó munkaerőt.

A munkavállalók képesek lesznek a saját igényeik alapján keresgálni a munkák között mivel állásajánlatok fogják sokszínűsíteni a lehetőségeiket, valamint lesznek a munkaadóktól hirdetések, amik ezt még jobban megkönnyítik.

Ezenfelül a munkáltatóknak is rengeteg előnyük származik a weboldal használatából. Regisztrálhatnak saját profilt, ahol számos kiegészítővel tudják inspirálni a munkát keresőknek, hogy őket válasszák.

Abstract

Carrerchooser

In our final thesis, we present a website that makes it easier for jobseekers and employers to find jobs. It makes it easier for workers to find a job and easier for employers to find good candidates.

Workers will be able to search for jobs based on their own needs as job offers will diversify their options and employers will have advertisements to make this even easier.

In addition, employers will also benefit greatly from using the website. They will be able to register their own profile, where they will be able to inspire jobseekers to choose them with a range of additional features.

Tartalomjegyzék

1. Bevezetés.....	4
1.1. Az EREDETI ötletünk így nézett ki :	4
1.2. Az eredeti ötlet bemutatása a tanároknak :	5
1.3. Újratervezés (sokszor) :	5
1.4. Végső ötlet bemutatása:.....	6
2. Hasonló alkalmazások.....	6
1. 2.1 :	6
2. 2.2.....	6
3. 2.3.....	7
3. Felhasználói dokumentáció.....	7
3.1.1 Lehetőségeink munkáltatóként:.....	7
3.1.2 Lehetőségeink magánszemélyként:.....	7
3.1.3 Lehetőségeink Supportként:.....	8
3.1.4 Lehetőségeink Adminként:.....	8
3.2. Belépés és regisztráció.....	8
3.2. Főoldal és tartalma.....	9
3.3. Funkciók ismertetése.....	10
3.4.2 Munkavállalótól Regisztrációs további adatok bekérése.....	11
3.4.3 Céges Regisztrációs további adatok bekérése.....	11
3.4 Profilok.....	12
3.4.1 Munkáltatói oldalról.....	12
3.4.2 Magánszemélyként.....	12
4. Fejlesztői dokumentáció.....	12
4.1. Az adatbázis táblái és kapcsolatai.....	12
4.1.1 Users tábla.....	13
4.1.2 Applikáció tábla.....	13
4.1.3 Employees tábla.....	13
4.1.4 Jobs tábla.....	13
4.1.5 Employer tábla.....	14
4.1.6 Role tábla.....	14
4.3. Metódusok, függvények és események.....	14
4.4. Osztályok és Modellek.....	16
4.5. Radzen.Blazor-os komponensek.....	16
4.6. Tesztdokumentáció.....	16
5. Az alkalmazás design-ja.....	17
6. Összefoglalás.....	18
6.1. A szakdolgozat célja.....	18
6.2. Megvalósítása.....	18

6.3. Fejlesztési lehetőségek.....	20
7. Irodalomjegyzék.....	21
7.1. Internetes Irodalomjegyzék.....	21
7.1.1 Programozási problémák megoldásai:.....	21
7.1.2 C#-os dokumentáció:.....	21
7.1.3 A.S.P.NET-es dokumentáció:.....	21
7.1.4: Blazoros dokumentáció:.....	21
7.1.5: blazor.radzen komponensek:.....	21

1. Bevezetés

1.1. Az EREDETI ötletünk így nézett ki :

Az általunk kigondolt Blazor-os Web-es alkalmazás az segíthet a fiataloknak a pályaválasztásban, ezért is viseli a Pályaválasztó nevet, a weboldalon a felhasználó által megadott konkrét információk alapján és egy kérdőív kitöltéséből kinyert eredmény alapján a program berangsorolná egy “érdeklődési kör” kategóriába is ami aztán az állásajánlatok személyre szabásnál hasznosíthatunk. Ezen adatok alapján szintúgy meghatározhatóak a felhasználó lehetőségei, mind munka, mind továbbtanulás téma körében, persze ez csak egy becslés amit egy külön oldalon tervezünk monitorozni, amit a felhasználónak készítünk hogy folyamatosan követni tudja az összes, számára elérhető és megtetsző lehetőségeket. Az alkalmazás lehet hasznos az általános iskolákban olyan gyerekeknél is akik még a pályaorientálódás elején keresik mivel szeretnének foglalkozni, valamint azoknak is akik intézményt keresnek továbbtanulás szempontjából és még az egyetemeket valamint a munkahelyeket is szerettük volna összegyűrni egy weboldalra. De amennyire ez az eredeti ötlet a komplexségtől bűzlik, annyira voltunk mi a megvalósításhoz való tudás hiányában.

1.2. Az eredeti ötlet bemutatása a tanároknak :

A Tanáraink megpróbáltak mindenben segíteni és olyan példákon keresztül megmutatni a programozásnak azt az oldalát ami érhetőbben tudta megvilágítani azt a problémát amibe botlottunk. Sajnálatos módon viszont egyik Tanárunk sem volt jártas az általunk választott technológiai megoldásban, ami a Radzen, Blazor, .NET8 valamint a környezetben sem ami a Radzen Blazor Stúdió volt. Ez a tény jócskán megnehezítette a feladatunkat az elejtől kezdve, hiszen még azok a technológiák is amikben Tanáraink jártasak voltak rejtély volt számunkra. Több órás közös beszélgetés és gondolkodás, rengeteg online és offline konzultálás a Tanárainkkal és hosszú YouTube tutorial videók sokszori megtekintése és végül de nem utolsó sorban a napjainkban folyamatosan fejlődő AI Chatbotokkal való folyamatos kommunikáció, ami jól jött ha már valamit rengeteget kerestünk az interneten, de már sehol sem találtuk akkor sok alternatívával szolgált. Mindezek együttesével mondhatom hogy a programozás logikai struktúráját ha nem is túl magabiztosan de viszonylag értjük. Készítettünk közösen egy kinézetet (illusztrációt) a Figma szerkesztő programban és bemutattuk azt Tanárainknak. Miután egyre többet kezdtünk arról beszálni hogy mennyire jó lesz, és milyen jó lehetne még ez a Webalkalmazás, a folyamatos tervezés közepette nem vettük észre hogy az idő az bizony nem áll meg.

1.3. Újratervezés (sokszor) :

Realizálnunk kellett, hogy biztosan nem lesz se időnk se elég információink az általunk választott technológiáról hogy minden eltervezett funkciót meg tudjuk valósítani. Több beszélgetés alapján a konzulensünkkel Varga Gáborral, úgy döntöttünk nem rugaszkodunk el annyira az eredeti ötlettől, viszont csak egy főbb funkcionálisra koncentrálunk ami esetünkben az álláskereső weboldal kategóriát képezi. A szerencsénk az volt hogy ez a viszonylag nagy változtatás, adott akkor egy olyan teret amit Magyarországon egyedüliként csak a Profession.hu foglalt el nagyrészt, aminek használatában már jártasak voltunk

mindannyian a csapatunkból, így könnyen kiszúrtuk azokat a hiányosságokat amiket a egyéb álláskereső oldalak használata után kiszúrtunk. Nem mellesleg a csapatunk egyik tagja Herczeg Marcell olyan családi kapcsolati tőkével rendelkezik akik több éves HR-es tapasztalatokkal rendelkeznek így tőlük is informálódott, hogy milyen hiányosságokat véltek felfedezni a mai álláskereső weboldalakban és hogy min lehetne javítani.

1.4. Végső ötlet bemutatása:

Az általunk választott ötlet egy olyan Blazor Webapplikáció-t takar ahol a felhasználók tudnak konkrétan keresni az álláshirdetésekben szereplő bármely adatra. Ahol lehetősége nyílik a felhasználónak regisztrálni akár egy Céges akár egy Munkavállalói profilt (a regisztráció befejezése egy profil-oldalt generál). Állásajánlatok randomizálva hármas oszlopban ahol minden hirdetés minden kártyáján minden fontos információ megtalálható, a kártyák egyben hivatkozások is a hirdetés teljes oldalára. Figyelemfelkeltő színek és animációdús Design. Főbb funkcionalitásai a saját profil oldal személyre szabhatósága, a hirdetések személyre szabhatósága, az állásajánlatok interaktív kártyás megoldása és pár jó tanács a főoldal tartalmában.

2 Hasonló alkalmazások

2.1 : Profession

A **Profession.hu** egy online álláskereső platform, amely elsősorban Magyarországon működik. A platform számos szolgáltatást kínál a felhasználóknak, hogy segítsen nekik a munkaerőpiac jobb megértésében és a megfelelő döntések meghozatalában. A felhasználók számos álláshirdetést találhatnak a platformon, és lehetőségük van önéletrajzuk feltöltésére és megosztására a munkaadókkal. A **Professional.hu** lehetőséget biztosít a felhasználóknak, hogy kapcsolatba lépjenek más szakemberekkel, és így bővíthetik szakmai hálózatukat. A felhasználók megoszthatják tapasztalataikat korábbi vagy jelenlegi munkahelyeikről, ami segíthet másoknak a munkahelyek jobb megértésében. A platform számos cikket és útmutatót kínál, amelyek segíthetnek a felhasználóknak a karrierjük fejlesztésében. Összefoglalva, a **Professional.hu** egy átfogó platform, amely számos eszközt és szolgáltatást kínál a munkaerőpiac jobb megértéséhez és a karrierfejlesztés elősegítéséhez.

2.2 : HVG Állásbörze

HVG Állásbörze - Jobline A **HVG Állásbörze - Jobline** egy online álláskereső platform, amely szintén Magyarországon működik. A platform számos szolgáltatást kínál a felhasználóknak, hogy segítsen nekik a munkaerőpiac jobb megértésében és a megfelelő döntések meghozatalában. A felhasználók számos álláshirdetést találhatnak a platformon. A **HVG Állásbörze - Jobline** lehetőséget biztosít a felhasználóknak, hogy kapcsolatba lépjenek más szakemberekkel, és így bővíthetik szakmai hálózatukat. A platform számos cikket és útmutatót kínál, amelyek segíthetnek a felhasználóknak a karrierjük fejlesztésében. Összefoglalva, a **HVG Állásbörze - Jobline** egy átfogó platform, amely számos eszközt és szolgáltatást kínál a munkaerőpiac jobb megértéséhez és a karrierfejlesztés elősegítéséhez.

2.3

Jobline.hu A **Jobline.hu** egy másik online álláskereső platform, amely szintén Magyarországon működik. A platform számos szolgáltatást kínál a felhasználóknak, hogy segítsen nekik a munkaerőpiac jobb megértésében és a megfelelő döntések meghozatalában. A felhasználók számos álláshirdetést találhatnak a platformon, és lehetőségük van önéletrajzuk feltöltésére és megosztására a munkaadókkal. A **Jobline.hu** lehetőséget biztosít a felhasználóknak, hogy kapcsolatba lépjenek más szakemberekkel, és így bővíthetik szakmai hálózatukat. A felhasználók megoszthatják tapasztalataikat korábbi vagy jelenlegi munkahelyeikről, ami segíthet másoknak a munkahelyek jobb megértésében. A platform számos cikket és útmutatót kínál, amelyek segíthetnek a felhasználóknak a karrierjük fejlesztésében. Összefoglalva, a **Jobline.hu** egy átfogó platform, amely számos eszközt és szolgáltatást kínál a munkaerőpiac jobb megértéséhez és a karrierfejlesztés elősegítéséhez.

3. Felhasználói dokumentáció

A weboldal fő célja hogy segítsen a cégeknek és vállalkozásoknak könnyen és hatékonyan munkaerőt találni, és nem mellesleg az hogy a munkavállalónak is olyan munkahelyi lehetőségeket biztosítson ami az egyénnek megfelelő.

3.1.1 Lehetőségeink munkáltatóként:

- Állás kártyák és Hirdetések teljes tartalmi személyre szabhatósága
- Több álláshirdetés létrehozása
- Céges profil személyre szabhatósága
- Jelentkezések elbírálása

3.1.2 Lehetőségeink magánszemélyként:

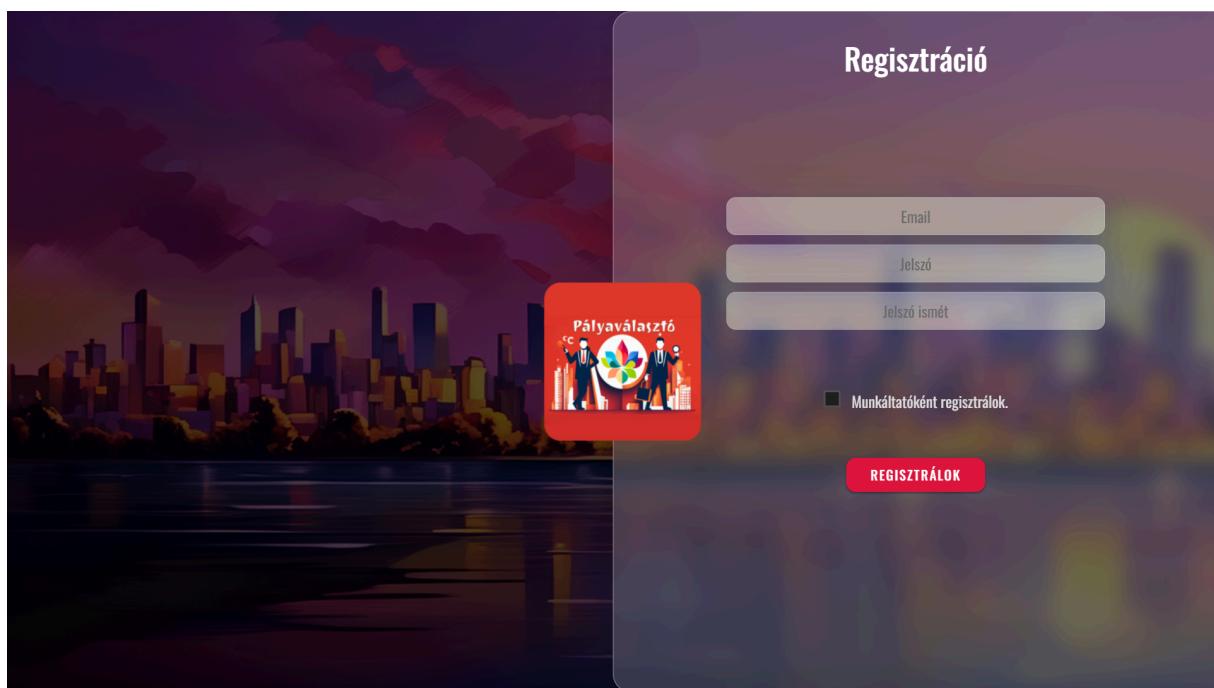
- Állások keresése
- Saját profil vagy céges profil készítése
- Álláshirdetések bér intervallumának megtekintése
- Álláshirdetések helyének megtekintése

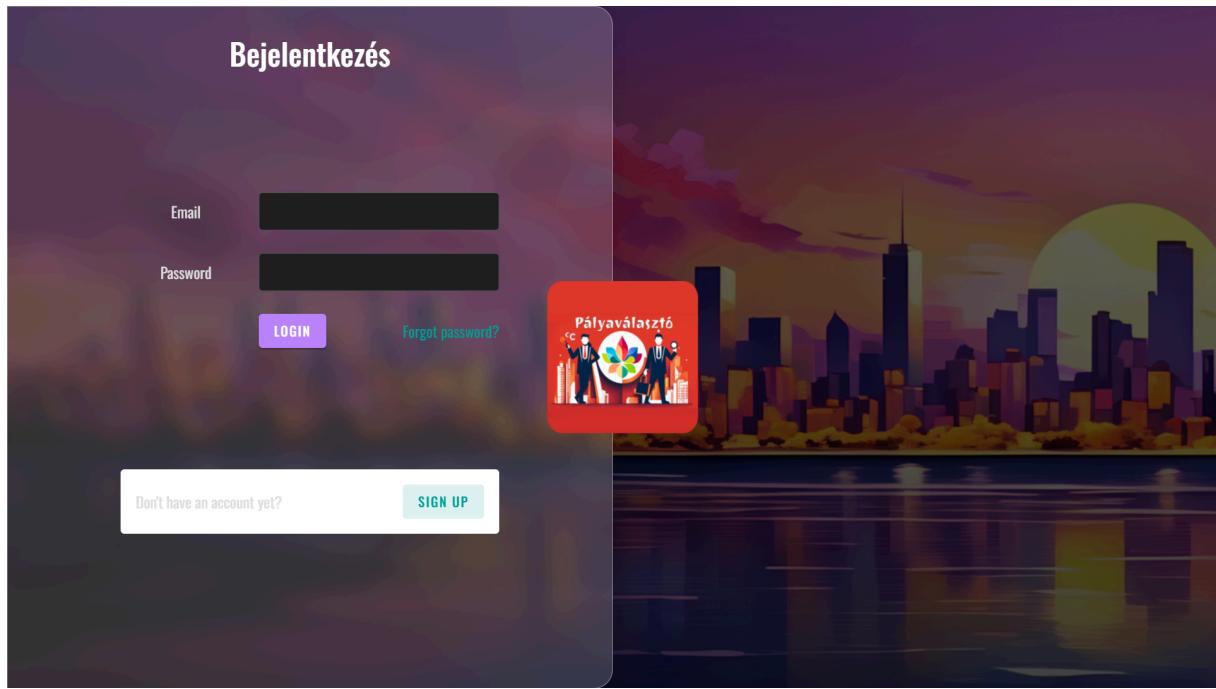
3.1.3 Lehetőségeink Adminként:

- Engedélyek szerkesztése
- A weboldal karbantartása
- Bármit tud szerkeszteni a weboldalon
- A felhasználókat inaktívvá tudja tenni
- A hirdetéseket tudja törlni

3.2. Belépés és regisztráció

A weboldalon bejelentkezés nélkül nem lehet hirdetésket létrehozni! Bárki képes fiókot létrehozni. A regisztrációhoz szükséges az email cím, jelszó, jelszó újra. Továbbá van egy jelölőnégyzet az oldalon, ahol a felhasználónak ki kell választania, hogy munkavállalóként vagy munkáltatóként szeretne-e fiókot létrehozni. Miután a felhasználó minden adatokat megadta, a választásuktól függően egy újabb oldal jelenik meg, ahol további információkat kell megadniuk. Ha munkavállalóként léptek be, akkor olyan információkat kell megadniuk, mint a telefonszámuk, portfóliójuk, vezeték- és keresztnévük, profilképük, beszélt nyelvük, iskolai végzettségük és annak helye, egy rövid bemutatkozás, és az előző munkahelyük. A munkáltatótól pedig olyan adatokat kérünk be mint a cégnak a neve, a cégnak a leírása, a cég logója, a székhelye, mérete, és a cég típusa. A belépéshoz szükséges az email cím és jelszó, majd a "Belépés" -re kattintva vagy az "Enter" lenyomásával megjelenik a főoldal.





3.3. Főoldal és tartalma

A főoldal minden felhasználó típusnál megegyezik. A főoldalon egy motivációs idézetet olvashatnak a felhasználók, a főoldalról lehetőség lesz átmenni a hirdetések oldalra, a bejelentkezés és regisztrációs oldalra, és az idő hiánya miatt nem befejezett blog oldalra.



3.4. Funkciók ismertetése

3.4.1 Hirdetések feladása/posztolása

Munkáltatók képesek lesznek hirdetéseket létrehozni poszt formájában. A cég profiljánál megfognak jelenni az állás lehetőségek. Továbbá a regisztrációjukat követően lesz lehetőség megadni a cégeknek az adatai, mint például a cégek neve, a cégről egy ismertető, a cégek székhelye, logója és sorolhatnám még

The screenshot shows a web application interface for job postings. At the top, there's a navigation bar with links for 'Állásajánlatok', 'Blog', 'Önéletrajz', 'Bejelentkezés!', and 'Regisztráció!'. Below the navigation, there's a sidebar on the left with icons for 'Pályaválasztó' (Profile), 'Állásajánlatok' (Jobs), 'Blog', 'Kapcsolat' (Contact), and 'Önéletrajz' (CV). The main content area features a job listing for a position titled 'Valami' with a salary range of '200.000Ft - 400.000Ft' and a note about working conditions. To the right of the job listing is a purple sidebar with a button to 'Fájl kiválasztása/módosítása' (File selection/modification) and a note: 'Ahhoz hogy szerkeszthesd a Hirdetés kártyát, kattints a szerkeszteni kívánt mezőre!' (To edit the advertisement card, click on the field you want to edit!). Below these are two buttons: 'ÁLLÁSHIRDETÉS MENTÉSE' (Save Advert) in green and 'MÉGSEM' (Nevermind) in red. A large central box is titled 'A Hirdetés főoldalának szerkesztése' (Edit the main page of the advertisement) and contains fields for 'Munka leírása' (Job description), 'Követelmények a munkavégzéshez' (Requirements for performance), 'Amit kínálunk / Munkahelyi extrák' (What we offer / Workplace extras), 'Minimum bér' (Minimum salary) set to '200.000Ft', 'Maximum bér' (Maximum salary) set to '400.000Ft', and 'Salgótarján' (Location).

3.5. Munkavállalótól Regisztrációs további adatok bekérése

További adatok

This is a registration form for additional information. It includes fields for 'First Name' (Last name), 'Portfolio' (Profile picture), 'Last Name' (Last name), 'Profilkép feltöltése' (Upload profile picture) with options for 'Fájl kiválasztása' (File selection) or 'Nincs fájl kiválasztva' (No file selected). There are also fields for 'Telefonszám' (Phone number), 'Előző munkahelyek és pozíciók' (Previous work experience and positions), 'Tanult nyelvek:' (Languages learned), 'Adja meg előző munkahelyét, pozícióját' (Provide details of previous employment, position), 'Legmagasabb végzettség:' (Highest level of education), 'Szakmai végzettség' (Professional qualifications), and 'Adja meg szakmai végzettségét' (Provide professional qualifications). A large blue 'Tovább' (Next) button is at the bottom.

3.6. Céges Regisztrációs további adatok bekérése

The screenshot shows a registration form titled "További adatok" (Additional data) with a cityscape background. The form includes fields for company name, physical address, phone number, logo upload, company size/type, sector/industry, website URL, and a brief description of the company. A "Tovább" (Next) button is at the bottom.

A profilba lesz egy keret amin belül tud a felhasználó képet vagy fájlt feltölteni az önéletrajzárról amelyet minden más fájlt a projektekben base64-é konvertálva text-be tároljuk el az adatbázisban.

3.7 Profilok

Egyszerre 2 profil is lehet, regisztrációkor ki lehet választani hogy magánszemélyként szeretnél-e regisztrálni vagy cég/munkáltató kent.

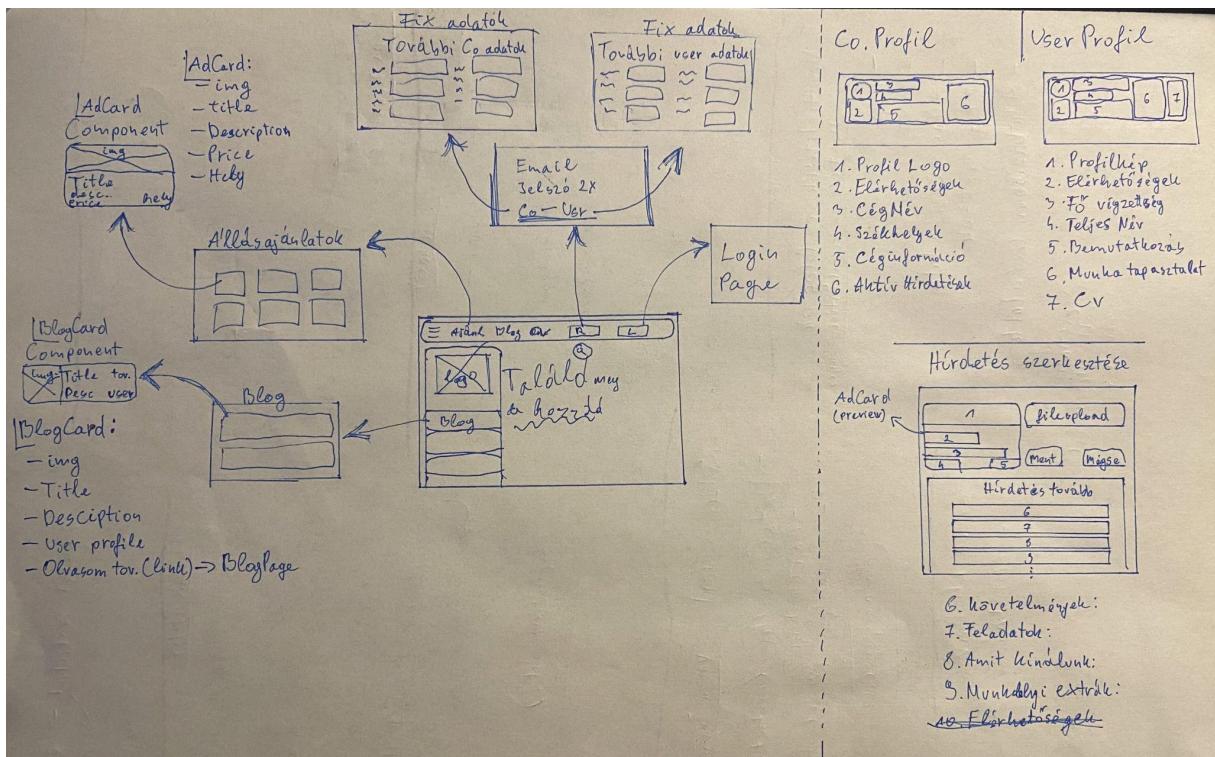
3.7.1 Munkáltatói oldalról

A munkáltató képes adatokat megadni a cégről, mint például helyszín, neve, foglalkoztatási köre illetve tud egy kis bemutatkozást is írni.

3.7.2 Magánszemélyként

Magánszemélyként a profilunk felér egy portfolióval. Megtudunk adni magunkról minden fontos információt arról hogy kik vagyunk mi. Önéletrajzot is tudunk feltölteni illetve képesek lesznek videókat, fényképeket is feltölteni magukról és a munkáikról is.

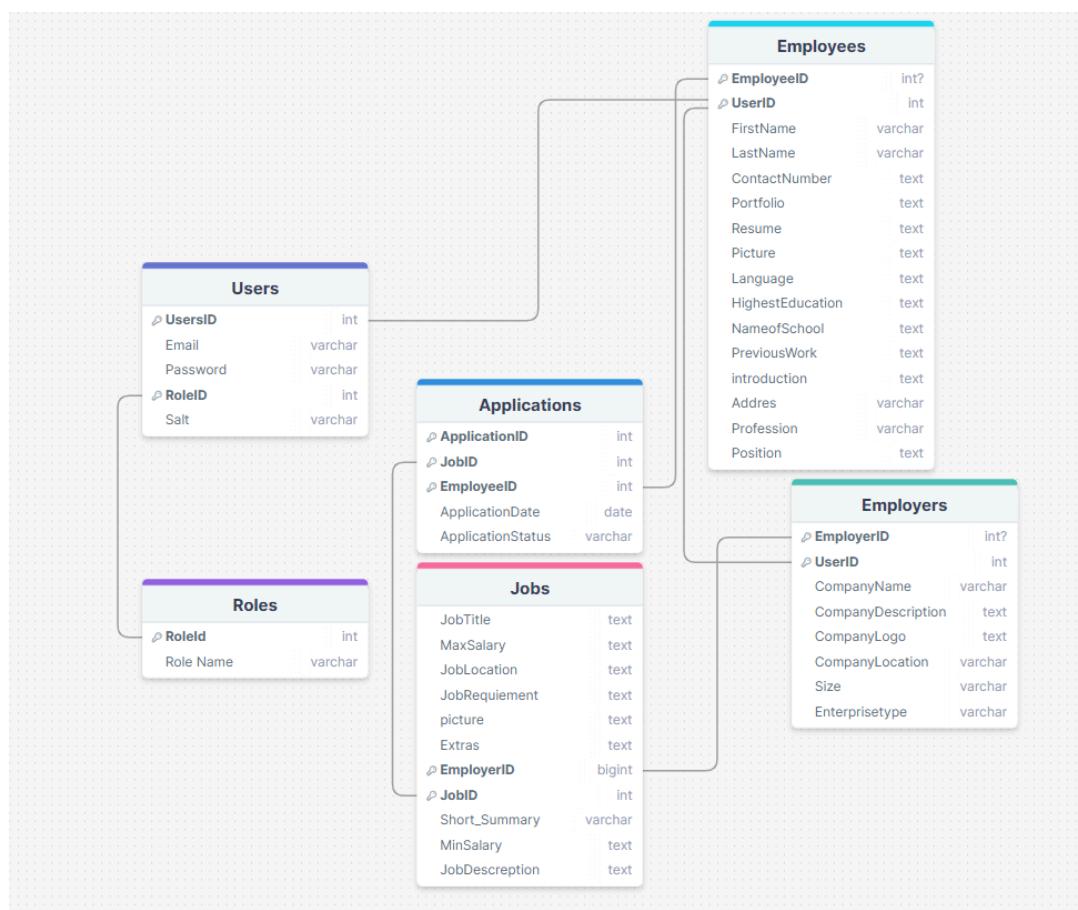
3.7.3 Kézzel rajzolt oldalak ismertetése



Ezen a képen jól látható melyik oldal hova navigál el és hogy hol minden információk lesznek beleérve a felhasználóktól valamint az is hogy minden adatot jelenítünk meg az adatbázisból.

4. Fejlesztői dokumentáció

4.1. Az adatbázis táblái és kapcsolatai



4.1.1 Users tábla

- A felhasználók tárolására szolgáló tábla.

Mezőnevek	Típus
UserID	INT,PK,NN
Salt	VARCHAR(255)
Password	VARCHAR(255)
Email	VARCHAR(255)
RoleID	INT

4.1.2 Applikáció tábla

- Az álláshirdetések jelentkezéseit tárolja

Mezőnevek	Típus
ApplicationID	INT,PK,NN
JobID	INT
EmployeeID	INT

ApplicationDate	DATE
ApplicationStatus	VARCHAR(255)

4.1.3 Employees tábla

- Employees adatait tárolja

Mezőnevek	Típus
EmployeeID	INT
UserID	INT
FirstName	VARCHAR(255)
LastName	VARCHAR(255)
ContactNumber	VARCHAR(255)
Portfolio	BLOB
Resume	BLOB
Language	TEXT
HighestEducation	TEXT
NameofSchool	TEXT
PreviousWork	TEXT
Introduction	TEXT
Address	VARCHAR(255)
Profession	VARCHAR(255)
Picture	TEXT
Position	TEXT

4.1.4 Jobs tábla

- Tárolja a hirdetett állások adatait

Mezőnevek	Típus
JobID	INT,PK,NN
EmployerID	INT
JobTitle	VARCHAR(255)
JobDescription	TEXT
JobRequirements	TEXT
JobLocation	VARCHAR(255)
JobSalary	INT
Picture	TEXT
MinSalary	TEXT
MaxSalary	TEXT
Short_Summary	VARCHAR(30)

4.1.5 Employer tábla

- Tárolja az állást hirdető cégek adatait

Mezőnevek	Típus
EmployerID	INT,PK,NN
UserID	INT
CompanyName	VARCHAR(255)
CompanyDescription	TEXT
CompanyLogo	BLOB
Size	VARCHAR(45)
Enterprisetype	VARCHAR(45)
CompanyURL	Text
ContactNumber	

4.1.6 Role tábla

- Ez a tábla segít elkülöníteni a jogosultságokat a felhasználók között.

Mezőnevek	Típus
RoleID	INT,PK,NN
RoleName	VARCHAR(255)

4.3. Metódusok, függvények és események

A Regisztrációt mi így oldottuk meg:

A FromUserRegistrationModelToUserMode metódus egy UserRegistrationDto objektumot vesz bemenetként, és egy user objektumot ad vissza. A metódus létrehoz egy új salt-ot, ami egy biztonsági intézkedés a jelszavak tárolásához. Ezután létrehoz egy új user objektumot, amelyben az Email mező értéke a UserRegistrationDto Email mezőjének értékét veszi fel. A Password mező értéke a UserRegistrationDto Password mezőjének értékét veszi fel, miután ezt a jelszót hash-elte a salt-tal. A salt mező értéke pedig a salt értékét veszi fel Base64 string formátumban.

A RegisterNewUserAsync metódus először ellenőrzi, hogy a megadott email című felhasználó már létezik-e az adatbázisban. Ezt az _worldDbContext.users.Any(_ => _.Email.ToLower() == userRegistration.Email.ToLower()) kifejezéssel teszi, ami true értéket ad vissza, ha a felhasználó már létezik, és false értéket, ha nem.

Ha a felhasználó még nem létezik, akkor a metódus létrehoz egy új user objektumot a FromUserRegistrationModelToUserMode(userRegistration) metódus segítségével, amely a UserRegistrationDto objektumot alakítja át user objektummá.

Az új user objektumot hozzáadják az adatbázishoz a `_worldDbContext.users.Add(newUser)` kifejezéssel. Végül a metódus frissíti az adatbázist az `await _worldDbContext.SaveChangesAsync()` kifejezéssel, ami elmenti az új felhasználót az adatbázisban.

Ha a felhasználó regisztrációja sikeres volt, akkor a metódus (`true, "Siker"`) értéket ad vissza. Ha a felhasználó már létezik, akkor (`false, "Az Email cím már létezik"`) értéket ad vissza. Ez a metódus aszinkron, ami azt jelenti, hogy nem blokkolja a program többi részét, míg várakozik az adatbázis műveletek befejezésére. Ezáltal a program többi része is folytatódhat, míg az adatbázis műveletek futnak. Ez a folyamat biztosítja, hogy a jelszavakat biztonságosan tároljuk és ellenőrzük. Ezért a salt-ot is el kell tárolni a felhasználóval együtt, hogy a jelszó ellenőrzésekor újra hash-elhessük a beírt jelszót a salt-tal, és összehasonlíthassuk a tárolt hash-elt jelszóval.

A bejelentkezést pedig így:

A `LoginUserAsync` metódus egy `UserLoginDto` objektumot vesz bemenetként, és végrehajtja a felhasználó bejelentkezési folyamatát. A metódus lépései a következők:

Felhasználó keresése: A metódus először megpróbálja megtalálni a felhasználót az adatbázisban az email cím alapján. Ezt az `_worldDbContext.users.FirstOrDefault(u => u.Email.ToLower() == userLogin.Email.ToLower())` kifejezéssel teszi.

Felhasználó létezésének ellenőrzése: Ha a felhasználó nem található az adatbázisban, akkor a metódus (`false, "A felhasználó nem található"`) értéket ad vissza.

Jelszó ellenőrzése: Ha a felhasználó létezik, akkor a metódus kinyeri a salt-ot a felhasználó adatbázisban tárolt adataiból, majd hash-eli a beírt jelszót a salt-tal. Ha a hash-elt jelszó nem egyezik meg a felhasználó adatbázisban tárolt jelszavával, akkor a metódus (`false, "Hibás jelszó"`) értéket ad vissza.

JWT token generálása: Ha a felhasználó létezik és a jelszó helyes, akkor a metódus generál egy JWT (JSON Web Token) tokent a felhasználó számára a `_jwtService.GenerateJwtToken(user)` kifejezéssel. A JWT token egy biztonságos módszer a felhasználó azonosítására a szerver és a kliens között.

Visszatérési érték: Ha a bejelentkezés sikeres volt, akkor a metódus (`true, jwtToken`) értéket ad vissza, ahol a `jwtToken` a generált JWT token. Ha a bejelentkezés sikertelen volt, akkor a metódus (`false, "A felhasználó nem található"`) vagy (`false, "Hibás jelszó"`) értéket ad vissza.

4.4. Osztályok és Modellek

A modelek teszik lehetővé az adatbázis adatainak tárolását a programon belül minden táblának megvan a megfelelő modelje, és dtoja(**Data Transfer Objectje**) amelyeket az adatok fogadására használtunk a klienstől(POST,PUT,GET,DELETE)

4.5. Radzen.Blazor-os komponensek

Mivel szakdolgozatunk során Blazort használtunk, így miután a konzulensünk megmutatta nekünk a Radzen-es komponenseket így elengedhetetlennek gondoltuk az ezeknek a használatát, ezen komponenseket használtuk a frontend elkészítése során

-login komponens ennek segítségével nem kellett nekünk külön egy teljes login oldalt írnunk, hanem csak ezt az egy komponenst kellett beletenni a programunkba, és az API-val való összekötése se volt nehéz

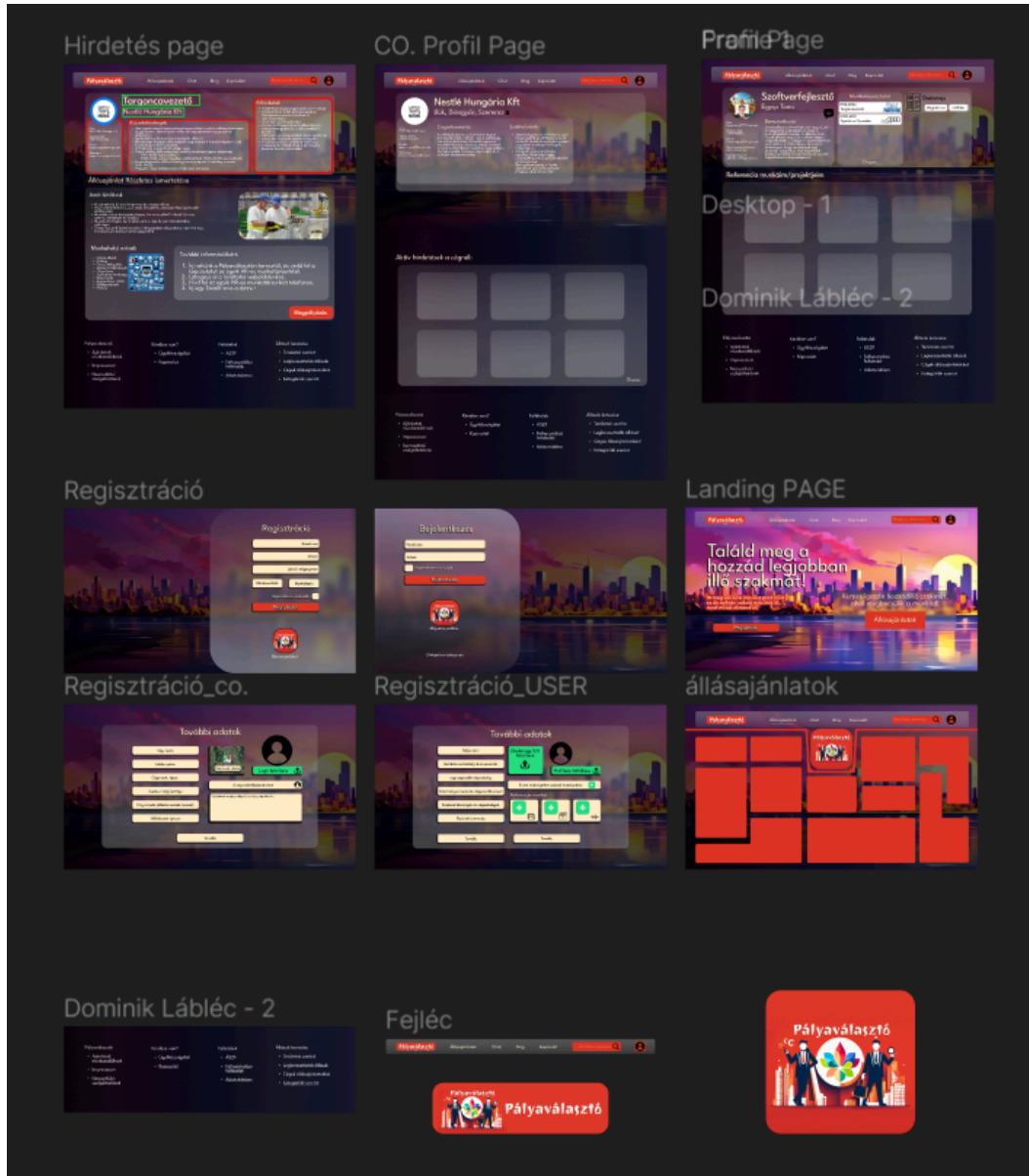
-Radzen Upload: enélkül minden bizonnal nehézkesnek bizonyult volna a file feltöltése megcsinálása, de így nagyon egyszerű volt, az itt kapott filet, base64-é konvertálva tároltuk el az adatbázisban

Radzen Searchbar: Ez sajnos idő hiányában nem lett befejezve, de az eredeti terv az lett volna, hogy a felhasználók tudnak keresni a feltöltött munkák tartalma alapján

4.6. Tesztdokumentáció

A tesztelés a fejlesztéssel párhuzamosan ment, majd elkészültekor többször kipróbáltuk a működését többféle helyzetben, és többféle operációs rendszeren, továbbá több ismerősünkkel is kipróbáltattuk és megkérdeztük, hogy mit gondolnak a mi álláskereső oldalunkról és mindegyiküknek tetszett, dicsérték a letisztultságát az oldalnak

5. Az alkalmazás design-ja



A képen azok a Figmában készült tervezetek láthatók amiket még az elején igazítottunk a változtatott ötlethez részben viszont itt is már botlottunk olyan akadályokba hogy olyan mozaikos kinézetű álláshirdetés oldal mint ami a harmadik oszlop harmadik sorában található a képen egyszerűen nem megvalósítható vagy ha igen akkor sem lett volna rá idő. A logókat AI képgenerátorral csináltuk amit aztán utánna csinosítottunk egy kis photoshop munkával. Egy pár elem nem teljesen úgy néz ki a gyakorlatban, mint ahogyan azt az elején elterveztük, de vannak olyan részek ahol jól látható hogy megmaradt a formalitás mert az elejtől tetszett az a megoldás mindenkinél, például a Login/Registration oldalak. Megpróbáltuk azt megvalósítani amit tervezünk, ez design szempontból sikerült is hiszen minden beviteli mező minden cím és bekezdés a figma design alapján a gyakorlatban is ott van ahol lennie kell. Megpróbáltuk a teljes designt minél inkább úgy kialakítani hogy egyszerűen kezelhető és érthető valamint interaktív legyen egybe az egész felület.

6. Összefoglalás

6.1. A szakdolgozat célja

A szakdolgozatunkban bemutatott web-alkalmazás célja, hogy összekössön munkaadókat és munkavállalókat, és segítsen nekik megtalálni a számukra legmegfelelőbb munkát vagy munkaerőt. Az alkalmazásunk egy olyan platformot kínál, ahol a munkaadók és a munkavállalók egymásra találhatnak, és ahol a munkaadók képesek megtalálni a számukra legideálisabb munkaerőt.

Ezt úgy valósítottuk meg, hogy a munkaadók véleményt tudnak írni a náluk dolgozókról. Ez a funkció segíti a többi céget abban, hogy kialakuljon egy kép a potenciális munkavállalóról. A vélemények alapján a munkaadók jobban megismerhetik a munkavállalók képességeit, tapasztalatait és munkamorálját, ami segíthet nekik a döntésben, hogy kit alkalmazzanak.

Ugyanakkor ez az állást kereső fél számára is előnyös. Minél jobb a munkavállaló visszajelzése, annál nagyobb a valószínűsége annak, hogy más cégek előbb adnak neki állást. A pozitív visszajelzések növelhetik a munkavállaló esélyeit a munkaerőpiacon, és segíthetnek neki abban, hogy gyorsabban találjon munkát.

Összefoglalva, a szakdolgozatunkban bemutatott web-alkalmazás egy olyan platformot kínál, ahol a munkaadók és a munkavállalók egymásra találhatnak, és ahol minden fél számára előnyös lehetőségeket kínálunk. Célunk, hogy segítsünk a munkaadóknak a legmegfelelőbb munkaerő megtalálásában, és a munkavállalóknak abban, hogy minél előbb találjanak munkát. Ezáltal hozzájárulunk a munkaerőpiac hatékonyabb működéséhez.

6.2. Megvalósítása

A munkánkat először is nagyon sok ötleteléssel és megbeszéléssel kezdtük, maga az ötlet is többször változott a tervezési folyamat során, majd végül az álláskereső webalkalmazásra esett a választásunk, majd ezután jött az oldal designjának az elkészítésével figmában, majd miután a több hét designolása véget ért jött a technológiák kiválasztása eleinte minden a frontendnél minden a backendnél több nyelv is szóba került végső során arra a döntésre jutottunk, hogy a frontendet Blazorban a backendet pedig A.S.P.Net-ben készítettük el, annak ellenére, hogy a Blazor lehetőséget ad a frontend és a backend fejlesztésre is (full stack) mégsem a Blazor backendjére esett a választásunk mivel az A.S.P.Net könnyebben illetve átláthatóbbnak tűnt, miután sikerült dőlőre jutni a használandó technológiákat illetően megkezdtük az adatbázist megtervezését MySQL Workbenchbe, miután ezek megvoltak hozzá láttunk a kódoláshoz is, az első oldal amit megcsináltunk az a Regisztrációs illetve a

Bejelentkezős oldal volt, mivel úgy gondoltuk, hogy ezzel lesz majd a legtöbb munka a backendben a jelszó titkosítással amely a következő lépésekkel áll:

Só generálása: A salt változó egy véletlenszerűen generált byte tömb, amit a jelszó hasheléséhez használnak. A sót minden felhasználóhoz külön generálják, hogy még ha két felhasználó ugyanazzt a jelszót használja is, a hashelt jelszavuk mégis különböző legyen. Ez növeli a rendszer biztonságát.

1. Jelszó hashelése: A HashPassword metódusban a bemeneti jelszót (plainPassword) és a sót használják a jelszó hasheléséhez. Itt használják az Rfc2898DeriveBytes osztályt, ami a PBKDF2 (Password-Based Key Derivation Function 2) algoritmust valósítja meg. Ez az algoritmus képes "sözött" hash értéket generálni a bemeneti jelszóból.
2. Hash és só összefűzése: A hashelt jelszót és a sót egyetlen byte tömbbe fűzik össze (passwordHash). Először a sót másolják a tömbbe, majd a hashelt jelszót. Ezután a byte tömböt Base64 formátumú stringgé alakítják, ami az adatbázisban tárolódik.

Ez a módszer biztosítja, hogy még ha valaki hozzáérne az adatbázisban tárolt jelszavakhoz, akkor sem tudná könnyen visszafejteni az eredeti jelszavakat. A sózás és a PBKDF2 algoritmus használata miatt a jelszó visszafejtése jelentős számítási erőforrást igényelne, ami tovább növeli a rendszer biztonságát.

A rendszerünkben a jogosultságok kezelését token alapú hitelisítéssel oldottuk meg. Ez lehetővé teszi számunkra, hogy korlátozzuk a hozzáférést bizonyos API végpontokhoz. Például csak a regisztrációs és bejelentkezős API végpontok érhetők el bárki számára, míg a többi végpont hozzáférése jogosultságokhoz van kötve. A tokenek generálását a JwtService osztály segítségével valósítottuk meg. A JwtService konstruktőrben beállítjuk a konfigurációt (_configuration), amit később felhasználunk a JWT kulcs (Jwt:Key) beolvásáshoz."

JWT generálása: A GenerateJwtToken metódusban generálják a JWT-t. Először létrehoznak egy JwtSecurityTokenHandler példányt, amit a token generálásához használnak. Ezután beolvassák a JWT kulcsát a konfigurációból, és byte tömbbe alakítják.

Token leíró: Létrehoznak egy SecurityTokenDescriptor példányt, amely leírja a generálandó tokent. Beállítják a token "tárgyát" (Subject), ami a tokenhez rendelt igényeket (claims) tartalmazza. Ebben az esetben egyetlen igényt adnak hozzá, amely a felhasználó e-mail címét tartalmazza. Beállítják a token lejáratát idejét (Expires), ami jelen esetben a jelenlegi időponttól számított 7 nap múlva lesz. Végül beállítják a token aláírási kulcsát (SigningCredentials), amit a token aláírásához használnak.

Token generálása és visszaadása: A JwtSecurityTokenHandler CreateToken metódusával létrehozzák a tokent a token leíró alapján, majd a WriteToken metódussal string formátumúvá alakítják a tokent, és visszaadják.

Miután ezzel végeztünk megcsináltuk minden táblának az API routeját amelyekhez létrehoztunk interfaceseket meg repositorykat

repository:-repositoriumot valósít meg, amely lehetővé teszi az entitásokkal kapcsolatos

műveletek végrehajtását, mint például létrehozás, olvasás, frissítés és törlés (CRUD műveletek).

interface: Az interfész meghatározza a műveleteket, amelyeket az alkalmazásokkal, munkákkal és alkalmazottakkal kapcsolatos adatbázis-repozitóriumnak meg kell valósítania. Ez magában foglalja az adatlekérési, létrehozási, frissítési, törlési és mentési műveleteket.

A kontrollereket ezek segítségével oldottuk meg MVC módszerrel.

Miután megvoltak a kontrollerek hozzá láttunk a frontedes oldalak folytatásához, sajnálatos módon idő hiányában sajnos az eredeti **ötletünkön** kicsit vissza kellett venni ugyanis, az álláshirdetések létrehozása mellé még akartunk egy blog oldalt is létrehozni, ahol szintúgy állásokat lehetett volna olvasni csak egy blog formájában.

A program készítése során több kisebb vagy nagyobb akadályba ütközöttünk amit így vagy úgy sikerült leküzdeni, amikor először nekiálltunk az API megcsinálásnak akkor eleinte auto mapper-el próbáltuk megoldani, de mivel az a megoldás nehéznek bizonyult így nem azt választottuk, továbbá nehézségeket okozott a tokenes autentikáció és az API szerver összekötése a frontend oldalal, de mindezeket sikeresen letudtuk a segítőkész konzulenseknek köszönhetően, az Ő segítségének köszönhetjük azt is, hogy az előre eltervezett featurek nagy részét sikerült implementálni

6.3. Fejlesztési lehetőségek

- Integrált önéletrajz generátor
- Állásajánlatok személyiségteszt alapján
- Blogok létrehozása



7. Irodalomjegyzék

7.1. Internetes Irodalomjegyzék

A weboldalak és videók amik segítettek megoldani a váratlan problémákat.

7.1.1 Programozási problémák megoldásai:

- https://www.youtube.com/watch?v=nVK14WnFcs0&list=PLnEA_YD8TqY3W3VJ_QBs8Cb2xU0Xte_1QS
- <https://chat.openai.com/>
- <https://www.radzen.com/documentation/blazor/get-started/>
- <https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-8/overview>
- <https://stackoverflow.com/>
- <https://www.geeksforgeeks.org/how-to-approach-a-coding-problem/>
- <https://www.w3schools.com/cs/index.php>
-

7.1.2 C#-os dokumentáció:

<https://adrenalinagol.app/>

7.1.3 A.S.P.NET-es dokumentáció:

<https://learn.microsoft.com/en-us/aspnet/web-api/>

7.1.4: Blazoros dokumentáció:

<https://learn.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-8.0>

7.1.5: blazor.radzen komponensek:

<https://blazor.radzen.com/>

8. Mellékletek (Programkódok)

8.1.1 A regisztráció kódja :

```
public class UserService : IUserService
{
    private readonly MyWorldDbContext _worldDbContext;
    private readonly IJwtService _jwtService;
    public UserService(MyWorldDbContext worldDbContext, IJwtService jwtService)
    {
        _worldDbContext = worldDbContext;
        _jwtService = jwtService;
    }

    private User FromUserRegistrationModelToUserMode(UserRegistrationDto
userRegistration)
    {

        byte[] salt = new byte[16];
        using (var rng = RandomNumberGenerator.Create())
        {
            rng.GetBytes(salt);
        }

        return new User
        {
            Email = userRegistration.Email,
```

```

        Password = HashPassword(userRegistration.Password, salt),
        salt = Convert.ToBase64String(salt)
    );
}

public async Task<(bool isUserRegistered, string Message)>
RegisterNewUserAsync(UserRegistrationDto userRegistration)
{
    var isUserExist = _worldDbContext.users.Any(_ => _.Email.ToLower() ==
userRegistration.Email.ToLower());

    if (isUserExist)
    {
        return (false, "Az Email cím már létezik");
    }

    var newUser = FromUserRegistrationModelToUserMode(userRegistration);
    _worldDbContext.users.Add(newUser);
    await _worldDbContext.SaveChangesAsync();
    return (true, "Siker");
}

```

8.1.2 Bejelentkezés kódja

```

public async Task<(bool isUserLoggedIn, string jwtToken)> LoginUserAsync(UserLoginDto
userLogin)
{
    var user = _worldDbContext.users.FirstOrDefault(u => u.Email.ToLower() ==
userLogin.Email.ToLower());

    if (user == null)
    {
        return (false, "A felhasználó nem található");
    }

```

```

    }

    var salt = Convert.FromBase64String(user.salt);
    var hashedPassword = HashPassword(userLogin.Password, salt);
    if (user.Password != hashedPassword)
    {
        return (false, "Hibás jelszó");
    }

    var jwtToken = _jwtService.GenerateJwtToken(user);
    return (true, jwtToken);
}

```

8.1.3 Ez a kód részlet az alkalmazás hitelesítési rendszerét állítja be a JWT (JSON Web Token) használatával :

```

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(options =>
{
    options.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidateLifetime = true,
        ValidateIssuerSigningKey = true,
        ValidIssuer = builder.Configuration["Jwt:Issuer"],
        ValidAudience = builder.Configuration["Jwt:Audience"],
        IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"]))
}

```

```
};  
});
```

8.1.4 Token generálása

A RegistrationLoginController a nevéből eredően a regisztrációért és a bejelentkezést felelős, azért lett ez külön véve mivel ez az kettő api route-ot amit mindenkinél el kell érnie

```
using JWTAuth.API.Dtos;  
using Microsoft.AspNetCore.Authorization;  
using Microsoft.AspNetCore.Mvc;  
using Palyavalaszto.Services;  
  
namespace Palyavalaszto.Controllers  
{  
    [AllowAnonymous]  
    [Route("api/[controller]")]  
    [ApiController]  
    public class UserRegistrationLoginController : ControllerBase  
    {  
        private readonly IUserService _userService;  
        public UserRegistrationLoginController(IUserService userService)  
        {  
            _userService = userService;  
        }  
        [HttpPost("register")]  
        public async Task<IActionResult> UserRegistration(UserRegistrationDto userRegistration)  
        {  
            var result = await _userService.RegisterNewUserAsync(userRegistration);  
            if (result.isUserRegistered)  
            {
```

```
        return Ok(result.Message);

    }

    ModelState.AddModelError("Email", result.Message);

    return BadRequest(ModelState);
}

[AllowAnonymous]

[HttpPost("login")]

public async Task<IActionResult> Login(UserLoginDto userLogin)

{
    var result = await _userService.LoginUserAsync(userLogin);

    if (result.isUserLoggedIn)
    {

        return Ok(new { Token = result.jwtToken });
    }

    else
    {

        Console.WriteLine("Sikertelen bejelentkezés a következő felhasználónével: " +
userLogin.Email);

        ModelState.AddModelError("Email", result.jwtToken);

        return BadRequest(ModelState);
    }
}

}
```

A token legenerálásáért felelős kód

```
using Microsoft.IdentityModel.Tokens;
using Palyavalaszto.Data.Entitites;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

namespace Palyavalaszto.Services
{
    public class JwtService : IJwtService
    {
        private readonly IConfiguration _configuration;

        public JwtService(IConfiguration configuration)
        {
            _configuration = configuration;
        }

        public string GenerateJwtToken(user users)
        {
            var tokenHandler = new JwtSecurityTokenHandler();
            var key = Encoding.ASCII.GetBytes(_configuration["Jwt:Key"]);
            var tokenDescriptor = new SecurityTokenDescriptor
            {
                Subject = new ClaimsIdentity(new Claim[]
                {
                    new Claim(ClaimTypes.Email, users.Email.ToString())
                }),
            };
            var token = tokenHandler.CreateToken(tokenDescriptor);
            var jwtToken = tokenHandler.WriteToken(token);
            return jwtToken;
        }
    }
}
```

```
Expires = DateTime.UtcNow.AddDays(7),  
SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key),  
SecurityAlgorithms.HmacSha256Signature)  
};  
var token = tokenHandler.CreateToken(tokenDescriptor);  
return tokenHandler.WriteToken(token);  
}  
}  
}
```

