



Freelancer Weboldal

Készítette

Krigovszki Bálint

Programtervező Informatikus BSc

Témavezető

Dr. Tajti Tibor

Egyetemi Docens

EGER, 2023

Tartalomjegyzék

Bevezetés	4
1. A rendszer alapjai	5
1.1. React.js frontend	5
1.2. Express.js és Node.js szerverszint	5
1.3. MongoDB adatbázisszint	6
1.4. Material UI	6
1.5. Csomagok	6
1.5.1. Axios	6
1.6. Middleware	7
1.6.1. Proxy	7
2. Telepítési útmutató	8
2.1. Technológiák és szükséges csomagok	8
3. Felhasználói felület	9
3.1. Főoldal	9
3.1.1. Header	9
3.1.2. Sidebar	10
3.1.3. Container	10
3.2. Bejelentkezés	11
3.3. Regisztráció	11
3.4. User Dashboard	12
3.5. Admin Dashboard	13
4. Adatbázis	15
4.1. Adatbázis táblák ismertetése	15
4.1.1. Users	15
4.1.2. Jobs	16
4.1.3. Jobtypes	16

5. Frontend	17
5.1. Komponensek	17
5.2. Redux	17
5.2.1. Constant	18
5.2.2. Reducer	18
5.2.3. Action	18
6. Backend	20
6.1. Jogosultsági rendszer	20
6.2. Hitelesítés	20
6.3. Kontrollerek	21
6.4. Modellek	22
6.5. Router	23
7. Tesztelés	25
7.1. Postman	25
7.2. Manuális tesztelés	26
8. Továbbfejlesztési lehetőségek	29
8.1. Szűrés	29
8.2. Üzenet küldés	29
8.3. Hirdetések mentése	29
8.4. Felhasználói adatok	29
8.5. Jogosultsági kör kibővítése	30
8.6. Kiemelt és népszerű hirdetések	30
8.7. Legfrissebb álláshirdetések	30
Összegzés	31
Irodalomjegyzék	33

Bevezetés

Egyetemi tanulmányaim során lehetőségem akadt számos technológiával megismerkednem, melyek közül több is felkeltette az érdeklődésem, azonban tetszésemet leginkább a webes alkalmazások illetve weboldalak fejlesztése nyerte el. Ebből az érdeklődésből adódóan szakmai gyakorlatom során is ilyen munkakörben tevékenykedtem, ami csak még jobban elmélyítette a webfejlesztés területe iránti érdeklődésemet.

Már korábbi tanulmányaim során is megismerkedhettem a webprogramozással, azonban igazán csak az egyetemi tanulmányaim megkezdését követően tudtam még mélyebben beleásni magam a programozás ezen területébe.

A téma iránti érdeklődésem következtében megfelelőnek láttam ezt a szakdolgozati témát választani, hiszen a már megszerzett tudásomat is hasznosítani tudom, valamint a kutatómunkának hála rengeteg új ismerettel bővíthet a tudástáram a dolgozatom elkészítésének folyamatában.

A szakdolgozati téma kiválasztása után egy konkrétabb megvalósítandó cél után kezdtem kutatni. Néhány nap kutatómunka után arra jutottam, hogy a hazánkban működő álláshirdető, szabadúszó munkákat kínáló portálok kínálata elég szűk, valamint a kezelésük sem optimális felhasználóbarát szempontból.

Ezeket az észrevételeket figyelembe véve, egyértelművé vált számomra, hogy szakdolgozatom célja egy olyan álláshirdető portál létrehozása, amellyel az internetes munkavállalás és munka ajánlás sokkal gördülékenyebben történik, a weboldal megjelenése modern és felhasználóbarát, továbbá megkönnyíti a munkavállalók és munkáltatók közötti megállapodást.

A technológiák kiválasztása során figyelembe vettem az azokkal való tapasztalataimat illetve előzetes ismereteimet annak érdekében, hogy a szakdolgozatom tükrözze a tudásom ezen témakörökben.

https://github.com/krigovszkibalint/KDPEQ8_Szakdolgozat

1. fejezet

A rendszer alapjai

A weboldal alapjaiban egy MERN stack applikáció. A MERN a MEAN stack[1] (MongoDB, Express, Angular, Node) számos változatának egyike, ahol a hagyomás Angular.js frontend keretrendszert a React.js váltja fel.

A MERN architektúra lehetővé teszi egy háromszintű architektúra (frontend, backend, adatbázis) könnyű felépítését teljes egészében JavaScript és JSON használatával.

Az Express és a Node alkotják a középső (alkalmazási) szintet. Az Express.js egy szerveroldali webes keretrendszer, a Node.js pedig egy JavaScript kiszolgálóplatform.

1.1. React.js frontend

A MERN stack legfelső szintje a React.js[2], a deklaratív JavaScript-keretrendszer dinamikus kliensoldali alkalmazások létrehozásához HTML-ben. A React segítségével egyszerű komponenseken keresztül összetett felületeket hozhatunk létre, összekapcsolhatjuk azokat a háttérkiszolgálón lévő adatokkal, és HTML-ként jeleníthetjük meg őket. Tanulmányaim során több alkalommal is használtam React.js frontend-et projektek elkészítéséhez, ennek tudatában választottam ezt a keretrendszert az előzetes ismereteimre hagyatkozva.

1.2. Express.js és Node.js szerverszint

A következő szint az Express.js[3] szerveroldali keretrendszer, amely egy Node.js[4] kiszolgálón belül fut. Az Express.js hatékony modellekkel rendelkezik az URL-útválasztáshoz (a bejövő URL-cím és a kiszolgáló funkciójának egyeztetéséhez), valamint a HTTP-kérések és -válaszok kezelésére.

Ha XML HTTP-kérelmeket (XHR-eket), GET-eket vagy POST-okat küldünk a React.js frontend-ről, csatlakozhatunk az alkalmazást működtető Express.js-függvényekhez. Ezek a funkciók pedig a MongoDB[5] Node.js illesztőprogramjait használják callback-

ek vagy promise-ok segítségével a MongoDB adatbázisban lévő adatok eléréséhez és frissítéséhez.

1.3. MongoDB adatbázisszint

A React.js kezelőfelületén létrehozott JSON-dokumentumok elküldhetők az Express.js-kiszolgálóra, ahol feldolgozhatók és (feltéve, hogy érvényesek) közvetlenül a MongoDB-ben tárolhatók későbbi visszakeresés céljából.

Főbb tulajdonságai közé tartoznak: Ad hoc lekérdezések, azaz támogatja a keresést mező alapján, érték-tartomány alapján vagy reguláris kifejezéssel. A lekérdezések visszaadhatják a dokumentum egy meghatározott részét és tartalmazhatnak javascript funkciókat is. Indexek, tehát a dokumentum bármelyik mezője alapján tudunk indexet készíteni. Továbbá a replikáció, terheléelosztás, fájl tárolás, és az aggregáció.[6]

1.4. Material UI

A Material UI[7] egy nyílt forráskódú React komponens könyvtár, amely megvalósítja a Google Material Design rendszerét. Előre elkészített komponensek átfogó gyűjteményét tartalmazza, melyek azonnal használatra készek.

1.5. Csomagok

A szakdolgozat elkészítéséhez felhasznált nagyobb jelentőséggel bíró csomagok: *axios*, *formik*, *yup* (Sémakészítő a futásidejű értékek elemzéséhez és ellenőrzéséhez), *moment* (JavaScript könyvtár a dátumok elemzéséhez, ellenőrzéséhez, manipulálásához és formázásához), *http-proxy-middleware*, *redux* (Lehetővé teszi, hogy a React komponensek beolvassák az adatokat egy Redux tárolóból, és műveleteket küldjenek a store-ba az állapot frissítéséhez), *react-router-dom*, *morgan* (HTTP request naplózás), *bcrypt* (Jelszó hash-elés), *jsonwebtoken*, *cookie-parser*, *cors*, *mongoose*, *nodemon*.

1.5.1. Axios

Az Axios[8] egy promise alapú HTTP-kliens a Node.js-hez és a böngészőhöz. Izomorf (futhat a böngészőben és a node.js-ben ugyanazzal a kódbázissal). Szerver oldalon a natív node.js http modult használja, míg a kliensen (böngészőn) az XMLHttpRequest-et használja.

1.6. Middleware

A Redux[9] *middleware* funkciója médiumot biztosít az elküldött műveletekkel való interakcióhoz, mielőtt azok elérnék a reduktort. Testreszabott middleware-függvények hozhatók létre magas sorrendű függvények írásával (egy olyan függvény, amely egy másik függvényt ad vissza), amely valamilyen logikát vesz körül. Több middleware kombinálható új funkciók hozzáadásához, és egyik köztes szoftvernek sem kell tudnia arról, hogy mi volt előtte és utána.

A feltételes küldés a middleware-be írható. Minden middleware megkapja a *store* küldését, hogy új műveleteket küldhessen, a *getState* pedig argumentumként funkcionál, hogy hozzáférhessen az aktuális állapothoz és visszaadhasson egy függvényt. A belső függvény bármely visszatérési értéke elérhető lesz magának a küldési függvénynek az értékeként.

1.6.1. Proxy

A dolgozat elkészítésének folyamatában egy olyan, problémába ütköztem, melynek ki-
küszöbölésére manuálisan kellett beállítanom egy proxy-t, különben a backend és a frontend nem volt képes együttműködni ami összeomláshoz vezetett, így ennek érdekében a *http-proxy-middleware* csomagot használtam fel.

1.1. kód. Proxy beállítása

```
1  const { createProxyMiddleware } = require('http-proxy-  
    ↪ middleware');  
2  
3  module.exports = function(app) {  
4    app.use(  
5      '/api',  
6      createProxyMiddleware({  
7        target: 'http://localhost:9000',  
8        changeOrigin: true,  
9      })  
10   );  
11  };
```

2. fejezet

Telepítési útmutató

A következőkben a webalkalmazás üzembe helyezéséről és az ehhez szükséges erőforrásokról, technológiákról lesz szó.

2.1. Technológiák és szükséges csomagok

A projekt sikeres működéséhez és bővítéséhez elsősorban egy *Node.js* telepítés szükséges. Az adatbázis úgy van beállítva, hogy ahhoz minden IP címről hozzá lehet férni, így ezt nem szükséges tovább konfigurálni.

Továbbá szükségünk van még bizonyos csomagokra: *bcryptjs*, *body-parser*, *cookie-parser*, *cors*, *dotenv*, *express*, *jsonwebtoken*, *mongoose*, *nodemon*, *morgan*, *@mui/material*, *redux*, *react-redux*, *redux-thunk*, *axios*, *react-toastify*, *react-router-dom*, *formik*, *yup*, *@redux-devtools/extension*, *http-proxy-middleware*, *react-pro-sidebar*, *moment*, *@mui/x-data-grid*, *react-google-charts*

```
Szerver elindult, PORT: 9000
DB csatlakozás sikeres
GET /api/type/jobs 304 133.157 ms - -
GET /api/type/jobs 304 115.057 ms - -
GET /api/jobs/show/?pageNumber=1&keyword=&cat=&location= 200 463.349 ms - 4292
```

2.1. ábra. Backend elindítása

A *backend* és a *frontend* elindítása az *npm start* paranccsal történik terminálból. Elsősorban a backend sikeres elindítása után két üzenet jelenik meg a terminálban, az első „Szerver elindult: PORT: XXXX”, a második pedig a DB csatlakozás. A futás közben a terminálon láthatók a HTTP kérések, URL-ek, valamint a státusz kódok és a műveletekhez felhasznált idő.

3. fejezet

Felhasználói felület

Az applikáció felhasználói felülete a Material UI segítségével valósult meg, mellyel lehetőségem volt könnyedén és viszonylag gyorsan modern, letisztult felhasználóbarát kezelőfelületet készíteni amin könnyen el lehet igazodni. A felület megalkotása során a legfontosabb szempontok a letisztultság és praktikusság voltak.[10]

3.1. Főoldal

3.1.1. Header

A főoldal header részében található a navbar, mellyel navigálhatunk az oldalak között. Itt található egy user avatar, melyre rákattintva megjelenik egy menü három opcióval: *Admin Dashboard*, *User Dashboard* és *Bejelentkezés / Kijelentkezés* (Az aktuális státusztól függően). Amennyiben a felhasználó nincs bejelentkezve úgy csak a Bejelentkezés opció áll a rendelkezésére, mivel a másik kettő a főoldalra irányítja vissza, továbbá ha a felhasználó nem adminisztrátor, akkor jogkörébe tartozik a User Dashboard megtekintése, azonban az Admin Dashboard-hoz nem fér hozzá jogosultság hiányában.



3.1. ábra. Header komponens

A navbar alatt található a címsor és egy kereső sáv, ami kulcsszavak segítségével szűri a megjelenített munkákat és aktívan módosítja a lapszámozást a megjelenített találatok számától függően.

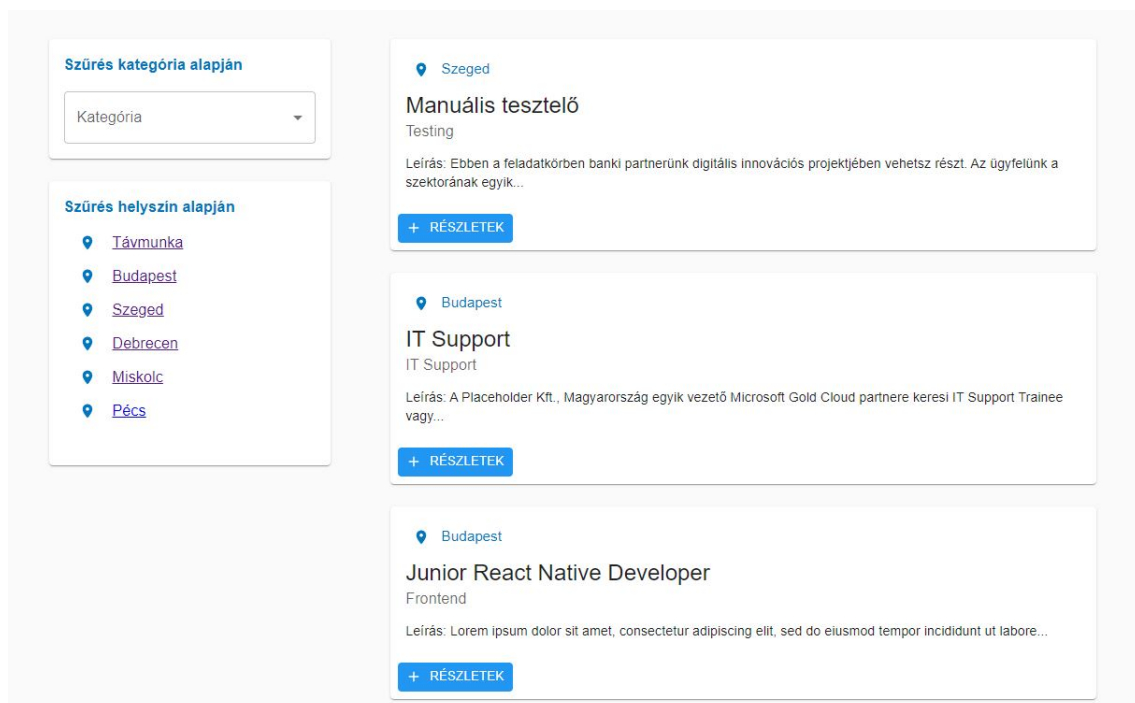
3.1.2. Sidebar

A sidebar komponens két részre bontható, mindkettő a kilistázandó álláshirdetések szűrésére szolgál. Az első részben egy dropdown menü található, melyre kattintva az összes az adatbázisban megtalálható állás kategória megjelenik, és bármelyiket kiválasztva módosul a megjelenített álláshirdetések listája az azonos kategóriába tartozókra.

A második komponens a sidebar-on belül lehetőséget kínál arra, hogy posztokat a munkavégzés helyszíne alapján szűrjünk meg, valamint a két szűrési opciót tudjuk kombinálni is, így egyszerre szűrhetünk helyszínre, kategóriára és a kereső sáv segítségével kulcsszavakra is.

3.1.3. Container

Ezen a komponensen belül jelennek meg az adatbázisból kilistázott állások. Időrendi sorrendben a legújabb hirdetés kerül az első helyre. Az egy oldalra eső hirdetések száma megszabható és a lapszámozás ennek függvényében változik.



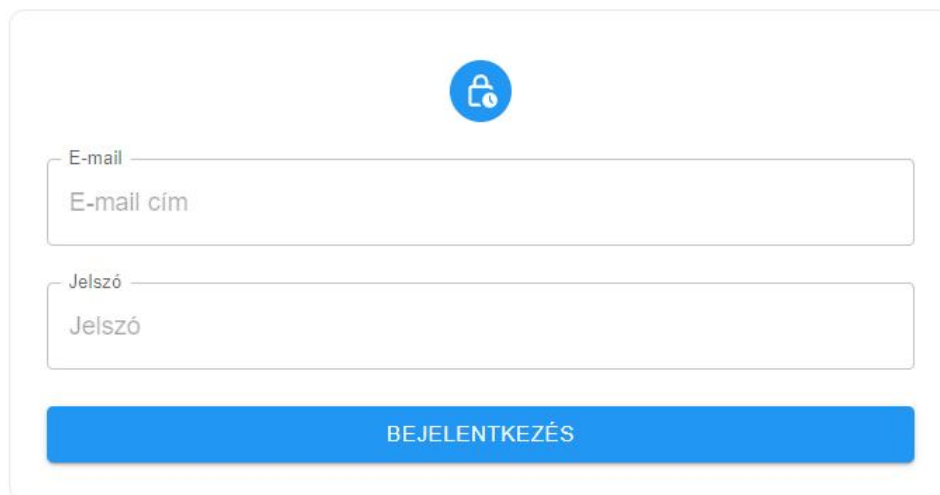
3.2. ábra. Sidebar és Container komponensek

A munkákkal kapcsolatban előnézetben látható a munkavégzés helye, a pozíció, a kategória, egy rövidített leírás és egy „Részletek” gomb amire kattintva tovább lép-

hetünk egy részletesebb leírás megtekintéséért, valamint itt már hirdetést létrehozó felhasználó által megszabott fizetés is láthatóvá válik a számunkra.

3.2. Bejelentkezés

Bejelentkezéshez a felhasználónak két adat megadására van szüksége, az első az e-mail címe amivel regisztrált a rendszerbe, a második a jelszava. A bejelentkezés form a *formik* csomag segítségével van ellenőrizve, azaz nem lehetséges megadni érvénytelen e-mail címet illetve a jelszónak is meg kell felelnie a követelményeknek. Az adatok sikeres bevitele után, a bejelentkezés gombra kattintást követően, a jogosultsági szinttől függően vagy a User Dashboard-ra vagy az Admin Dashboard-ra lesz átirányítva a felhasználó a további munkálatok érdekében.

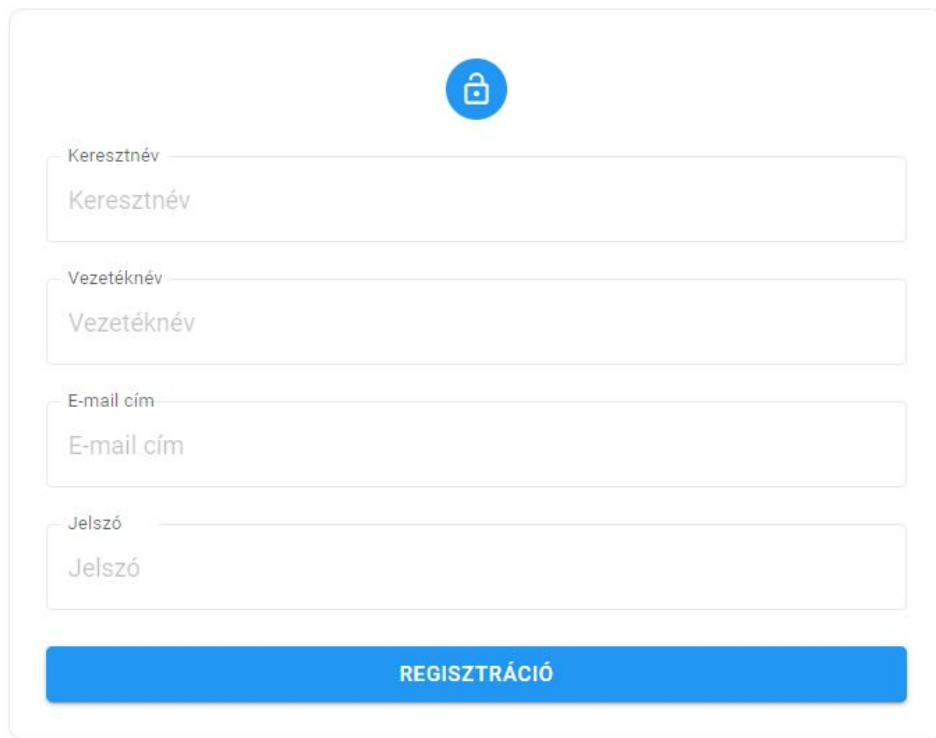
A bejelentkezési felület egy egyszerű, modern dizájnban. A felület közepén van egy kék zárolt ajtó ikon. Alatta két input mező található: az első "E-mail" címmel, a második "Jelszó" címmel. Mindkét mezőben a címetek szürke betűkben vannak megadva. Az input mezők alatt egy nagy, kék gomb van, amelyen fehér betűkkel "BEJELENTKEZÉS" van írva.

3.3. ábra. Bejelentkezési felület

Amennyiben hibás adatokat ad meg a felhasználó bejelentkezési form-on vagy a formai követelményeknek nem felelnek meg az általa begépeltek, azt a megfelelő hibaüzenet fogja jelezni a felhasználó felé.

3.3. Regisztráció

Regisztráció során a felhasználónak kötelező megadnia a keresztnévét, vezetéknévét, egy érvényes e-mail címet és egy legalább 6 karakterből álló jelszót. Amennyiben valamelyik feltétel nem teljesül nem jön létre új fiók és a felhasználó hibaüzenet formájában értesül.



The image shows a registration form within a light gray rounded rectangle. At the top center is a blue circular icon with a white padlock. Below it are four input fields, each with a label above it: 'Keresztnév' (First Name), 'Vezetéknév' (Last Name), 'E-mail cím' (Email address), and 'Jelszó' (Password). Each field contains a placeholder text matching its label. At the bottom of the form is a solid blue button with the white text 'REGISZTRÁCIÓ'.

3.4. ábra. Regisztrációs felület

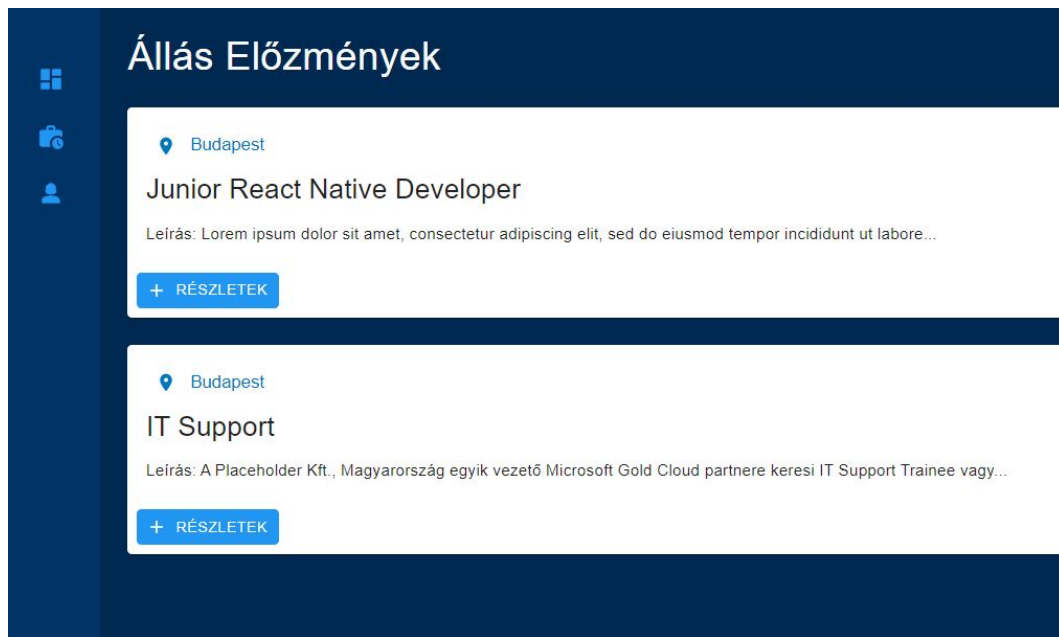
3.4. User Dashboard

Sikeres bejelentkezés eredményeképpen azt egy értesítés jelzi a jobb felső sarokban, a *toastify* csomagnak köszönhetően.

A User Dashboard szintén két nagyobb részre bontható, az egyik a sidebar a másik a container. A Dashboard-on belül a sidebar opcióival lehet navigálni, alapértelmezett módon az „Írányítópult” menübe kerül a felhasználó, ahol a Container-en belül láthatja regisztrációjának dátumát és azt, hogy hány álláshirdetést hozott létre a weboldalon.

A jelentkezett állások menüben azokat az álláshirdetéseket a listáját találhatja meg a felhasználó, amikre korábban jelentkezett. Az adatok menüpontra kattintva a Container-ben megtalálhatók a felhasználói fiókhoz tartozó személyes adataink, azaz kereszt és -vezetéknév, azonosításra használt e-mail cím és a státusz, vagyis a jogosultsági kör az oldalon.

Az utolsó opció az oldalsávon belül a kijelentkezés, mely a főoldalra navigálja a felhasználót.

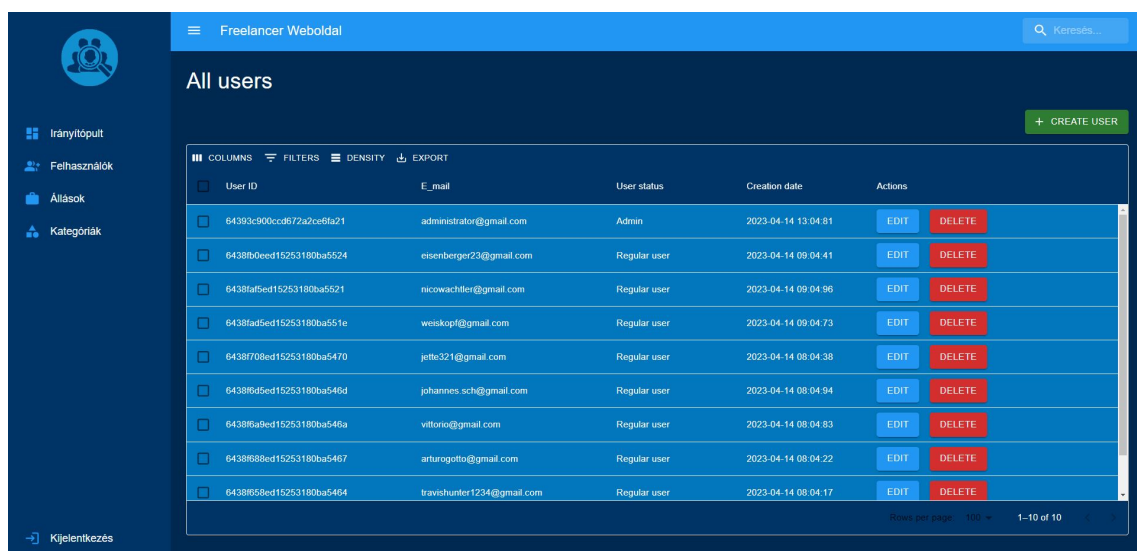


3.5. ábra. Állás előzmények

3.5. Admin Dashboard

Adminisztrátorként innen lehetséges menedzselni a weboldalon megjelenő tartalmakat, azaz a hirdetéseket, az állások kategóriáit és a felhasználókat is.

A felhasználókat ki lehet listázni egy táblázatba, ahol megjelenik az azonosítójuk, e-mail címük, a jogkörük, regisztrációjuk ideje, valamint az „Actions” oszlopban két gomb amikkel szerkeszthetők az adataik, illetve törölni is lehet egyes felhasználókat.



3.6. ábra. Admin Dashboard - Felhasználók

A táblázatot ezen kívül még lehet rendezni több szempont szerint is. Minden oszlopra lehetséges a növekvő illetve csökkenő rendezés, az oszlopokat el lehet rejtetni,

szűrni lehet kulcsszavakkal, a lapszámozást lehet módosítani és az adminisztrátor rendelkezésére áll még egy exportálás opció is, ami a felhasználó listát .csv formátumba exportálja. Mindezen funkciók a Material UI X Datagrid[11] komponensének köszönhetőek.

<input type="checkbox"/> ID ↓	Állás címe	Kategória	Felhasználó	Aktív	Fizetés	Actions
<input type="checkbox"/> 643902460ccd672a2ce6f8cc	Expert Frontend Engineer	Frontend	Admin	Igen	HUF 1.8M	<button>EDIT</button> <button>DELETE</button>
<input type="checkbox"/> 6438ff150ccd672a2ce6f77b	Senior Frontend Engineer	Frontend	Admin	Igen	HUF 1.8M	<button>EDIT</button> <button>DELETE</button>
<input type="checkbox"/> 6438fe950ccd672a2ce6f700	Medior Java Backend Developer	Backend	Admin	Igen	HUF 1.51M	<button>EDIT</button> <button>DELETE</button>
<input type="checkbox"/> 6438fe3f0ccd672a2ce6f6e3	Lead Solution Architect	Architecture	Admin	Igen	HUF 1.8M	<button>EDIT</button> <button>DELETE</button>
<input type="checkbox"/> 6438fcd4ed15253180ba5570	Data Analyst	Data	Admin	Igen	HUF 1.5M	<button>EDIT</button> <button>DELETE</button>

3.7. ábra. Admin Dashboard - Álláshirdetések

Az állásokra is vonatkoznak a fentebb említettek annyi eltéréssel, hogy a táblázatban az álláshirdetés azonosítója, címe, kategóriája, a felhasználó aki létrehozta a hirdetést, aktív-e a hirdetés és a fizetés összege jelenik meg.

4. fejezet

Adatbázis

Az adatbázis elkészítéshez MongoDB-t a validációhoz pedig Mongoose-t használtam, ugyanis korábbi projektekből már volt szerencsém ezekkel a technológiákkal tapasztalatot szerezni. A Mongoose[12] egy egyszerű, séma alapú megoldást kínál az alkalmazások modellezésére. Tartalmazza a beépített típus cast-olást, érvényesítést, *query* építést, üzleti logikai *hook*-okat és egyebeket.

4.1. Adatbázis táblák ismertetése

A MongoDB az adatrekordokat dokumentumokként tárolja, amelyeket gyűjteményekben gyűjtenek össze. Egy adatbázis egy vagy több dokumentumgyűjteményt tárol. A gyűjtemények hasonlóak a relációs adatbázisok tábláihoz. A táblák mindegyikére alkalmazva van a *timestamps* mező, mellyel láthatóvá válik számunkra, hogy egy adott rekordot mikor hoztak létre, illetve módosítottak utoljára.

4.1.1. Users

- **__id**: ObjectId, felhasználók azonosítására szolgál
- **role**: Number, a jogosultsági szintet reprezentálja
 - alapértelmezett értéke 0 (átlagos felhasználó)
- **firstName**: String, a felhasználó keresztnéve
 - kötelező megadni, maximum 32 karakter
- **lastName**: String, a felhasználó vezetéknéve
 - kötelező megadni, maximum 32 karakter
- **email**: String, validált e-mail cím

- kötelező megadni, egyedinek kell lennie, e-mail cím formátumnak megfelel
- **password**: String, hashelt jelszó
 - kötelező megadni, legalább 6 karakter legyen

4.1.2. Jobs

- **__id**: ObjectId, állások azonosítására szolgál
- **title**: String, az álláshirdetés címe
 - kötelező megadni, maximum 64 karakter
- **description**: String, az állás leírása
 - kötelező megadni
- **salary**: String, a fizetés összege
 - kötelező megadni
- **location**: String, a munkavégzés helyszíne
 - kötelező megadni
- **available**: Boolean, az állás elérhetőségét jelzi
 - alapértelmezett érték: true
- **jobType**: ObjectId, a kategóriát azonosítja
 - kötelező, *jobType*-ra mutat
- **user**: ObjectId, a hirdetést létrehozó felhasználót azonosítja
 - kötelező, *User*-re mutat

4.1.3. Jobtypes

- **__id**: ObjectId, állás kategóriák azonosítására szolgál
- **jobTypeName**: String, állás kategória neve
 - kötelező megadni, maximum 64 karakter
- **user**: ObjectId, a kategóriát létrehozó felhasználót azonosítja
 - kötelező, *User*-re mutat

5. fejezet

Frontend

5.1. Komponensek

A komponensek független és újrafelhasználható kód részletek. Ugyanazt a célt szolgálják, mint a JavaScript függvények, de elszigetelten működnek, és HTML-t adnak vissza. A komponensek kimenetükben más komponensekre is hivatkozhatnak. Ez lehetővé teszi, hogy ugyanazt a komponens-absztrakciót használjuk bármilyen részlet szinthez.

5.1. kód. AdminRoute komponens

```
1 import React from 'react'
2 import { useSelector } from 'react-redux';
3 import { Navigate } from 'react-router-dom';
4
5 const AdminRoute = ({ children }) => {
6
7     const { userInfo } = useSelector((state) => state.
        ↪ signIn);
8     return userInfo && userInfo.role === 1 ? children
        ↪ : <Navigate to="/" />;
9 }
10
11 export default AdminRoute
```

5.2. Redux

A Redux egy nyílt forráskódú JavaScript-könyvtár az alkalmazások állapotának kezelésére és központosítására. Leggyakrabban olyan könyvtárakkal használják, mint a React vagy az Angular a felhasználói felületek felépítésére.[13]

5.2.1. Constant

A Reduxban számos műveletet és reduktort határozunk meg, miközben egy alkalmazást készítünk és kezeljük annak állapotát. A konstansok egy módot adnak a műveletek és a reduktorok típusának meghatározására egy fájlban vagy egy helyen. A konstansok figyelembevételének több oka is van:

- A műveletek típusa és a reduktorok két különböző fájlban használatosak. A konstansok segítenek importálni és egyetlen oldalról használni őket.
- A kód olvashatósága növekszik, mivel az összes konstans egy fájlban van felsorolva.
- Segít csökkenteni a kisebb gépelési hibákat írás közben.

5.2.2. Reducer

A reduktorok tiszta függvények a Reduxban. A reduktorok az egyetlen mód az állapotok megváltoztatására a Reduxban. Ez az egyetlen környezet, ahol a logikát és számításainkat megírhatjuk. A reduktor funkció elfogadja az alkalmazás és a művelet előző állapotát, kiszámítja a következő állapotot, és visszaadja az új objektumot.

A dolgozatban a reduktorokon belül egy switch case szerkezettel vizsgálom meg a paraméterben átadott action objektumot, majd az ennek megfelelő visszatérési értékkel tér vissza a szerkezet.

Mivel az alkalmazás egyre összetettebbé vált, a reduktorokat célszerű volt különálló függvényekre felosztani, amelyek mindegyike az állapot egy független részét kezeli. Ezt *combineReducers* függvény segítségével valósítottam meg, ami egy helper függvény, mely egy olyan objektumot, amelynek értékei különböző redukáló függvények, egyetlen redukáló függvénnyé alakít át, amelyet átadhatunk a *createStore*-nak.

A kapott reduktor minden leszármazott reduktort meghív, és az eredményeket egyetlen állapotobjektumba gyűjti. A *combineReducers()* névterek által generált állapot az egyes reduktorok állapotait a kulcsaik alatt adja meg, ahogyan átadja a *combineReducers()*-nek.

5.2.3. Action

Az *action* egy egyszerű JavaScript objektum, amely egy *type* mezővel rendelkezik. A *type* mezőnek egy olyan String-nek kell lennie, amely leíró nevet ad ennek a műveletnek. A String első része a funkció vagy kategória, amihez a művelet tartozik, a második rész pedig a konkrét dolog ami történt.

Egy *action* objektumnak lehetnek további mezői a történetekről szóló kiegészítő információkkal. Konvenció alapján ezeket az információkat a *payload* nevű mezőbe helyezzük.

Ezek alapján például az állás hirdetések betöltése a főoldalon is ilyen módon történik. A *jobLoadAction* paraméterben megkapja a lapszámozásnak megfelelő oldalszámot, az esetleges keresésben használt kulcsszót, a kategóriát amelybe az állás tartozik, valamint a hirdetéshez tartozó helyszínt. Ezután a függvény aszinkron módon folytatódik. Elsősorban elküldésre kerül a *JOB_LOAD_REQUEST*, mely az összes álláshirdetés megjelenítési kérelmét foglalja magában, majd egy try-catch szerkezet try részében az *await* kulcsszót követően egy *axios* GET request segítségével megkapjuk az URL-t. Ezt követően a *dispatch* metódust használva továbbítunk két dolgot, az elsőt, hogy sikeres volt az állások betöltése (*JOB_LOAD_SUCCESS*), valamint a másodikat a *payload* mezőben, ami a kérésre vonatkozó adatokat tartalmazza. Amennyiben sikertelen a betöltés, úgy a *type* mezőbe *JOB_LOAD_FAIL* kerül, valamint a *payload* egy hibaüzenettel tér vissza.

5.2. kód. Hirdetések betöltése

```
1 export const jobLoadAction = (pageNumber, keyword = ''  
  ↪ , cat = '', location = '') => async (dispatch)  
  ↪ => {  
2   dispatch({ type: JOB_LOAD_REQUEST });  
3   try {  
4     const { data } = await axios.get(`/api/jobs/  
      ↪ show/?pageNumber=${pageNumber}&keyword=${  
      ↪ {keyword}&cat=${cat}&location=${location  
      ↪ }`)  
5     dispatch({  
6       type: JOB_LOAD_SUCCESS,  
7       payload: data  
8     });  
9   } catch (error) {  
10    dispatch({  
11      type: JOB_LOAD_FAIL,  
12      payload: error.response.data.error  
13    });  
14  }  
15 }
```

6. fejezet

Backend

6.1. Jogosultsági rendszer

A jogosultsági rendszer két részre osztható, az adminisztrátorokra és az átlagos felhasználókra. Az átlagos felhasználóknak jogkörébe tartozik az állások megtekintése, szűrése és keresése, valamint a jelentkezés egyes hirdetésekre. Megtekinthetik a User Dashboard-ot ahol láthatják, a felhasználói fiók adatait, mint például a regisztráció idejét és a létrehozott hirdetések számát, továbbá megnézheti az előzményeit, amiben azok az álláshirdetések találhatók, melyre a felhasználó korábban jelentkezett.

Az adminisztrátorok az Admin Dashboard-on keresztül lekérhetik az adatbázisból az összes felhasználót egy táblázatba, melyre különböző szűrőket alkalmazhatnak, felhasználói adatokat tudnak módosítani és törölni, új felhasználót tudnak létrehozni. Új állás kategóriákat hozhatnak létre, amiken belül új álláshirdetéseket tudnak létrehozni, melyekhez szintén rendelkezésre áll egy a felhasználó listához hasonló táblázat, ugyanazokkal a funkciókkal.

6.2. Hitelesítés

A dolgozatban az adatok hitelesítése JSON Web Token[14] segítségével történik. A JSON Web Token (JWT) egy nyílt szabvány (RFC 7519) az információk biztonságos továbbítására a felek között JSON-objektumként. Kompakt, olvasható és digitálisan aláírt privát kulcs vagy nyilvános kulcspár segítségével az Identity Provider (IdP) által. Így a token sértetlenségét és hitelességét más érintett felek is ellenőrizhetik. A JWT használatának célja nem az adatok elrejtése, hanem az adatok hitelességének biztosítása. A JSON Web Token aláírt és kódolt, nem titkosított.

A JWT egy token alapú állapot nélküli hitelesítési mechanizmus. Mivel ez egy kliensoldali állapot nélküli munkamenet, a szervernek nem kell teljes mértékben egy adattárra (adatbázisra) támaszkodnia a munkamenet információinak mentéséhez.

6.3. Kontrollerek

A *controller* egy JavaScript objektum, amely attribútumokat/tulajdonságokat és függvényeket tartalmaz. Minden vezérlő elfogad egy *scope* paramétert, ami arra az alkalmazásra/modulra vonatkozik, amelyet a vezérlőnek kezelnie kell.

A kontrollereket említve kerülnek a képbe a middleware függvények. A middleware-függvények olyan függvények, amelyek hozzáféréssel rendelkeznek a *request* objektumhoz (req), a *response* objektumhoz (res) és az alkalmazás *request-response* ciklusának következő függvényéhez. A következő függvény az *Express router* egyik funkciója, amely meghívásakor végrehajtja az aktuális middleware-t követő middleware-t.

Ilyen módon kerül például létrehozásra egy új állás kategória. A függvény legelején az *exports* kulcsszóval exportáljuk aszinkron függvényünket *createJobType* néven, hogy azt egy másik állományon belül felhasználhassuk az átláthatóbb kód érdekében. A függvény törzsén belül egy try-catch szerkezettel létrehozzuk az állás kategóriát ami a *request* törzsén belül egy *jobTypeName* és egy felhasználói azonosítóból áll, annak érdekében, hogy nyomon tudjuk követni melyik felhasználó hozta létre az adott kategóriát. Sikeres létrehozás esetén egy 201-es státusz kódot és a létrehozott kategóriát kapjuk eredményül, ellenkező esetben pedig a művelet jellegéhez megfelelő hibaüzenetet kapjuk.

6.1. kód. Kategória létrehozása middleware függvénnyel

```
1 exports.createJobType = async(req, res, next) => {
2   try {
3     const jobT = await JobType.create({
4       jobTypeName: req.body.jobTypeName,
5       user: req.user.id
6     });
7     res.status(201).json({
8       success: true,
9       jobT
10    })
11  } catch (error) {
12    next(error);
13  }
14 }
```

A middleware függvények a következő feladatokat hajthatják végre:

- bármilyen kód végrehajtása
- *request* és *response* objektumok módosítása
- *request-response* ciklus lezárása
- a következő middleware stack meghívása

Ha az aktuális middleware függvény nem zárja le a *request-response* ciklust, meg kell hívnia a *next()* függvényt, hogy átadja a vezérlést a következő middleware függvénynek. Ellenkező esetben a kérés függőben marad.

Ezekben a controller-ekben implementáljuk a CRUD műveleteket és azoknak variációit (egy felhasználó lekérése id alapján, minden felhasználó lekérdezése, stb...). Update kérés során figyelembe kell vennünk, hogy a kérés akkor járjon adatmódosítással, ha a beszúrni kívánt adat eddig nem volt jelen.

6.4. Modellek

A modellek olyan adatstruktúrák, amelyeket adataink alakjának meghatározására használunk. A modell a felhasználói felület központi eleme. Ez az egyetlen hely, ahol szűrhetjük és kezelhetjük az alkalmazáson áthaladó összes adatot.

6.2. kód. User séma a user modellből

```
1  const userSchema = new mongoose.Schema({
2
3    firstName: {
4      type: String,
5      trim: true,
6      required: [true, 'Keresztnév megadása kötelező'],
7      maxlength: 32,
8    },
9    lastName: {
10     type: String,
11     trim: true,
12     required: [true, 'Vezetéknév megadása kötelező'],
13     maxlength: 32,
14   },
15   email: {
16     type: String,
17     trim: true,
18     required: [true, 'E-mail cím megadása kötelező'],
19     unique: true,
20     match: [
21       /^([a-zA-Z0-9_+&'-]+)@([a-zA-Z0-9_+&'-]+)\.([a-zA-Z0-9_+&'-]+)$/i,
22       'Kérem érvényes e-mail címet adjon meg'
23     ]
24   },
25   password: {
```

```

26     type: String,
27     trim: true,
28     required: [true, 'Jelszó megadása kötelező'],
29     minlength: [6, 'A jelszó legalább 6
        ↪ karakterből kell álljon'],
30   },
31
32   jobsHistory: [jobsHistorySchema],
33
34   role: {
35     type: Number,
36     default: 0
37   }
38
39 }, { timestamps: true })

```

A User Modellen belül kezeljük a felhasználóhoz tartozó összes adatot, mint vezetéknév, keresztnév, e-mail cím, jelszó, korábbi állások, jogosultság és egy timestamp is tartozik minden felhasználóhoz, ami megadja a létrejöttének idejét, valamint azt az időpontot amikor frissítették a felhasználót. Az adatain kívül ebben a modellben található még a jelszó titkosítását végző függvény, ami az adatbázisba való mentés előtt hash-eli a felhasználó jelszavát. A feltöltött álláshirdetések és kategóriák is ilyen struktúrájú modelleket használnak.

6.5. Router

A React Router egy JavaScript-keretrendszer, amely lehetővé teszi, hogy kezeljük a kliens- és szerveroldali útválasztást a React alkalmazásokban. Lehetővé teszi egyoldalas web- vagy mobilalkalmazások létrehozását, amelyek lehetővé teszik az oldal frissítése nélküli navigálást. Azt is lehetővé teszi számunkra, hogy a megfelelő alkalmazásnézet megőrzése mellett használjuk a böngészési előzmények funkcióit.

Egyszerűvé teszi az alkalmazás URL-címének és állapotának kezelését, segítségével megadhatjuk az összes lehetséges URL-mintát az alkalmazásban, és azt, hogy melyik UI-összetevőt kell megjeleníteni. Ez az útválasztó csökkenti az alkalmazásnak az állapot fenntartásához szükséges kód mennyiségét, és elérhetőbbé teszi az új funkciók hozzáadását.

Egyes route-okhoz tartoznak `IsAuthenticated` és `isAdmin` feltételek, melyek arra szolgálnak, hogy bizonyos oldalakat csak a bejelentkezett felhasználók, illetve csak az adminisztrátorok érjének el, továbbá az is látható, hogy a route-ok végén a controller-ben szereplő függvények kerülnek meghívásra.

6.3. kód. User Route

```
1 // /api/allusers
2 router.get('/allusers', isAuthenticated, isAdmin,
    ↪ allUsers);
3 // /api/user/id
4 router.get('/user/:id', isAuthenticated, singleUser);
5 // /api/user/edit/id
6 router.put('/user/edit/:id', isAuthenticated, editUser
    ↪ );
7 // /api/admin/user/delete/id
8 router.delete('/admin/user/delete/:id',
    ↪ isAuthenticated, isAdmin, deleteUser);
9 // /api/user/jobhistory
10 router.post('/user/jobhistory', isAuthenticated,
    ↪ createUserJobsHistory);
```


7. fejezet

Tesztelés

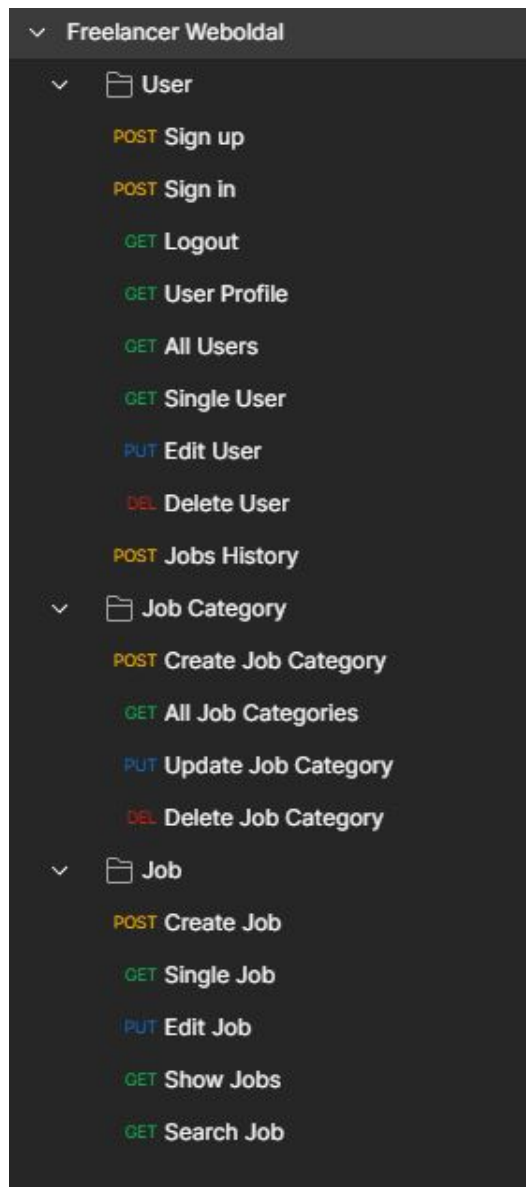
7.1. Postman

A Postman[15] egy API-platform API-k létrehozására és használatára. A Postman leegyszerűsíti az API életciklusának minden lépését, és egyszerűsíti az együttműködést, így jobb API-kat hozhatunk létre gyorsabban. Kéréseket küldhetünk az általunk használt API-khoz való csatlakozáshoz.

Ezen kérések segítségével lekérdezhetünk, hozzáadhatunk, törölhetünk és frissíthetünk adatokat. A kérésekkel küldhetünk paramétereket, felhatalmazással kapcsolatos adatokat és bármilyen törzsadatot amire szükségünk lehet a későbbiek során.

A szoftver segítségével teszteltem az API működését GET, POST, PUT és DELETE requestek futtatásával, a megfelelő adatok és környezet biztosításával.

Ezek a kérések három kategóriára bonthatók, az első a Felhasználó kategória, amibe a felhasználóval kapcsolatos kérések tartoznak. Ilyen kérések a regisztráció, bejelentkezés, kijelentkezés, felhasználói profil megtekintése, egy felhasználó lekérése, összes felhasználó lekérése, felhasználó adatainak módosítása, felhasználó törlése, valamint a felhasználó összekötése olyan állásokkal amire korábban jelentkezett. A második részben az állások kategóriájára vonatkozó request-ek vannak, ezek az egyszerű CRUD (Create, Read, Update, Delete) műveletek. A harmadik kategória magukra az állásokra vonatkozik, itt is megtalálhatók a CRUD műveletek, továbbá egy vagy több állás hirdetés megjelenítése, illetve keresés az állások között bizonyos kulcsszavakkal.



7.1. ábra. Postman API tesztelés

7.2. Manuális tesztelés

A dolgozat tesztelését manuálisan is elvégeztem végfelhasználóként és adminisztrátorként is. A tesztelés eredményeiről készítettem egy táblázatot, melyben látható, hogy milyen tevékenységet teszteltem, mi volt az elvárt eredmény, az aktuális eredmény és a böngésző amiben végeztem a tesztet. Az átlag felhasználó és az adminisztrátor teszt eredményeit külön táblázatban helyeztem el az átláthatóság kedvéért.[16]

7.1. táblázat. Általános felhasználó tesztelés

Tesztelt funkció	Elvárt eredmény	Aktuális eredmény	Teszt környezet
Álláshirdetések listázása a főoldalon	Adatbázisban szereplő összes hirdetés megjelenik	Az adatbázisban szereplő összes hirdetés megjelent	Windows 10 Home, Google Chrome 112.0
Keresés frontend állásokra	Csak olyan munkák jelennek meg, melyben a „frontend” kulcsszó szerepel	Csak frontend állások jelennek meg	Windows 10 Home, Google Chrome 112.0
Állások szűrése „Architecture” kategóriára	Csak „Architecture” kategóriába tartozó állások jelennek meg	Az „Architecture” kategória hirdetései jelennek meg	Windows 10 Home, Google Chrome 112.0
Állások szűrése helyszíntre	Csak a kiválasztott helyszíntre jelennek meg hirdetések	A kiválasztott helyszíntre jelennek meg hirdetések	Windows 10 Home, Google Chrome 112.0
Bejelentkezés	Sikeres bejelentkezés, a felhasználó a User Dashboardra kerül átirányításra	Sikeres bejelentkezés után, a felhasználó a User Dashboardra került	Windows 10 Home, Google Chrome 112.0
Regisztráció	Sikeres regisztráció	A regisztráció sikerült, a fiók létrejött	Windows 10 Home, Google Chrome 112.0
Kijelentkezés	Kijelentkezés után a felhasználó a főoldalra kerül	Sikeres kijelentkezés, átirányítás a főoldalra	Windows 10 Home, Google Chrome 112.0
Állás előzmények megtekintése	A felhasználóhoz tartozó állás előzmények kilistázódnak	A felhasználó állás előzményei sikeresen kilistázódtak	Windows 10 Home, Google Chrome 112.0
Admin Dashboard hozzáférés	Nem megfelelő jogosultság, a felhasználó nem fér hozzá az oldalhoz	Jogosultság hiányában a felhasználó a főoldalra kerül	Windows 10 Home, Google Chrome 112.0

7.2. táblázat. Adminisztrátor tesztelés

Tesztelt funkció	Elvárt eredmény	Aktuális eredmény	Teszt környezet
Állások kilistázása az admin felületen	Az álláshirdetések és a kapcsolatos adatok megjelennek táblázatban	Az álláshirdetések és a hozzá tartozó adatok megjelentek	Windows 10 Home, Google Chrome 112.0
Új álláshirdetés létrehozása	Új hirdetés jön létre a megadott adatokkal	A hirdetés létrejött a helyes adatokkal	Windows 10 Home, Google Chrome 112.0
Hirdetés módosítása	Az adott hirdetés megfelelői részei módosulnak	A megfelelő adatok módosultak	Windows 10 Home, Google Chrome 112.0
Álláshirdetés törlése a rendszerből	A kijelölt hirdetés törlődik	A hirdetés törlésre került	Windows 10 Home, Google Chrome 112.0
Felhasználók kilistázása az admin felületen	A felhasználók és adataik megjelennek táblázatban	A felhasználók adataikkal megjelentek a táblázatban	Windows 10 Home, Firefox 110.0.1 64 bit
Új felhasználó létrehozása	Új felhasználó jön létre a megadott adatokkal	A felhasználó létrejött a megadott adatokkal	Windows 10 Home, Google Chrome 112.0
Felhasználó módosítása	A megfelelő adatok frissülnek	A felhasználó adatai frissültek	Windows 10 Home, Google Chrome 112.0
Felhasználó törlése	Az adott felhasználó törlődik az adatbázisból	A felhasználó törlésre került	Windows 10 Home, Firefox 110.0.1 64 bit
Új állás kategória létrehozása	Új kategória jön létre	A kategória létrejött	Windows 10 Home, Firefox 110.0.1 64 bit
Kategória módosítása	A kategória a megfelelő adatokkal módosul	A kategória rendben frissült	Windows 10 Home, Firefox 110.0.1 64 bit
Állás kategória törlése	A kategória törlésre kerül a rendszerből	A kijelölt kategória törölve lett	Windows 10 Home, Firefox 110.0.1 64 bit

8. fejezet

Továbbfejlesztési lehetőségek

8.1. Szűrés

A meghirdetett állásokra alkalmazható szűrők listájának kibővítése illetve kombinálása. Több tényező bevezetése, mint például: technológiák, fizetés intervallum, szerződés típusa, ajánlott illetve elvárt tapasztalat, munkavégzés nyelve, különböző juttatások.

8.2. Üzenet küldés

Egy egyszerű üzenetküldő rendszer bevezetése mely lehetőséget biztosít a munkavállalók és a munkát kínáló közötti kapcsolatfelvételre, egyeztetésre, valamint az adminisztrátorokkal való könnyű és közvetlen kommunikációra.

8.3. Hirdetések mentése

Lehetőséget biztosítani a felhasználó számára, hogy a neki tetsző hirdetéseket egy külön listához adva el tudja azokat tárolni, így később könnyen visszakereshető lesz az ajánlat. Az álláshirdetés létrehozója felé pedig valamilyen módon jelezni, hogy hányan mentették le a bejegyzését, így számon tudja tartani az érdeklődést a bejegyzése iránt.

8.4. Felhasználói adatok

Felhasználói adatok kibővítése a munkavállalás szempontjából fontos adatokkal. Ilyen adatok lehetnek például a korábban végzett munkák, egyes területeken és technológiákkal szerzett tapasztalat és eltöltött idő, a munkavállaló által beszélt nyelvek, tartózkodási helye, utazási lehetőségei, önéletrajza és referenciái.

8.5. Jogosultsági kör kibővítése

Az átlagos felhasználó és az adminisztrátor közé egy moderátori jogkör bevezetése, ami a weboldalon megjelenített tartalmakat ellenőrzi, illetve moderálja. Jogkörébe tartozna a hirdetések módosítása és törlése, valamint egyfajta kapcsolattartóként is működhetne átlag felhasználó és adminisztrátor között.

8.6. Kiemelt és népszerű hirdetések

Lehetőséget biztosítani egyes felhasználóknak valamilyen módon egy kiemelt hirdetést létrehozni az oldalon, mely a listán felül egy jól látható pozícióban helyezkedik el, illetve a sok felhasználó által megtekintett hirdetéseket külön kiemelni népszerű álláshirdetesként.

8.7. Legfrissebb álláshirdetések

Egy külön szekcióban kiemelni a mai napon feladott álláshirdetéseket, így a felhasználóknak lehetőségük lenne rögtön aktuális hirdetéseket látni, az esetleg több napja vagy hete létrehozott bejegyzések helyett.

Összegzés

Az alkalmazás elkészítése során használt technológiáknak hála a weboldal konzisztensen működik, rugalmas és további felületek bevezetése sem okoz problémát a rendszerbe, így új igények felmerülése esetén nem szükséges egy új rendszer fejlesztésébe belefogni, hiszen könnyűszerrel bővíthető, továbbfejleszthető. Az elkészült produktum egy nyílt rendszer, így amennyiben szükséges, úgy a közösség rendelkezésére áll továbbfejleszthetőség szempontjából.

A webalkalmazásnak a célja, hogy az online munkavállalás és hirdetés sokkal elérhetőbb és könnyebb legyen mindenki számára, ezért is fordítottam nagyobb hangsúlyt a felhasználói felület megtervezésére, hogy az a lehető legkényelmesebb használatot biztosítsa és letisztult legyen. A munkavállalóknak könnyebb dolga legyen és a meghirdetett állások közül könnyűszerrel ki tudják válogatni azokat, melyek nekik szimpatikusak és legtöbb igényüknek megfelelnek.

A weboldal jelenlegi állapotában egy egyszerűbb álláshirdető portál funkcióját képes ellátni, azonban a továbbfejlesztési lehetőségekben felderített hiányosságok implementálása a dolgozatban, valamint a jelenlegi működési logika refaktorálása és a felhasználói felület továbbfejlesztése egy pozitív lépés lenne a dolgozat állapotát tekintve. Így az alkalmazás fejlesztése az egyetemi tanulmányaim befejezése után sem áll meg.

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani egyetemi oktatóimnak az évek során nyújtott segítségükért, minden átadott tudásért és érdeklődési köröm kibővítéséért, valamint konzulens tanáromnak, Dr. Tajti Tibornak, aki a konzultációk során biztosított tanácsaival és szakértelmével hatalmas segítséget nyújtott a szakdolgozatom elkészítéséhez.

További köszönetet szeretnék mondani a tanulmányaim alatt megismert hallgatótársaimnak és barátaimnak, akiknek minden alkalommal megmutatkozott a segítőkészsége és összetartása.

Szintén köszönöm minden közeli barátomnak és ismerősömnek, az évek során nyújtott folyamatos támogatásukért és segítségükért.

Végül de nem utolsó sorban, szeretnék köszönetet mondani családom minden tagjának, akik a tanulmányaim alatt végig mellettem álltak, segítettek és támogattak.

Irodalomjegyzék

- [1] MEAN STACK: [https://en.wikipedia.org/wiki/MEAN_\(solution_stack\)](https://en.wikipedia.org/wiki/MEAN_(solution_stack))
- [2] REACT.JS FORRÁS: <https://react.dev>
- [3] EXPRESS.JS FORRÁS: <https://expressjs.com>
- [4] NODE.JS FORRÁS: <https://nodejs.org>
- [5] MONGODB: <https://www.mongodb.com>
- [6] ERIC BUSH: *Node.js, MongoDB, React, React Native Full-Stack Fundamentals and Beyond*, 2018.
- [7] MATERIAL UI FORRÁS: <https://mui.com>
- [8] AXIOS FORRÁS: <https://axios-http.com>
- [9] REACT-REDUX FORRÁS: <https://react-redux.js.org>
- [10] ADHAM DANNAWAY: *Practical UI*, 2022.
- [11] MATERIAL UI X DATAGRID: <https://mui.com/x/react-data-grid>
- [12] MONGOOSE: <https://mongoosejs.com/docs>
- [13] DANIEL BUGL: *Learning Redux*, 2017.
- [14] JSON WEB TOKEN FORRÁS: <https://jwt.io>
- [15] POSTMAN: <https://learning.postman.com/docs/introduction/overview/>
- [16] MANUAL TESTING: https://en.wikipedia.org/wiki/Manual_testing

NYILATKOZAT

Alulírott Kugovszki Bálint, büntetőjogi felelősségem tudatában kijelentem, hogy az általam benyújtott, Freelancer Weboldal című szakdolgozat (diplomamunka) önálló szellemi termékem. Amennyiben mások munkáját felhasználtam, azokra megfelelően hivatkozom, beleértve a nyomtatott és az internetes forrásokat is.

Tudomásul veszem, hogy a szakdolgozat elektronikus példánya a védelem után az Eszterházy Károly Katolikus Egyetem könyvtárába kerül elhelyezésre, ahol a könyvtár olvasói hozzájuthatnak.

Kelt: Gyöngyös, 2023 év 04 hó 04 nap.

Kugovszki Bálint

aláírás