

# MyHealthCare

**Szerzők:**

Bedő László

Donáth Dániel-Zsolt

Barabási Róbert Zsolt

**Vezető tanár:**

Dr. Szántó Zoltán

Szoftver rendszerek tervezése 2021

Sapientia Erdélyi Magyar Tudományegyetem

# 1.Bevezetés

Az emberi szervezet egy igen komplex, összhangban mozgó,létező rendszer amelynek minden más rendszerhez hasonlóan energiára van szüksége a megfelelő működés érdekében. Akárcsak minden más működő rendszer, az emberi szervezet is meghibásodhat, ezt a folyamatot nevezzük megbetegedésnek. A történelem során, a technológia fejlődésével egyre modernebb, hatásosabb gyógymódok születtek. Jelenleg a 21. században, a legelterjedtebb és leghatásosabb módszer a különböző betegségek kezelésére, az úgynevezett kórházi, egészségügyi ellátás. Bizonyos esetekben, nem megfelelő logisztikával rendelkező településeken egy kórházi időpontfoglalás is nehézkes, bonyolult folyamat lehet. Ennek ellenére a modern társadalom nagyon nagy része rendelkezik okostelefonokkal, amely egy lehetséges megoldást kínálhat az előzőleg megemlített problémára.

A mi alkalmazásunk a csúcstechnológia segítségével elhozhatja a megoldást a kórházi időpontfoglalás problémájára. Az alkalmazás használatával drasztikusan csökkenteni lehet a foglalással eltöltött időt, valamint a kívánt foglalást bármikor, bárholnan megtehetjük, nélkülözve a kórház meglátogatását, valamint a rendelőben töltött időt. Az alkalmazás tervezése során minél inkább arra törekedtünk, hogy az időpontfoglalás gyors és gördülékeny keretek között történjen felhasználói korosztálytól függetlenül.

Az alkalmazásunkat egy már meglévő alkalmazás mintájára hoztuk létre, azzal a feltétellel, hogy minimalizáljuk az időpontfoglaláshoz szükséges funkciókat.

## 1.1. Az alkalmazás rövid ismertetése

Az alkalmazás használata során kórházi időpont foglalásokat lehet létrehozni, a felhasználó választhat kórházat, egészségügyi részleget és akár orvost is. Orvos felhasználóként pedig, meg lehet tekinteni azokat a hozzánk rendelt vizsgálatokat.

## 1.2. Az alkalmazás logika menete

Belépés után, amely történhet bejelentkezéssel vagy új felhasználó esetében regisztrációval (regisztráció csak kliensként lehetséges, az orvost az adatbázis adminisztrátor állítja be a rendszerbe) az applikáció fő oldalán találja magát a felhasználó.

Az időpontfoglalás érdekében elsőre kórházat kell választania a felhasználónak. Ezt követően a kiválasztott kórház részlegei közül kell választania, majd ha ez is megtörtént, orvost is választ akinél szeretné elvégezni a kívánt vizsgálatot.

Hibás választás esetén a felhasználó visszaléphet és kijavíthatja döntését, valamint a létrehozott foglalást megtekintheti a 'My Appointments' menüpont alatt. Az elvégzett vizsgálatok a 'Feedbacks' menüpontban elérhetőek.

A felhasználó megtekintheti saját profilját, valamint ugyan itt ki is jelentkezhet az aktuális munkamenetből.

Orvos felhasználó esetében meg lehet tekinteni a hozzánk rendelt időpont foglalásokat.

## 2. Célkitűzések

Az applikáció tervezése közben számos célt tűztünk ki magunk elé amelyeket úgy gondolunk, hogy fontos az szoftver működése szempontjából illetve a jó felhasználói élmény eléréséhez szükségesek.

Ezen célok közé tartozik a felhasználói profil, amely lehetőséget ad arra, hogy a felhasználók nyomon kövessék, rendszerezze, személyre szabják előző adataikat.

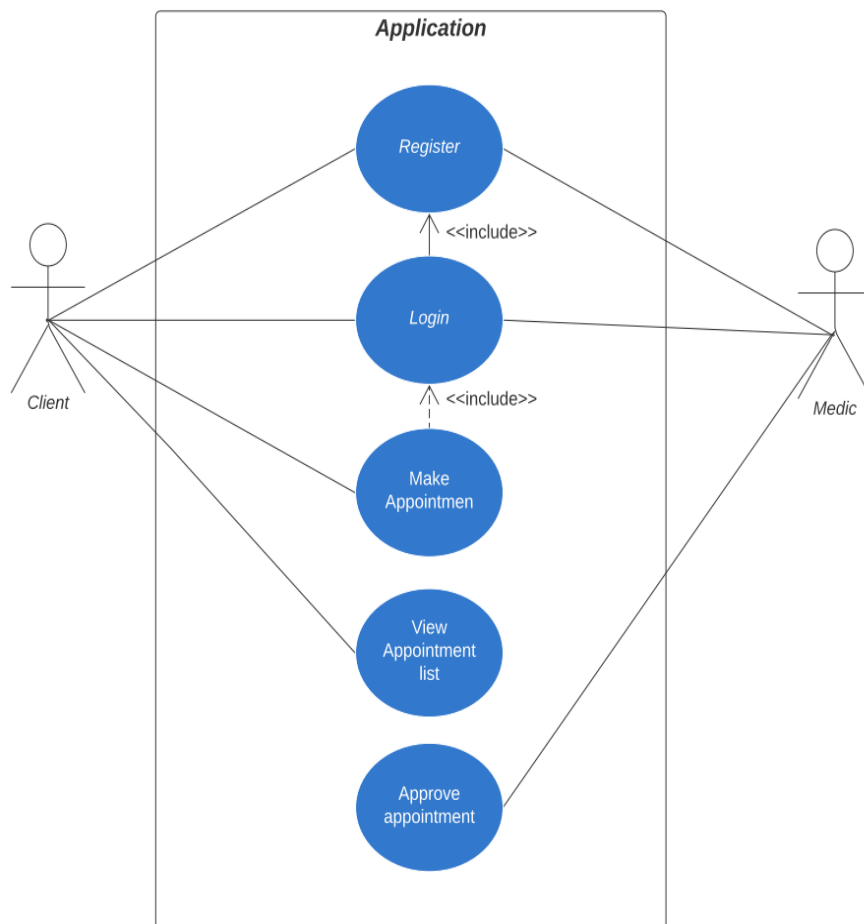
Fontosnak tartottuk azt is, hogy az alkalmazást mint kliens és mint orvos is egyaránt tudja majd használni.

Céljaink között szerepelt az egyszerű, gyors de aprólékos időpontfoglalás, ennek érdekében a felhasználó választhat kórházat, kórházi részleget valamint orvost, azon belül pedig szabad időpontot, mint ezt csupán néhány kattintással megteheti.

További céljaink között szerepelt a már meglévő időpontok megtekintése valamint az elvégzett konzultációk számontartása.

## 3.Követelmény specifikáció

### 3.1. Felhasználói követelmények



1. Ábra: Use-case diagram

Az applikációnak két fajta szerepköre van, két fajta felhasználó lehet. Egy kliens felhasználó és egy orvos felhasználó. (1. Ábra: Use-case diagram)

A kliens felhasználó funkcionalitásai:

- Bejelentkezés: a felhasználó bejelentkezhet a már meglévő fiókjaiba, ha helyesek a bejelentkezéskor megadott adatok.
- Regisztráció: a felhasználó létrehozhat új fiókokat, ha a regisztrációnál kért és megadott adatok helyesek.
- Időpontfoglalás: a felhasználó létrehozhat új orvosi időpontokat, megtekintheti a már meglévőket, valamint megnézheti a már elvégzett konzultációk visszajelzéseit.

Orvos felhasználó funkcionalitásai:

- Bejelentkezés: ugyanaz mint a kliens felhasználó esetében.
- Konzultációk megtekintése: megtekintheti a hozzá rendelt konzultációkat és azok adatait.

## 3.2. Rendszer követelmények

### 3.2.1. Funkcionális követelmények

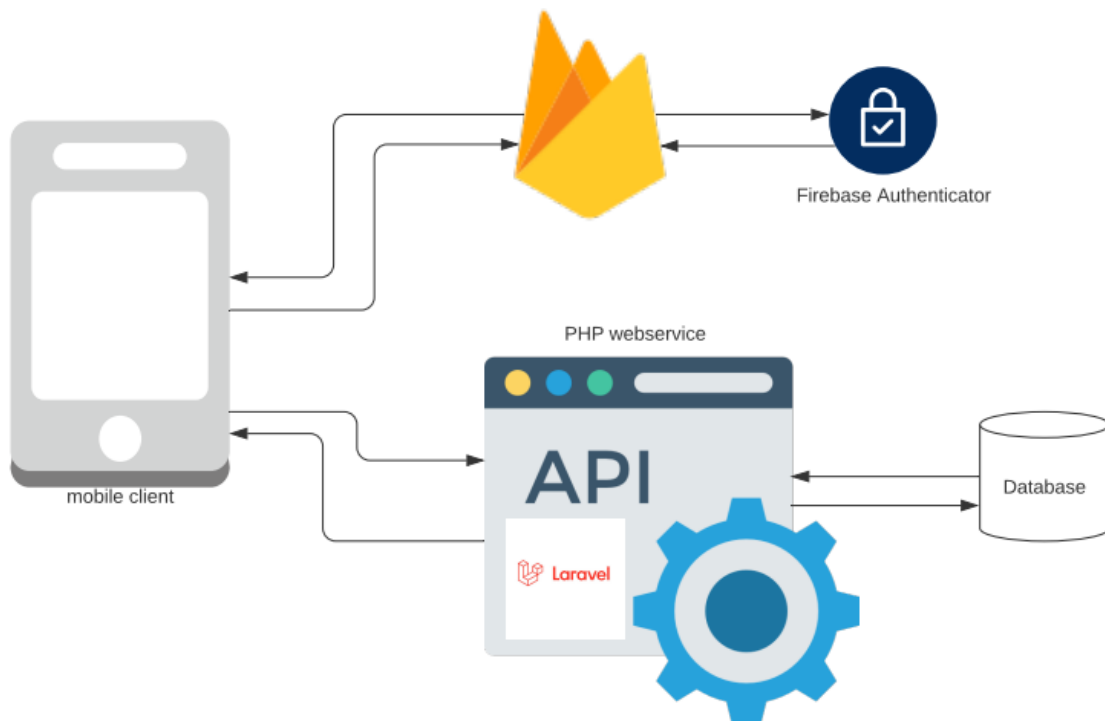
Az alkalmazás indításakor a kezdő oldalon találja magát a felhasználó, ahol tud regisztrálni, be tud jelentkezni. Bejelentkezés után új időpont foglalást hozhat létre, megtekintheti a már meglévőket, illetve a visszajelzéseket a már elvégzettekre.

### 3.2.2. Nem funkcionális követelmények

- Okostelefon, tablet
- Android: 10+
- API level: 29+
- Internet connection: 3G. 4G. 5G
- Free storage: ~15Mb
- Free memory: ~750Mb - 1000Mb

## 4. Tervezés

### 4.1. Alkalmazás felépítése (Architektúrája)



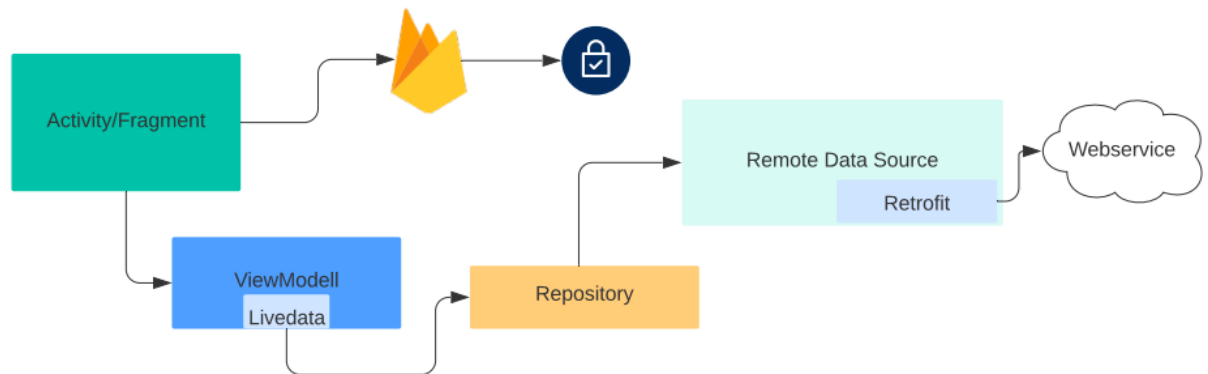
2. Ábra: Architektúra diagram

Az alkalmazás tervezésének első lépéseiben azt próbáltuk meghatározni, hogy a milyen funkciókkal szeretnénk, hogy bírjon az alkalmazásunk, ezek meghatározása során pedig sikerült elkülöníteni, hogy milyen felhasználó típusok lesznek. (1. Ábra: Use-case diagram)

Az applikációt két nagy részre bontottunk, eszerint a részek szerint osztottuk szét egymás közt a munkát is. (2. Ábra: Architektúra diagram) Az egyik ilyen rész a back-

end része az applikációnak, ez magába foglalja a szerver, adatbázis létrehozását, valamint az API elkészítését és deploy-olását. A másik modul pedig az applikációt tartalmazza, az applikáció logikájának megvalósítását.

Szerver oldalon a bejelentkezés és regisztráció esetében a Firebase Authenticator-t használtuk. Az időpontfoglalás esetében egy PHP-Laravel-ben írott API-t használtunk amit Heroku-ra deploy-oltunk. Az API eltárolja az adott felhasználó konzultációit valamint az időpont foglaláshoz szükséges egyéb adatokat (kórházak, kórházi részlegek, orvosok)



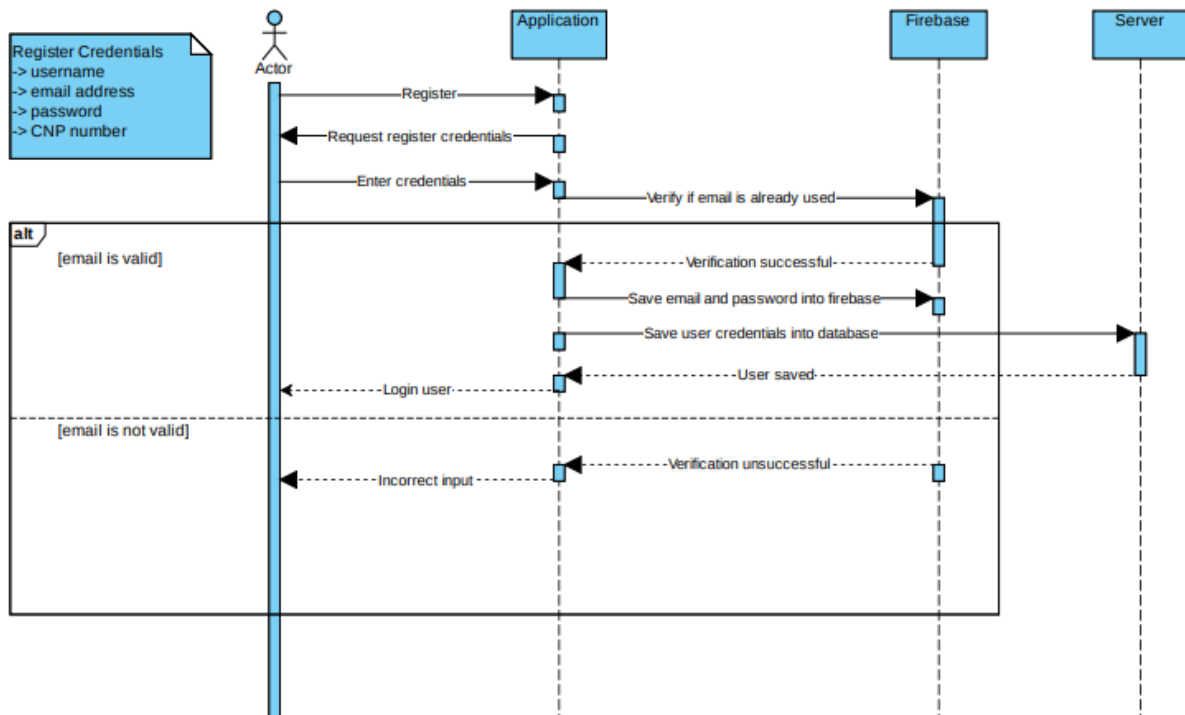
3. Ábra: MVVM diagram

Kliens oldalon egy android applikációt készítettünk, melynek során követtük az MVVM (Model-View-ViewModel) elvet. (3. Ábra: MVVM diagram)

Ezek után minden egyes fő funkciót lebontottunk, illetve megnéztük, hogy mikor és hogyan teremt kapcsolatot a felhasználó az applikációval, illetve az applikáció a szerverrel, meghatároztunk a kommunikációkat közöttük.

## 4.2. Szekvencia diagramok

### 4.2.1. Regisztráció



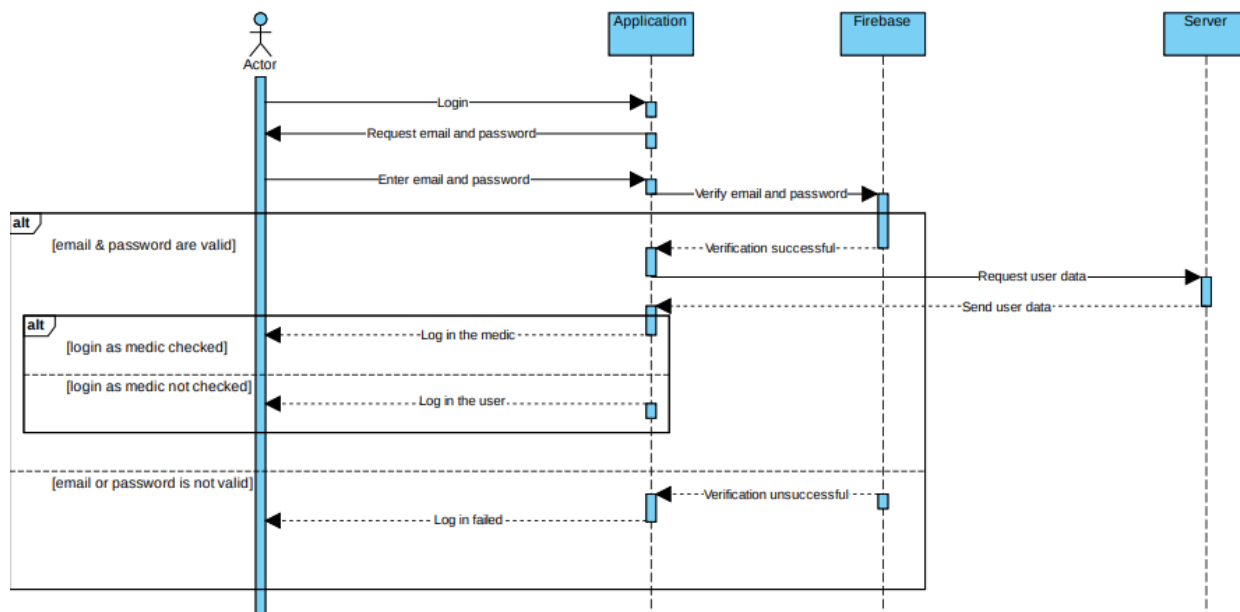
4. Ábra: Regisztráció szekvencia diagram

A regisztráció során a felhasználó, az ábrán "Actor" egy regisztrációs kérést küld az applikáció felé egy gomb lenyomásával, ezután az applikáció bekéri a regisztrációs adatokat, amelyet egy gomb lenyomásával elküld a felhasználó az alkalmazásnak, az applikáció csak akkor engedi meg az elküldést, hogyha kitöltött a felhasználó minden mezőt illetve rendelkezik a megfelelő karakterszámmal és karakter típusokkal. Ezután az applikáció le ellenőrzi, hogy az adatbázisban szerepel-e ilyen e-mail című felhasználó (Firebase Authenticator), ha nem szerepel ilyen felhasználó akkor visszaküldi, hogy rendben van, a felhasználó betevődik az adatbázisba (Firebase + API), majd az applikáció automatikusan bejelentkeztetni a felhasználót a megadott regisztrációs adatok szerint. Ha van már ilyen felhasználó az adatbázisban akkor az



applikáció visszakapja hogy, sikertelen a regisztráció, majd lehetőséget kínál az adatok kijavítására.

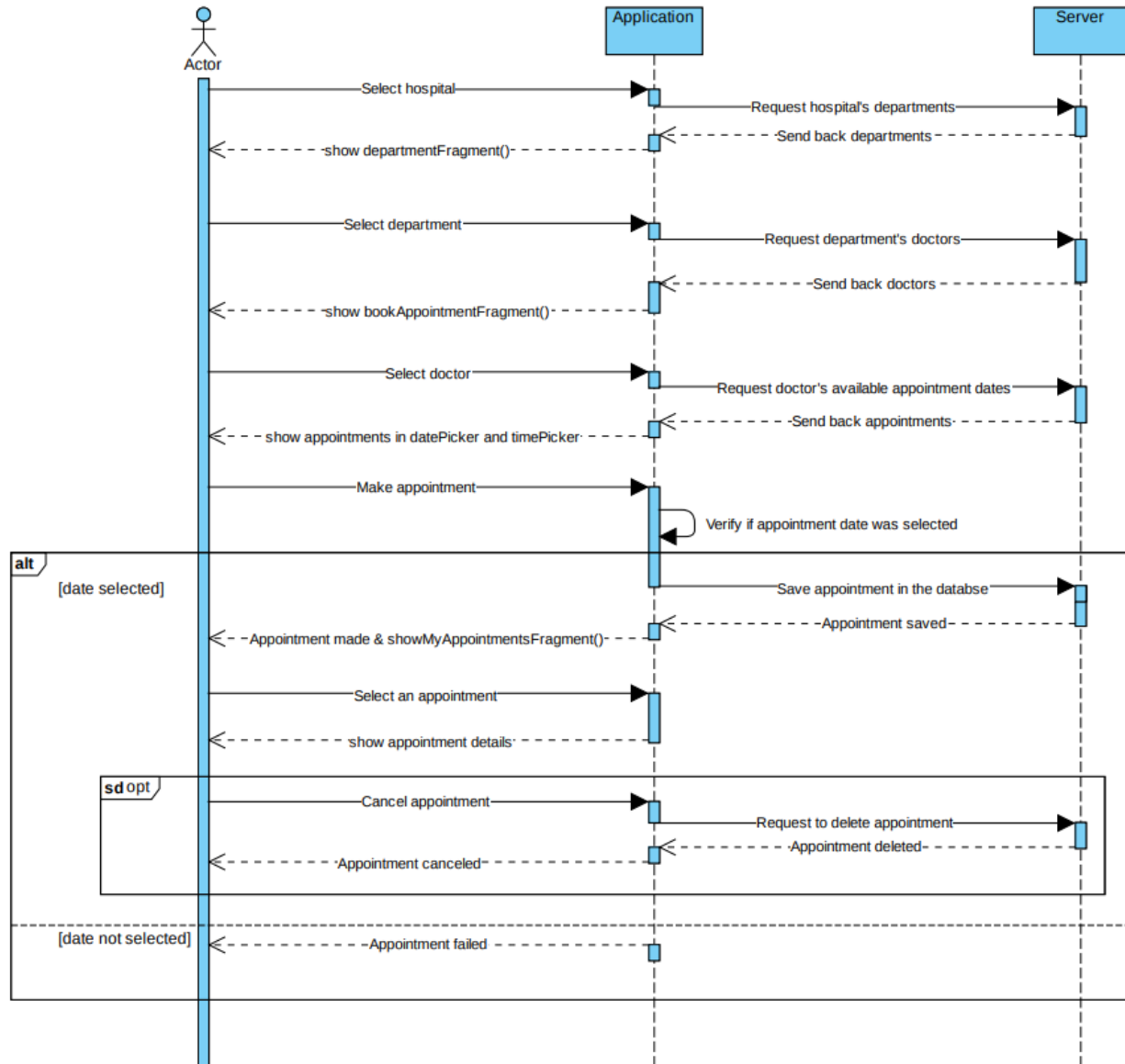
#### 4.2.2. Bejelentkezés



5. Ábra: Bejelentkezés szekvencia diagram

Bejelentkezés során a felhasználó egy bejelentkezési kérést küld az applikáció fele, a bejelentkezés gomb lenyomásával amely kéri a felhasználót, hogy adja meg az e-mail címét valamint jelszavát. Lehetőség szerint ki lehet választani, hogy az aktuális felhasználó kliensként vagy orvosként lépjen be, a 'login as medic' opció elfogadásával. Az applikáció ez után továbbítja a kérést a Firebase Authenticator fele, amely ellenőrizni a megadott adatok helyességét. Ha minden helyes akkor az applikáció a szervertől lekéri az aktuális felhasználó adatait, valamint bejelentkezteti. Ellenkező esetben jelezve lesz a felhasználónak a sikertelen bejelentkezés.

### 4.2.3. Időpontfoglalás

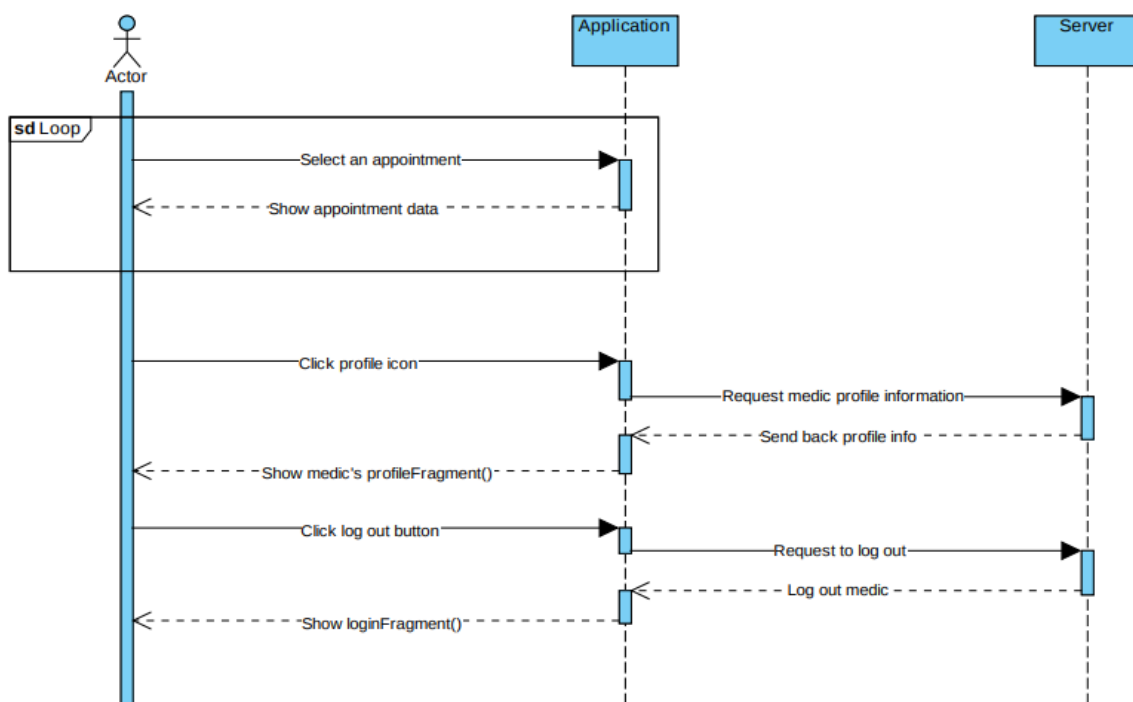


6. Ábra: Időpontfoglalás szekvencia diagram

Időpontfoglalás előfeltétele az, hogy a felhasználó kliens kell legyen és be kell legyen jelentkezve a munkamenetbe. A felhasználó kiválasztja első lépésben a kórházat, a megfelelő elemre kattintva, majd a kérést továbbítja az applikáció a szerver felé, amely visszaadja az adott kórház kórházi részlegeit. Ezt az elvet követően a kórházi részleg kiválasztása után a felhasználó megkapja a részleg orvosait valamint eljut az időpontfoglalás oldalára.

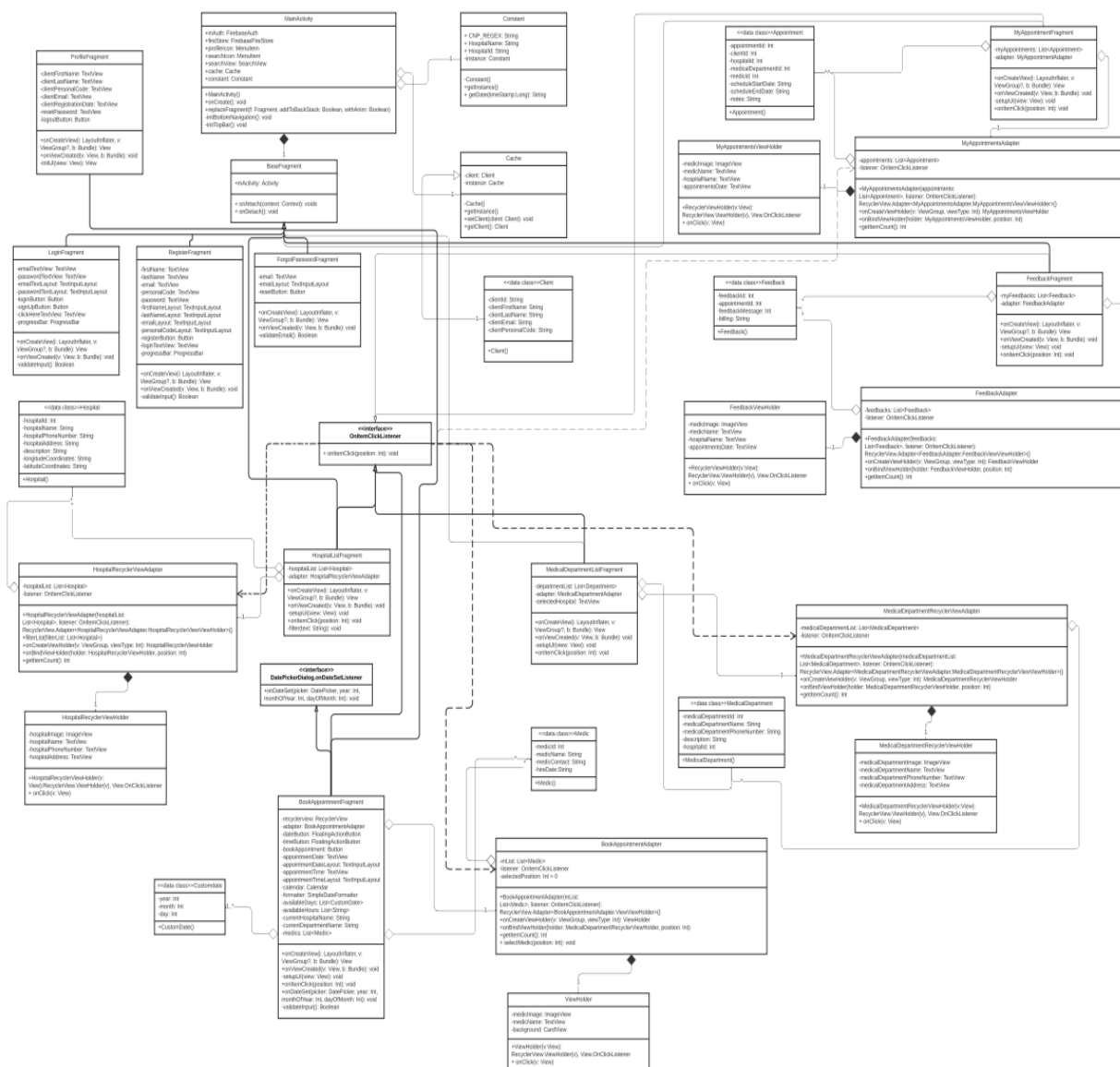
Itt kiválasztva a választott orvost, a szerver visszaküldi az orvos szabad időpontjait, amelyet az applikáció megjelenít. Ha a felhasználó választott időpontot (napot és órát) akkor egy gomb lenyomásával megtörténik az időpontfoglalás, az adatok elmentődnek szerver oldalon. Ha nem választott időpontot, az esetben az applikáció jelzi, hogy a művelet sikertelen, nem lehet időpontot foglalni. Az időpont foglalást követően a kliens megtekintheti a meglévő időpontjait, tetszés szerint elvethet is időpontot (amennyiben rákattint a kívánt időpontra és az elvetés opciót választja), melynek értelmében az adatbázisból is törölve lesz az időpont.

#### 4.2.4. Orvoshoz rendelt időpontok megtekintése



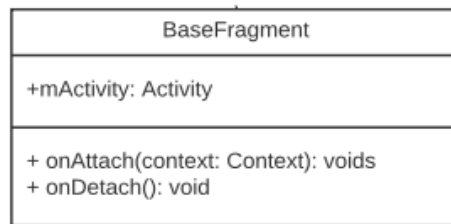
7. Ábra: Orvosi időpontok megtekintése szekvencia diagram

### 4.3. Osztály diagramok



## 8. Ábra: Kliens felhasználó osztálydiagram

Az alkalmazás egyik legnagyobb és legösszetettebb osztály diagramja a kliens felhasználó osztálydiagram. A diagram szerepe, hogy ábrázolja az android fő komponensek közötti kapcsolatot. Dióhéjban az applikáció magja a *MainActivity* osztály, amely az applikáció állapotától függően mindig megelenít egy adott *Fragment* osztályt, ennek értelmében a *MainActivity* és *Fragment*ek között szoros tartalmazási kapcsolat létezik, mivel az android architektúra értelmében *Fragment* nem létezik *Activity* nélkül.

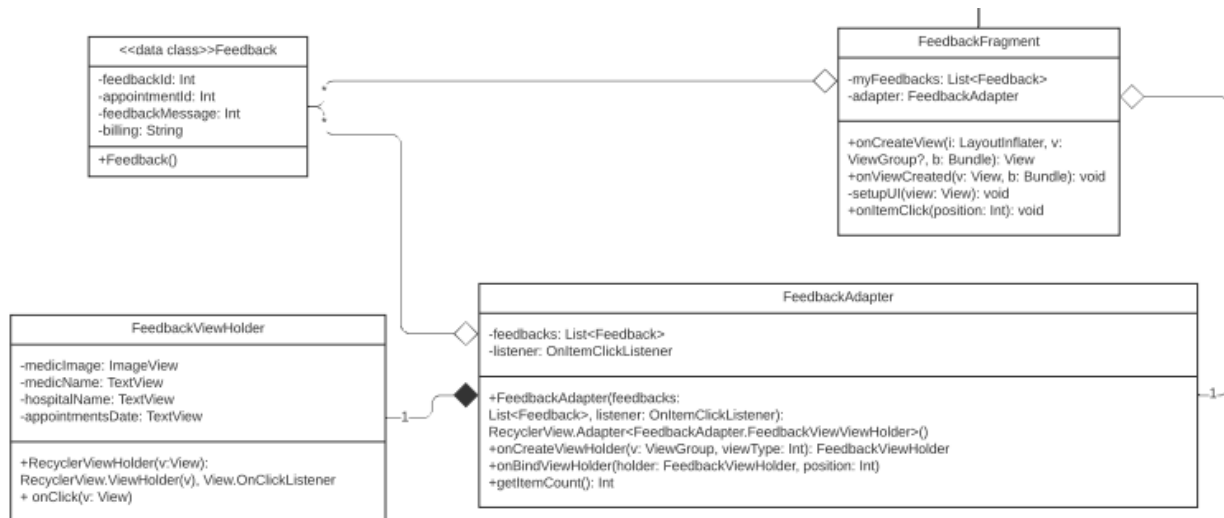


## 9. Ábra: BaseFragment osztály

A *Fragment*ek ősosztálya egy *BaseFragment* osztály (9. Ábra: *BaseFragment* osztály), ebből származik az összes többi *Fragment*. A *BaseFragment* szerepe az, hogy referenciát tart az alkalmazás kontextusára, amely szükséges a különböző android feladatok végrehajtásához (*MainActivity* attribútumainak kezelése).

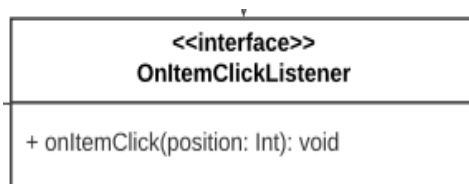
A *Fragment*ek 2 nagy kategóriába sorolhatóak:

- Belépést, regisztrációt, felhasználó profilt kezelő *fragment*ek
- Időpontfoglalást (foglalás, meglévők megnézése, visszajelzések az elvégzett konzultációkról) kezelő *fragment*ek



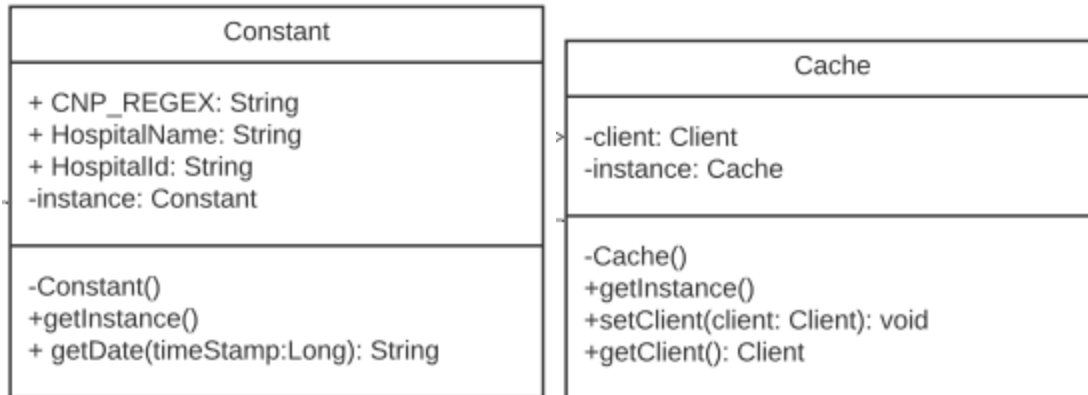
10. Ábra: FeedBack Fragment osztálydiagram

Bizonyos Fragmentek, amelyek nagy számú adatot kell megjelenítsenek (10. Ábra: Feedback Fragment osztálydiagram), RecyclerView-t (görgethető listát használnak). Ezen listák egy a sokhoz kapcsolatot tartanak a megjelenítendő elemekkel, valamint a görgethető lista osztályok rendelkeznek egy belső osztállyal amely egy adott lista elem vizuális megjelenítéséért felel. Minden sok elemet kezelő Fragment, referenciát tartalmaz egy görgethető lista osztályra valamint egy a sokhoz kapcsolatot alkot a megjelenítendő adat listájával.



11. Ábra: OnItemClickListener Interface

Minden olyan Fragment esetén, ahol nagy elemszámú adatot kell megjeleníteni (10. Ábra: FeedBack Fragment osztálydiagram), egy adott elemmel történő interakciót nem a görgethető lista osztály old meg, hanem az őt tartalmazó fragment, amely implementálja a fentebbi ábra interface-ét.

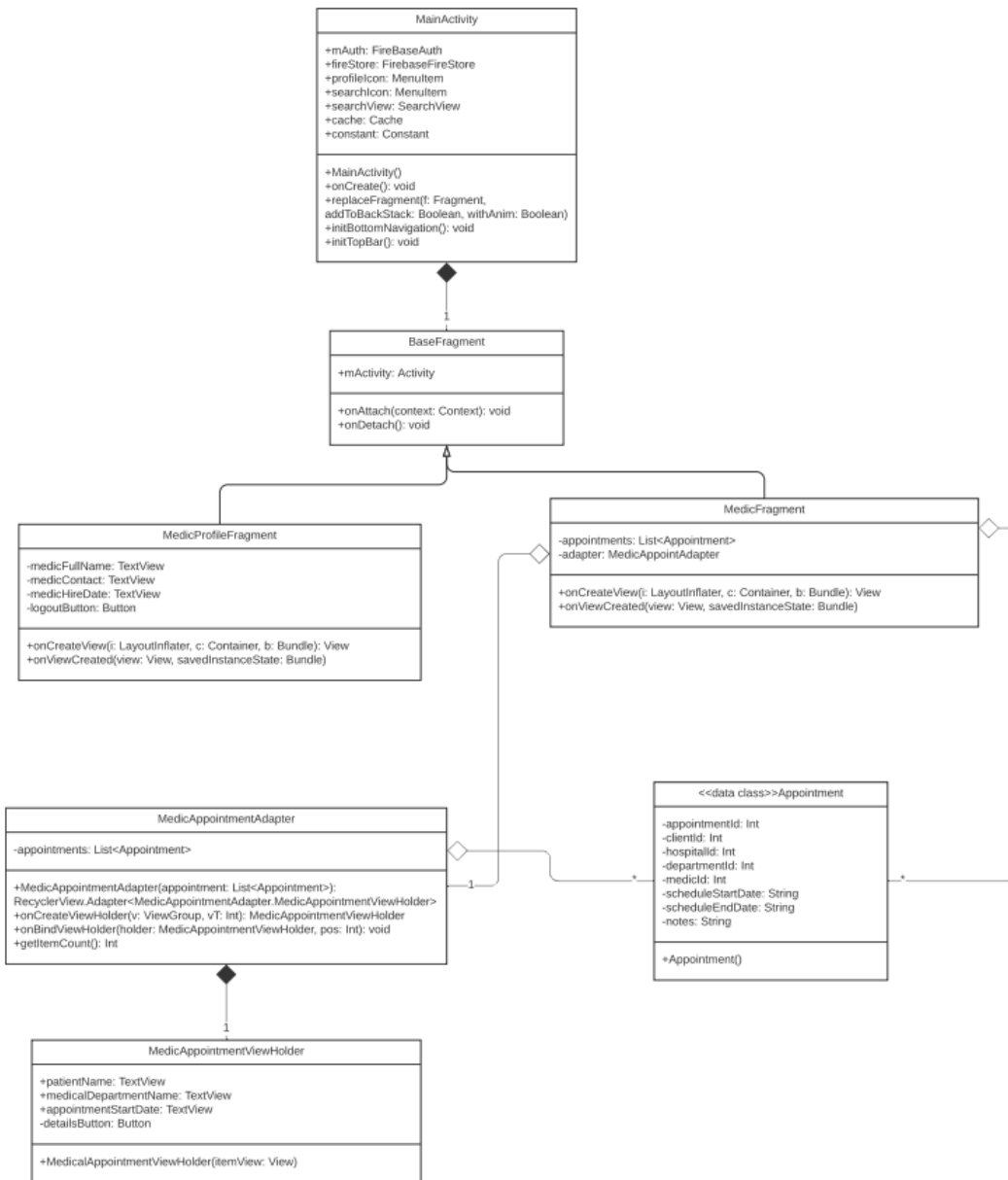


12. Ábra: Constant és Cache osztályok

A Fragmentek mellett, a MainActivity továbbá referenciát tartalmaz két segédly osztályra: *Constant* és *Cache* (12. Ábra: Constant és Cache osztályok)

A Constant osztály feladata, az applikáció során használt konstanst változók összegyűjtése és managelése.

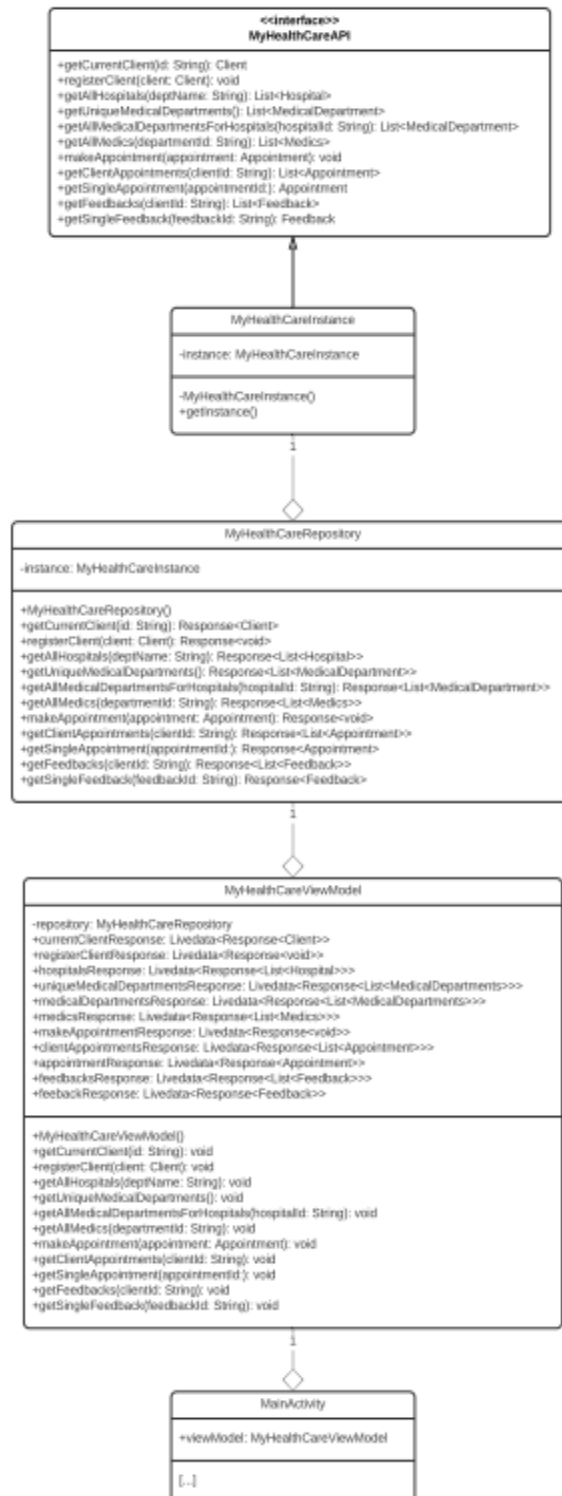
A Cache osztály lehetővé teszi a webservice által visszaadott adatok ideiglenes elmentését a memóriába. Ez akkor lehet hasznos ha egy adatra többször is szükségünk van az applikáció során, így nem kell mindig HTTP kérést végeznünk



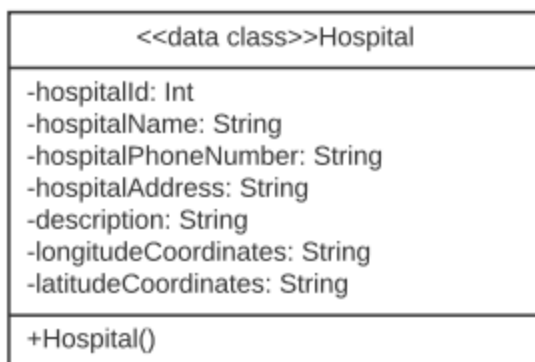
13. Ábra: Orvos felhasználó osztály diagram

Az orvos felhasználó osztály diagramja teljes mértékben hasonló a klienséhez, azzal a különbséggel, hogy kevesebb Fragment van jelen a relációban.





14. Ábra: Web service osztály diagram



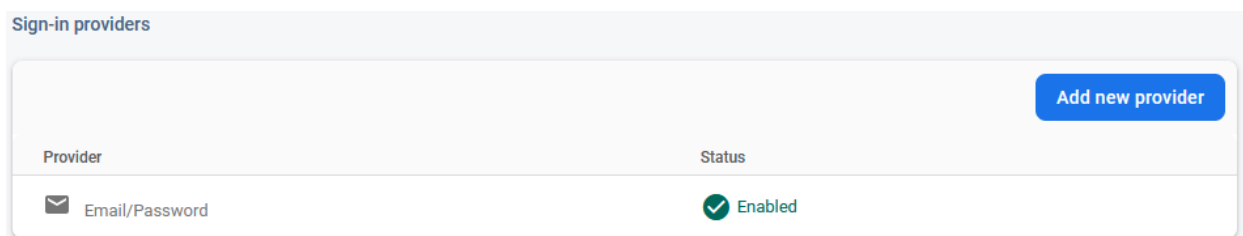
15. Ábra: Példa Model osztályra (Hospital)

A HTTP kérések megvalósításának diagramja, amely 3 fő osztályra tagolható. (14. Ábra: Web service osztály diagram) Ezen osztályok segítségével az adat egy irányban mozog, a szervertől az applikációig és ez idő alatt az adat immutable.

- WebService Interface:
  - Meghatározza a csatlakozási pontokat a távoli szerverrel.
- DAO (Data Access Object) osztály:
  - Implementálja a MyHealthCare interface-t.
  - Singleton, Thread safe osztály.
  - Átalakítja az API JSON formátumú válaszát a megfelelő kotlin model osztályokra. (15. Ábra: Példa Model osztályra (Hospital))
- Repository osztály:
  - Referenciát tartalmaz a dao osztályra.
  - Köztes osztály a dao és a viewmodel között, ez által a viewmodel független tud lenni a dao felépítésétől.
- ViewModel osztály:
  - Observer design pattern használata.
  - Itt történik a HTTP kérés egy background thread-en (esetünkben coroutine-t használva).
  - A választ egy *observable* változóba zárja, ennek köszönhetően aki feliratkozik erre a változóra, mindig értesül ha frissül az adat.

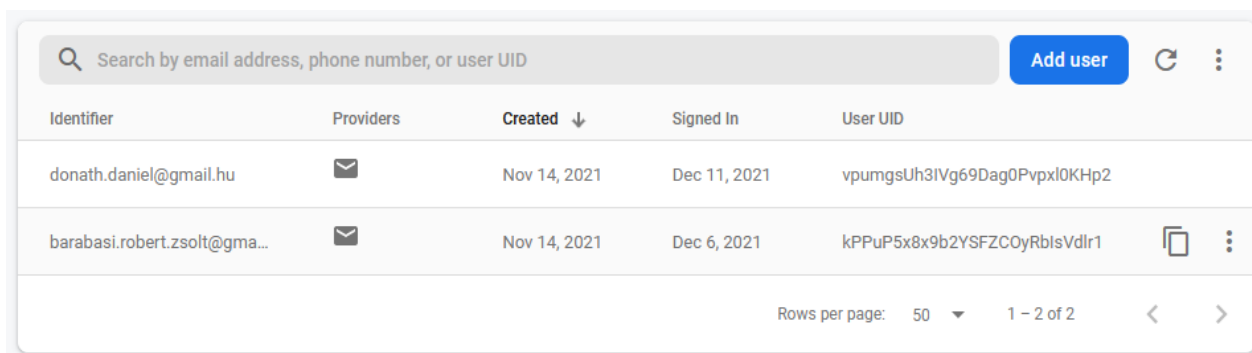
Az adatok tárolása szempontjából, egy általunk készített web service-t használtunk, amely PHP-ban, a Laravel keretrendszer segítségével készült.  
[TODO: class diagram, endpointok leirasa]

## 4.4. Bejelentkezés, regisztráció hitelesítése



17. Ábra: Firebase bejelentkezési metódus

A bejelentkezési, regisztrációs folyamatok menedzselését a Firebase Authentication-ra bíztuk. Bejelentkezési opció során, a Firebase által biztosított *email and password* (17. Ábra: Firebase bejelentkezési metódus) lehetőséget választottuk. A Firebase nem csak a bejelentkezést könnyítette meg, hanem lehetővé tette a munkamenet megőrzését az applikáció bezárása után. Ez azt jelenti, ha nem történik explicit kijelentkezés, az alkalmazás megőrzi a bejelentkezett felhasználót, tehát ha újraindítjuk az applikációt nem jelentkeztet be újra, amely egy nagyon felhasználói élményt biztosít.



18. Ábra: Regisztrált felhasználók adatai

A bejelentkezés után, a Firebase által visszaadott UID-t (18. Ábra: Regisztrált felhasználók adatai) használtuk a web service során is a Client táblánál, így azonosítottuk a felhasználókat és a hozzájuk tartozó információkat.

## 4.5. Endpoints

Az alkalmazás szerver oldali felépítését követően, az API-t *Heroku*-ra publikáltuk, a következő URL-el: <https://myhealthcare-app.herokuapp.com>

Az alkalmazás szerver oldali tervezése során a következő endpoint-okat határoztuk meg:

### **@GET**

- **/api/clients/{id}**
  - -> a megadott id alapján vissza adja a keresett klienst

### **@POST**

- **/api/clients/registerClient**
  - -> új kliens regisztrálása

### **@GET**

- **/api/hospitals/{medical\_dep\_name?}**
  - -> vissza adja a kórházak listáját
  - -> medical\_dep\_name, opcionális paraméter, ha megadjuk a kórházi részleg nevét akkor csak azon kórházakat adja vissza, ahol szerepel a keresett részleg

### **@GET**

- **/api/medicaldepartments/unique**
  - -> vissza adja az egyedi kórházi részlege nevét

### **@GET**

- **/api/medicalDepartments/hospital/{hospital\_id?}**
  - -> vissza adja a kórházi részlegek listáját
  - -> hospital\_id, opcionális paraméter, ha megadjuk a kórház id-t, akkor csak, az adott kórházhoz tartozó részlegeket adja vissza

### **@GET**

- **api/medics/{medical\_dep\_id?}**
  - -> vissza adja az orvosok listáját
  - -> medical\_dep\_id, opcionális paraméter, ha megadjuk akkor csak az adott részleghez tartozó orvosokat adja vissza

### **@POST**

- **api/appointments/makeAppointment**
  - -> új időpont foglalás létrehozása

### **@GET**

- **api/appointments/client/{id}**
  - -> vissza adja a megadott kliens id alapján a kliens időpont foglalásait

### **@GET**

- **api/appointments/{id}**
  - -> vissza adja a megadott id alapján a neki megfelelő időpont foglalást

### **@GET**

- **api/feedbacks/client/{id}**
  - -> vissza adja a megadott kliens id alapján, a neki megfelelő visszajelzéseket a konzultációkról

### **@GET**

- **api/feedbacks/{id}**
  - -> vissza adja a megadott id alapján a neki megfelelő konzultáció visszajelzését

```
{
  "id": 185,
  "name": "Prof. Stanford Barton",
  "contact": "711-023-132",
  "hired_date": "2014-01-15",
  "email": "katelyn.rau@hotmail.com",
  "password": ">@\"M\"5P@e6J8\\R8"
},
```

19. Ábra: Példa az API által visszaadott válaszra

Az API tervezése során, az általa visszaadott választ JSON formátumba adtuk meg (19. Ábra: Példa az API által visszaadott válaszra), ez által biztosítva a könnyű és gyors összekapcsolást a mobil klienssel.

## 4.6. Alkalmazás Managelése

### 4.6.1. Verziókövetés

Az alkalmazás fejlesztése több lépésben, több héten keresztül történt. Ahhoz, hogy maximálisan ki tudjuk használni a technológia nyújtotta lehetőségeket, a verzió követést a *Git* egy nyílt forráskódú, elosztott verziókezelő szoftverre bíztuk. A Git mellett, A Github-ot használtuk, mint adattár hosting szolgáltatás. A két említett szoftver segítségével, közösen tudtunk dolgozni ugyanazon a projekten, akár egy időben is képesek voltunk párhuzamosan fejleszteni az alkalmazásunkat. A közös munka mellett, nyugodtak lehettünk abban is, hogy a projekt egy biztos helyen van és ha egy adott pillanatban véletlenül elrontottunk valamit, könnyedén vissza tudtunk térni egy előző verzióra.

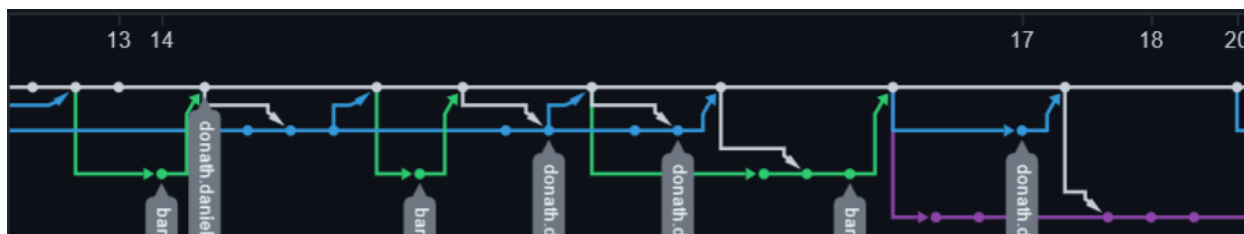
A projekt mappaszerkezet a következőképpen épül fel:

- > docs
  - > diagrams
  - > general\_idea.docx
  - > data\_model.pdf
  - > software\_req\_spec.docs
- > src
  - > api [php-laravel web service]
  - > clients [android applikáció]
- > .gitignore

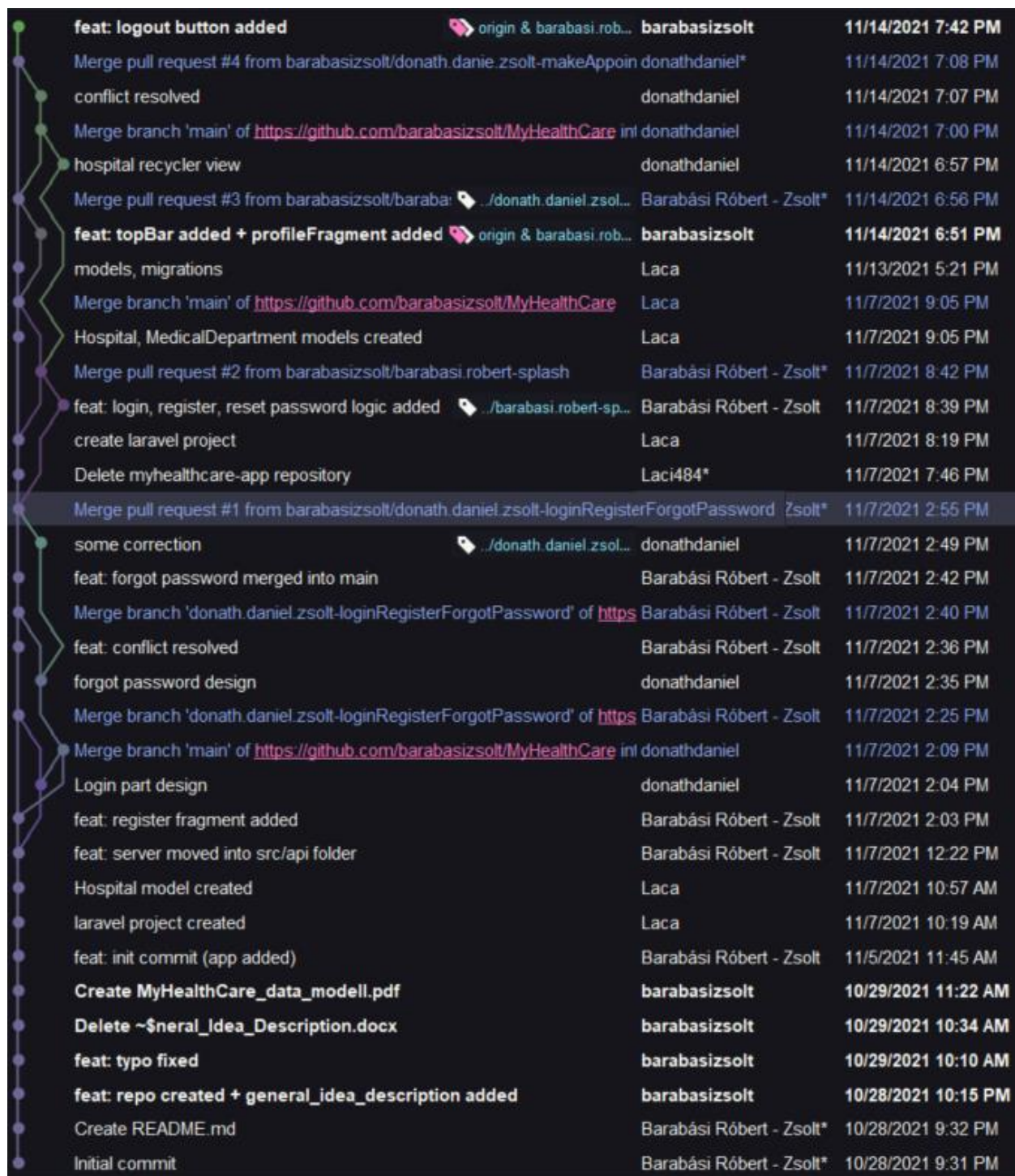
Fontosnak tartottuk, hogy mappa szinten is legyen egy rendszer, így még könnyebben lehetett párhuzamosan dolgozni a projekten.

A projekt fő ága a *main* nevet kapta, minden egyéb új funkcionalitást egy-egy külön ágon implementáltunk le, majd miután az egészen jóvá hagytuk, a feature bekerült a main ágba.

A következő kép, a fejlesztés során keletkezett ágakat ábrázolja egy kisebb intervallumon belül.



20. Ábra: Ágak eloszlása

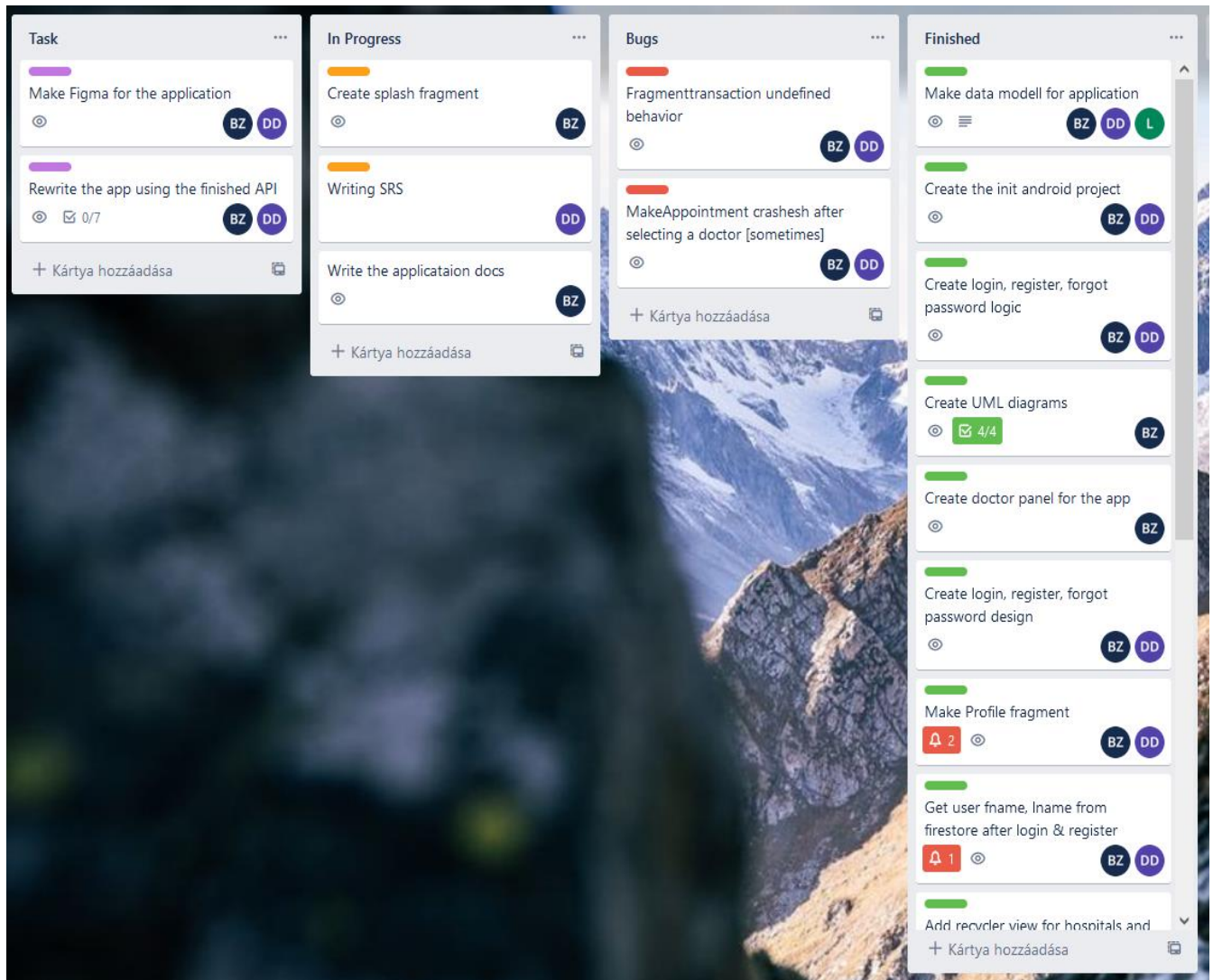


21. Ábra: Verziókövetés ábrázolása, egy kisebb intervallumon



### 4.6.2. Egyéni task-ok managelése

Annak érdekében, hogy nyomon tudjuk követni, ki milyen feladatot végez éppen, erre a célra a *Trello* felületét választottuk. A trello segítségével könnyedén nyomon tudtuk követni, ki milyen feladaton dolgozik, valamint meg tudtuk jelölni a jövőbeli elvégzendő feladatokat és számon tudtuk követni az esetleges hibákat, bug-okat is.








22. Ábra: Trello management klines oldalon

## 5. Alkalmazás bemutatása


## 5.1. Bejelentkezés

8:54



LTE  

# Log in



☐ Log in as medic

LOG IN

Forgot password?

[Click here](#)

---

New to MyHealthCare?

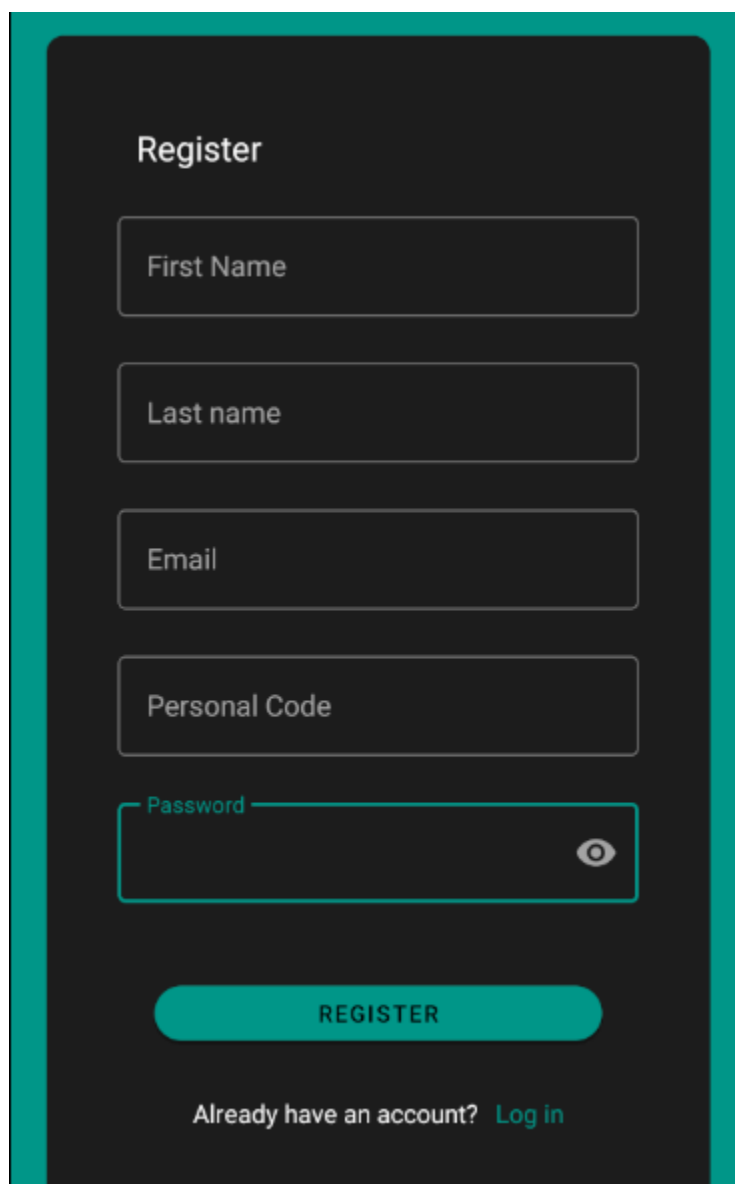
SIGN UP

### 23. Ábra: Bejelentkezés

Bejelentkezés során, a felhasználó eldöntheti, hogy kliens vagy orvosként szeretné használni az alkalmazást. Amennyiben az utóbbit választja, be kell pipálni a

*Log in as medic* opciót. Ha a megadott bejelentkezési adatok helyesek a felhasználó bekerül a munkamenetbe, ellenkező esetben jelzi az applikáció a sikertelen belépést.

## 5.2. Regisztráció



Register

First Name

Last name

Email

Personal Code

Password

REGISTER

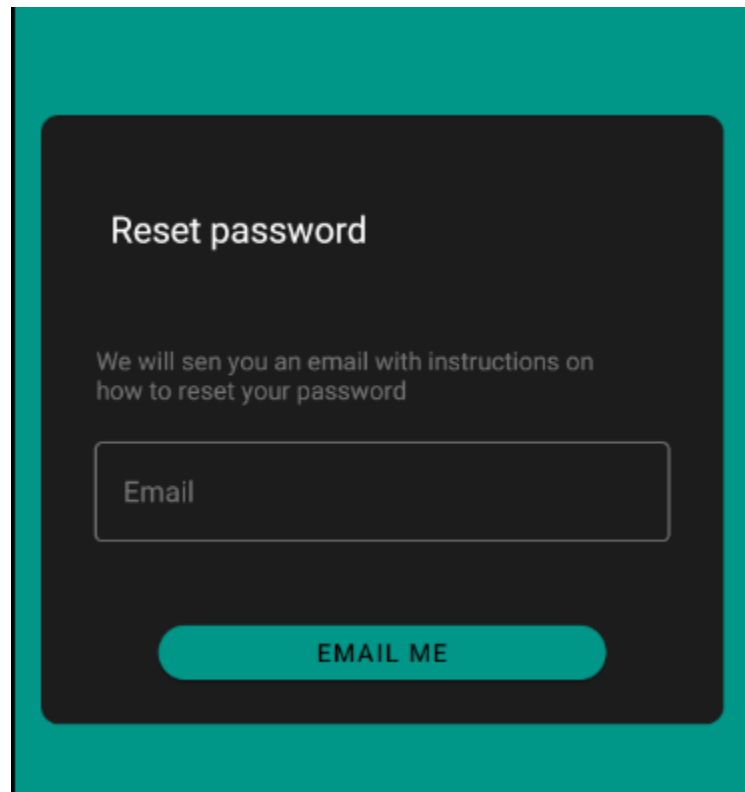
Already have an account? [Log in](#)

24. Ábra: Regisztráció

Regisztrálni csak kliens felhasználó képes, az orvos felhasználókat az adatbázis adminisztrátor állítja be. Hasonlóan a bejelentkezéshez, ha sikeresek a megadott adatok, akkor a felhasználó regisztrálódik, valamint ezt követően automatikusan

belépteti az alkalmazás. Ellenkező esetben az alkalmazás jelzi, a sikertelen regisztrációt, lehetőséget ad az újrapróbálkozásra.

### 5.3. Elfelejtett jelszó

A screenshot of a 'Reset password' form. The form is centered on a teal background. It has a dark gray rounded rectangle containing the title 'Reset password' in white. Below the title is a line of text: 'We will sen you an email with instructions on how to reset your password'. Underneath this is a text input field with the placeholder 'Email'. At the bottom of the form is a teal button with the text 'EMAIL ME' in white.

25. Ábra: Elfelejtett jelszó

Abban az esetben, ha a felhasználó elfelejtette a jelszavát, az email címe megadásával egy új jelszót igényelhet. Ha helyes email címet adott meg és megnyitja postafiókját, akkor a Firebase segítségével új jelszót állíthat be. Fontos kiemelni, hogy ez a funkcionalitás jelenleg csak kliens felhasználóként működik.

## 5.4. Időpontfoglalás

### 5.4.1. Kórház kiválasztása

### Select Hospital



Hospital Name

Policlinica 2\_0

Phone Number

 0744077777

Hospital Address

 Bulevardul 1 Decembrie 1918, Târgu M



 Make appointment

 Feedback

 My appointments

26. Ábra: Főoldal

Sikeres bejelentkezés után kliens szemszögből a következő oldallal találkozunk. Az alsó menüsáv azt jelzi, hogy jelenleg az időpontfoglalás oldalon vagyunk, ugyanott balról jobbra olvasva találjuk a visszajelzéseket és a saját már lefoglalt konzultációkat. A felső menüsáv jelzi, hogy első körben kórházat kell választanunk. Ugyanitt a kereső ikont használva kereshetünk a kórházak között, valamint a profile ikonra kattintva megnézhetjük a profilunkat. Ha rákattintunk egy kórházra, akkor az időpontfoglalás egy további szakaszához érünk (5.4.2 Kórházi részleg kiválasztása)

#### 5.4.2. Kórházi részleg kiválasztása

Select Department

Selected Hospital

☞ Policlinica 2\_0

AMERICAN COLLEGE of CARDIOLOGY

Medical Department Name

Neurology\_0\_0

Phone Number

☎ 0744077777

Medical Department Address

🏠 Floor 3, room 314

☞ Make appointment

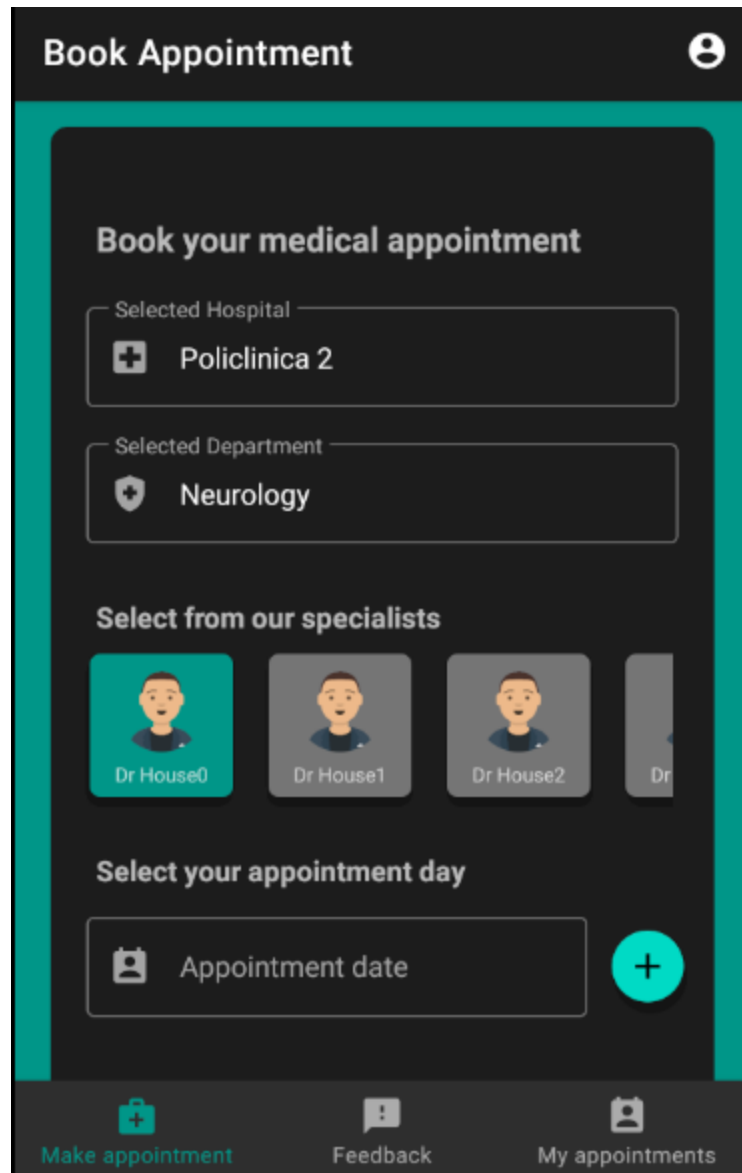
💬 Feedback

👤 My appointments

27. Ábra: Kórházi részleg kiválasztása

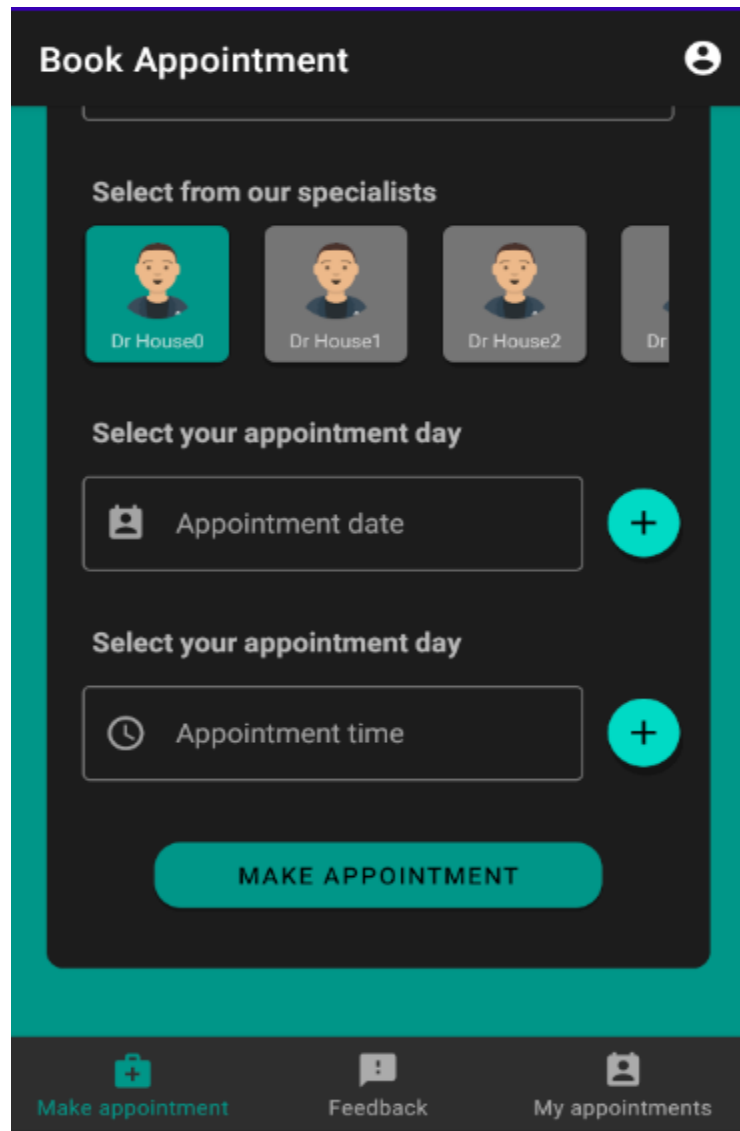
Hasonlóan az előző oldalhoz, a két menüsáv megmarad. Ugyanúgy van lehetőségünk keresni a kórházi részlegek között és innen is elérhetjük a profil oldalunkat. Ha rákattintunk egy kórházi részlegre, akkor az időpontfoglalás utolsó részéhez érünk (5.4.3 Időpontfoglalás elvégzése)

#### 5.4.3. Időpontfoglalás elvégzése



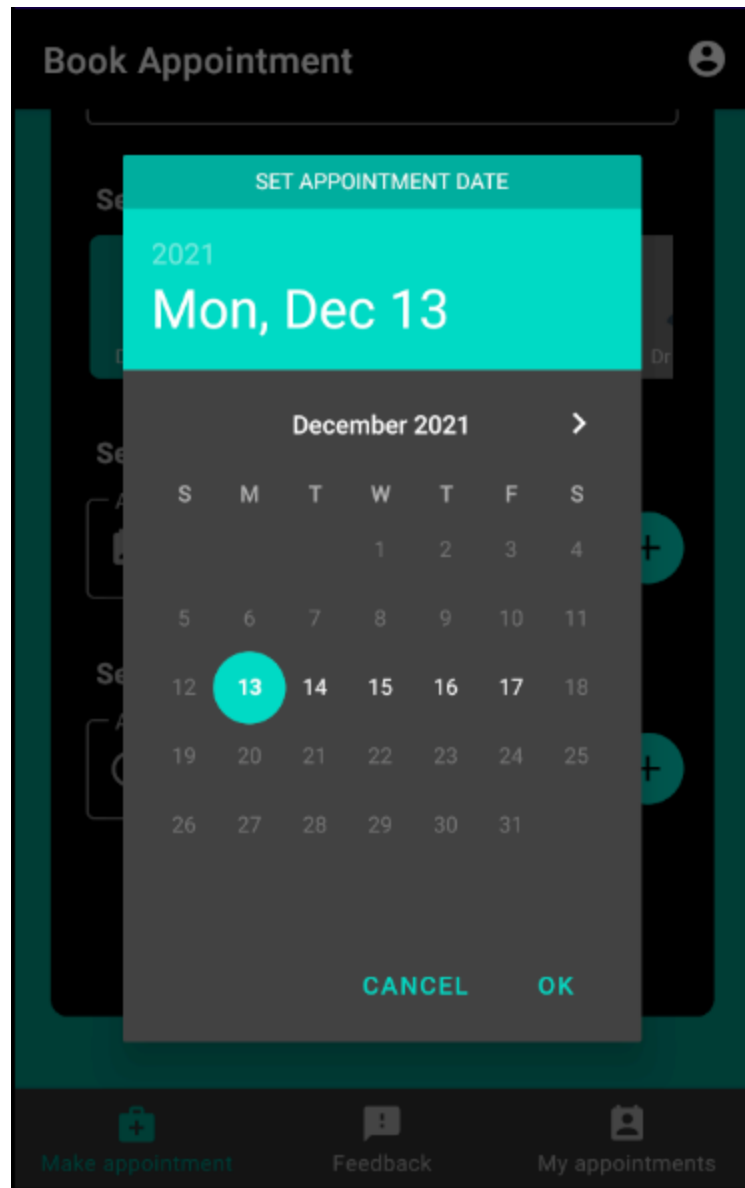
28. Ábra: Időpontfoglalás



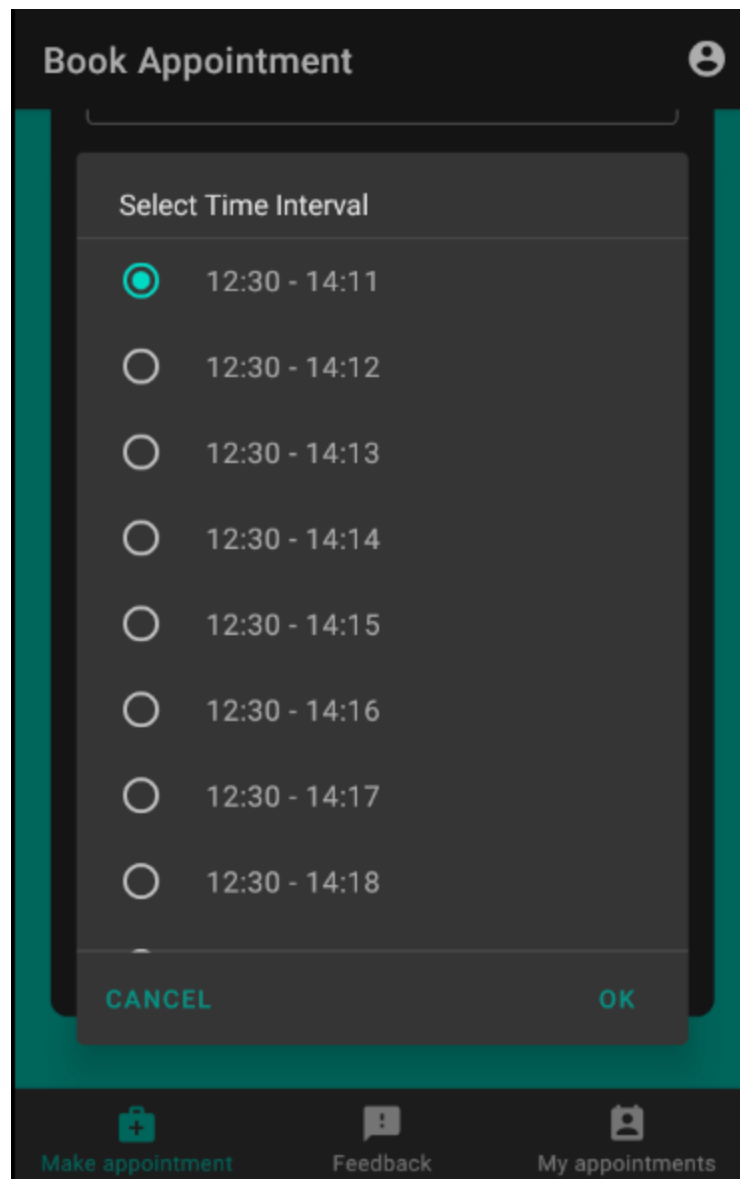


29. Ábra: Időpontfoglalás

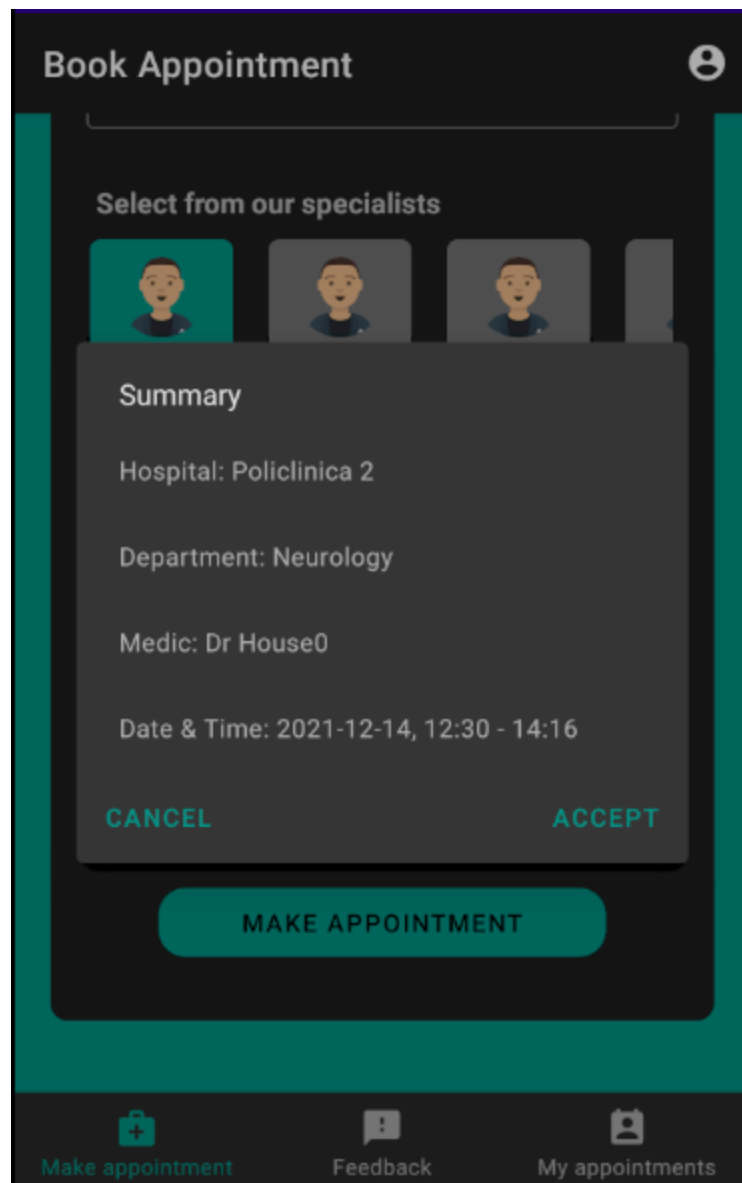
Az első ábrán (28. Ábra: Időpontfoglalás) megfigyelhető, hogy az applikáció jelzi, az eddig kiválasztott kórházat és kórházi részleget. Továbbá lehetőséget ad, hogy orvost is válasszunk (alapértelmezetten az első orvos van kiválasztva). Miután választottunk orvost is, választanunk kell szabad időpontot (nap és óra), (29. Ábra: Időpontfoglalás, 30. Ábra: Nap kiválasztása, 31. Ábra: Óra kiválasztása). Ha mindent kiválasztottunk akkor, a Make Appointment gombra kattintva még egyszer összegezve láthatjuk az időpontfoglalás részleteit (32. Ábra), majd jóváhagyhatjuk. Ellenkező esetben ha valamilyen mező üresen maradt, az applikáció jelzi és ki lehet javítani. Ha sikeres az időpontfoglalás, az applikáció a saját időpontjainkhoz navigál (34. Ábra: Saját időpontok).



30. Ábra: Nap kiválasztása.

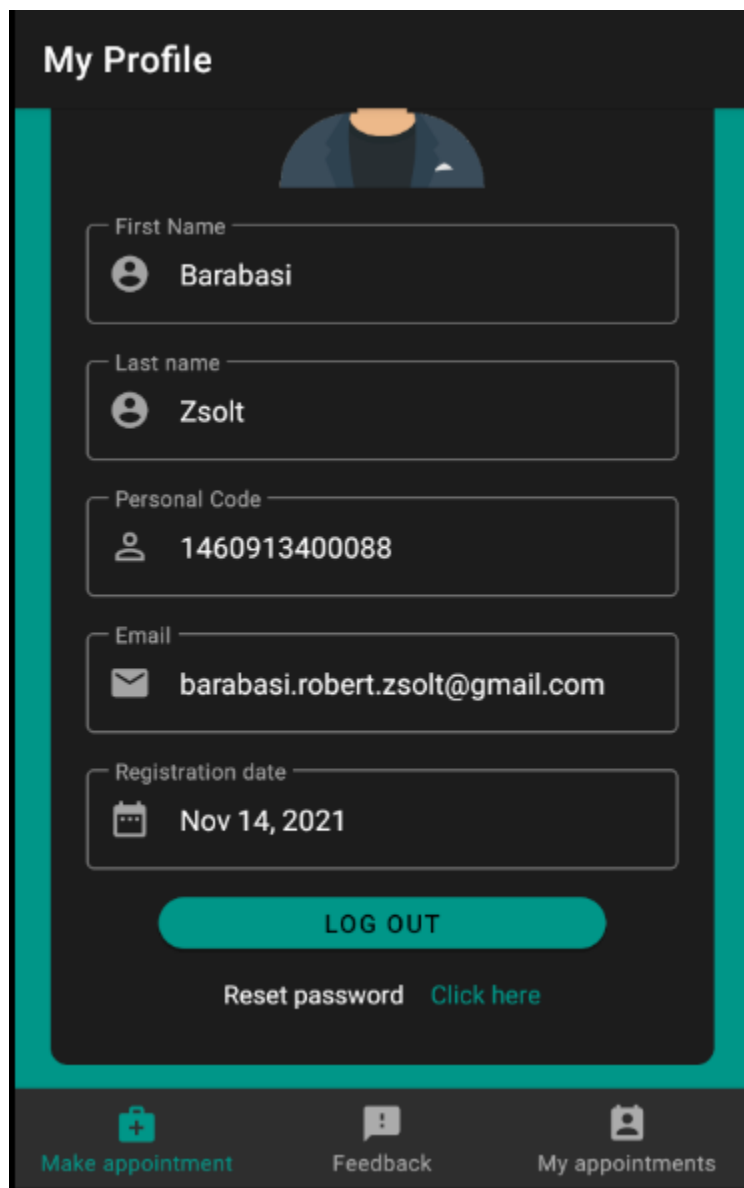


31. Ábra: Óra kiválasztása.



32. Ábra: Időpontfoglalás összegzése, jóváhagyás

## 5.5. Profil megtekintése



33. Ábra: Profil

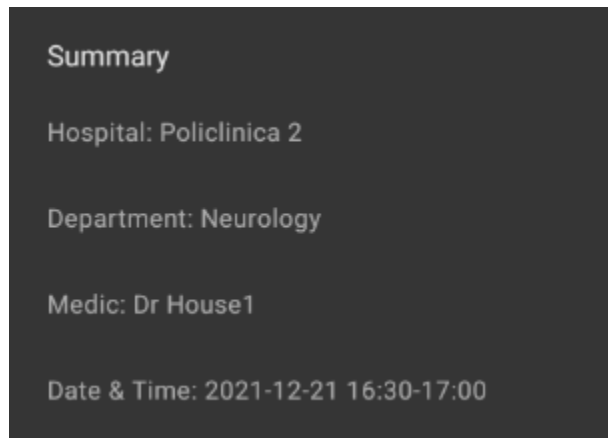
Ahogy az fentebb kiderült a profil ikon segítségével el lehet érni a felhasználói profilt. Itt a felhasználó megtekintheti adatait, jelszót cserélhet (25. Ábra: Elfelejtett jelszó), valamint a kijelentkezés gombra kattintva kiléphet az aktuális munkamenetből.

## 5.6. Saját időpontok



34. Ábra: Saját időpontok

Ezen az oldalon a kliens felhasználó megtekintheti a már lefoglalt időpontjait. Az egyes időpontok egy általános leírást adnak az adott konzultációról, ha rákattint egy időpontra, akkor egy részletesebb leírás fogadja majd a felhasználót (35. Ábra: Időpont részletei)

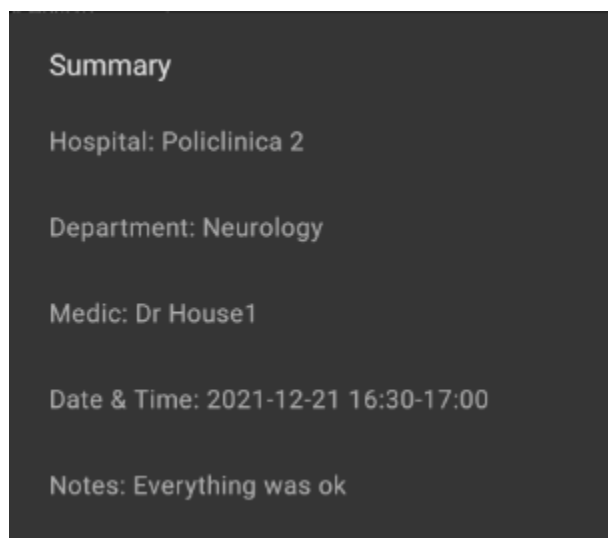


35. Ábra: Időpont részletei.

A felhasználó itt megtekintheti a konzultáció részletei.

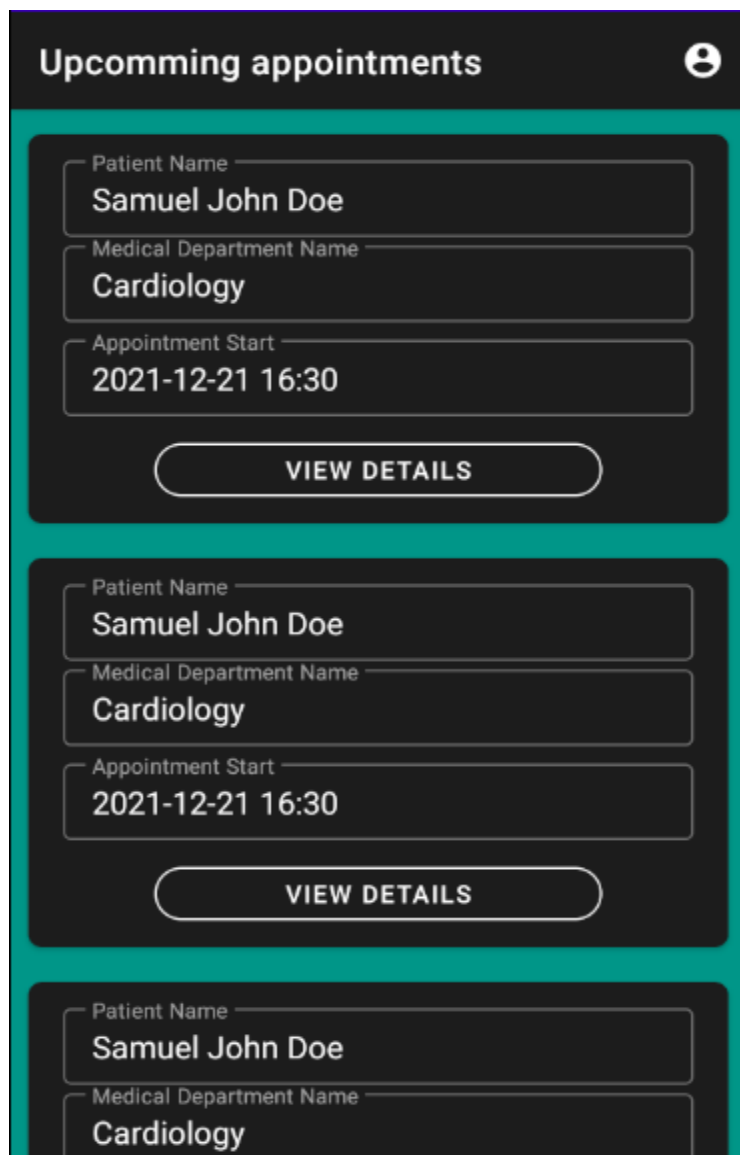
## 5.7. Visszajelzések

Ez az oldal design tekintetében teljesen ugyanaz a Saját időpontok oldallal. (34. Ábra: Saját időpontok), annyi az eltérés csupán, ha a felhasználó rákattint egy adott visszajelzésre a megjelenő részlet kicsit eltér az előbbinél (36. Ábra: Visszajelzés részletei).



36. Ábra: Visszajelzés részletei.

## 5.8. Orvosi felhasználó



37. Ábra: Orvosi felhasználói oldal

Az orvosi felhasználói oldal is hasonló a klienséhez, azzal az eltéréssel, hogy itt az alsó menüsáv eltűnik, hiszen csak egyetlen oldalról van szó. A felhasználó megtekintheti a hozzá rendelt konzultációkat, a *View Details* gombra kattintva egy



részletes leírás jelenik meg (35. Ábra: Időpont részletei). Ugyanúgy érvényes a profil ikon funkcionalitása, a már előzőleg ismertetett oldalt (33. Ábra: Profil) jeleníti meg az orvos adataival.

## 6.Összegzés

### 6.1. Továbbfejlesztési lehetőségek

A projekt céljának és implementációja közben jópár fejlesztési lehetőségre gondoltunk, melyek nagyságrendekkel javíthatják a felhasználói élményt, és tágíthatják a felhasználó lehetőségeit, ezáltal mégjobban megkönnyítve a applikáció nyújtotta szolgáltatások elérhetőségét.

Továbbfejlesztési lehetőségek pontokba szedve:

- A kórházak lista különböző szűrők alapján tudja rangsorolni, illetve szűrni őket, mint például távolság(a felhasználó jelenlegi helyzete szerint távolság szerint csökkenő, illetve növekvő sorrendbe tudja helyezni).
- A kórházak listáján a felhasználónak van lehetősége megtekinteni minden kórház adatait, ahol azt is láthatja, hogy a térképen, hol helyezkedik el.
- A kórházi részlegek listáján a felhasználónak van lehetősége megtekinteni minden kórházi részleg adatait a részleteit
- Az időpontfoglalás során a felhasználó megtekintheti az orvos adatait, illetve az orvos értékelését, 1-től 10-es skálán, továbbá megjegyzést küldhet az időpontfoglalással együtt
- Az orvos a beérkező időpontfoglalás részleteit megtekintheti, majd elfogadhatja vagy elutasíthatja azt. A felhasználó értesítést kap az orvos visszajelzéséről, majd ezután jelenik meg az időpont Saját időpontok listáján
- A felhasználó a lejárt időpontját értékelheti, és megjegyzést fűzhet hozzá
- Az orvos a készíti el a visszajelzéseket, melyek során a állományokat is tud csatolni, melyek a vizsgálatok eredményeit tartalmazzák
- A felhasználó a kiszámlázott értékeket online is kifizetheti
- Egy adminisztrátori felület létrehozása, ahol az adminisztrátor frisiitheti, törölheti a kórházak, azok részlegei és az egészségügyi dolgozók adatait, és újakat is létre tud hozni.

- Egy beállítások menüoldal létrehozása, ahol a felhasználó személyre szabhatja az applikációt, illetve a személyes adatait frissítheti
- Az applikációt használó személy, megváltoztathatja az applikáció nyelvét(angol, magyar, román)

# 7. Irodalomjegyzék

1. ....	Bevezetés	2
1.1. Az alkalmazás rövid ismertetése .....		2
1.2. Az alkalmazás logika menete .....		3
2. ....	Célkitűzések	3
3. ....	Követelmény specifikáció	4
3.1. Felhasználói követelmények.....		4
3.2. Rendszer követelmények <sup>[60]</sup> .....		5
3.2.1. Funkcionális követelmények .....		5
3.2.2. Nem funkcionális követelmények.....		5
4. ....	Tervezés	6
4.1. Alkalmazás felépítése (Architektúrája) .....		6
4.2. Szekvencia diagramok.....		8
4.2.1. Regisztráció.....		8
4.2.2. Bejelentkezés .....		9
4.2.3. Időpontfoglalás .....		10
4.2.4. Orvoshoz rendelt időpontok megtekintése .....		11
4.3. Osztály diagramok .....		12
4.4. Bejelentkezés, regisztráció hitelesítése.....		20
4.5. Endpoints .....		21
4.6. Alkalmazás Managelése .....		23
4.6.1. Verziókövetés .....		23
4.6.2. Egyéni task-ok managelése .....		25
5. ....	Alkalmazás bemutatása	26
5.1. Bejelentkezés .....		26
5.2. Regisztráció.....		27
5.3. Elfelejtett jelszó .....		28
5.4. Időpontfoglalás.....		29
5.4.1. Kórház kiválasztása .....		29

5.4.2. Kórházi részleg kiválasztása.....	30
5.4.3. Időpontfoglalás elvégzése.....	31
5.5. Profil megtekintése.....	37
5.6. Saját időpontok .....	38
5.7. Visszajelzések.....	39
5.8. Orvosi felhasználó.....	40
6. ....	Összegzés
.....	41
6.1. Továbbfejlesztési lehetőségek.....	41
7. ....	Irodalomjegyzék
.....	43