SAPIENTIA
ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM

Qliniqueue fejlesztői dokumentáció

Tavaszi félév

2021-2022

Fejlesztés .NET környezetben

Deák Adrienn

Köllő Zsolt

Kristóf István-Levente

Számítástechnika IV.

Tartalomjegyzék

1. Bevezető és motiváció

Napjainkban egyre nagyobb teret hódítanak a telefonos alkalmazások, mindent azon szeretnénk elvégezni a kényelem érdekében. Sok embernek nehezére esik a telefonos időpontfoglalás, így szívesebben foglalnak időpontot valamilyen online rendszeren keresztül. Ezek a rendszerek nem csak a szégyenlős személyek életét könnyebítik meg, hanem azokét is, akiknek rohanó napjukba nem fér bele a plusz telefonálás, főleg ha elsőre nem is fogadják hívását. Ezekből az okokból kifolyólag fontosnak tartjuk az ilyen vagy az ehhez hasonló rendszerek jelenlétét mindennapi életünkbe. Ötletünk egy olyan Android alkalmazás, mely egy klinikán rendelő szakorvosokhoz való online programálást tesz lehetővé. Az alkalmazásunk neve Qliniqueue, mely tükrözi a megcélzott terület nevét.

2. Csapatfelosztás és projektmenedzsment

Csapatunk tagjai Deák Adrienn, Köllő Zsolt, Kristóf István-Levente számítástechnika szakos, negyed éves hallgatók. A feladatokat igyekeztünk azonos mértékben kiosztani, illetve amennyire lehetőségünk volt közösen dolgoztunk, így az ötletet egy hosszas beszélgetés után határoztuk meg. A közös munka hol online körülmények között, hol pedig személyesen zajlott.

A csoporton belül a projektmenedzser szerepét Deák Adrienn kapta, aki munkája nagy részét tervezéssel, menedzseléssel és dokumentálással töltötte, így az programozás nagyobb részét Köllő Zsolt és Kristóf István-Levente végezték el.

Annak érdekében, hogy a munka gördülékenyen folyhasson Github segítségével menedzseltük projektünk, melyen belül egy Kanban board-ot is létrehoztunk, hogy felügyelni tudjuk a folyamatokat

[1].

A felhasználói felülettel kapcsolatos ötletelések is közösen történt és a megbeszéltek alapján Deák Adrienn Figma tervezőprogram segítségével valósította meg azt
[2]

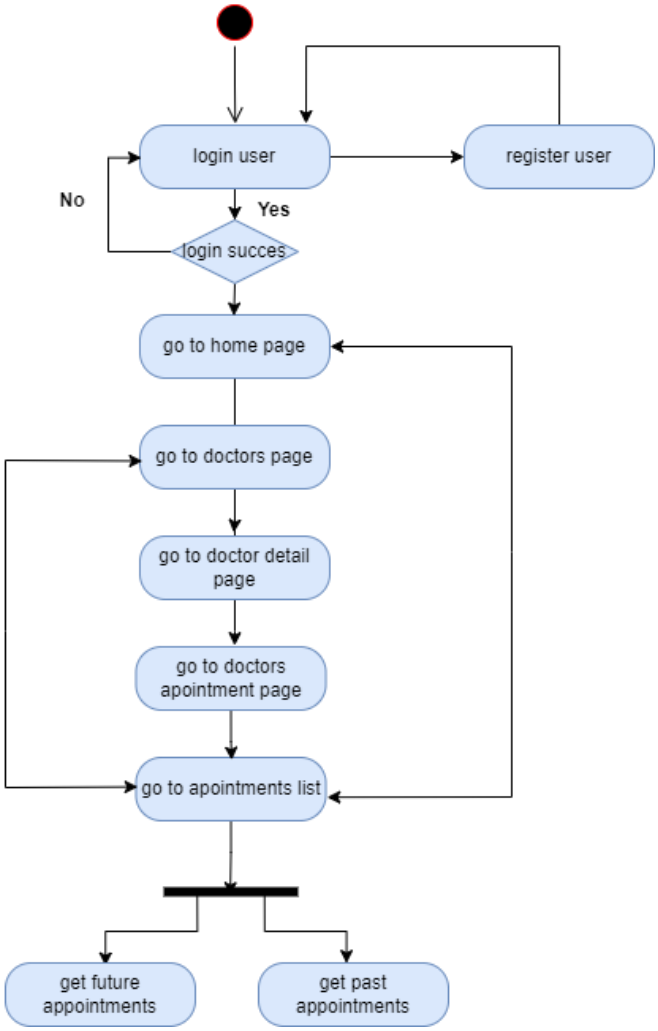
[1].

projekt megvalósítása során a teszteket Zsolt és Levente írták meg. A regisztráció és a bejelentkezés Adrienn feladata volt. Levente feladatai közé tartozott még a menü létrehozása. Zsolt

az időpontfoglalás frontend és backend részén dolgozott. A „Fake REST API” megvalósítását Levente és Zsolt közösen végezték.

3. Követelményespecifikáció

3.1. Funkcionális követelmények



1. ábra - Activity diagram

A felhasználó az applikáció elindítási után a bejelentkezési oldalon találja magát, itt két lehetőség áll előtte, bejelentkezik a saját felhasználójával, vagy pedig tovább navigál a regisztrációs

oldalra. Regisztráció után a felhasználó vissza kerül a bejelentkezési oldalra, ahol a létrehozott felhasználóval bejelentkezhet.

A bejelentkezés után a felhasználó a kezdőlapon találja magát, itt két lehetőség áll előtte. Az ,,Orvosaink” gomb lenyomásával megjelenítheti az orvosok listáját, vagy pedig megnyithatja az úgynevezett ,,navigation drawer”-t, melynek segítségével átnavigálhat az applikáció más részeire.

A ,,navigation drawer” megnyitásával a következő oldalakra tudunk navigálni: ,,Kezdőlap”, ,,Orvosok”, illetve ,,Programálásaim”.

Az ,,Orvosok” oldalon egy listát találunk az applikációban szereplő orvosokkal, abban az esetben, ha rákattintunk egy orvosra a listában, akkor átkerülünk a kiválasztott orvos adatlapjára, ahol bővebb információt találunk az orvossal kapcsolatosan. Az adatlapon található egy ,,Időpontfoglalás” gomb, ezt megnyomva átkerülünk egy másik oldalra, ahol programálni tudjuk magunkat az adott orvoshoz.

Az ,,Időpontfoglalás” oldalt ki kell tölteni a saját információinkal, majd ha kiválasztottuk a dátumot és az időpontot is programálni tudjuk magunkat a ,,Foglalás” gombra kattintván.

A sikeres időpontfoglalások megjelennek a ,,Programálásaim” oldalon, ahol meg tudjuk tekinteni a soron következő időpontokat, illetve azokat emelyek már lejártak.

3.2. Nem funkcionális követelmények

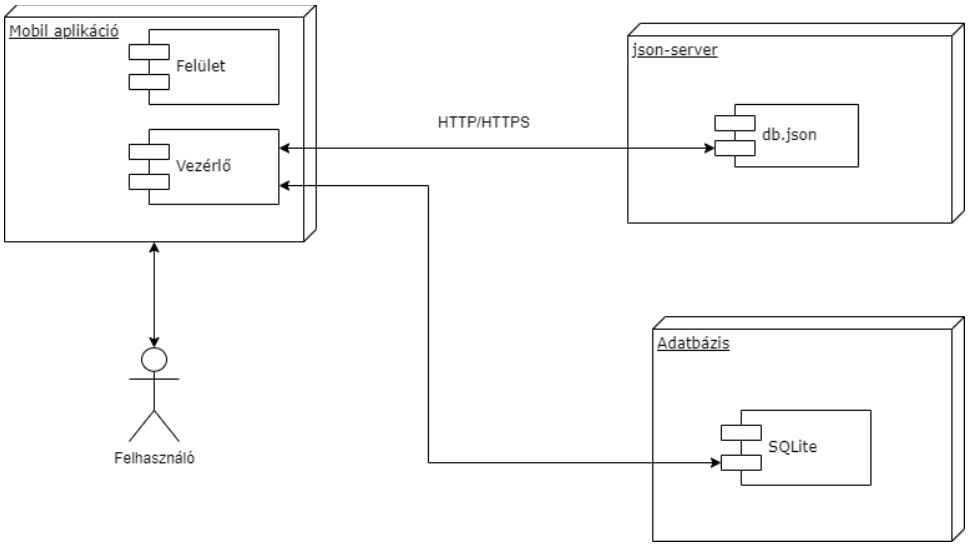
- Könnyen kezelhetőség
- Gyorsaság
- Megbízhatóság
- Személyes adatok biztonsága
- Elérhetőség
- Skálázhatóság
- Github verziókövetés

4. Rendszerarchitektúra

A C# programozási nyelv egyike a legjobb választásoknak, amikor Windows operációs rendszeren szeretnénk aplikációkat fejleszteni. Nagyon széles körben használt nyelvnek számít az objektum orientáltsága és a magas szintű nyelvezete miatt. Ezen programozási nyelvet használja a

Xamarin környezet is, amely lehetőséget ad Android, IOS és Windows alkalmazások fejlesztésére .Net keretrendszerben. Xamarinban a programcsomagok könnyen kezelhetők és frissíthetők NuGet segítségével. A .Net keretrendszer használata lehetőséget nyújt arra is, hogy asszinkron módon programozzunk, ez Xamarin alkalmazások esetén sokszor a hasznunkra válik.

A rendszer kiépítése során igyekeztünk az MVC (Model-View-Controller) architektúrát követni. Az MVC egy külön vezérlést biztosít, amely a felhasználói kéréseket kiszolgálja. A Controller fogadja a kérést a felhasználótól, és ezt feldolgozva előállítja a megfelelő nézetet. Ezen módszer használatával a nézet teljesen mentesül a kérések fogadásától.



2. ábra - Kontent diagram

5. Rendszerspecifikáció

Az általunk felépített rendszer egy Xamarin környezetben elkészített applikáció, amely orvosokhoz való online programálást tesz lehetővé. Az applikáció használatához szükséges, hogy a felhasználó regisztrálni tudjon a felületen keresztül. A felhasználó adatai lokálisan tárolódnak el egy SQLite adatbázisban, melyet az sqlite-net-pcl NuGet csomag telepítésével építettük be az alkalmazásba. Megemlítendő, hogy az applikáció további fejlesztése során már más adat tárolási módszert használtunk, viszont fontosnak tartottuk, hogy a felhasználó adatai a legnagyobb biztonságnak örvendjenek, ezért választottuk a regisztráció és a bejelentkezés esetében az SQLite adatbázist.

A továbbiakban egy úgynevezett „Fake REST API”-t használtunk az adatok tárolására és lekérdezésére. Az adatokat JSON formátumban tároltuk el. Az adatok egy lokális .json fájlban lesznek eltárolva amelyből fogjuk tudni lekérni a nekünk megfelelő adatokat. A rendszer működtetéséhez a json-server-t kell telepítenünk lokálisan. A package.json fájlban adhatjuk a rendszerünkhöz szükséges

információkat, mint például a json-server verziószámát, vagy futtatni kívánt parancsokat. Ezt követően az „npm install” parancs kiadásával települni fognak a szükséges függőségek. Az „npm run json:server” parancs kiadásával indíthatjuk el a rendszerünket. Ehhez viszont szükségünk van arra, hogy a package.json fájlba megadjuk, hogy a rendszer futásakor melyik fájlt használja fel az adatok eltárolására, és esetünkben szükség volt arra is, hogy megadjuk azt az ip címet amelyiken induljon a rendszer az alapértelmezett localhost helyett. Erre azért volt szükség, hogy azonos hálózaton lévő mobileszközzel el tudjuk érni az adatainkat. A fenti két funkcionalitást a package.json „scripts”-en belül kell beállítanunk a következő sor megadásával: ”json:server”: ”json-server –watch db.json –host <ip:address> db.json”. Ezt követően a telepített függőség biztosítsa számunkra a CRUD műveleteket és helyettünk elvégzi az adatok kezelését.

A navigáció az egyik legfontosabb része az Android és IOS applikációknak. A navigation drawer segítségével pár kattintással el tudjuk érni az applikáció bármely részét. Ennek a megvalósításához Shell navigációt használtunk. A Shell navigáció könnyen megvalósítható, létrehozunk egy XAML fájlt, ami az esetünkben az AppShell fájl, ebben valósítjuk meg a navigációt a ShellItem, illetve a FlyoutItem segítségével. A ShellItem-ben megadjuk az első oldalt amit szeretnénk, hogy betöltsön az applikáció, míg a FlyoutItem jelöli azokat az oldalakat, ahová szeretnénk majd navigálni az első oldalról. Ha az alkalmazás egy adott pontján szeretnénk elkezdeni a Shell navigációt (esetünkben a bejelentkezési oldal után), akkor a MainPage-et át kell állítani az AppShell.xaml fájlra.

6. Az almalmazás tesztelése

Annak érdekében, hogy egy minőségibb alkalmazást készíthessük különböző teszteket végeztünk. Bár a tesztek alapvetőek, de ezzel mégis biztonságosabbá tehetjük rendszerünk. A tesztek nagy részét az egységtesztek teszik ki, ezen belül inkább a felhasználói felület elemeinek a tesztelése. Azért gondoltuk, hogy inkább a felhasználói felületet teszteljük, mivel ezen a felületen keresztül lép interakcióba a felhasználó az alkalmazással, így az jól tervezett és megvalósított kell legyen, hogy a felhasználó elvárásainak megfeleljünk.

7. Továbbfejlesztési lehetősége

Továbbfejlesztési lehetőségnek látjuk a komolyabb támadások elleni védelem bevezetését, mivel az alkalmazás személyes információkat tartalmaz. E mellett számos funkcionalitás bevezetése is lehetséges, mint például szóljon, hogy 1 év után ideje újra rutinellenőrzésre mennünk, vagy akár

megkönnyítve az orvos és asszisztense munkáját a felhasználónak lehetne olyan lehetősége is, hogy feltölthesse már meglevő beteglapjait, receptjeit. Ezzel a funkcionalitással nem csak a klinika munkáját könnyebbítenénk meg, de a beteg is rendszerezni tudná orvosi iratait.

8. Ábrajegyzék

9. Hivatkozások

[1]https://github.com/dkadrienn/Qliniqueue.git

[2] https://www.figma.com/file/5t5rbzofPKJQyMAWZXUrSW/QTlinik?node-id=0%3A1