**Baranya Vármegyei Szakképzési Centrum Simonyi Károly Technikum és Szakképző Iskola**

**Vizsgaremek**

**Készítették: Szabó Gergely**

**Bogdán Balázs**

**Pécs**

**2025**

**Baranya Vármegyei Szakképzési Centrum Simonyi Károly Technikum és Szakképző Iskola**

*Szakma megnevezése: Szoftverfejlesztő és –tesztelő*

*A szakma azonosító száma: 5 0613 12 03*

**Vizsgaremek**

GB Medical – Időpontfoglaló

Készítették: Szabó Gergely

Bogdán Balázs

**Pécs**

**2025**

# Nyilatkozat

Alulírott büntetőjogi felelősségünk tudatában nyilatkozunk és aláírásunkkal igazoljuk, hogy a benyújtott vizsgaremekünk saját, önálló munkánk. Az abban hivatkozott nyomtatott és elektronikus szakirodalom felhasználása a szerzői jogok szabályainak megfelelően készült. Tudomásul vesszük, hogy vizsgaremek esetén plágiumnak számít:

* Szószerinti, vagy attól kismértékben eltérő idézet közlése idézőjel és hivatkozás megjelölése nélkül.
* Tartalmi idézet hivatkozás megjelölése nélkül.
* Más publikált gondolatainak (cikk, dolgozat) sajátomként való feltüntetése.

Alulírott **Szabó Gergely, Bogdán Balázs** kijelentjük, hogy a plágium fogalmát megismertük, és tudomásul vesszük, hogy plágium esetén vizsgaremekünk visszautasításra kerül.

|  |  |
| --- | --- |
| ..................................  Szabó Gergely | ..................................  Bogdán Balázs |

Pécs, 2025.04.06

# Tartalomjegyzék

[Nyilatkozat 2](#_Toc195264282)

[Tartalomjegyzék 3](#_Toc195264283)

[I. Bevezető, a feladat rövid ismertetése 5](#_Toc195264284)

[1. Munkamegosztás 6](#_Toc195264285)

[II. A felhasználói dokumentáció 7](#_Toc195264286)

[1. A program általános specifikációja 7](#_Toc195264287)

[2. Rendszerkövetelmények 8](#_Toc195264288)

[3. A programok telepítése 10](#_Toc195264289)

[3.1. MAMP telepítése és konfigurálása: 10](#_Toc195264290)

[3.2. JDK telepítése a NetBeans előtt: 11](#_Toc195264291)

[3.3. NetBeans letöltése és konfigurálása: 12](#_Toc195264292)

[3.4. WildFly szerver konfigurálása 12](#_Toc195264293)

[3.5. Frontend (Angular) telepítése 15](#_Toc195264294)

[4. A program használatának részletes leírása 17](#_Toc195264295)

[4.1 Bevezetés 17](#_Toc195264296)

[4.2 Regisztráció és emailes megerősítés 17](#_Toc195264297)

[4.3 Bejelentkezés 18](#_Toc195264298)

[4.4 Szolgáltatás vagy orvos választása foglalás előtt 19](#_Toc195264299)

[4.5 Időpontfoglalás elérhető időpontokra 19](#_Toc195264300)

[4.6 Lefoglalt időpont megjelenítése és lemondása 21](#_Toc195264301)

[4.7 Orvosok értékelése és visszajelzések megtekintése 22](#_Toc195264302)

[4.8 Orvoskeresés szűrőmező segítségével 23](#_Toc195264303)

[4.9 Szolgáltatás adatainak megtekintése legördülő menüvel 24](#_Toc195264304)

[4.10 Jelszó helyreállítás 25](#_Toc195264305)

[4.11 Oldal tetejére visszaugró gomb 25](#_Toc195264306)

[4.12 Admin oldal és felhasználó törlése 26](#_Toc195264307)

[III. A fejlesztői dokumentáció 27](#_Toc195264308)

[1. Témaválasztás indoklása 27](#_Toc195264309)

[2. Az alkalmazott fejlesztői eszközök 27](#_Toc195264310)

[3. Tervezési módszer 28](#_Toc195264311)

[3.1 Alkalmazás tervezése 29](#_Toc195264312)

[3.2 Tervezési módszertan ismertetése 29](#_Toc195264313)

[3.3 Vizuális modellezés 30](#_Toc195264314)

[3.4 Objektum-orientált programozás (OOP) megközelítése az MVC struktúrában 30](#_Toc195264315)

[4. Adatmodell leírása 31](#_Toc195264316)

[4.1 Áttekintés 31](#_Toc195264317)

[4.2 Adatbázis táblák és kapcsolatok 32](#_Toc195264318)

[4.3 Kiegészítő táblák (fejlesztés során létrehozott) 35](#_Toc195264319)

[4.4 Normalizálás folyamata 36](#_Toc195264320)

[4.5 OOP jellegű megvalósítás: 37](#_Toc195264321)

[4.6 Biztonsági kérdések, titkosítás és GDPR 38](#_Toc195264322)

[5. Részletes feladatspecifikáció, algoritmusok 38](#_Toc195264323)

[6. Tesztelési dokumentáció 44](#_Toc195264324)

[7. Továbbfejlesztési lehetőségek 44](#_Toc195264325)

[7.1. Nem megvalósított, de tervezett funkciók 45](#_Toc195264326)

[7.2. Jövőbeli fejlesztési lehetőségek 45](#_Toc195264327)

[IV. Összegzés 47](#_Toc195264328)

[Irodalomjegyzék, forrásmegjelölés 48](#_Toc195264329)

# I. Bevezető, a feladat rövid ismertetése

A GB Medical - időpontfoglaló projekt célja egy modern, felhasználóbarát alkalmazás kifejlesztése, amely lehetővé teszi a betegek számára, hogy online időpontot foglaljanak az orvosi rendelőkben, értékeléseket is leadhassanak az orvosokra, valamint emailben megerősítést kapjanak a foglalásról.

A projektet több okból is választottuk:

* **Valós problémára adott megoldás:** Az egészségügyi szolgáltatásokban az időpontfoglalás, az orvosok értékelése és a foglalásról szóló emailes visszaigazolás gyakran okoz nehézségeket, ezért egy jól megtervezett online rendszer jelentős segítséget nyújthat mind az orvosi praxisok, mind a páciensek számára.
* **Technológiai kihívások:** A rendszer fejlesztése során számos modern technológiát és fejlesztési módszert alkalmaztunk, például RESTful API-k, reszponzív webes felületek, adatbázis-kezelő rendszerek, email értesítési megoldások és verziókövetés (GitHub) integrációját, amelyek révén alaposabban megismerkedtünk a skálázható, biztonságos és felhasználóbarát megoldások kialakításának kihívásaival.
* **Csapatmunka és projektszervezés:** A projekt megvalósításához egy kétfős csapatot alkottunk, ahol a tagok egymást kiegészítve, külön felelősségi köröket vállaltak. A fejlesztési folyamat során olyan eszközöket használtunk, mint a GitHub a verziókövetéshez, Jira a projektmenedzsmenthez és a munkaidő logoláshoz.

A projekt során szerzett tapasztalataink között szerepel a komplex rendszertervezés, a moduláris kódolás és a tiszta kód elveinek alkalmazása. Ezek az ismeretek nem csak a jelenlegi projekt sikeréhez járultak hozzá, hanem alapot szolgáltatnak a jövőbeli szoftverfejlesztési feladatainkhoz is.

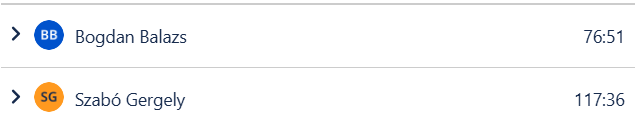
## 1. Munkamegosztás

* Szabó Gergely:
  + **Adatbázisterv:** Adatbázismodell-diagram készítése a dbdiagram.io segítségével
  + **Adatbázis-kezelés:** phpMyAdmin használata
  + **Backend fejlesztés:** NetBeans 19 segítségével Java alapú szerveroldali fejlesztés
  + **Frontend fejlesztés:** Angular alapú kliensoldali fejlesztés
  + **Projektmenedzsment:** Jira eszköz használata, taskok kezelése
  + Dokumentáció elkészítése
* Bogdán Balázs:
  + **Webdesign:** Figma segítségével a vizuális tervezés
  + **Frontend fejlesztés:** Angular alapú kliensoldali fejlesztés

Mindkét tag aktívan részt vett a **GitHub** használatában a kód verziókövetésére és együttműködésre, illetve **Jirában** a munkaidő logolásában.

**GitHub repository:** https://github.com/SzGeri25/vizsgaremek-bogdan-szabo.git

Összességében a GB Medical - időpontfoglaló projekt nemcsak egy technikai kihívást jelentett, hanem lehetőséget biztosított arra is, hogy fejlesszük a csapatmunkát, a projektszervezést és a modern szoftverfejlesztési módszerek alkalmazását.



Jira munkaidők

# II. A felhasználói dokumentáció

## 1. A program általános specifikációja

|  |  |
| --- | --- |
| **Funkció neve** | **Leírás** |
| Szolgáltatás vagy orvos választása foglalás előtt | A felhasználó a foglalás megkezdése előtt választhat szolgáltatás vagy konkrét orvos alapján, hogy a megfelelő időpontok jelenjenek meg a naptárban. Emellett megjelenítheti az összes időpontot egy gomb megnyomásával is. |
| Időpontfoglalás elérhető időpontokra | A naptárban megjelenő szabad időpontokra a felhasználó rákattintva foglalhat. A foglalás kizárólag bejelentkezett felhasználó számára elérhető. A rendszer automatikusan visszaigazoló emailt küld, amely tartalmazza a foglalás részleteit. |
| Lefoglalt időpont megjelenítése és lemondása | Az oldal frissítése után a felhasználó által lefoglalt időpont más színnel jelenik meg. Erre kattintva a felhasználó lemondhatja azt. Lemondásról is emailben érkezik megerősítés. Csak saját foglalás mondható le, kizárólag bejelentkezés után. |
| Orvosok értékelése és visszajelzések megtekintése | Bejelentkezett felhasználók csillagos (numerikus) és opcionálisan szöveges értékeléseket adhatnak az orvosokra. A korábbi értékelések nyilvánosan megtekinthetőek. |
| Orvoskeresés szűrőmező segítségével | A felhasználó keresőmezőben szűrheti az orvosokat név, email cím, telefonszám vagy praxis alapján. |
| Szolgáltatás adatainak megtekintése legördülő menüvel | A felhasználó egy legördülő menü segítségével megtekintheti az elérhető szolgáltatásokat. A kívánt szolgáltatás kiválasztása után megjelennek a hozzá tartozó részletes adatok, például leírás, ár, időtartam és a szolgáltatáshoz tartozó orvos(ok). |
| Regisztráció és emailes megerősítés | Új felhasználók regisztrálhatnak a rendszerbe. A regisztráció után a rendszer emailt küld egy megerősítő linkkel, melyre kattintva aktiválható a fiók. |
| Jelszó helyreállítás | Elfelejtett jelszó esetén a felhasználó emailben kérhet helyreállító linket. A linkre kattintva új jelszót adhat meg egy külön oldalon. |
| Oldal tetejére visszaugró gomb | Az oldal jobb alsó sarkában egy fix pozíciójú gomb jelenik meg, amelyre kattintva a felhasználó automatikusan visszajut az oldal tetejére. |

## 2. Rendszerkövetelmények

**Hardver követelmények:**

**Minimális hardver konfiguráció:**

* Processzor: 2 GHz-es kétszálas (Dual-Core) processzor
* RAM: 4 GB
* Szabad merevlemez terület: 2 GB
* Képernyő felbontás: 1024 x 768
* Grafikus kártya: DirectX kompatibilis videókártya

**Ajánlott hardver konfiguráció:**

* Processzor: 3 GHz-es négymagos (Quad-Core) processzor
* RAM: 8 GB
* Szabad merevlemez terület: 5 GB vagy több
* Képernyő felbontás: 1920 x 1080
* Grafikuskártya: DirectX 11 kompatibilis videókártya

**Szoftver követelmények:**

**Operációs rendszerek:**

* Windows 10 vagy újabb
* macOS 10.12 (Sierra) vagy újabb
* Linux (Ubuntu 20.04 vagy újabb)

**Egyéb szoftver komponensek:**

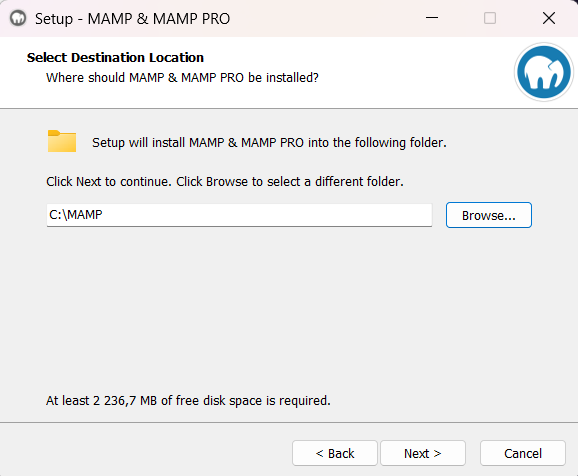
* Adatbázis-kezelés:  
  + phpMyAdmin (a MAMP csomag részeként) az adatbázis kezeléséhez.
* Backend fejlesztés:  
  + **NetBeans 19**: A backend fejlesztése Java alapú, és a NetBeans 19 IDE-t használja a szerveroldali kód megírásához és karbantartásához.
* Frontend fejlesztés:  
  + **Angular**: Az Angular 19 alapú frontend fejlesztéshez szükséges a Node.js és npm telepítése.
  + A frontend fejlesztéshez **Visual Studio Code**-ot használunk, amelyben az Angular keretrendszert könnyedén lehet kezelni.
* Projektmenedzsment:
  + **Jira**: A projektmenedzsmenthez a Jira weboldalát használtuk, amely lehetővé teszi a havi sprintek kezelését, a taskok nyomon követését és az egyes feladatok státuszának ellenőrzését.

## 3. A programok telepítése

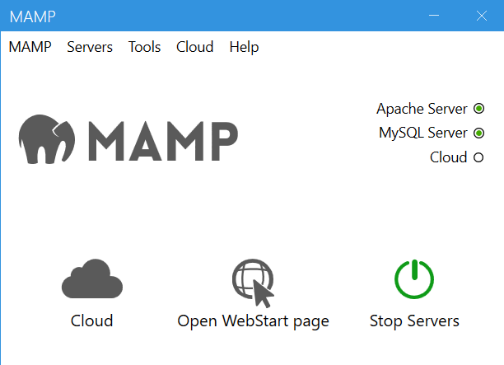
### 3.1. MAMP telepítése és konfigurálása:

* + Telepítés: Kövesse a mellékelt telepítő varázsló utasításait a MAMP telepítéséhez.

MAMP telepítése



* + phpMyAdmin használata: Indítsa el a MAMP-ot, majd a böngészőben nyissa meg a “http://localhost/phpMyAdmin” címet.



MAMP indítása

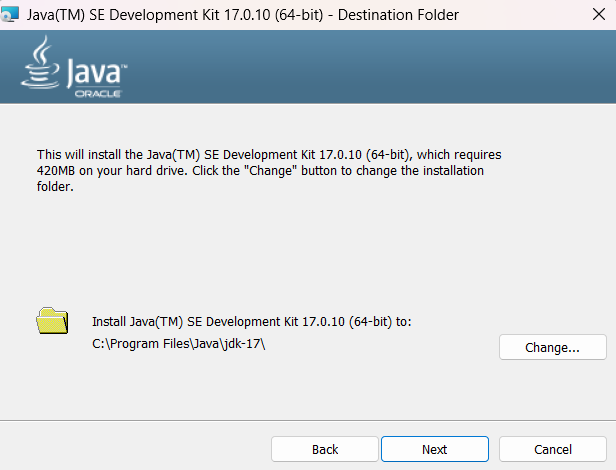
* + Importálja a mellékelt SQL fájlt (gb\_medical\_idopontfoglalo.sql) az új adatbázisba, majd az oldal alján kattintson az **Indítás** gombra.



Adatbázis importálása

### 3.2. JDK telepítése a NetBeans előtt:

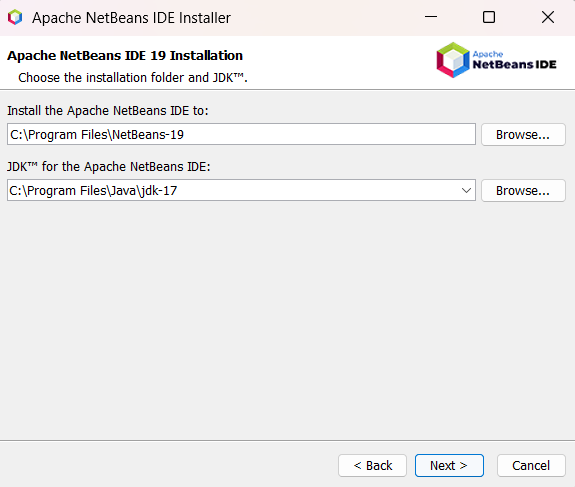
* + Telepítse mindkét JDK-t egy közös mappába, például a C:\Program Files\Java könyvtárba a mellékelt telepítők segítségével.
  + JDK 17: jdk-17.0.10\_windows-x64\_bin.exe
  + JDK 1.8.0: jdk-8u202-windows-x64.exe



JDK telepítés

### 3.3. NetBeans letöltése és konfigurálása:

* + A mellékelt telepítőcsomag tartalmazza a NetBeans 19 IDE telepítőjét. Indítsa el a NetBeans telepítőt.



NetBeans telepítés

* + A telepítés során válassza a JDK 17-et a legördülő menüben.

### 3.4. WildFly szerver konfigurálása

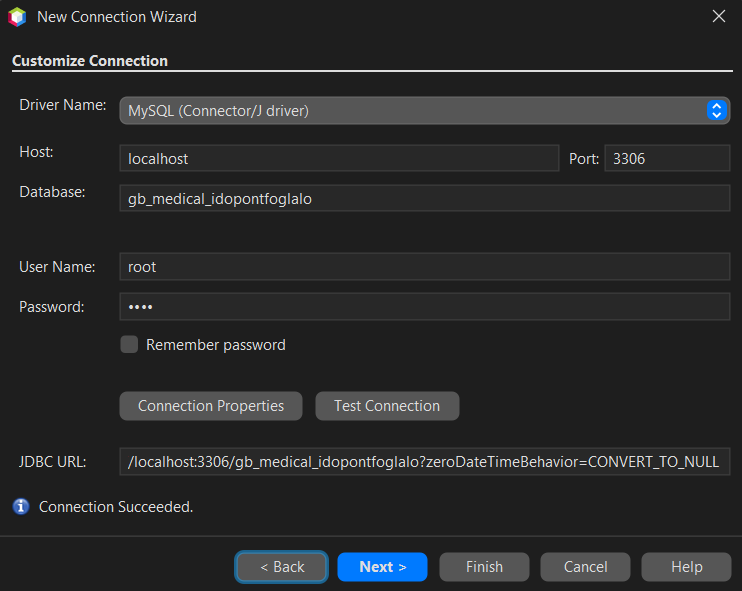
* + - 1. MySQL Connector elhelyezése
* Másolja a mellékelt **mysql-connector-java-8.0.23.jar** fájlt abba a mappába, ahol a JDK-k találhatók, például a **C:\Program Files\Java** mappába.

1. WildFly kicsomagolása

* Csomagolja ki a mellékelt **wildfly-preview-26.1.1.Final.zip** fájlt egy tetszőleges helyre, például a **C:\Program Files\wildfly-26.1.1.Final** mappába.

1. Adatbázis hozzáadása a NetBeans-ben

* Nyissa meg a **NetBeans 19** alkalmazást rendszergazdaként.
* Váltson a **Services** fülre.
* Nyissa le a **Databases** menüt.
* Kattintson jobb gombbal, és válassza a **New Connection...** opciót.
* A megjelenő ablakban válassza ki a **MySQL Connector** lehetőséget a **Driver** mezőben.
* A Driver File opciónál tallózza be a mysql-connector-java-8.0.23.jar fájlt.
* Kattintson a **Next** gombra.
* Adja meg az adatbázis csatlakozási adatokat:
  + - Database**:** gb\_medical\_idopontfoglalo
    - User Name: root
    - Password: root



Adatbázis csatlakoztatása

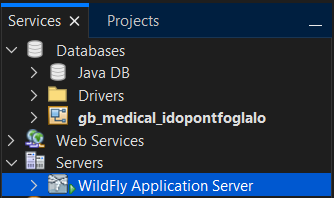
* A „**Test Connection**” gombbal ellenőrizheti a kapcsolat sikerességét.
* Kattintson a **Next** gombra.
* A Schema kiválasztásánál hagyja **üresen (No Schema).**
* A Connection Name mezőbe írja be: gb\_medical\_idopontfoglalo.
* Kattintson a **Finish** gombra.
* Az újonnan létrehozott adatbázis kapcsolatnál kattintson jobb gombbal, és válassza a **Connect** opciót.

4. WildFly szerver hozzáadása a NetBeans-ben

* Nyissa meg a NetBeans 19 alkalmazást rendszergazdaként.
* Váltson a **Services** fülre.
* Bontsa ki a **Servers** részt.
* Kattintson jobb gombbal, és válassza az **Add Server...** opciót.
* A megjelenő listából válassza a **WildFly** lehetőséget.
* A **Server Location** mezőben adja meg a kicsomagolt **WildFly** mappa elérési útját, például **C:\Program Files\wildfly-26.1.1.Final**.
* Kattintson a **Finish** gombra.

5. Szerver indítása

* A hozzáadott **WildFly** szerveren kattintson jobb gombbal.
* Válassza a **Start** opciót.
* Ha minden beállítás megfelelő, a szerver néhány másodperc alatt elindul.



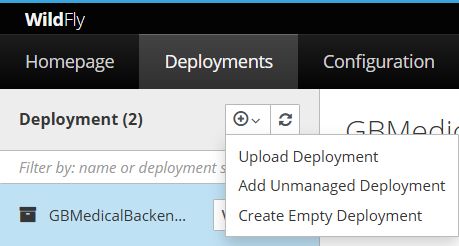
Szerver és adatbázis kapcsolat

6. Admin felület megnyitása

* Nyisson meg egy böngészőt, és navigáljon a következő URL-re:  
  [**http://127.0.0.1:9990/console/index.html#home**](http://127.0.0.1:9990/console/index.html#home)
* Bejelentkezéshez használja az alábbi adatokat:
  + - **Felhasználónév**: admin
    - **Jelszó:** Simonyi\_2024

7. Projekt deployolása a WildFly konzolon keresztül

* Nyissa meg a **GBMedicalBackend** projektet a **NetBeans-ben**. File > Open Project
* Kattintson jobb gombbal a projekt nevére, majd válassza a **Clean and Build** opciót, hogy a szükséges **.war** fájl elkészüljön.
* Nyissa meg a **WildFly admin konzolt** a fenti URL-en, és jelentkezzen be.
* Válassza a **Deployments** menüpontot.
* Kattintson a **+** jelre, majd **Upload Deployment**.

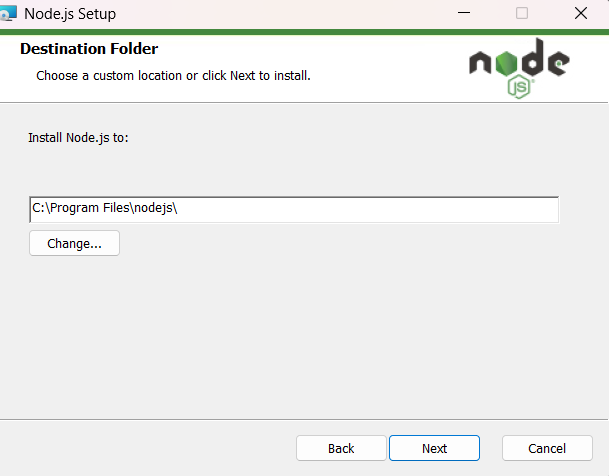


Backend deployolása szerverre

* Tallózással keresse meg és válassza ki a buildelt fájlt (GBMedicalBackend-1.0-SNAPSHOT.war) a Backend projekt **target** mappájában.
* Kattintson a **Next** > **Finish** gombra.

### 3.5. Frontend (Angular) telepítése

1. Node.js telepítése:
   * Töltse le a Node.js 22.9.0 verzióját a mellékelt telepítő segítségével. Csak a licenszt kell elfogadni, minden más „Next”.



Node.js telepítés

1. VS Code telepítése:
   * Töltse le a Visual Studio Code-ot a mellékelt telepítőcsomag segítségével.
2. Angular CLI telepítése:
   * Nyisson meg egy terminált vagy parancssort, és futtassa az alábbi parancsot:

**npm install -g @angular/cli**

* Ezután ellenőrizze a verziót a **ng version** paranccsal

1. Frontend projekt konfigurálása és buildelése:
   * Nyissa meg a VS Code-ot, majd nyissa meg a mellékelt Angular projekt mappáját. File > Open Folder > GBMedical
   * Nyomjon egy CTRL+J-t.
   * A projekt gyökerében futtassa az **npm install** parancsot.
   * Az Angular projekt buildeléséhez futtassa az **ng s -o** parancsot. Néhány másodperc múlva automatikusan megnyílik a weboldal a böngészőben <http://localhost:4200/home> címen.
   * A biztos működés érdekében frissítsen rá az oldalra.

A képen szöveg, képernyőkép, Webhely, multimédia látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

**Készen van! Üdvözlünk a weboldalunkon! 😊**

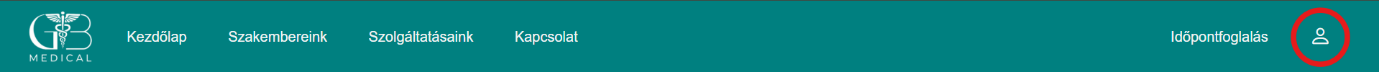
## 4. A program használatának részletes leírása

### 4.1 Bevezetés

Ez a program lehetővé teszi az időpontfoglalást, a foglalás lemondását, az orvosok értékelését, regisztrációt-bejelentkezést, illetve a kapcsolódó szolgáltatások megtekintését. A következő fejezetekben lépésről lépésre ismertetjük a program használatát, kezdve a felhasználói fiók létrehozásától egészen a foglalások kezeléséig.

### 4.2 Regisztráció és emailes megerősítés

Regisztráció folyamata:

1. Regisztrációs űrlap elérése:
   * A navigációs sávon kattintson a „felhasználó” ikonra. Keresse meg a „Regisztrálj!” aláhúzott szöveget és nyomjon rá.
2. Adatok megadása:
   * A képen szöveg, képernyőkép, Betűtípus, szám látható

     Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Töltse ki az űrlapot a következő mezőkkel: keresztnév, vezetéknév, érvényes email cím, telefonszám, jelszó. Ha már létezik az email cím vagy telefonszám, azt jelzi a rendszer egy felugró ablakkal.
3. Regisztráció elküldése:
   * Kattintson a "Regisztráció" gombra a folyamat befejezéséhez. A sikeres regisztráció után automatikusan átirányít a bejelentkező felületre.
4. Emailes megerősítés:
   * A képen szöveg, Betűtípus, fehér, képernyőkép látható

     Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A képen szöveg, képernyőkép, Betűtípus, embléma látható

     Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A rendszer automatikusan küld egy megerősítő emailt. Nyissa meg az email fiókját, keresse meg a levelet, majd kattintson a benne található aktiváló linkre „Megerősítés” felirattal.

Sikertelen megerősítés

Sikeres megerősítés

### 4.3 Bejelentkezés

Bejelentkezés lépései:

1. Bejelentkezési űrlap:
   * A navigációs sávon kattintson a „felhasználó” ikonra.
2. Adatok megadása:
   * Írja be a regisztráció során megadott email címet és jelszót.
3. Bejelentkezés:
   * A képen szöveg, képernyőkép, Betűtípus, szám látható

     Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Kattintson a "Bejelentkezés" gombra, ezután a rendszer ellenőrzi az adatokat. Sikeres bejelentkezés esetén visszairányít a főoldalra.

### 4.4 Szolgáltatás vagy orvos választása foglalás előtt

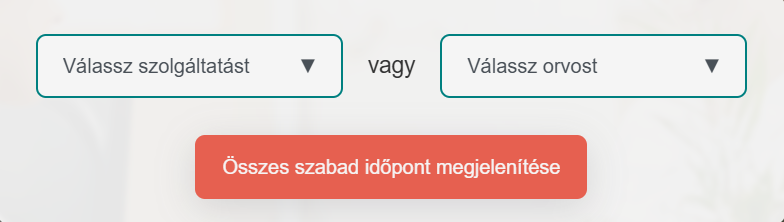
Funkció leírása:

* A felhasználó a foglalás megkezdése előtt választhat, hogy melyik szolgáltatáshoz vagy orvoshoz tartozó időpontra kíván foglalni. Az alatta lévő gombra kattintva megjelenítheti az összes időpontot. Ez a választás befolyásolja, hogy mely időpontok jelennek meg a naptárban.

Használat lépései:

1. Választási felület:
   * A főoldalon található két legördülő menü, ahol kiválaszthatja a kívánt szolgáltatást vagy orvost. A gomb a választástól függetlenül jeleníti meg az összes időpontot.
   * Válassza ki a kívánt opciót a listából vagy kattintson a gombra.

Választási lehetőségek



1. Időpontok frissítése:
   * A választás után a naptárban automatikusan megjelennek az időpontok.

### 4.5 Időpontfoglalás elérhető időpontokra

Funkció leírása:

* A naptárban megjelenő szabad időpontokra a felhasználó kattintással tud foglalást kezdeményezni. Fontos, hogy a foglalás csak bejelentkezett felhasználók számára érhető el, és a rendszer automatikusan emailben visszaigazolást küld a foglalás részleteivel.

Használat lépései:

1. Naptár megtekintése:
   * A képen szöveg, képernyőkép, Betűtípus, szám látható

     Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A felhasználó a választás után a naptárban látja a szabad időpontokat.

Szabad időpontok a naptárban

1. Időpont kiválasztása:
   * Kattintson a kívánt, szabad időpontra.
   * A képen szöveg, képernyőkép, Betűtípus, embléma látható

     Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Ha nincs bejelentkezve:

A „Bejelentkezés!” gombra kattintva a rendszer átirányít a bejelentkező felületre, a „Mégsem” gombra kattintva bezárja az ablakot.

1. A képen szöveg, képernyőkép, Betűtípus látható

   Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Foglalás megerősítése:
   * A rendszer egy felugró ablakban jeleníti meg a foglalás részleteit.
2. Visszaigazolás:
   * A képen szöveg, képernyőkép, Betűtípus, embléma látható

     Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A foglalás sikeres végrehajtása után a rendszer automatikusan visszaigazoló emailt küld a foglalás részleteivel.

### 4.6 Lefoglalt időpont megjelenítése és lemondása

Funkció leírása:

* A foglalás után a felhasználó által lefoglalt időpont a naptárban piros színnel jelenik meg. Erre az időpontra kattintva a felhasználó lemondhatja a foglalást, melyről szintén emailben kap visszaigazolást.

Használat lépései:

1. Foglalás megtekintése:
   * Az oldal frissítése után a lefoglalt időpont egy elkülönített, piros színnel jelenik meg a naptárban.
2. Lemondás indítása:
   * A képen szöveg, képernyőkép, Betűtípus, szám látható

     Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Kattintson a lefoglalt időpontra. Egy hasonló ablak jelenik meg, mint a foglalásnál.
3. Lemondás megerősítése:
   * A képen szöveg, képernyőkép, Betűtípus, embléma látható

     Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Erősítse meg a lemondást a megjelenő ablakban. A felhasználó csak a saját időpontját tudja lemondani.
4. Email visszaigazolás:
   * A rendszer emailben értesíti a felhasználót a foglalás lemondásáról, a foglalás részleteivel.

### 4.7 Orvosok értékelése és visszajelzések megtekintése

Funkció leírása:

* A bejelentkezett felhasználók csillagos (numerikus) és opcionálisan szöveges értékelést adhatnak az orvosokra. A korábbi értékelések nyilvánosan megtekinthetőek.

A képen szöveg, képernyőkép, Betűtípus, embléma látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Ha nincs bejelentkezve:

Használat lépései:

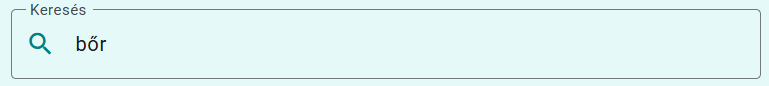
1. Értékelési felület elérése:
   * Miután bejelentkezett, kattintson a „Szakembereink” menüpontra. Az értékelni kívánt orvos alatt kattintson az „Értékelés hozzáadása” gombra.
2. Értékelés megadása:
   * Válassza ki a csillagok számát, és (opcionálisan) írja be a szöveges visszajelzést.
3. Értékelés beküldése:
   * Kattintson a "Küldés" gombra.
4. Értékelések megtekintése:
   * A képen szöveg, képernyőkép, Betűtípus, névjegykártya látható

     Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A képen szöveg, képernyőkép, Betűtípus, embléma látható

     Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A korábbi értékelések a kiválasztott orvos adatlapján jelennek meg, így a felhasználó összehasonlíthatja mások véleményét.

### 4.8 Orvoskeresés szűrőmező segítségével

Funkció leírása:

* A felhasználó egy keresőmező segítségével tudja szűrni az orvosokat különböző paraméterek alapján, mint név, email cím, telefonszám vagy praxis.

Használat lépései:

1. Keresőmező használata:
   * A „Szakembereink” oldalon található keresőmezőbe írja be a kívánt szűrőkifejezést.
2. Szűrés:
   * A rendszer azonnal frissíti az eredményeket és kiemeli a találatokat a megadott keresőkifejezés alapján.
3. A képen szöveg, Emberi arc, ruházat, személy látható

   Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Találatok megtekintése:
   * A felhasználó megtekintheti azokat az orvosokat, akik megfelelnek a keresési feltételeknek.

### 4.9 Szolgáltatás adatainak megtekintése legördülő menüvel

Funkció leírása:

* A felhasználó egy legördülő menü segítségével tekintheti meg az elérhető szolgáltatásokat. A kiválasztott szolgáltatáshoz tartozó részletes adatok – leírás, ár, időtartam, a hozzá tartozó orvos(ok)– automatikusan megjelennek.

Használat lépései:

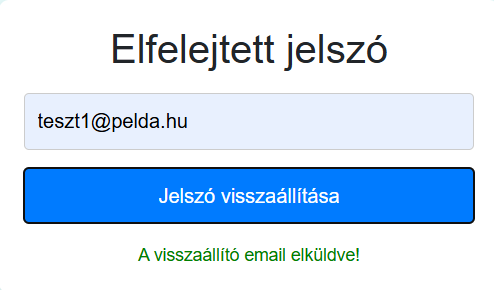
1. Legördülő menü megnyitása:
   * Kattintson a szolgáltatások legördülő menüjére.
2. Szolgáltatás kiválasztása:
   * Válassza ki a kívánt szolgáltatást a listából.
3. Részletes adatok megtekintése:
   * A kiválasztott szolgáltatás adatlapja megjelenik, ahol az összes részlet látható.

### 4.10 Jelszó helyreállítás

Funkció leírása:

* Elfelejtett jelszó esetén a felhasználó képes helyreállító linket kérni az email címére. A linkre kattintva egy külön oldalon új jelszót adhat meg.

Használat lépései:

1. Jelszó helyreállítási űrlap:
   * A bejelentkezési oldalon kattintson az "Elfelejtetted a jelszavad?" linkre.
2. Email megadása:
   * Írja be az email címét a helyreállítási űrlapon.
3. Link fogadása:
   * A rendszer egy helyreállító linket küld az adott email címre.
4. Új jelszó megadása:
   * A képen szöveg, képernyőkép, Betűtípus, szám látható

     Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Kattintson a kapott linkre, majd az új jelszó mezőbe írja be a kívánt jelszót, és erősítse meg azt.

### 4.11 Oldal tetejére visszaugró gomb

Funkció leírása:

* Az oldal jobb alsó sarkában található fix pozíciójú gomb segít, hogy a felhasználó bármikor visszaugorjon az oldal tetejére. Ez különösen hasznos hosszú oldalak böngészésekor.

Használat lépései:

1. Gomb megjelenése:
   * Miután az oldal görgetése során az oldal aljára ér, a visszaugró gomb megjelenik.
2. Gomb használata:
   * ****Kattintson a gombra, és a rendszer automatikusan visszagörget az oldal tetejére.

### 4.12 Admin oldal és felhasználó törlése

Funkció leírása:

* Ha admin fiókkal lépünk be, a rendszer automatikusan a **/admin** oldalra irányít. Itt egy táblázatban megjelenik az összes aktív és törölt felhasználó. Az utolsó oszlopban egy „kuka” ikon található, amelynek megnyomásával a kiválasztott felhasználó logikailag törlésre kerül.

**Fontos**: Ha kijelentkezve az URL-be beírjuk, hogy /admin, átírányít a bejelentkező felületre.  
Ha sima felhasználóként vagyunk bejelentkezve és úgy próbáljuk beírni, kiírja, hogy nincs ehhez jogosultságod és visszadob a főoldalra.

A törlés során:

* Az adott felhasználó **isDeleted** mezője **1-re** állítódik az adatbázisban, így a rendszer inaktívként kezeli.
* A felhasználó ezután nem tud belépni a fiókjába.
* A táblázat isDeleted mezője frissül, de a felhasználó továbbra is látható a táblázatban.

# III. A fejlesztői dokumentáció

## 1. Témaválasztás indoklása

A választott témakör – egy online időpontfoglaló rendszer fejlesztése, amely magában foglalja a szolgáltatások és az orvosok adatainak kezelését, valamint a felhasználói értékeléseket és visszajelzéseket – több okból is releváns és aktuális. Elsősorban a modern egészségügyi rendszerek digitalizációja, az online szolgáltatások iránti növekvő igény, valamint a felhasználói élmény javítása indokolja a projekt megvalósítását. A rendszer lehetővé teszi a páciensek számára, hogy egyszerűen és átlátható módon foglaljanak időpontot, illetve visszajelzéseket adjanak az orvosokról, ami hozzájárul a minőség folyamatos javításához. Emellett a projekt fejlesztése során számos modern technológiai és programozási eszközt alkalmaztunk, amelyek segítségével a kód jól strukturált, könnyen karbantartható és bővíthető lett. Ezen indoklások tükrében a választott téma nemcsak piacképes, hanem szakmailag is kihívást jelent, amely hozzájárul a fejlesztői ismeretek bővítéséhez és a jövőbeli innovációk támogatásához.

## 2. Az alkalmazott fejlesztői eszközök

A projekt fejlesztése során több különböző eszközt és technológiát alkalmaztunk, amelyek mindegyike hozzájárult a rendszer funkcionalitásának és minőségének megvalósításához. Az alábbiakban felsoroljuk a legfontosabb fejlesztői eszközöket és komponenseket:

* Programozási nyelv és platformok:
  + **Java:** A backend fejlesztéshez használtuk, amely a szerveroldali logika megvalósítását szolgálja.
  + **Angular 19:** A frontend fejlesztéshez alkalmazott modern, TypeScript-alapú keretrendszer, amely komponensekkel építhető fel.
* Fejlesztői környezetek:
  + **NetBeans 19 IDE:** A Java alapú backend fejlesztéshez használt integrált fejlesztőkörnyezet, amely támogatja a kód szervezését és hibakeresését.
  + **Visual Studio Code:** A frontend fejlesztéshez alkalmazott könnyen használható és testreszabható szerkesztő.
* Adatbázis-kezelő rendszer:
  + **MAMP:** A phpMyAdmin felület segítségével végezzük az adatbázis importálását és kezelését. Az SQL fájlokban definiált adatmodell a rendszer működésének alapját képezi.
  + **MySQL:** A projekt adatbázisa, amelyben a felhasználói, orvosi és foglalási adatok tárolásra kerülnek.
* Szerver környezet:
  + **WildFly:** A Java alapú backend alkalmazás futtatásához használt szerver, amely a deployolási folyamat során biztosítja az alkalmazás stabil futását.
  + **MySQL Connector/J:** A Java és a MySQL adatbázis közötti kapcsolatot biztosító JDBC driver, amely a mellékelt „mysql-connector-java-8.0.23.jar” fájl segítségével integrálódik.
* Egyéb eszközök és modulok:
  + **Angular CLI:** Az Angular projekt gyorsabb buildelését és fejlesztését segítő parancssori eszköz.
  + **Jira:** A projektmenedzsmenthez és a feladatok nyomon követéséhez alkalmazott szoftver, amely támogatja a fejlesztési folyamat átláthatóságát.
  + **GitHub**: A forráskód verziókövetésére és csapatmunka támogatására használt platform. Lehetővé teszi a kód tárolását, verziókezelését, valamint az együttműködést pull requestek, issue-k és branch-ek használatával. A projekt GitHub repository-jában került tárolásra a backend, a frontend, valamint az adatbázis sémák és a dokumentáció egyes részei is külön brancheken.
  + **Postman**: Endpointok teszteléséhez használt program
* Dokumentációs eszközök:
  + **Google Docs / MS Word:** A felhasználói és fejlesztői dokumentáció elkészítéséhez használt szövegszerkesztő, amely lehetővé teszi a dokumentumok könnyű szerkesztését és megosztását.

## 3. Tervezési módszer

Ebben a részben ismertetjük az alkalmazás tervezési megközelítését, a választott módszertant, a vizuális modellezést, valamint a rendszer architektúráját, különös tekintettel az MVC mintára és a konfigurációs modulban megvalósított JWT generálás és validáció funkciókra.

### 3.1 Alkalmazás tervezése

A projekt célja egy online időpontfoglaló rendszer létrehozása, amely lehetővé teszi, hogy a felhasználók szolgáltatás vagy konkrét orvos alapján válasszanak időpontot, illetve foglalást, értékeléseket adjanak és visszajelzéseket olvassanak. A tervezés során a skálázhatóság, a moduláris felépítés és a könnyű karbantarthatóság voltak a fő szempontok. A rendszer három fő komponensre épül: a frontend felhasználói felületre, a backend üzleti logikára, valamint az adattárolást biztosító adatbázisra, amelyek RESTful API-kon keresztül kommunikálnak.

### 3.2 Tervezési módszertan ismertetése

A projekt fejlesztése során az alábbi megközelítéseket alkalmaztuk:

* **Moduláris és rétegezett architektúra:**  
  A rendszer különálló modulokra tagolódik, melyek egymástól elkülönítve kezelik a felhasználói, foglalási, és értékelési funkciókat. Ez elősegíti a kód újrahasznosíthatóságát és a rendszer könnyű karbantartását.
* **MVC architektúra:**  
  Az alkalmazás a Model-View-Controller (MVC) minta mentén lett kialakítva, melyben:
  + **Model:** Az adatstruktúrák, entitások és adatbázis-interakciók kezelése történik.
  + **Service:** Az üzleti logika és a különböző funkciók közötti összeköttetést biztosító réteg.
  + **Controller:** Az API-k és felhasználói kérések fogadásáért, valamint a válaszok elkészítéséért felelős réteg.
  + **Config:** A rendszer konfigurációs beállításait tartalmazza, beleértve a biztonsági konfigurációkat. Itt kerül megvalósításra a JWT (JSON Web Token) generálás és validáció, amely a felhasználók hitelesítését és az API-k biztonságát szolgálja. A JWT konfigurációs modul felelős a tokenek aláírásáért, a lejárati idő beállításáért, és a tokenek validációjáért, így biztosítva, hogy csak hitelesített felhasználók férjenek hozzá a védett erőforrásokhoz.
  + **+1 Filters:** Ezen a rétegen keresztül történik a bejövő kérések és a kimenő válaszok előfeldolgozása, például a CORS (Cross-Origin Resource Sharing) beállítások kezelése. A *CorsFilter.java* osztályban a CORS fejléc beállításait valósítjuk meg, hogy engedélyezzük a kliens (például az Angular alapú <http://localhost:4200>) és a szerver közötti biztonságos kommunikációt.
* **Iteratív fejlesztési ciklus:**  
  Az iteratív és inkrementális fejlesztési modell alkalmazásával a rendszer kisebb részegységeit folyamatosan fejlesztettük és teszteltük, így biztosítva a fokozatos finomítást és a hibák korai elhárítását.

### 3.3 Vizuális modellezés

A rendszer vizuális modellezését több eszközzel végeztük el:

* **Adatbázis modellezés:**  
  Az adatbázis struktúráját a dbdiagram.io segítségével modelleztük, amely áttekinthetően mutatja a táblák, kapcsolatok és attribútumok felépítését.
* **Design terv:**  
  A felhasználói felület és az alkalmazás vizuális megjelenítését a Figma-ban terveztük meg, amely támogatja a fejlesztők és tervezők közötti hatékony kommunikációt. A kezdeti designtervek irányt mutattak az Angular frontend fejlesztés során, azonban a megvalósítás során bizonyos elemek módosultak. Egyes funkciókat egyszerűsítettünk vagy átalakítottunk a felhasználói élmény javítása és a technikai megvalósíthatóság érdekében. Ezeket a változtatásokat iteratív módon, a fejlesztési folyamat részeként vezettük be.

### 3.4 Objektum-orientált programozás (OOP) megközelítése az MVC struktúrában

Az OOP alapelveit – mint az enkapszuláció, öröklődés, polimorfizmus és absztrakció – integráltuk a rendszer fejlesztésébe, különösen az MVC architektúra keretében:

* **Enkapszuláció:**  
  Az adatok és a hozzájuk tartozó metódusok elkülönítése érdekében az egyes osztályokban privát adattagokat és nyilvános interfészeket alkalmaztunk. Ez biztosítja az adatok integritását.
* **Öröklődés és polimorfizmus:**  
  Az általános működés leírására létrehozott ősosztályokból származtatott konkrét osztályok segítségével a kód újrahasznosítható és könnyen bővíthető maradt. Bár a hangsúly az MVC tiszta elkülönítésén van, ez a megközelítés támogatja a moduláris fejlesztést.
* **Absztrakció:**  
  Interfészek és absztrakt osztályok használatával meghatároztuk a rendszer fő funkcionális követelményeit, amelyek alapján az egyes komponensek konkrét implementációja kialakult.
* **JWT konfiguráció:**  
  A **Config** modulban implementáltuk a JWT generálást és validációt. Ebben a részben:
  + A felhasználói hitelesítés során egy titkos kulcs segítségével generálódnak a tokenek.
  + A tokenek tartalmazzák a felhasználó azonosítóját és egyéb szükséges információkat.
  + A rendszer minden bejövő kérést ellenőriz a tokenek validációja révén, biztosítva, hogy csak hitelesített felhasználók férjenek hozzá a védett API-khoz.

Ezzel a megoldással a biztonságos kommunikáció és a jogosultság-ellenőrzés integrálva van az alkalmazásba, ami kulcsfontosságú a modern webalkalmazások esetében.

## 4. Adatmodell leírása

### 4.1 Áttekintés

A projekt adatmodellje egy relációs adatbázis struktúrán alapul, amelyet a dbdiagram.io segítségével terveztünk meg. Az adatbázis célja, hogy tárolja az időpontfoglalással, felhasználói adatokkal, orvosi információkkal, foglalásokkal, értékelésekkel és egyéb, a rendszer működéséhez szükséges adatszerkezetekkel kapcsolatos információkat. A modell a normalizáció alapelveit követi, így a redundancia minimalizálva van, és az adatok integritása garantált.

### 4.2 Adatbázis táblák és kapcsolatok

DBdiagram-modell

**Patients tábla:**

* **id:** Egyedi azonosító, autoinkrementálva (elsődleges kulcs).
* **name:** A páciens teljes neve.
* **email:** A páciens email címe, egyedi azonosítóként is szolgál (pl. bejelentkezéshez).
* **phone\_number:** Egyedi telefonszám, amely a páciens elérhetőségét biztosítja.
* **password:** A páciens jelszava, ideális esetben hash-elve tárolva a biztonság érdekében.
* **isAdmin:** Logikai érték, amely jelzi, hogy a felhasználó adminisztrátori jogosultságokkal rendelkezik.
* **is\_deleted:** Logikai jelző, amely azt mutatja, hogy a rekord törlésre került-e.
* **created\_at:** A rekord létrehozásának időpontja.
* **updated\_at:** A rekord utolsó módosításának időpontja.
* **deleted\_at:** A rekord törlésének időpontja (ha nem törölt, akkor értéke NULL).

**Doctors tábla:**

* **id:** Egyedi azonosító, autoinkrementálva (elsődleges kulcs).
* **name:** Az orvos neve.
* **email:** Az orvos email címe, egyedi és a kommunikációhoz is szükséges.
* **phone\_number:** Egyedi telefonszám az orvos elérhetőségéhez.
* **password:** Az orvos jelszava, biztonsági okokból hash-elve.
* **bio:** Rövid életrajzi leírás vagy szakmai összefoglaló az orvosról.
* **is\_deleted:** Logikai jelző, amely megmutatja, hogy a rekord törlésre került-e.
* **created\_at:** A rekord létrehozásának időpontja.
* **updated\_at:** A rekord utolsó módosításának időpontja.
* **deleted\_at:** A rekord törlésének időpontja (ha nem törölt, akkor NULL).

**Appointments tábla:**

* **id:** Egyedi azonosító, autoinkrementálva.
* **doctor\_id:** Idegen kulcs, amely az orvos azonosítójára hivatkozik.
* **patient\_id:** Idegen kulcs, amely a páciens azonosítójára hivatkozik.
* **start\_time:** Az időpontfoglalás kezdete.
* **end\_time:** Az időpontfoglalás vége.
* **duration:** A foglalás időtartama percekben.
* **status:** A foglalás aktuális állapota (például booked, cancelled, completed).
* **is\_deleted:** Logikai jelző, hogy a foglalás törölve lett-e.
* **created\_at:** A foglalás létrehozásának időpontja.
* **updated\_at:** A foglalás utolsó módosításának időpontja.
* **deleted\_at:** A foglalás törlésének időpontja (ha érvénytelen, akkor nem NULL).

**Payments tábla**:

* **id:** Egyedi azonosító, autoinkrementálva.
* **appointment\_id:** Idegen kulcs, az Appointments tábla azonosítójára hivatkozik.
* **amount:** A fizetendő összeg.
* **payment\_method:** A fizetési mód (például online vagy in-person).
* **payment\_status:** A fizetés állapota (például paid, pending, failed).
* **is\_deleted:** Logikai jelző, amely a törölt fizetési tranzakciókat jelzi.
* **created\_at:** A tranzakció létrehozásának időpontja.
* **updated\_at:** A tranzakció utolsó módosításának időpontja.
* **deleted\_at:** A tranzakció törlésének időpontja (ha alkalmazandó).

**Reminders tábla**:

* **id:** Egyedi azonosító, autoinkrementálva.
* **appointment\_id:** Idegen kulcs, az Appointments tábla azonosítójára hivatkozik.
* **reminder\_time:** Az emlékeztető küldésének időpontja.
* **reminder\_method:** Az emlékeztető küldésének módja (pl. email, SMS, push értesítés).
* **sent:** Logikai érték, amely jelzi, hogy az emlékeztető már elküldésre került-e.
* **is\_deleted:** Logikai jelző, hogy az emlékeztető törölve lett-e.
* **created\_at:** A rekord létrehozásának időpontja.
* **updated\_at:** A rekord utolsó módosításának időpontja.
* **deleted\_at:** A rekord törlésének időpontja.

**Schedules tábla**:

* **id:** Egyedi azonosító, autoinkrementálva.
* **doctor\_id:** Idegen kulcs, az orvos azonosítójára hivatkozik.
* **start\_time:** A rendelkezésre állás kezdete.
* **end\_time:** A rendelkezésre állás vége.
* **available\_slots:** Az adott időintervallumban elérhető időpontok száma.
* **is\_deleted:** Logikai jelző, hogy a rekord törlésre került-e.
* **created\_at:** A rekord létrehozásának időpontja.
* **updated\_at:** A rekord utolsó módosításának időpontja.
* **deleted\_at:** A rekord törlésének időpontja.

**Reviews tábla**:

* **id:** Egyedi azonosító, autoinkrementálva.
* **doctor\_id:** Idegen kulcs, az orvos azonosítójára hivatkozik.
* **patient\_id:** Idegen kulcs, a páciens azonosítójára hivatkozik.
* **rating:** Az értékelés számszerűsített formája (1-től 5-ig).
* **review\_text:** A felhasználó által írt szöveges visszajelzés.
* **created\_at:** Az értékelés rögzítésének időpontja.

**Services tábla**:

* **id:** Egyedi azonosító, autoinkrementálva.
* **name:** A szolgáltatás neve.
* **description:** Részletes leírás a szolgáltatásról.
* **price:** A szolgáltatás ára.
* **duration:** A szolgáltatás időtartama percekben.
* **is\_deleted:** Logikai jelző, amely azt mutatja, ha a szolgáltatás törölve lett.
* **created\_at:** A rekord létrehozásának időpontja.
* **updated\_at:** A rekord utolsó módosításának időpontja.
* **deleted\_at:** A rekord törlésének időpontja.

**Doctors\_x\_Services tábla**:

* **doctor\_id:** Idegen kulcs, az orvos azonosítójára hivatkozik.
* **service\_id:** Idegen kulcs, a szolgáltatás azonosítójára hivatkozik.
* **Funkció:** Ez a kompozit elsődleges kulcsot alkotó tábla a sok-sok kapcsolat kezelésére szolgál, meghatározva, hogy melyik orvos mely szolgáltatásokat kínálja.

### 4.3 Kiegészítő táblák (fejlesztés során létrehozott)

**password\_reset\_tokens tábla**:

* **id**: Egyedi azonosító, elsődleges kulcs, autoinkrementálva.
* **email**: A felhasználó email címe, amelyhez a token tartozik.
* **token**: A generált egyedi token, amely a jelszó visszaállításához szükséges.
* **expires\_at**: A token lejárati ideje, amely után a token érvénytelen.
* **used**: Logikai (boolean) érték, amely jelzi, hogy a token már fel lett-e használva.
* **created\_at**: A token létrehozásának időpontja.

**patient\_verifications tábla**:

* **id**: Egyedi azonosító, elsődleges kulcs, autoinkrementálva.
* **patient\_id**: A kapcsolódó páciens azonosítója (idegen kulcs a Patients táblából).
* **token**: A regisztráció megerősítéséhez generált token.
* **verified**: Logikai (boolean) érték, amely megmutatja, hogy a regisztrációt megerősítették-e.
* **created\_at**: A token létrehozásának időpontja.
* **expires\_at**: A token érvényességi ideje.
* **verified\_at**: Az az időpont, amikor a regisztráció megerősítése megtörtént.

Funkcióik:

* A **password\_reset\_tokens** tábla a felhasználók jelszavának visszaállításához szükséges tokeneket tárolja. Az email címmel összekapcsolt token segítségével biztosítható, hogy csak az arra jogosult felhasználó tudja a jelszót visszaállítani.
* A **patient\_verifications** tábla az emailes regisztráció megerősítéséhez szükséges tokeneket és azok állapotát tartalmazza. A regisztráció során generált token alapján a felhasználó megerősítheti email címét, így aktiválva fiókját.

### 4.4 Normalizálás folyamata

Az adatmodell kialakítása során törekedtünk a normalizációra, elsősorban a harmadik normálformáig (3NF):

**Első normálforma (1NF)**

* **Atomi értékek:** Minden tábla oszlopában csak egyszerű (atomikus) értékek szerepelnek, tehát nem tartalmazhat például tömböket vagy listákat.
* **Egyediség:** Minden rekord (sor) egyedi, így nincs duplikáció az adatok között. Ez általában elsődleges kulccsal van biztosítva.

**Második normálforma (2NF)**

* **Teljes függőség:** Az 2NF-ben arra törekszünk, hogy minden nem kulcs attribútum (oszlop) teljesen függjen az elsődleges kulcstól. Ez azt jelenti, hogy ha a tábla összetett elsődleges kulccsal rendelkezik (több oszlop alkotta), akkor semmilyen nem kulcs oszlop nem függhet csak az elsődleges kulcs egy részétől.
* **Részleges függőség elkerülése:** Ha egy nem kulcs attribútum csak egy részhez kapcsolódna az összetett kulcsból, az adatismétléshez és frissítési problémákhoz vezethet. A 2NF ezt kizárja, így az adatok egységesebbek és könnyebben karbantarthatók.

**Harmadik normálforma (3NF)**

* **Transzitív függések megszüntetése:** A 3NF-ben biztosítjuk, hogy minden nem kulcs attribútum közvetlenül a kulcstól függjön, és ne legyenek olyan esetek, amikor egy nem kulcs attribútum egy másik nem kulcs attribútumtól függ. Ez a transzitív függés kizárása segít elkerülni, hogy egy változás (például egy adat módosítása) több helyen is hatással legyen, így csökkenti az adatinkonzisztencia kockázatát.
* **Közvetlen kapcsolat:** Az adatok közvetlenül kapcsolódnak az elsődleges kulcshoz, ami a logikus és egyértelmű adatszerkezetet támogatja.

A Doctors\_x\_Services tábla például a sok-sok kapcsolat normalizálását segíti elő, elkerülve a duplikációkat és redundanciát a Doctors és Services táblákban.

### 4.5 OOP jellegű megvalósítás:

* **Model osztályok:** Entitások, melyek reprezentálják az adatbázis tábláit.
* **Controller osztályok:** Az API-k és a felhasználói kérések kezelésére szolgálnak.
* **Service osztályok:** Az üzleti logikát és a modell adatainak feldolgozását végzik.
* **Config osztályok:** A rendszer konfigurációs beállításait, köztük a JWT token generálás és validáció funkcióit tartalmazzák.
* **Filters:** Ezen a rétegen keresztül történik a bejövő kérések és a kimenő válaszok előfeldolgozása, például a CORS (Cross-Origin Resource Sharing) beállítások kezelése. A *CorsFilter.java* osztályban a CORS fejléc beállításait valósítjuk meg, hogy engedélyezzük a kliens (például az Angular alapú <http://localhost:4200>) és a szerver közötti biztonságos kommunikációt.

### 4.6 Biztonsági kérdések, titkosítás és GDPR

A rendszer fejlesztése során különös figyelmet fordítottunk a biztonsági szempontokra:

* **Adatvédelem és GDPR:**  
  Az adatbázisban tárolt személyes adatok (például email, telefonszám, jelszó) kezelését a GDPR előírásainak megfelelően valósítottuk meg.
  + A felhasználói jelszavakat hash-elés (**SHA1**) módszerével védjük, így azok nem tárolódnak plain text formában.
  + Csak a szükséges adatokat kérjük be a felhasználótól, és biztosítjuk, hogy az adatokhoz csak jogosultsággal rendelkező személyek férjenek hozzá.
* **Titkosítás:**  
  A kommunikáció biztonságos HTTPS protokollon keresztül történik, így az adatátvitel során az érzékeny információk titkosítva kerülnek továbbításra.
* **JWT és API biztonság:**  
  A backend konfigurációs moduljában implementáltuk a JWT (JSON Web Token) generálást és validációt. Ez biztosítja, hogy az API-khoz csak hitelesített felhasználók férjenek hozzá, és megakadályozza az illetéktelen hozzáférést.

## 5. Részletes feladatspecifikáció, algoritmusok

Az alábbi pseudokódok bemutatják a rendszer legfontosabb függvényeit/eljárásait.

**JWT Token Generálás – Pseudokód:**

**Függvény**: createJWT(páciens) – Backend, config réteg, JWT.java class

**Bemenet**: páciens objektum (id, isAdmin, createdAt)

**Kimenet**: JWT token (String)

1. Állítsd be az aktuális időt: now = most()

2. Állítsd be az aláírás titkos kulcsát (SECRET)

3. Építsd fel a tokent a következő adatokkal:

- issuer: "GBmedical"

- subject: "test"

- claim: id = páciens.id

- claim: isAdmin = páciens.isAdmin

- claim: createdAt = páciens.createdAt

- issuedAt = now

- expiration = now + 1 nap

4. Írd alá a tokent HS256 algoritmussal és SECRET kulccsal

5. Térj vissza a legenerált tokennel

**JWT Token Validálás – Pseudokód**

**Függvény**: validateJWT(token) – Backend, config réteg, JWT.java class

**Bemenet**: token (String)

**Kimenet**: validáció eredménye (int)

1. Próbáld meg dekódolni a JWT tokent a SECRET kulccsal

2. Ha sikeres:

a. Olvasd ki az "id" értékét a tokenből

b. Hozz létre egy új Patients objektum azonosítóval

c. Ha a tokenből kiolvasott id megegyezik a Patients objektumban lévővel:

- térj vissza 1 (érvényes token)

d. Egyébként térj vissza 2 (érvénytelen token)

3. Ha hiba történik (lejárt, hibás aláírás, rossz formátum stb.):

- Térj vissza 3 (lejárt vagy hibás token)

**Felhasználó ID-jának kiolvasása**

A getUserId() függvény célja, hogy a frontend oldalon a felhasználó azonosítóját visszakérdezze a böngésző helyi tárolójából (**localStorage**) elmentett JWT token alapján. Ez az azonosító kulcsfontosságú a foglalási és értékelési funkciók során, hiszen a rendszer így tudja, melyik felhasználó végezte az adott műveletet.

**Függvény**: getUserId() – Frontend, auth.service.ts

**Bemenet**: Nincs (a localStorage használata miatt globális adatforrás)

**Kimenet**: Felhasználó azonosító (number) vagy null

1. Olvasd ki a token-t a localStorage-ból az 'authToken' kulcs segítségével.

2. Ha a token nem létezik, térj vissza null értékkel.

3. Dekódold a token-t a decodeToken metódussal.

4. Ha a dekódolt token tartalmaz egy 'id' mezőt:

- Alakítsd az 'id'-t számmá.

- Térj vissza az 'id' értékével.

Egyébként térj vissza null értékkel.

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

**Időpontfoglalás és értesítés – Pseudokód (AppointmentService)**

**Függvény**: addAppointmentWithNotification (doctorId, patientId, startTime, endTime) – Backend (Service réteg)

**Bemenet**: orvosId, páciensId, kezdési idő, befejezési idő

**Kimenet**: JSON objektum válasz

1. Hozz létre egy JSON objektumot visszatérési értékként

2. Alapértelmezetten állítsd be a státuszt: "success", státuszkód: 200

3. Ha bármelyik bemeneti paraméter érvénytelen:

- Állítsd a státuszt: "invalidInput"

- Állítsd a státuszkódot: 400

4. Egyébként:

a. Hívd meg a model.addAppointmentWithNotification metódust az időpont létrehozásához

b. Ha a metódus hamisat ad vissza:

- Állítsd a státuszt: "modelException"

- Állítsd a státuszkódot: 500

c. Ha sikeres:

- Lekérjük a páciens teljes nevét, email címét

- Lekérjük az orvos nevét és a szolgáltatás nevét

- Összeállítjuk a HTML formátumú értesítő email tartalmát

- Meghívjuk az EmailService.sendEmail függvényt a levél kiküldéséhez

- A válaszba beírjuk a foglalás adatait: orvosId, páciensId, kezdési és befejezési idő

5. Visszatérünk a JSON objektummal, amely tartalmazza a státuszt, státuszkódot, és ha sikeres, a foglalás részleteit

**Időpontfoglalás lemondása és értesítés – Pseudokód (AppointmentService)  
Függvény:** cancelAppointment (appointmentId, patientId) – Backend (Service réteg)  
**Bemenet:** appointmentId, patientId **Kimenet:** JSON objektum válasz

1. Hozz létre egy JSON objektumot visszatérési értékként.
2. Alapértelmezetten állítsd be a státuszt: "success", és a státuszkódot: 200.
3. Próbáld meg: a. Hívd meg a model.cancelAppointment metódust a foglalás lemondásához. b. Ha a metódus visszaadja, hogy a művelet sikertelen (modelResult["success"] hamis):
   * Dobj kivételt a modelResult["error"] üzenettel. c. Ha a lemondás sikeres:
   * Kérd le a foglalás részleteit a modelResult-ból (appointmentDetails).
   * Állítsd össze a HTML formátumú értesítő email tartalmát, amely tartalmazza:
     + A páciens nevét.
     + Az orvos nevét.
     + A szolgáltatásokat.
     + Az időpontot (kezdési és befejezési idő).
   * Hívd meg az EmailService.sendEmail függvényt a következő paraméterekkel:
     + Címzett: details["patientEmail"]
     + Email típus: APPOINTMENT\_CANCELLATION
     + Tartalom: a fenti HTML emailContent
   * Illeszd be a válasz JSON objektumba a modelResult-t és az email küldés eredményét.
4. Ha kivétel keletkezik:
   * Állítsd be a státuszt "error"-ra, és a státuszkódot 500-ra.
   * Tedd bele a JSON objektumba a hibaüzenetet (e.getMessage()).
5. Állítsd be a JSON objektumban a végső státuszt és státuszkódot.
6. Térj vissza a JSON objektummal, amely tartalmazza a státuszt, a státuszkódot, és ha sikeres, a foglalás részleteit.

**Szabad időpontok megjelenítése – getAvailableSlots (tárolt eljárás):**  
**Függvény**: getAvailableSlots()

**Bemenet**: Nincs

**Kimenet**: Lista a szabad időpontokról, tartalmazza a slot\_start, slot\_end, doctor\_id, doctor\_name, service\_id, service\_name értékeket

1. Hozz létre egy ideiglenes táblát "TimeSlots" a következő lépések szerint:

a. Kérdezd le a "schedules" tábla rekordjait.

b. Minden schedule rekordhoz generálj 30 perces slotokat:

- Definiálj egy iterációs változót n.x, amely 0-tól 23-ig megy (azaz 24 slotot jelenthet).

- Számítsd ki:

- slot\_start = s.start\_time + (n.x \* 30 perc)

- slot\_end = s.start\_time + ((n.x + 1) \* 30 perc)

- Csak azokat a slotokat add hozzá, ahol (n.x \* 30) kevesebb, mint a s.start\_time és s.end\_time közötti különbség percekben.

2. Lekérdezés a szabad időpontok meghatározására:

a. Végezzen LEFT JOIN-t a TimeSlots és az Appointments táblák között a következő feltételekkel:

- Azonos doctor\_id érték.

- A slot időintervallum (slot\_start, slot\_end) ne fedje le egy meglévő foglalás (a.start\_time és a.end\_time) időtartamát, figyelembe véve, hogy a foglalás státusza ne legyen "cancelled".

b. Csatlakoztasd az orvosok és szolgáltatások adatait:

- Csatlakoztasd a doctors\_x\_services táblát a doctor\_id alapján.

- Csatlakoztasd a services táblát a service\_id alapján.

- Csatlakoztasd a doctors táblát a doctor\_id alapján, hogy megkapd az orvos nevét.

c. Szűrd ki azokat a slotokat, amelyeknél:

- Nem található foglalás (Appointments rekord nincs, azaz a.id NULL).

- A slot kezdő időpontja (slot\_start) későbbi, mint a jelenlegi idő (NOW()).

3. Rendezd az eredményeket növekvő sorrendben a slot\_start alapján.

4. Töröld az ideiglenes "TimeSlots" táblát.

5. Térj vissza a lekérdezés eredményével (lista a szabad időpontokról).

**Megjegyzések**:

* A tárolt eljárás biztosítja a 30 perces időintervallumokra bontott, jövőbeli szabad időpontok lekérdezését.
* Amennyiben a rendszerben szükség van a dátum szerinti szűrésre, azt a backend rétegben (az eljárás eredményének feldolgozásakor) kell elvégezni.
* Az eredményt egy JSON objektumban vagy megfelelő adatstruktúrában adhatjuk vissza, amelyet a frontend az időpontfoglalási folyamat során felhasznál.

## 6. Tesztelési dokumentáció

A projekt során manuális tesztelést alkalmaztunk. A tesztelési folyamat során készítettünk egy tesztkatalógust Excel-ben, amely kb. 50 tesztesetet tartalmaz. Ezek a tesztesetek a felhasználói funkciók helyes működésére összpontosítottak. A katalógust mellékeltük „**Tesztesetek**” néven.

## 7. Továbbfejlesztési lehetőségek

A GB Medical projekt továbbfejlesztéséhez több lehetőség is adódik, amelyek vagy időhiány miatt nem valósultak meg, vagy a későbbi fejlesztési fázisokra lettek halasztva. Az alábbiakban összegyűjtöttünk néhány olyan fejlesztési ötletet, amelyeket érdemes lehet a jövőben megvalósítani.

### 7.1. Nem megvalósított, de tervezett funkciók

* + - 1. **Fejlettebb admin felület**

Egy részletesebb adminisztrációs felületet, ahol az adminok **statisztikákat láthatnak** a rendszer működéséről, például:

* Összes foglalás száma egy adott időszakban
* Legnépszerűbb orvosok és rendelési idők
* Lemondott időpontok aránya

Ezen felül az adminok közvetlenül kezelhették volna az időpontokat, értékeléseket, felhasználókat és orvosokat egy átláthatóbb, dinamikus kezelőfelületen.

* + - 1. **Orvosok saját bejelentkező felülete**

Egy külön **orvosi dashboard** kialakítása is tervben volt, ahol az orvosok a saját fiókjukba belépve:

* Megnézhették volna a hozzájuk foglalt időpontokat
* Kezelhették volna az elérhetőségüket
* Szükség esetén törölhettek vagy módosíthattak volna foglalásokat
* Módosíthatták volna a munkaidejüket

Mindezek a funkciók segítették volna az orvosokat abban, hogy hatékonyabban kezeljék a saját rendelési idejüket.

* + - 1. **Automatikus értesítések SMS-ben**

Jelenleg az értesítések e-mailben érkeznek. Egy **SMS gateway** (pl. Twilio) használatával a páciensek telefonos értesítést is kaphatnának a közelgő időpontjukról vagy egy lemondásról.

### 7.2. Jövőbeli fejlesztési lehetőségek

**Mobilalkalmazás fejlesztése**

A rendszer jelenleg egy **reszponzív webalkalmazás**, de egy **Android és iOS** mobilalkalmazás növelhetné a felhasználói élményt. A mobilapp push értesítéseket küldhetne a pácienseknek és orvosoknak az időpontjaikról.

**Többnyelvű támogatás**

A rendszer jelenleg magyar nyelvű, de egy többnyelvű funkció (pl. **angol, német**) lehetővé tenné a külföldi páciensek számára is az időpontfoglalást.

**Online fizetési integráció**

Jelenleg az időpontfoglalás után a fizetés manuálisan történik. Egy online fizetési rendszer (pl. Stripe vagy PayPal) integrálásával a páciensek azonnal kifizethetnék a konzultációs díjat, csökkentve ezzel a lemondások számát.

**AI-alapú időpontajánlás**

Egy mesterséges intelligencia alapú algoritmus segíthetne a pácienseknek a számukra legmegfelelőbb időpont kiválasztásában az orvos elérhetősége és a korábbi foglalási szokások alapján.

**Elektronikus recept és dokumentumkezelés**

Az orvosok a rendszerben kiállíthatnák az elektronikus recepteket és diagnózisokat, amelyeket a páciensek letölthetnek, így csökkentve a papíralapú adminisztrációt.

**Felhasználói profil**

A felhasználói profil oldalon a páciens egyszerűen megtekintheti és módosíthatja személyes adatait, valamint hozzáférhet a korábbi és jövőbeni időpontjaihoz, diagnosztikai információihoz és orvosi dokumentumaihoz.

# IV. Összegzés

  A GB Medical projekt megvalósítása során számos területen jelentős fejlődést értünk el. Szakmailag mélyítettük tudásunkat az adatbázistervezésben, a backend fejlesztésben Java nyelven WildFly szerver környezetében, valamint az Angular alapú frontend fejlesztésében. Emellett a projektmenedzsment és a csapatmunka eszköztárát is bővítettük: a JIRA rendszer segítségével hatékonyan nyomon tudtuk követni a feladatokat, míg a GitHub használata elősegítette a verziókezelést és az együttműködést.A projekt során számos kihívással kellett szembenéznünk. Az adatbázis szerkezetének és a tárolt eljárások optimalizálásának megtervezése kulcsfontosságú volt annak érdekében, hogy egy hatékony és skálázható rendszert hozzunk létre. A backend és a frontend integrációja, a különböző bejelentkezési szerepek kezelése, valamint az időpontfoglalási funkciók – ideértve a foglalás létrehozását és lemondását – megvalósítása során sok kreatív megoldásra és alapos tervezésre volt szükség.

Jövőbeli célkitűzéseink között szerepel a projekt további fejlesztése és finomítása:

* **Adminisztrációs felület bővítése:** Az adminisztrátorok számára egy átfogó, könnyen kezelhető felület kialakítása, amely segít a rendszer erőforrásainak és a felhasználói adatok kezelésében.
* **Külön bejelentkezési rendszer az orvosoknak:** Az orvosok számára egy dedikált felület és jogosultsági rendszer kialakítása, amely lehetővé teszi a saját időbeosztásuk és szolgáltatásaik kezelését.
* **Online fizetés és SMS-értesítések integrálása:** Az időpontfoglalási folyamat zökkenőmentessége érdekében bevezetjük az online fizetési lehetőségeket, valamint az automatikus SMS értesítéseket, amelyek javítják a kommunikációt a felhasználók és az egészségügyi szolgáltatók között.
* **Mobilalkalmazás és többnyelvű támogatás:** Hosszú távon tervezzük a rendszer mobilalkalmazás formájában történő elérhetővé tételét, valamint a többnyelvű felület kialakítását, hogy szélesebb körben legyen használható és még jobban igazodjon a felhasználói igényekhez.

Ezzel a megközelítéssel nemcsak egy stabil, jól működő alapot építettünk ki, hanem egy olyan rugalmas rendszert hoztunk létre, amely a jövőben könnyen bővíthető, fejleszthető és adaptálható az egészségügyi szolgáltatások egyre változó igényeihez.

# Irodalomjegyzék, forrásmegjelölés

* MyClinic weboldala (inspirációként): <https://www.myclinic.com>
* MySQL dokumentáció és Oracle Java Tutorials (adatbázis-kezelés és SQL lekérdezések): <https://dev.mysql.com/doc/> és <https://docs.oracle.com/javaee/>
* Angular hivatalos dokumentáció és Angular Material útmutatók (moduláris felépítéshez és UI fejlesztéshez): <https://angular.dev/overview> és <https://material.angular.io/>
* Java EE és WildFly dokumentáció, valamint Oracle Java SE és EE kézikönyvek (backend fejlesztés és szerveroldali megoldások): <https://docs.wildfly.org/> és <https://docs.oracle.com/javase/tutorial/>
* GitHub verziókezelési útmutató:
* <https://docs.github.com/en> és releváns cikkek a témában
* W3Schools és Stack Overflow (általános webfejlesztési technikák, hibakeresési tanácsok): <https://www.w3schools.com> és <https://stackoverflow.com>
* FullCalendar dokumentáció (naptárkezeléshez): <https://fullcalendar.io/>
* SweetAlert dokumentáció (figyelmeztető üzenetek testre szabásához): <https://sweetalert2.github.io/>
* ChatGPT (általános fejlesztési kérdésekhez és hibakeresési javaslatokhoz): <https://chat.openai.com>