**Pannonhalmi Főapátság Szegedi SOB Technikuma**

****

**Dolgozói Beléptető Rendszer**

Szakképzés neve és kódja: Szoftverfejlesztő és -tesztelő (506131203)

Témavezető: Rédai Dávid  
Készítők: Adamovits Otília Alexandra: Frontend Backend, Molnár Szabolcs Szilveszter: Backend, Maui App

Szeged, 2025.

Tartalomjegyzék

[1. Bevezetés 5](#_iorcl6vmp6y5)

[2.1 Fejlesztőkörnyezet ismertetése és a választás indoklása 6](#_trqj1nl4qjq8)

[2.2 Adatszerkezet bemutatása 8](#_ykc21o0ilt)

[2.3 Egyedi és érdekesebb algoritmusok bemutatása 11](#_vjcx7utkf97d)

[2.3.1 Be- és kiléptetési folyamat 11](#_69ghdqb6zbx)

[2.3.2 Autentikáció JWT tokennel 12](#_60iw411272xv)

[2.3.3 Swagger + Authorization fejlesztés 12](#_mljzt42rr7ut)

[2.4 Tesztelés leírása 14](#_ftt4jpv4szw4)

[2.5 Tesztdokumentáció 14](#_cky38k995vuo)

[Hardver: 15](#_lc6qexhxqu4e)

[Operációs rendszerek: 15](#_ykoop1ynaphe)

[Böngészők: 16](#_uew0mbi269ih)

[Felbontások: 16](#_ys14j4cgsrbt)

[Hibakezelési és szélsőséges esetek tesztelése 17](#_5js6825w9h2w)

[Felhasználónév-jelszó páros a teszteléshez: 18](#_i34rg2ao1mk)

[Megjegyzések a működésről 19](#_ffe6jtuofeuk)

[Fejlesztési lehetőségek 20](#_m6y3sn31hxno)

[4. Felhasználói dokumentáció 21](#_4bbfyfpqanit)

[4.1 A program céljának és lényegesebb funkcióinak összefoglalása 21](#_2v056f41guyk)

[4.2 Szükséges hardvereszközök és szoftverek felsorolása 21](#_uxrzy5tidi30)

[Hardverigény: 21](#_7ws8mvxc3fvf)

[Szoftverigény: 22](#_s5m05r47q372)

[4.3 Telepítés és indítás lépéseinek ismertetése 22](#_new5n4tkcwfq)

[Backend telepítése: 22](#_60x3gzxwl3pr)

[Frontend telepítése: 23](#_fc1laoq47uce)

[Felhasználói oldal megnyitása: 23](#_71pu1s9thdis)

[Swagger elérés: 23](#_5kthnbxdwf3p)

[4.4 Telepítés és indítás lépéseinek ismertetése MAUI APP 23](#_fgacthh84svg)

[4.5 a A program részletes bemutatása 24](#_ydqycear51q5)

[Dolgozói felület: 24](#_7n1v60w8m1ke)

[Admin felület: 24](#_kasfs97gz37y)

[Bejelentkezés 25](#_k5fsnyomj9ea)

[Dolgozói nézet funkciói 25](#_hpwlscb8foj8)

[Admin nézet funkciói 25](#_8hocjh1rkr0y)

[Kijelentkezés 25](#_7w2n752847jw)

[4.6 Helytelen használatból adódó hibajelzések magyarázata 26](#_c2dfh2sbxgcm)

[Erősségek (Strengths) 28](#_8lnqqmt5a7cy)

[Gyengeségek (Weaknesses) 28](#_vf3pmb2uqazj)

[Lehetőségek (Opportunities) 29](#_d2j2z2yvj656)

[Veszélyek (Threats) 29](#_mgjfcxbn46ez)

[5. Összefoglalás, köszönetnyilvánítás 29](#_5ct31o9hry2c)

### **1. Bevezetés**

A dolgozói beléptetőrendszer létrehozásának célja egy olyan digitális megoldás megvalósítása volt, amely hatékonyan kezeli és nyomon követi a munkavállalók munkaidejét, belépési és kilépési adatait. A rendszer fő feladata a dolgozók munkaóráinak pontos nyilvántartása, valamint a ledolgozott idő napi és havi szintű összesítése. A megvalósított alkalmazás egy ASP.NET Core alapú Web API backendből áll, amelyhez egy SvelteKit frontend csatlakozik, és biztosítja a felhasználók számára az egyszerű és átlátható kezelőfelületet.

A témaválasztás indoklása több tényezőre vezethető vissza. Egyrészt, mivel a modern vállalkozások számára kiemelt jelentőséggel bír a munkaidő-nyilvántartás hatékony kezelése, ezért fontosnak tartottuk egy olyan rendszer kidolgozását, amely automatizálja és digitalizálja ezt a folyamatot. Másrészt, közös szakmai fejlődésünk szempontjából is releváns volt a projekt, hiszen a backend- és frontend-fejlesztés, valamint az adatbázis-kezelés területén szerettünk volna mélyebb ismereteket szerezni és gyakorlati tapasztalatokat gyűjteni.

A rendszer fejlesztése során célkitűzésünk az volt, hogy egy biztonságos és megbízható alkalmazást hozzunk létre, amely támogatja a beléptetési és kijelentkezési folyamatokat, valamint lehetőséget biztosít az adminisztrátorok számára a dolgozók adatainak kezelésére. A projekt során külön figyelmet fordítottunk a jogosultságkezelésre és a felhasználói hitelesítés megvalósítására is, amelyhez JWT (JSON Web Token) alapú autentikációs rendszert alkalmaztunk.

A projektet gyakorlati hasznosulás szempontjából is relevánsnak tartjuk, hiszen egy ilyen rendszer alkalmazható lehet akár kis- és középvállalkozások, akár nagyobb szervezetek esetében is, ahol fontos a munkaidő pontos rögzítése és a munkavállalók munkaidejének átlátható nyilvántartása. Ezen túlmenően a projekt során megszerzett tudás és tapasztalatok hasznos alapot biztosítanak számunkra a jövőbeli szoftverfejlesztési feladatok során.   
A MAUI alkalmazás lehetővé teszi, hogy a dolgozók mobileszközről kezeljék a munkanapjaikat – belépjenek, elindítsák és befejezzék a munkavégzést, valamint megtekinthessék az adott napi és havi ledolgozott munkaidejüket.  
A választott projekt jól illeszkedik a szoftverfejlesztői szakma gyakorlati követelményeihez, mivel több, valós környezetben is alkalmazható technológiát és architektúrát ötvöz: REST API-kat, adatbázis-kezelést (MYSQL Server), platformfüggetlen mobilalkalmazás-fejlesztést (MAUI), valamint frontend- és backendkommunikációt.

**2. Fejlesztői dokumentáció**

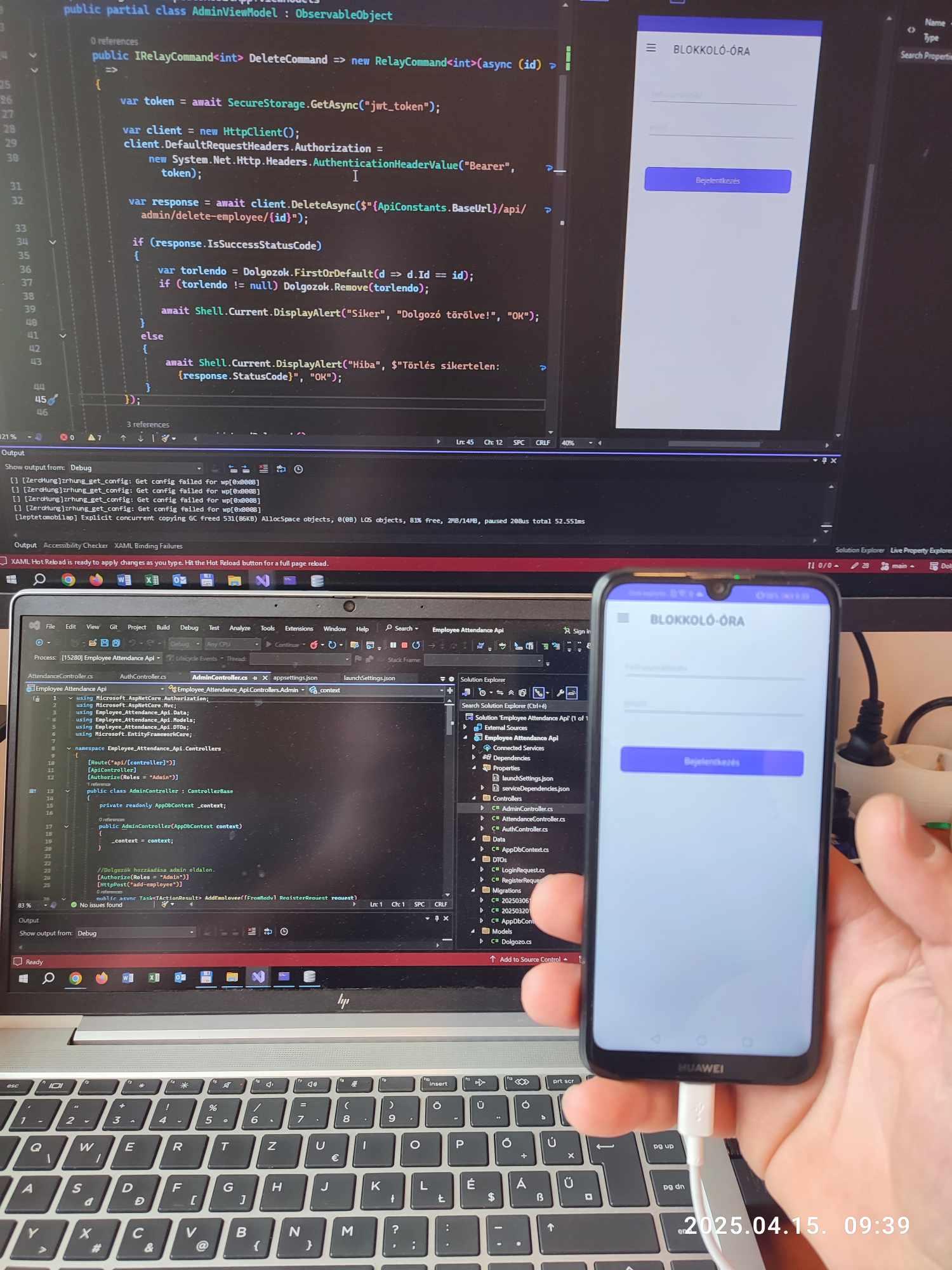
### **2.1 Fejlesztőkörnyezet ismertetése és a választás indoklása**

A dolgozói beléptetőrendszer fejlesztését egy **ASP.NET Core Web API 8.0** alapú backend és egy **SvelteKit** frontend környezetben valósítottuk meg. A backendhez **C#** nyelvet használtunk, míg a frontend fejlesztés **JavaScript** alapokon történt.

A választásunkat az indokolta, hogy az ASP.NET Core stabil, ipari szintű keretrendszer, amely kiválóan támogatja a REST API-k gyors és biztonságos fejlesztését, beépített autentikációs és jogosultságkezelési lehetőségekkel. A **SvelteKit** pedig modern, gyors és egyszerűen testreszabható frontend keretrendszer, amely jól integrálható REST API-khoz.

A fejlesztés során az alábbi környezetet és eszközöket használtuk:

* **Visual Studio 2022** (ASP.NET Core API fejlesztés)
* **VS Code** (SvelteKit frontend fejlesztés)
* **Postman** (API tesztelés)
* **MYSQLite** adatbázis (lokális, könnyen kezelhető fejlesztési környezethez)
* **MyMYSQL**  adatbázis (felhő alapú, komplex fejlesztési környezethez)
* **Swagger** (API dokumentáció és tesztelés)
* **Git** verziókezeléshez (GitHub tároló)
* Tesztelés mobilkészüléken: Android 11/12-es rendszerű telefon

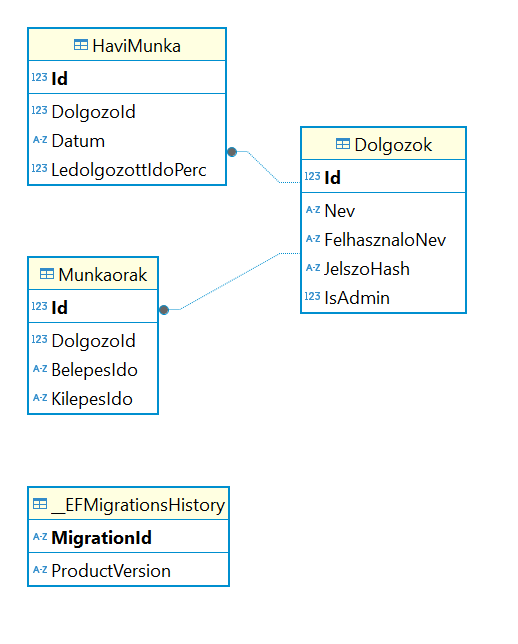
A .NET MAUI választása lehetővé teszi a multiplatformos mobilalkalmazás-fejlesztést (Android, iOS), a Visual Studio pedig beépített támogatást kínál ehhez. A MYSQL Server könnyen integrálható az ASP.NET API-val, és jól támogatja a relációs adatmodellezést.  


A végleges alkalmazás üzemeltetését **Aiven** felhőalapú szolgáltatásban terveztük meg, amely biztosítja a skálázhatóságot és a megbízhatóságot.

### **2.2 Adatszerkezet bemutatása**

Az alkalmazás három fő táblát használ az MYSQL adatbázisban, amelyek az **Entity Framework Core** segítségével kerültek implementálásra:

* **Dolgozo**
  + Id: int (Elsődleges kulcs)
  + Nev: string
  + FelhasznaloNev: string
  + JelszoHash: string (BCrypt-tel hashelt jelszó)
  + IsAdmin: bool (adminisztrátori jogosultság megjelölése)
* **Munkaora**
  + Id: int (Elsődleges kulcs)
  + DolgozoId: int (idegen kulcs a Dolgozo táblára)
  + BelepesIdo: DateTime (belépés ideje)
  + KilepesIdo: DateTime? (kilépés ideje, null lehet, ha nincs még kilépés)
* **HaviMunka**
  + Id: int (Elsődleges kulcs)
  + DolgozoId: int (idegen kulcs a Dolgozo táblára)
  + Datum: DateTime (az adott nap)
  + LedolgozottIdoPerc: int (az adott napi ledolgozott idő percekben)



Az alkalmazásban DTO-k (Data Transfer Object) is alkalmazásra kerültek a bejelentkezési és regisztrációs folyamatokhoz (pl. LoginRequest, RegisterRequest).

**Főbb változók a MAUI ViewModelben**:

* IsAdmin (bool) – jogosultság alapján állítja be a felületet
* IsWorking (bool) – állapotváltozó munkavégzéshez
* StartWorkCommand, EndWorkCommand – MVVM parancsok a munkavégzéshez
* DailyHours, MonthlyStats – UI megjelenítéshez használt ObservableProperty-k

### **2.3 Egyedi és érdekesebb algoritmusok bemutatása**

#### **2.3.1 Be- és kiléptetési folyamat**

Az **AttendanceController** tartalmazza a dolgozók beléptetési és kiléptetési műveleteit. A belépéskor egy új Munkaora rekord keletkezik a belépés időpontjával, míg kilépéskor a rendszer az adott nap aktív műszakját (ahol KilepesIdo == null) frissíti a kilépés időpontjával.

MásolásSzerkesztés

var munkaora = await \_context.Munkaorak.FirstOrDefaultAsync(m => m.DolgozoId == dolgozoId && m.KilepesIdo == null);

munkaora.KilepesIdo = DateTime.Now;

await \_context.SaveChangesAsync();

#### **2.3.2 Autentikáció JWT tokennel**

Az **AuthController** felel a bejelentkezési logikáért és a token generálásáért. A dolgozók hitelesítéséhez BCrypt jelszó-hash validációt használok, majd sikeres hitelesítés esetén JWT token kerül generálásra, amely a felhasználó nevét és szerepkörét (pl. Admin) tartalmazza.

MásolásSzerkesztés

var tokenDescriptor = new SecurityTokenDescriptor

{

Subject = new ClaimsIdentity(claims),

Expires = DateTime.UtcNow.AddHours(1),

SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)

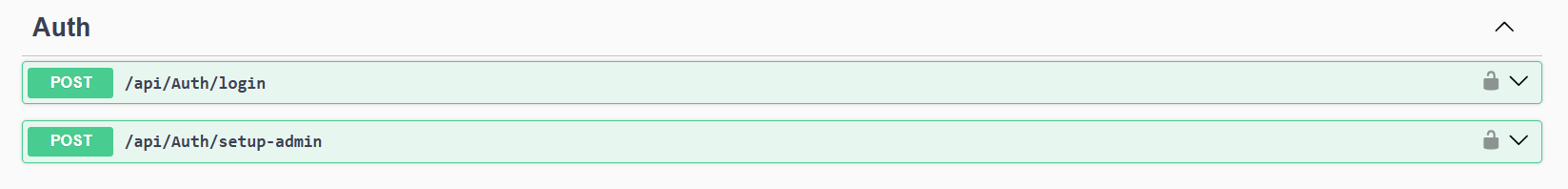
};

A tokenben szereplő szerepkörök (Admin vagy User) az API végpontokhoz való hozzáférést szabályozzák.

#### **2.3.3 Swagger + Authorization fejlesztés**

A Swagger UI-t kiegészítettük egy Bearer tokenes hitelesítési lehetőséggel, amely lehetővé teszi, hogy a dokumentált API végpontokat már közvetlenül az interfészen belül is tesztelhessük, token megadásával.

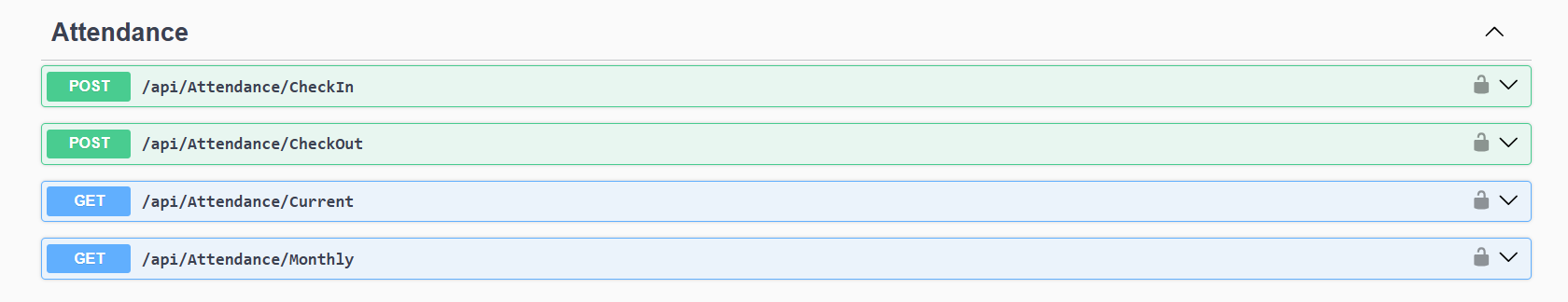
*Authentication:*

**

*POST /api/auth/login - User login*

*POST /api/auth/register - User registration*

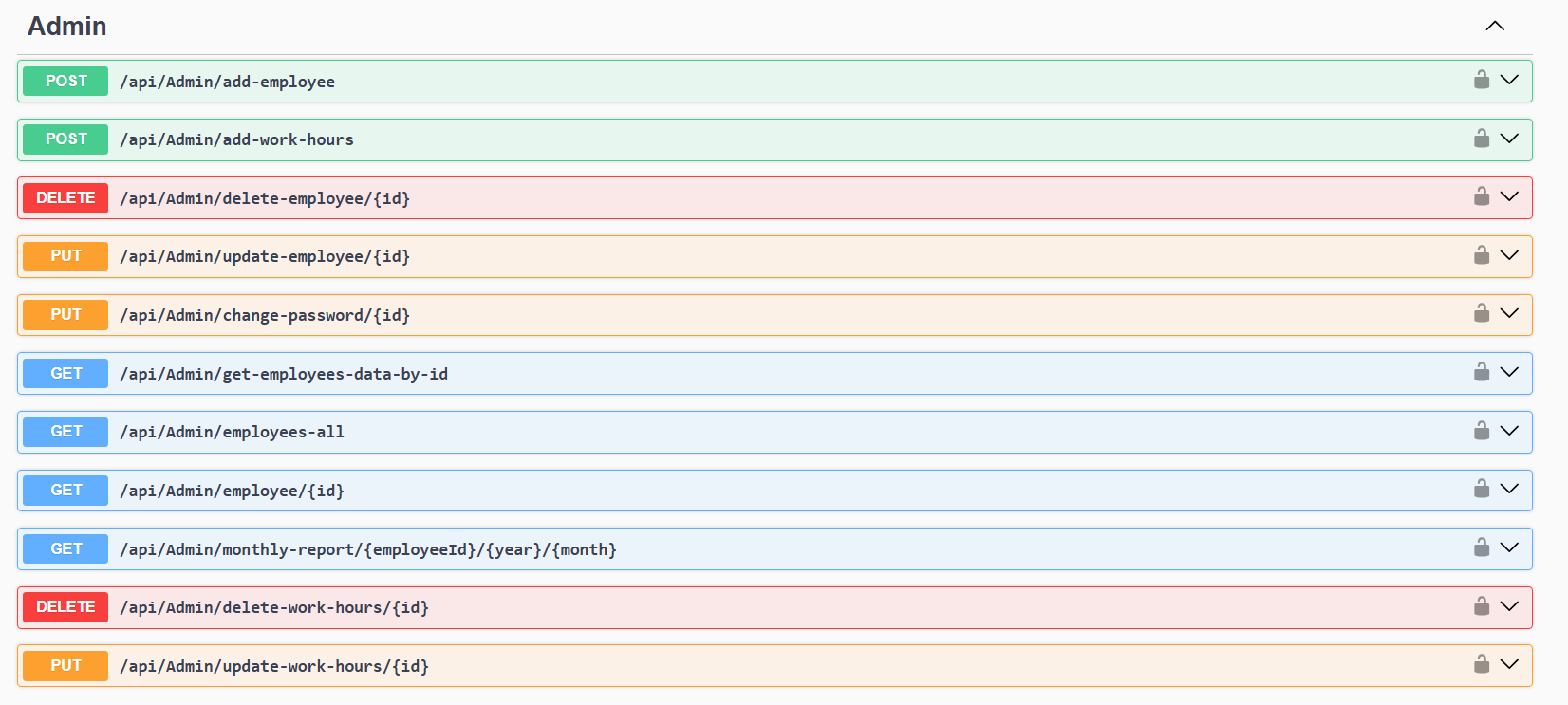
*Attendance:*

**

*POST /api/attendance/checkin - Check in*

*POST /api/attendance/checkout - Check out*

*GET /api/attendance/current - Get current session*

*Admin:  
*

*GET /api/admin/employees - List all employees*

*POST /api/admin/add-employee - Add new employee*

*PUT /api/admin/update-employee - Update employee details*

*GET /api/Admin*

*POST /api/Admin*

*PUT /api/Admin/{id}*

*DELETE /api/Admin/{id}*

### **2.4 Tesztelés leírása**

A tesztelést főként **Postman** segítségével végeztem, ahol az alábbi fő funkciókat ellenőriztem:

* Sikeres és sikertelen bejelentkezés admin és felhasználói jogosultságokkal.
* Dolgozó regisztrációs folyamat.
* Belépés és kilépés rögzítése (beleértve az edge-case-eket, pl. kilépés nélküli napok).
* Jogosultság alapú végpont hozzáférés (Admin-only funkciók tesztelése).

Emellett manuális teszteléssel a SvelteKit frontend oldalán is validáltam az API-val való kommunikációt, különös tekintettel a CORS szabályozás és token-alapú hitelesítés helyes működésére.

### **2.5 Tesztdokumentáció**

* **Bejelentkezés**:  
  + Input: helyes felhasználónév és jelszó → Elvárt: token visszaadása.
  + Input: hibás jelszó → Elvárt: 401 Unauthorized.
* **Belépés rögzítése**:  
  + Input: létező dolgozó ID → Elvárt: „Bejelentkezés rögzítve!”
  + Input: nem létező dolgozó ID → Elvárt: „Nincs aktív műszak!” hibakezelés.
* **Kilépés rögzítése**:  
  + Input: aktív műszakkal rendelkező dolgozó → Elvárt: kilépési idő rögzítése.
  + Input: aktív műszak nélküli dolgozó → Elvárt: hibaüzenet.
* **Admin jogosultság ellenőrzése**:  
  + Admin felhasználó → hozzáférés Admin-only végpontokhoz.
  + Normál felhasználó → 403 Forbidden válasz.

**Tesztkörnyezetek:**

Az alkalmazást különböző hardver- és szoftverkörnyezetekben teszteltem annak érdekében, hogy megbizonyosodjak a stabil működéséről több platformon:

#### **Hardver:**

* Asztali számítógép (Intel Core i5-12400, 16 GB RAM, SSD)
* Laptop (AMD Ryzen 5 5500U, 8 GB RAM)
* Android okostelefon (Samsung Galaxy A52, 6 GB RAM)

#### **Operációs rendszerek:**

* Windows 11 Pro (22H2)
* Ubuntu 22.04 LTS
* Android 13 (Chrome böngészőn keresztül)

#### **Böngészők:**

* Google Chrome (v123)
* Mozilla Firefox (v124)
* Microsoft Edge (v122)
* Android Chrome (mobil teszteléshez)

#### **Felbontások:**

* FullHD (1920x1080)
* HD Ready (1366x768)
* Mobil (1080x2400)

**Megjegyzés**: A rendszer reszponzív módon viselkedik, a SvelteKit frontend a különböző képernyőméretekhez alkalmazkodik, mobil eszközön is megfelelően jelenik meg.

### **Hibakezelési és szélsőséges esetek tesztelése**

Az alábbi esetekben vizsgáltam a program helyes működését:

|  |  |  |  |
| --- | --- | --- | --- |
| **Teszt eset** | **Bevitt adat** | **Elvárt működés** | **Teszt eredménye** |
| Hibás bejelentkezés | Hibás jelszó | 401 Unauthorized válasz üzenet: „Érvénytelen felhasználónév vagy jelszó!” | Sikeres |
| Ismeretlen felhasználónév | Nem létező felhasználónév | 401 Unauthorized válasz | Sikeres |
| Üres bejelentkezési adatok | üres mezők | 400 Bad Request vagy 401 Unauthorized válasz | Sikeres |
| Kilépés aktív műszak nélkül | nincs aktív „Munkaora” rekord | 404 Not Found válasz: „Nincs aktív műszak!” | Sikeres |
| Belépés nem létező dolgozó ID-val | ID nem létezik | 200 OK válasz, de az adatbázisban nincs változás | Sikeres |
| Üres tokennel történő API hívás | Token nélküli fetch | 401 Unauthorized válasz | Sikeres |

### **Felhasználónév-jelszó páros a teszteléshez:**

Teszteléshez létrehozott adminisztrátor:

* Felhasználónév: **admin2**
* Jelszó: **1234**

Teszteléshez létrehozott dolgozó:

* Felhasználónév: **feknor**
* Jelszó: **1234**

A teszt felhasználó regisztrálása során a jelszó automatikusan BCrypt-tel került titkosításra, így valós környezetben is biztonságosan tárolódik.

### **Megjegyzések a működésről**

* A rendszer minden tesztelt környezetben megbízhatóan működött.
* Mobil böngészőkön a belépési/kilépési funkciók gyorsan reagálnak, a MAUI APP frontend pedig jól alkalmazkodik az érintőképernyőhöz.
* A REST API végpontok helyes hibakódokat és érthető válaszokat adnak, amelyek megkönnyítik a hibaelhárítást a felhasználók és fejlesztők számára is.
* Külön figyelmet fordítottunk a JWT alapú hitelesítésre, így jogosultság nélkül nem lehet hozzáférni védett végpontokhoz.

### 

### **Fejlesztési lehetőségek**

A jelenlegi rendszer stabil alapot biztosít, azonban a jövőben további funkciókkal szeretnénk bővíteni:

1. **Dolgozói profiloldal**: ahol a dolgozók megtekinthetik az összes eddigi munkaórájukat és havi összesítéseiket.
2. **Admin felület fejlesztése**: teljes CRUD-műveletek dolgozók, munkaórák és havi ledolgozott idők kezelésére (SvelteKit frontend bővítése).
3. **Automatizált havi riportok**: PDF export funkció, amely havi jelentéseket generál a dolgozók ledolgozott munkaidejéről.
4. **E-mail értesítések**: belépés és kilépés után automatikus e-mail küldése a dolgozónak és az adminnak.
5. **Többnyelvűség támogatása**: angol-magyar felület közötti váltás a frontend oldalon.
6. **Adatbázis bővítése**: PostgreMYSQL vagy MYSQL Server migráció a nagyvállalati környezethez való igazítás érdekében.

## 

## **4. Felhasználói dokumentáció**

### **4.1 A program céljának és lényegesebb funkcióinak összefoglalása**

Ez a szoftver egy **webalapú dolgozói beléptető és munkaidő-nyilvántartó rendszer**. Célja, hogy digitálisan nyilvántartsa a vállalkozás munkavállalóinak munkaidejét: mikor léptek be és ki a munkahelyről, illetve mennyi időt töltöttek el munkával egy adott napon vagy hónapban.

A rendszer **két fő szereplőt** különböztet meg:

* **Dolgozó**: beléphet, kiléphet és megtekintheti havi ledolgozott óráit.
* **Adminisztrátor**: hozzáfér az admin felülethez, ahol dolgozókat vehet fel, törölhet, és a fejlesztések után láthatja az összes dolgozó munkanaplóját.

A program a böngészőben használható, nem szükséges telepíteni semmilyen további alkalmazást a számítógépre.

### **4.2 Szükséges hardvereszközök és szoftverek felsorolása**

#### **Hardverigény:**

* Asztali PC vagy laptop
* Minimum: 2 GB RAM, ajánlott: 4 GB RAM vagy több
* Stabil internetkapcsolat
* Alternatíva: modern okostelefon vagy tablet (Android 9 vagy újabb)

#### **Szoftverigény:**

* **Böngésző**: Google Chrome (ajánlott), Mozilla Firefox, Microsoft Edge, Safari
* **Backend**: .NET 8 SDK + ASP.NET Core Web API futtatása (csak szerveroldalon szükséges)
* **Frontend**: SvelteKit alapú webalkalmazás, amely a böngészőben fut
* **Adatbázis**: MYSQL (fejlesztéshez), opcionálisan MYSQL Server üzemi környezetben
* **Swagger UI**: API teszteléshez, fejlesztői vagy admin felhasználók számára

### **4.3 Telepítés és indítás lépéseinek ismertetése**

#### **Backend telepítése:**

1. Töltse le vagy klónozza a projektet GitHub-ról.
2. Nyissa meg a projektet Visual Studio 2022-ben.
3. Ellenőrizze az appsettings.json fájlt, hogy az adatbázis elérési út helyes legyen (pl. MYSQLite helyi fájl vagy MYSQL Server).
4. Nyomjon F5-öt vagy indítsa el dotnet run paranccsal.
5. A rendszer alapértelmezés szerint a http://localhost:7032 címen fog futni.

#### **Frontend telepítése:**

1. Navigáljunk a frontend mappába parancssorban vagy terminálban.
2. Futtassuk: npm install
3. Indítsuk el a fejlesztői szervert: npm run dev
4. Az alkalmazás a http://localhost:3000 címen lesz elérhető böngészőben.

#### **Felhasználói oldal megnyitása:**

* A dolgozó vagy adminisztrátor egyszerűen böngészőből megnyitja a frontend URL-t: http://localhost:3000

#### **Swagger elérés:**

* A backend API a http://localhost:7032/swagger címen dokumentálva van, ahol tesztelhetők a végpontok.

### **4.4 Telepítés és indítás lépéseinek ismertetése MAUI APP**

1. Töltsük le az APK fájlt a fejlesztőtől (pl. e-mail, fájlmegosztó link).
2. A telefon beállításaiban engedélyezzük az **ismeretlen forrásból származó alkalmazások** telepítését.
3. Keressük meg a letöltött fájlt, és kattintsunk rá a telepítéshez.
4. A telepítés után megjelenik az alkalmazás ikonja a telefon főmenüjében.
5. Kattintsunk az ikonra az alkalmazás elindításához.
6. Jelenleg teszt környezetben működik az app verzió tehát közös Wifi hálózaton kell legyen a backend és a tesztelésre használt mobil készülék, ez azonban (ha ez a kívánság) előnyt is jelenthet mivel például csak a munkahelyen tudnak a dolgozók becsekkolni.

### **4.5 a A program részletes bemutatása**

#### **Dolgozói felület:**

* **Bejelentkezés**: felhasználónév és jelszó megadása után a rendszer bejelentkezteti a dolgozót.
* **Munka indítása**: "Belépés" gomb megnyomásával rögzíti a rendszer a belépési időt.
* **Munka befejezése**: "Kilépés" gombbal rögzíti a kilépési időt.
* **Havi munkaóra**: egy táblázatban a dolgozó látja az adott hónap napi bontású munkaidejét (órában és percben).

#### **Admin felület:**

* **Dolgozók kezelése**: dolgozók hozzáadása, törlése, adatainak módosítása.
* **Munkaidő riportok**: az admin láthatja a dolgozók teljes munkaidejét havi bontásban. (jövőbeni fejlesztés lesz)

**4.5 b A program részletes bemutatása MAUI APP**

#### **Bejelentkezés**

Az indító képernyőn be kell írni a felhasználónevet és jelszót. Amennyiben a bejelentkezés sikeres:

* Admin felhasználók az **Admin oldalra** kerülnek.
* Dolgozók a **Dashboard oldalra** jutnak.

#### **Dolgozói nézet funkciói**

* **Munka indítása**: az aktuális időpont mentésre kerül.
* **Munka befejezése**: kiszámítja és eltárolja az aznapi munkaidőt.
* **Napi számláló**: megmutatja, mennyi időt dolgoztunk aznap.
* **Havi ledolgozott napok**: listázza az adott hónap minden napját a ledolgozott idővel.

#### **Admin nézet funkciói**

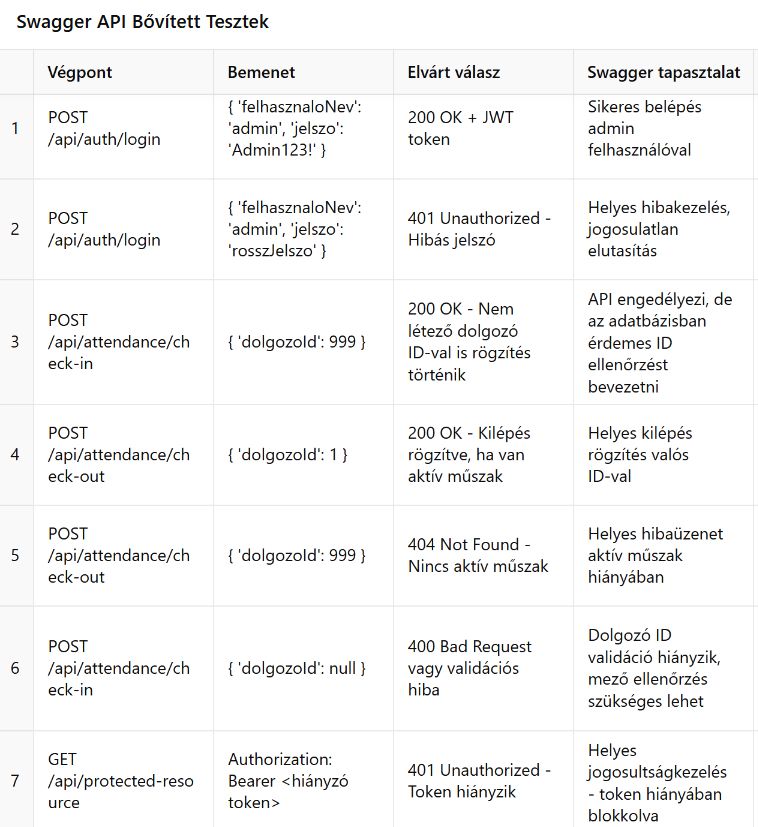
* Dolgozók kezelése (felhasználók listázása)
* Munkaidők megtekintése (statisztikák, összesített adatok)  
  (jövőbeni fejlesztés)

#### **Kijelentkezés**

A hamburger menüben található „Kijelentkezés” gombbal visszatérhetünk a bejelentkező képernyőre.

### **4.6 Helytelen használatból adódó hibajelzések magyarázata**

* **Hibás bejelentkezés**: ha a megadott felhasználónév vagy jelszó helytelen, a rendszer a következőt írja ki:  
   „Érvénytelen felhasználónév vagy jelszó!”
* **Belépés nélküli kilépés**: ha a dolgozó belépés nélkül próbál kilépni, a rendszer válasza:  
   „Nincs aktív műszak!”
* **Token hiánya**: ha valaki nem jelentkezett be, és úgy szeretne a szerverhez hozzáférni, a rendszer üzenete:  
   401 Unauthorized
* **Hibás adatbevitel**:  
  + Üres bejelentkezési mezők esetén a mezők piros kerettel jelennek meg, és figyelmeztető szöveg látható:  
     „A mező kitöltése kötelező!”
  + Hibás dolgozó ID megadása backend hívásnál: nincs blokkolva, de a rendszer később nem találja az ID-hoz tartozó rekordot.



Hibás jelszóval történő bejelentkezés kezelése

Nem létező dolgozó ID-val való belépés/kilépés tesztelése

Null vagy hiányzó input mezők kezelése

Token nélküli végpont elérés tiltása

**SWOT-analízis a dolgozói beléptető rendszerhez**

#### **Erősségek (Strengths)**

* Biztonságos bejelentkezés JWT token alapú hitelesítéssel és bcrypt jelszóhasheléssel.
* Külön admin és dolgozó szerepkörök kezelése.
* Automatikus munkaóra nyilvántartás (belépés és kilépés rögzítése).
* API alapú backend, amely könnyen integrálható a Svelte frontenddel.
* Swagger integráció, amely megkönnyíti az API tesztelését és dokumentálását.
* Platformfüggetlen futtathatóság (ASP.NET Core backend, MYSQLite vagy MYSQL Server adatbázis).

#### **Gyengeségek (Weaknesses)**

* Jelenleg csak alap szintű riportok és statisztikák (pl. nincs grafikus megjelenítés vagy részletes kimutatások a dolgozók számára).
* A munkaórák kezelése még nem tartalmaz túlóra vagy szabadságkezelési funkciókat.
* A rendszer skálázhatósága korlátozott lehet nagyobb céges környezetekben (pl. több száz dolgozó esetén).
* A frontend oldalon még szükség van egy teljes értékű admin felület és dolgozói dashboard kialakítására.

#### **Lehetőségek (Opportunities)**

* Integráció RFID/NFC olvasókkal a gyorsabb beléptetés érdekében. Ez a projekt nemcsak egy vizsgafeladat, hanem egy olyan mintaalkalmazás, amely könnyen adaptálható más, hasonló célú rendszerekhez is, és amelyet tovább lehet fejleszteni – például RFID vagy NFC-alapú beléptetés integrálásával, vagy felhőalapú adattárolással.
* Bővíthetőség további modulokkal, pl. szabadságkezelés, túlóra-jóváhagyás, havi/jelenléti kimutatások.
* Felhős hoszting (pl. Aiven) és CI/CD pipeline bevezetése a hatékonyabb üzemeltetés érdekében.

#### **Veszélyek (Threats)**

* Biztonsági kockázatok: amennyiben nem megfelelően frissíted a függőségeket és a rendszer sérülékenységeit (pl. JWT token kezelése).
* Adatvédelmi kihívások (GDPR megfelelés), különösen ha személyes adatokat kezelsz (pl. név, munkaidő).
* Nagyvállalati környezetben versenyhelyzet más, kész, kereskedelmi rendszerekkel szemben (pl. SAP vagy munkaidő-nyilvántartó szoftverek).
* Technikai elavulás, ha a rendszer hosszabb távon nem kap folyamatos karbantartást vagy fejlesztést.

## **5. Összefoglalás, köszönetnyilvánítás**

A dolgozói beléptető- és munkaidő-nyilvántartó rendszer fejlesztése során sikerült elérnünk azokat a szakmai célokat, amelyeket a projekt elején megfogalmaztunk. A rendszer teljes körűen lefedi az alapvető munkaidő-kezelési folyamatokat: biztosítja a dolgozók számára a napi belépés és kilépés regisztrálását, a havi ledolgozott idő nyomon követését, és lehetőséget ad az adminisztrátoroknak a dolgozók adatainak kezelésére is. Fontos célunk volt, hogy a rendszer egy könnyen használható, webes felületen keresztül legyen elérhető, ami mind az adminisztrátorok, mind a dolgozók számára egyszerű és átlátható élményt nyújt.

A projekt során rengeteg hasznos tapasztalattal gazdagodtunk. Kiemelnénk például az ASP.NET Core Web API használatának mélyebb megismerését, különös tekintettel a JWT token-alapú autentikációra és a REST API biztonsági megoldásaira. Szintén sokat tanultunk a SvelteKit frontend keretrendszer alkalmazásából, amely segített abban, hogy modern és gyors webes felületet tudjunk kialakítani.

A legnagyobb kihívást talán az jelentette, hogy az API és a frontend oldal közötti kommunikációt biztonságosan és hatékonyan alakítsuk ki, figyelve a CORS problémákra, a tokenek kezelésére és az adminisztrátori jogosultságok megfelelő elkülönítésére.

A projekt utóéletét illetően a célunk az, hogy a rendszert tovább bővítsük új funkciókkal. Szeretnénk fejleszteni például a dolgozói profilfelületet, bevezetni a PDF formátumú havi riportok exportálását, és integrálni egy e-mail értesítési rendszert is.

Végül szeretnénk megköszönni mindazoknak a támogatását, akik közvetve vagy közvetlenül hozzájárultak a projekt sikeres megvalósításához. Külön köszönettel tartozunk oktatóinknak a szakmai iránymutatásért, valamint családjainknak és barátainknak a türelemért és támogatásért, amelyet a fejlesztési időszak során nyújtottak számunkra. Az ő segítségük nélkül nem tudtuk volna ennyi energiát és figyelmet szentelni a projekt tökéletesítésének.

**6. Irodalomjegyzék**

Microsoft. (n.d.). *ASP.NET Core*. Retrieved from<https://learn.microsoft.com/en-us/aspnet/core/>

Microsoft. (n.d.). *Entity Framework Core*. Retrieved from<https://learn.microsoft.com/en-us/ef/core/>

Microsoft. (n.d.). *ASP.NET Core Web API with Swagger*. Retrieved from<https://learn.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-swashbuckle>

BCrypt.Net. (n.d.). *BCrypt.NET-Next - Simple .NET password hashing*. GitHub. Retrieved from<https://github.com/BcryptNet/bcrypt.net>

MYSQLite Consortium. (n.d.). *MYSQLite*. Retrieved from [https://www.MYSQLite.org/](https://www.sqlite.org/)

JWT.io. (n.d.). *JSON Web Tokens - Introduction*. Retrieved from https://jwt.io/introduction

Svelte. (n.d.). *Svelte – Cybernetically enhanced web apps*. Retrieved from<https://svelte.dev/>

OpenAPI Initiative. (n.d.). *Swagger: OpenAPI Specification*. Retrieved from https://swagger.io/specification/