Pannon Egyetem

Műszaki Informatikai Kar

Rendszer- és Számítástudományi Tanszék

programtervező informatikus BSc

SZAKDOLGOZAT

Windows Phone alkalmazás tervezése és fejlesztése betegadatok kezeléséhez

Mód Márton

Témavezető: Frits Márton

2014

# Feladatkiírás

# Nyilatkozat

Alulírott Mód Mártonhallgató, kijelentem, hogy a dolgozatot a Pannon Egyetem Rendszer- és Számítástudományi tanszékén készítettem a programtervező informatikus végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozatban lévő érdemi rész saját munkám eredménye, az érdemi részen kívül csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy a dolgozatban foglalt eredményeket a Pannon Egyetem, valamint a feladatot kiíró szervezeti egység saját céljaira szabadon felhasználhatja.

Veszprém, 2014. december 12.

Mód Márton

Alulírott Frits Márton témavezető kijelentem, hogy a dolgozatot Mód Mártona Pannon Egyetem Rendszer- és Számítástudományi tanszékén készítette programtervező informatikus végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozat védésre bocsátását engedélyezem.

Veszprém, 2014. december 12.

Frits Márton

# Köszönetnyilvánítás

Szeretnék köszönetet mondani Frits Márton témavezetőmnek a segítségért és iránymutatásért, amit a szakdolgozatom elkészítése alatt nyújtott.

Szintén szeretném megköszönni a családomnak, barátaimnak a sok támogatást a dolgozat elkészítéséhez.

Nem szeretnék megfeledkezni Bob Taborról sem, akinek remek online kurzusai vezettek be a Windows Phone alkalmazás fejlesztés világába.

# tartalmi összefoglaló

Napjainkban életünk meghatározó részét képezi az okostelefonok használata. Ezen készülékek jelentős hatást gyakorolhatnak egyesek életvitelére különböző alkalmazások használatával. Ilyen alkalmazások közé sorolhatóak az egészségügyi applikációk is. Feladatom céljául egy olyan alkalmazás fejlesztését tűztem ki, amivel a betegek saját mért adataikat rögzíthetik, amiket az orvosa nyomon követhet. Az applikáció segítségével hatékonyabbá és sikeresebbé tehető az orvosok munkája. Az alkalmazás használatával időt és energiát spórolhatnak mind a betegek és az orvosok is.

Az alkalmazás az orvosok és pácienseik között levő kommunikációs határokat hivatott legyőzni. Az applikáció használatával elkerülhetőek a gyakoribb vizitek, az orvosnak kevesebb pácienst kell fogadnia munkája során, és egyszerűen, szinte bárhonnan nyomon követheti egyes betegeinek állapotát.

Az applikációt Doctor-Patient-nek neveztem el. A Doctor-Patient alkalmazás segítségével a betegek rögzíthetik vérnyomás és vércukor mérésük eredményeit. Ezeket a rögzített értékeket a beteg orvosa megtekintheti és szaktudása szerint következtetéseket vonhat le az összegyűlt adatokból. Hogy a kommunikáció két irányú lehessen, az orvos különböző űrlapokat küldhet betegeinek, amit a beteg kitöltve visszaküldhet az orvosa felé, aki ebből levonhatja a megfelelő következtetéseket. A méréseket az alkalmazás képes grafikonon ábrázolni különböző időszakokra lekérve, hogy átfogóbb képet lehessen kapni a páciens állapotáról.

Az alkalmazást Windows Phone 8.1 platformra készítettem el Windows Runtime alkalmazásként. Elkészítettem az alkalmazást kiszolgáló szerver prototípusát és a hozzá tartozó adatbázist, ugyan ez nem képezi szakdolgozatom részét, de szükségét éreztem egy ilyen elkészítésének.

Az elkészült szoftver segítségével elértem a feladat által kitűzött célt. Az applikáció sikeresen kommunikál a teszt szerverrel és a tervezett funkciókat megvalósítottam.

**Kulcsszavak:** Windows Phone 8.1 alkalmazás, egészség, beteg felügyelés, vércukor, vérnyomás

# abstract

Nowadays, using smart phones are essential part of our daily lives. These devices can have a deep impact on one’s lifestyle through a wide range of applications. Healthcare applications can be rated into this category. The aim of my task is to develop such an application that people can use to store their own recorded medical data which can be tracked by their doctor. This application can make any doctor’s job more successful and efficient. With the use of this software we can save time for both doctor and patient.

The application’s purpose is to eliminate communication problems between patients and medical personnel. With it, the number of doctor’s appointments can be reduced, a fewer number of sickened people have to be treated during work time because their current health status can be tracked with the application.

I have given the application the “Doctor-Patient” name. With the use of this, people can record their own blood pressure and blood-sugar level results. These records can be obtained by their doctors and may be able to be led to conclusions from the recorded data for further therapy. In order for this to be a two-way communication, the doctors can send various forms to their patients that can be sent back after filling it. These results can be illustrated with charts and spreadsheets for different periods of time in order the get a more comprehensive picture of the patient’s illness.

The app is developed on Windows Phone 8.1 platform as a Windows Runtime application. I have created a prototype for the server and a database connected to it, and although this wasn’t part of my dissertation, I felt the need for making one.

With the finished software I have achieved my aim. The application can successfuly communicate with the test server and contains all planned features.

**Keywords**: Windows Phone 8.1 application, health, patient tending, blood-sugar level, blood pressure level

Tartalomjegyzék

[Feladatkiírás II](#_Toc406076025)

[Nyilatkozat III](#_Toc406076026)

[Köszönetnyilvánítás IV](#_Toc406076027)

[tartalmi összefoglaló V](#_Toc406076028)

[abstract VI](#_Toc406076029)

[1. Bevezetés 1](#_Toc406076030)

[2. Egészségügyi alkalmazások 2](#_Toc406076031)

[2.1. Kórház és orvos kereső alkalmazások 2](#_Toc406076034)

[2.2. Segélyhívó alkalmazások 3](#_Toc406076035)

[2.3. Beteg adatokat tároló alkalmazások 3](#_Toc406076036)

[2.4. Egészségügyi tudásbázis alkalmazások 3](#_Toc406076037)

[2.5. Vérnyomás érték rögzítő alkalmazások 4](#_Toc406076038)

[2.6. Vércukor érték rögzítő alkalmazások 4](#_Toc406076039)

[2.7. Konklúzió 4](#_Toc406076040)

[3. Windows Phone platform 6](#_Toc406076041)

[3.1. Röviden a platform történetéről 6](#_Toc406076043)

[3.1.1. Windows Phone 7 6](#_Toc406076044)

[3.1.2. Windows Phone 7 frissítései 8](#_Toc406076045)

[3.1.3. Windows Phone 8 10](#_Toc406076046)

[3.1.4. Windows Phone 8 frissítései 12](#_Toc406076047)

[3.1.5. Windows Phone 8.1 12](#_Toc406076048)

[3.1.6. Windows Phone 8.1-es frissítések 14](#_Toc406076049)

[3.2. Fejlesztés Windows Phone-ra 15](#_Toc406076050)

[3.2.1. Előfeltételek 15](#_Toc406076051)

[3.2.2. Alkalmazás fajták 16](#_Toc406076052)

[3.2.3. Alkalmazás sablonok 16](#_Toc406076053)

[3.2.4. Támogatott programozási nyelvek 17](#_Toc406076054)

[4. Feladat specifikáció 18](#_Toc406076055)

[4.1. Rendszer követelmények 18](#_Toc406076057)

[4.2. Funkcionális követelmények 18](#_Toc406076058)

[5. Tervezés 21](#_Toc406076059)

[5.1. Teszt szerver 21](#_Toc406076061)

[5.2. Adatbázis 22](#_Toc406076062)

[5.2.1. Az adatbázis felépítése 22](#_Toc406076063)

[5.3. Alkalmazás 24](#_Toc406076064)

[5.3.1. Használati esetek 24](#_Toc406076065)

[5.3.2. Folyamatok 25](#_Toc406076066)

[5.3.3. Struktúra 27](#_Toc406076067)

[5.3.4. Interakció 28](#_Toc406076068)

[5.4. JSON 30](#_Toc406076069)

[6. Implementálás 31](#_Toc406076070)

[6.1. Model View ViewModel 31](#_Toc406076072)

[6.2. Univerzális Hub alkalmazás sablon 32](#_Toc406076073)

[6.3. Felhasznált programkönyvtárak 32](#_Toc406076074)

[6.3.1. Json.NET 33](#_Toc406076075)

[6.3.2. QKit 33](#_Toc406076076)

[6.3.3. NotificationsExtensions 33](#_Toc406076077)

[6.3.4. NodeTime és DaveSmits.ThemeManager 34](#_Toc406076078)

[6.4. Megosztott kód 34](#_Toc406076079)

[6.4.1. Segédosztályok 35](#_Toc406076080)

[6.4.2. Átalakítók 40](#_Toc406076081)

[6.4.3. Adat modell 40](#_Toc406076082)

[6.4.4. Nyelvi támogatás 43](#_Toc406076083)

[6.5. Háttérfolyamatok 44](#_Toc406076084)

[6.6. Megjelenítés 44](#_Toc406076085)

[6.6.1. HubPage 45](#_Toc406076086)

[6.6.2. Űralap generálás 47](#_Toc406076087)

[7. Alkalmazás bemutatása 49](#_Toc406076088)

[7.1. Bejelentkezési és regisztrációs felület 49](#_Toc406076090)

[7.2. Páciens felületei 51](#_Toc406076091)

[7.3. Orvos felületei 56](#_Toc406076092)

[8. Tesztelés 61](#_Toc406076093)

[8.1. Fejlesztői környezet 61](#_Toc406076095)

[8.2. Teszt szerver 61](#_Toc406076096)

[8.3. Alkalmazás teszt 61](#_Toc406076097)

[9. Összefoglalás 63](#_Toc406076098)

[*Irodalomjegyzék* 64](#_Toc406076099)

[*Ábrajegyzék* 66](#_Toc406076100)

[*Mellékletek* 67](#_Toc406076101)

# Bevezetés

A szakdolgozatom témája egy egészségügyi alkalmazás elkészítése Windows Phone platformra, amiben betegek rögzíthetik mért adataikat, amiket a betegek orvosai is megtekinthetnek. Az alkalmazás segíti az orvosok és páciensek közötti kommunikációt. Szakdolgozatomban ismertetni fogom alkalmazásom fejlesztésének részleteit, ami a Doctor-Patient nevet kapta.

A munkám kezdetén a feladatom a Pannon Egyetemen futó GE (General Electric) Világszínvonalú Intelligens és Inkluzív Egészségügyi Információs és Döntéstámogató Keretrendszer kutatási projekt keretében indult. A projekt célja egy elektronikus beteg– és egészségügyi adatokat kezelő informatikai rendszer létrehozásának megalapozása, egy ilyen rendszer lehetőségeinek kutatása. Ehhez a rendszerhez készítettem a mobil klienst. Azonban a projektben tervezett háttérrendszer nem készült még el a szakdolgozatom írásakor. Az alkalmazás megfelelő működéséhez szükség van egy szerverre, amivel adatokat cserélhet és kommunikálhat. Hogy megvalósíthassam a szakdolgozatom feladatát egy projekten kívüli megoldásban kezdtem el gondolkodni.

Az utóbbi pár év folyamán, a piacon kapható okostelefonok egyre elérhetőbbé váltak, napjainkban az emberek többsége rendelkezik ilyen eszközzel. Ezen készülékek jelentős hatást gyakorolhatnak egyesek életvitelére különböző alkalmazások használatával. Az egészségügyi alkalmazásoknak egyre nagyobb potenciálja van. Feladatom céljául egy olyan applikáció fejlesztését tűztem ki, aminek segítségével hatékonyabbá és sikeresebbé tehető az orvosok munkája. Az alkalmazás használatával időt és energiát spórolhatnak mind a betegek és az orvosok is.

Az alkalmazásom elsősorban diabéteszes és magas vérnyomásban szenvedő betegek számára készült. Gyakorlatban főleg magán klinikák és házi orvosi rendelők alkalmazottai és betegei által lehetne használni. A szoftver segítségével az orvosok felügyelhetik betegeiket a rögzített értékek alapján. Különböző kérdőívek és űrlapok küldésével további információkat kaphatnak a páciens állapotról. Az alkalmazással szerzett adatok alapján az orvos konklúziókat vonhat le, minél több információ áll a rendelkezésére, annál jobb döntéseket hozhat beteg kezelése során.

# Egészségügyi alkalmazások

Azokat az alkalmazásokat, amelyek egészségünk megőrzésében, egészségünk javításában, betegségek leküzdésében segítenek, az egészségügyi alkalmazások kategóriájába soroljuk. Ez egy rendkívül tág terület. Ide sorolhatóak a futást segítő alkalmazásoktól kezdve, a helyes táplálkozást segítő alkalmazásokon át, különböző orvosi alkalmazásokig. Ebben a fejezetben az utóbbi csoportot fogom megvizsgálni. A különböző okostelefon alkalmazás boltokban számos ilyen alkalmazást találhatunk. Kutatásomat a Windows Phone alkalmazás áruházra korlátoztam, mivel a szakdolgozatom témájában szereplő alkalmazást is erre a platformra készítettem. Vizsgálatom egyik szempontja volt, hogy funkciókban hasonló alkalmazásokat keressek a Doctor-Patient alkalmazáshoz.



## Kórház és orvos kereső alkalmazások

Ezeknek az alkalmazásoknak a segítségével kórházakat, orvosi rendelőket, és orvosokat kereshetünk közelünkben. Segítségükkel lekérhetjük az orvosok rendelési időpontjaikat.

Említésre méltó alkalmazások:

* *Carolinas*¸ applikácó a *Carolinas HealthCare System*től. Főbb funkciói: térképes kórház és sürgősségi osztály keresés, várakozási idők megtekintése, orvos keresés, időpont foglalás. Angol nyelvű és csak USA-ban levő kórházakkal működik.
* *Doctor-OK*, *ALLWIN Informatika* készítette. Segítségével Budapest orvosaihoz lehet bejelentkezni. Magyar nyelvű.
* *Doctors Online*, az *AirPersons* szolgáltatásaihoz készült mobil kliens. A szolgáltatásban regisztrált egészségügyi szakemberektől lehet kérdezni, tanácsokat kérni. Van magyar nyelvi támogatás.

Egyedül egy darab olyan alkalmazást sikerült találnom, amivel magyar orvosokat lehet keresni, de azt is csak Budapesten. Magyarországi kórházakat a *Mobil Kisokos* nevű alkalmazással lehet megtalálni, de ezt az alkalmazást nem sorolnám bele az egészségügyi alkalmazások kategóriájába.

## Segélyhívó alkalmazások

Több segélyhívó alkalmazást is találtam, amik közül kiemelkedett az *Emergency Call* nevű applikácó *Randonmobile*-tól. Segítségével a világon szinte bárhonnan felhívhatjuk a helyileg megfelelő segélyhívó számot, mivel helymeghatározással megállapítja, hogy melyik segélyhívó számot kell hívnunk. A hívás során bediktálhatjuk a megállapított helyzetünk koordinátáját. Angol nyelvű.

Másik kiemelkedő applikáció az *Emergency Kit* névre hallgat. Rendkívül hasznos alkalmazás lehet vészhelyzetek esetén. Az alkalmazásban beállíthatjuk, hogy kit értesítsenek vészhelyzet esetén, megadhatjuk, milyen vércsoportunk van, milyen gyógyszereket szedünk, mire vagyunk allergiásak. Ezek az információk rendkívül értékesek lehetnek, ha balesetet szenvedtünk.

## Beteg adatokat tároló alkalmazások

Ezen alkalmazásokkal tárolhatjuk kórtörténetünket és szedet gyógyszereinket. Említésre méltó alkalmazások:

* *Med Tracker, Qwinix Technologies Inc*-től. Főbb funkciói: kórtörténetet és szedett gyógyszereket tárulhatunk vele, emlékeztetőt állíthatunk be a gyógyszer beszedésének időpontjára, jelszavas védelem. Angol nyelvű.
* *Documed, Roberto Cataudella*-tól. Beteg adatok tárolására használhatjuk. Az alkalmazás képes lementeni a tárolt adatokat számítógépünkre.

## Egészségügyi tudásbázis alkalmazások

Ide azokat az alkalmazásokat soroltam be, amik valamilyen egészségügyi tudást oszthatnak meg a felhasználóval, például betegségek tüneteit, elsősegély nyújtás folyamatát.

Említésre méltó alkalmazások:

* *Gyógyszerkereső*, *EvilHedgehog*tól. Magyarországon forgalmazott gyógyszerekre kereshetünk rá, az adatokat folyamatosan frissülő gyógyszer adatbázisokból nyeri. Közli a gyógyszer árát, kiszerelését és betegtájékoztatóját is elolvashatjuk.
* *Mobile First Aid*, *Stefano Driussi*-tól. Elsősegély nyújtási folyamatban segédkező alkalmazás.
* *First Aid kit*, *JOLTA TECHNOLOGIES*-től. Orvosi és háztartási tanácsok és praktikák gyűjteményét foglalja magába. Betegségekről részletes leírásokat olvashat a felhasználója.

## Vérnyomás érték rögzítő alkalmazások

Ilyen alkalmazásokat nagy számban lehet találni Windows Phone-ra. Legtöbb alkalmazás képes a rögzített értékekről statisztikát készíteni és grafikonon ábrázolni azokat. Talált alkalmazások (zárójelben a kiadó neve): *BloodPressureDB* (*Horst* *Klier*), *Medical Assistant* (*Teos*), *Arterial* (*DevelopingStudent*), *Blood Pressure Pro* (*Thorsten* *Lange* *IT-Solutions*), *Blood Pressure* (*SilverCode Solutions Ltd*), *My Blood Pressure* (*Code Forge LLC*), *Blood Pressure* (*vue*). Kiemelném közülük *BloodPressureDB* és *Blood Pressure Pro* alkalmazásokat, mivel lehetőséget adnak a jelentések exportálására PDF vagy Excel fájlformátumokban.

## Vércukor érték rögzítő alkalmazások

Több mint egy tucat vércukor érték rögzítő alkalmazásra bukkantam kutatásom során. Ezekben az alkalmazásokban a felhasználó rögzítheti mért vércukor szintjének értékét, amit az alkalmazás eltárol, amikből statisztikákat készít vagy grafikonon ábrázolja. Talált igényesebb alkalmazások (zárójelben a kiadó neve): *Diabetes Journal* (*xanosoft*), *DiabetesControl* (*Espirit*), *myDiabetes* (*Christian Müller*), *Glucose Tracker* (*Bo9 Consulting*), *Diabetes* (*Areon*), *Diabetes Manager* (*Collective Wisdom Now*), *Diabetic Diary* (*Lamex Software*), *DiabetesDiary* (*Nicola Scapin*), *Pulse* (*123Developer*). Itt is találni olyan applikációt, amiben van adat exportálási lehetőség, ilyen például a *Diabetes Manager*. Ebben a kategóriában nem találtam magyar nyelvű applikációt.

## Konklúzió

Kutatásom alapján kijelenthetem, hogy a Windows Phone alkalmazás áruházban hiányos a kínálat jó minőségű, magyar nyelvű orvosi alkalmazásokból. Angol nyelvű alkalmazásokból széles kínálat áll rendelkezésre, de nem mindenki beszél angol, azért fontosnak tartom, hogy az angolul nem tudó betegeknek is legyen lehetőségük orvosi alkalmazások igénybevitelére. Ugyan csak nincs olyan alkalmazás, amivel a páciens saját orvosával tarthat közvetett kapcsolatot. Az általam készített alkalmazás ezt az űrt próbálja betölteni.

A Technische Universität Dresden elvégzett egy kutatást az 50 év feletti diabéteszes betegek mobil alkalmazás használatának elfogadottságáról [9]. A kutatás eredménye alapján nagyon fontos, hogy olyan alkalmazást dobjunk piacra, aminek használata jelentős előnnyel jár. Fontos az egyszerű és könnyen megtanulható kezelés. Érdemes támogatást beépíteni az alkalmazásba a gyengén látók számára. A kutatás alapján az is kiderült, hogy az idősebb embereknek fontos tényező, hogy az alkalmazást a saját nyelvükön használhassák, mivel zavaró számukra az angol nyelv használata. Másik számukra zavaró tényező volt a mobil készülékek magas ára. Az általam fejlesztett alkalmazás ingyenes és az alsó kategóriás Windows Phone készülékekhez relatíve olcsó áron hozzá lehet jutni.

# Windows Phone platform

Ebben a fejezetben a Windows Phone-ról, a platform történetéről, fontosabb eseményeiről lesz szó. A fejezet végén bemutatásra kerül a Windows Phone alkalmazásfejlesztés.

A Windows Phone egy Microsoft által okostelefonokra fejlesztett operációs rendszer. Elődje a Windows Mobile, mely zseb PC-ken futott. Windows Mobile-lal ellentétben a Windows Phone-t fogyasztói piacra szánták. Az International Data Corporation kutatása szerint, 2014 második negyedévben a harmadik legnagyobb piaci részesedéssel rendelkezik a mobil operációs rendszerek között az Android és iOS után [7].



## Röviden a platform történetéről

2010. február környékén az Apple dominálta a mobil piacot, Android még csak most lépett bele a versengésbe, de Microsoftnak még nem sikerült piacra dobnia egy okostelefonokra szánt operációs rendszert. Azonban Microsoftnak ott volt a Windows Mobile és a Kin. Utóbbi egy rövid életű mobil telefon sorozat volt, melyet a Sharppal közösen fejlesztettek, de sajnos az eladási statisztikák miatt megbukott. Windows Mobile platform üzleti, vállalati közeg számára készült, így a Microsoftnak nem volt mit adnia a hétköznapi fogyasztók számára.

### Windows Phone 7

Microsoftnak lépni kellet, ezt a 2010-ben tette meg Barcelonában az MWC-n (Mobile World Congress), ahol február 15-én bejelentették új mobil operációs rendszerüket, a Windows Phone 7-et. De hivatalosan csak 2010. november 8-án került forgalomba.

A Windows Phone 7 egy erős alapja volt egy mobil operációs rendszer platformnak. Sok mindenben különbözött a többi operációs rendszertől. A leglátványosabb jegye a kezdőképernyő volt. A legtöbb okostelefon kezdőképernyőjén ikonok sorait találjuk, Windows Phone esetén úgy nevezett élő csempék (*live tiles*) helyezkednek el. Microsoft egy új felhasználói felületet készített, mely a "*Modern*" nevű (korábbi nevén "*Metro*") formanyelven alapszik. Az elő csempék célja az volt, hogy a felhasználó csak egy pillantással hozzájuthasson a kellő információkhoz, és a csempékkel nyithassa meg a számára szükséges alkalmazást. Például az e-mail kliens élő csempéje az olvasatlan üzenetek számát tartalmazza. Sajnos a Windows Phone 7.0-ás verziójában csak a natív alkalmazásoknak voltak igazi élő csempéi, külső fél által fejlesztett alkalmazásokból hiányzott ez a funkció.



3.1. ábra Windows Phone 7-es készülék

Másik lényeges különbség volt az úgy nevezett *hub*ok jelenléte. Ezek a központok megfelelő szintű integrációval rendelkeztek és jó alapot biztosítottak a következő frissítések számára. A kezdeti *hub*ok a következők voltak: *People* *Hub*, *Zune*, játékok, fénykép *hub*ok. A *People Hub* egy telefonkönyvnek felel meg, azzal a kivétellel, hogy itt nem csak a telefonszámot és e-mail címeket lehet tárolni, hanem a Windows Phone 7.0 ezen kapcsolatok szociális jellegére is fókuszált. Miután a felhasználó belépett a különböző e-mail és közösségi oldalakhoz tartozó fiókjaival, a *hub* ezen oldalakról megjelenítette a képeket, állapot információkat, és üzeneteket. Mind a mai napig ez a *hub* a Windows Phone telefonok fontos részét képezi.

A *Zune* *hub*ban volt megtalálható az összes média tartalom. Rendelkezett zene és videó lejátszóval, és FM rádióval is. Az operációs rendszer későbbi verzióiból kivették a Zune zenelejátszó támogatását.

A következő a leginkább reklámozott Windows Phone funkció a beépített *Office Hub* volt. Üzleti felhasználók számára ajánlott szolgáltatás segítségével hozzáférhettek Office dokumentumaikhoz telefonjukon. Bár több volt, mint semmi, de csak nagyon alapvető funkcionalitással rendelkezett, limitálva ezzel vállalati használhatóságát.

A két utolsó *hub* a játékok és fényképek *hub*. A *Pictures Hub* kezelte a fényképeket, de nem csak a helyileg tárolt képeket, hanem amiket a felhőben és a közösségi oldalakon vannak tárolva. Az *Xbox* *hub*ban kaptak helyet a játék alkalmazások.

A *hub* szolgáltatásoktól eltekintve az operációs rendszer többi része szokásos volt. Volt egy Windows Phone alkalmazásbolt, egy szokványos tárcsázó, térképek alkalmazás, amit a *Bing Maps* hajtott, egyszerű kamera és e-mail alkalmazás és további szokásos funkciók, amik megtalálhatóak egy modern operációs rendszerben. Különböző értesítésekről a felhasználó az elő csempék és az úgy nevezett *toast* értesítések formájában szerzett tudomást. *Toast* értesítések a készülék képernyőjének tetején jelentek meg pár másodperc erejéig, sajnos nem volt mód arra, hogy ezeket az értesítéseket később vissza lehessen nézni.

Microsoft több gyártóval is szerződött, köztük a Samsunggal, HTC-vel, Dell-lel, és az LG-vel. Számos piacra dobott készülék és egy nagyvállalat, mint a Microsoft, ellenére a Windows Phone 7 nem szerepelt sikeresen az eladások terén. Ennek két fő oka volt, Microsoft és a fejlesztők. Microsoft egy nagyon lezárt platformot készített, ahol a felhasználónak és a fejlesztőknek egyaránt kevés mozgásteret hagyott. A fejlesztők, a számukra bocsátott API segítségével nem voltak képesek egyes alkalmazások, mint például egy videó szerkesztő vagy egy komolyabb zene lejátszó alkalmazás elkészítésére. Egy gyors, folyékony, de funkcionalitásban limitált operációs rendszert alkotott a Microsoft, ami sürgős frissítésre szorult, ha versenyben akart maradni az iOS és az Androiddal szemben.

### Windows Phone 7 frissítései

Az első frissítés csomag 2011-ben érkezett *NoDo* kódnéven, számos kisebb újítást hozott a platformnak, köztük a másolás és beillesztés funkció, tovább fejlesztett többszálú munkavégzés és számos hibajavítás az operációs rendszer számára.

2011. február 11-én Londonban Steve Ballmer, a Microsoft akkori vezérigazgatója, és Stephen Elop, a Nokia akkori vezérigazgatója bejelentették vállalataik közös együttműködését, mely során Windows Phone vált a Nokia készülékek első számú operációs rendszerévé lecserélve a Symbiant. Ezáltal Nokia a Windows Phone első számú támogatójává vált és meghozta azt a löketet a platformnak, amire szüksége volt.

Egy évvel a Windows Phone 7 leleplezése után megérkezett az első jelentős frissítés, a Windows Phone 7.5. Ez a frissítés a *Mango* nevet kapta és 500 új funkcióval büszkélkedhetett. Az élő csempéket újra élesztették, *Mango* segítségével gyakrabban frissíthetik az alkalmazások csempéiket és lehetővé tették a hivatkozást az alkalmazás különböző pontjaira. A kezdőképernyő sokkal változatosabbá vált ezen módosítások során. Az alkalmazáslistában helyet kapott egy ugrólista, aminek segítségével egy betű megnyomásával, az ezzel a betűvel kezdődő nevű alkalmazásokhoz ugorhat a felhasználó. Az üzenetek alkalmazást felváltotta az üzenetek *hub*, amiben az sms, Facebook és Windows *messenger* üzenetek kerültek egy fedél alá. Új funkciókkal bővült a *Bing* kereső szolgáltatása is, zenefelismerés, QR kód szkennelés, és helyzet alapú keresés. Az alkalmazások közötti váltást jelentősen megkönnyítette a *Mango* frissítés, a vissza gomb lent tartásával könnyedén válthat a felhasználó a már futó alkalmazások között.

A platform egy évvel a megjelenés után még mindig nagyon kevés alkalmazással rendelkezett, sok ismert alkalmazásnak nem volt Windows Phone változata vagy csak egy nagyon gyenge változata volt az Android és iOS párjának. Ezzel egyetemben a Windows Phone 7.5-ből hiányzott néhány funkció, melyek hiánya okozta frusztráció hozzáadódott az alkalmazás hézaghoz és lassan elvette a kedvét azoktól is, akik pozitívan álltak a rendszerhez.

Nokia is hozzálátott a Windows Phone készülékeinek piacra dobáshoz, mint például a Nokia Lumia 800 és 900. A Tango frissítés lehetővé tette az alsó kategóriás készülékek támogatását Windows Phone operációs rendszerrel. Nokia ezt kihasználva piacra dobott három Lumia készüléket, az 510-et, 610-et és a 710-et.

Az utolsó frissítés a 7.8-as volt. A felhasználói felületet változtatták meg, hogy jobban hasonlítson a Windows Phone 8-hoz, a csempék új színeket és méreteket kaptak. Hardveres okok miatt a Windows Phone 7-es készülékek nem frissíthetők Windows Phone 8-ra.

### Windows Phone 8

A platform következő generációját 2012. október 29-én mutatták be. A Windows Phone 8 lecserélte a Windows CE alapú architektúrát Windows NT kernel alapúra, ezáltal sok közös komponenst megoszt a Windows 8-cal, közelebb hozva a két platformot egymáshoz. Ez az egyesítés azzal az árral járt, hogy a Windows Phone 7-ről nem lehetett 8-ra frissíteni.



3.2. ábra Windows Phone 8-as készülékek, balról jobbra: Nokia Lumia 920, Nokia Lumia 820, Nokia Lumia 620

Windows Phone 8 számos módon korszerűsítette a platformot. Elsők között hardveres újítások kaptak helyet. Lehetővé tette a készülékek bővítését SD kártyákkal, az operációs rendszer támogatja az NFC, HD kijelzők és 16 GB-nál nagyobb tárhelyek használatát. Ezen változtatások segítségével a készülékek gyártói a legújabb hardveres specifikációkkal dobhatták piacra termékeiket.

Szoftveres szempontból a legszembetűnőbb változás a kezdőképernyőn volt látható. Több testre szabhatási lehetőség állt rendelkezésre. 20 féle szín közül lehetett választani. Windows Phone 8 lehetővé tette az élő csempék méretének megváltoztatását. Három méret közül választhat a felhasználó, a legkisebb csempe méret egy ikonnak felelt meg, a közepes méretű csempékre kevés információt, míg a legnagyobb, avagy széles csempére több információt lehet elhelyezni.

Microsoft kibővítette *Xbox* szolgáltatásait, most már nem csak a játékok *hub*ban található meg, hanem a Zenék és Videók *hub*ban is. Utóbbi váltotta fel a Zune-t, egyesek örömére, mások bosszúságára.

Családi funkciók is kerültek bevezetésre, köztük az úgy nevezett Gyerek sarok. Ez egy a telefon többi részétől elzárt részt képez az operációs rendszerben, ahol a telefon gazdája beállíthatja, hogy milyen alkalmazásokhoz, milyen zenékhez, videókhoz lehet hozzá férni innen a Gyerek sarokból. Ezáltal megóvva a gyermeket az ártalmas tartalmaktól és kizárja a lehetőséget, hogy a gyermek átállítsa a telefon beállításait, vagy nem kívánt hívást kezdeményezzen a telefonon.

Windows Phone 8 beépített pénztárca alkalmazással is rendelkezik, a felhasználó megadhat különféle bank és hitel kártyáinak adatait. Így az alkalmazásboltban az itt megadott kártyákkal tud fizetni.

Az operációs rendszer alapértelmezett internetes böngészője az Internet Explorer 10 lett, ami sajnos sok alapvető funkció nélkül érkezett a rendszerre. Apró újításokat kapott az Office hub és javult a OneDrive integráció is. Lehetőség van térképek letöltésére, hogy kapcsolat nélküli módban is használhatóak legyenek.

Minden előnyével és hátrányával együtt az operációs rendszer még mindig zárt volt. Ennek egyik hátránya az volt, hogy a felhasználók képtelenek voltak megoldani a tárhely menedzsment hibáját. Használat során az átmenetileg tárolt adatok nem törlődtek a telefonról, lassan megtelítve annak belső tárhelyét, míg nem szinte használhatatlanná téve a készüléket. Csak is a telefon visszaállítása gyári beállításokra oldotta meg a problémát, ilyenkor minden felhasználói adat törlődik a készülékről. Ez a hiba nem volt annyira súlyos a 16 GB vagy annál nagyobb belső tárhellyel rendelkező készülékeknél. A hiba egészen a 8.1-es frissítésig fent állt.

Windows Phone 8 hasonló gondokkal küszködött, mint elődje. A motorház alatt ugyan nagy változások mentek végbe, de a felhasználói élmény alapvetően nem változott meg. Még mindig nem volt értesítési központ, nem volt támogatás videó fájlok szerkesztésére. Az alkalmazások hiánya itt is ugyan úgy érezhető volt, mint elődjénél.

Csak négy készülékgyártó képviselte magát Windows Phone 8 fronton, köztük elsőként a Nokia, őt követte a HTC, a Samsung és a Huawei. Nokia készülékek teljességgel dominálták a piacot Windows Phone berkekben, oly annyira, hogy a Lumia és a Windows Phone név szinte összeforrt. Az AdDuplex 2014. október 22-ei felmérése alapján a Windows Phone 8-as készülékek 94,99%-ka Nokia [1].

### Windows Phone 8 frissítései

Windows Phone 8-ra érkező frissítésekre *General Distribution Release*-ként (GDR) hivatkozott a Microsoft. Az első frissítés, a GDR 1, a *Portico* nevet kapta és 2012 decemberében jelent meg. Apró fejlesztéseket és hibajavításokat tartalmazott.

A GDR 2-es frissítés 2013 júliusától kezdve érkeztek ki a telefonokra. Apró újdonságok és hibajavítások mellett ebbe a frissítésbe került bele az FM rádió támogatás. Új alapértelmezett kamera alkalmazás beállításának lehetőségét is elhozta ez a frissítés. *CalDAV* és *CardDAV* támogatást is kapott a rendszer, amik egyes Google szolgáltatások eléréséhez szükségesek. Az úgy nevezett *Data* *Sense* funkció segítségével a felhasználó felügyelhette az adatkapcsolat során felhasznált adatmennyiséget, havi korlátot állíthat be magának és akár a háttérben történő adatcserét is letilthatta.

2013. október 14-én adta ki a GDR 3-as frissítést a Microsoft. Ez volt a harmadik és egyben az utolsó frissítés Windows Phone 8-ra. Új hardveres támogatást hozott az operációs rendszernek, amivel elérhetővé vált a 1080p felbontású kijelzők és négy magos processzorok használata. A nagyobb kijelzővel rendelkező készülékek képesek lettek egy harmadik élő csempe oszlop megjelenítésére is. Szoftveres újdonságok között volt az új autós üzemmód, képernyőtartalom tájolásának rögzítése, és a lehetőség, hogy bezárjunk alkalmazásokat az alkalmazás váltó felületen. Ez volt az első frissítés, ami részt vett a *Developer* *Preview* programban. Ennek a programnak a keretében, a regisztrált Windows Phone fejlesztők hamarabb letölthették az új frissítést, mint az átlagos felhasználók.

### Windows Phone 8.1

Windows Phone 8.1-et 2014. április 2-án jelentették be, amihez a fejlesztők két héttel később hozzá is férhettek a *Developer* *Preview* program keretében. Rengeteg újdonsággal bővült az operációs rendszer. Ez az új operációs rendszer nagyobb lépést jelentett egy átlagos felhasználó számára, mint a Windows Phone 8. A felhasználói felület maradt a régi, de új beállításokkal még testre szabhatóbbá tették. Minden készüléken elérhetővé vált a harmadik csempe oszlop. A 8.1 lehetővé tette a csempék színének átlátszóra állítását, és egy háttérkép elhelyezését a csempék mögé. A rendszer alkalmazások betűinek méretét is lecsökkentették, így több hely maradt tartalmak megjelenítésére. Az operációs rendszer megkapta az új műveletközpontját is, ahol a felhasználó megtekinthette a kapott értesítéseket és négy darab gyorsműveletet közül választhat, amik segítségével be és ki kapcsolhatja az adott funkciót.



3.3. ábra Nokia Lumia 930 Windows Phone 8.1-el

A 8.1 az Internet Explorer 11-es verziójával érkezett, ami az előző változat hiányosságainak nagy részét bepótolta. Megszűnt a böngésző fülek limitje, WebGL és új HTML 5-ös videó lejátszó támogatást kapott és bekerült az inkognitó böngészési mód is. Az operációs rendszer képes lett szinkronizálni a munkamenetet a telefon és egy Windows 8-as számítógép között. Rendszer beállítások szinkronizálása is lehetővé vált, például azonos téma szín használata.

Egyik legnagyobb újdonság Cortana, a Windows Phone 8.1 digitális személyi asszisztense volt. Képes kezelni a felhasználó emlékeztetőit, naptár időpontjait sok egyéb hasznos funkció mellett. A készülék kereső gombjáról lehet elérni, ahol a felhasználó beszéd útját kommunikálhat vele. A készülék használata során Cortana megtanulja, hol dolgozik a felhasználó és milyen útvonalakon szokott mozogni. Ezen információk és a forgalmi jelentések alapján értesítheti a felhasználót, hogy időben induljon el a munkahelyére. Windows Phone 8.1 egyik legnagyobb dobása lett, azonban csak az Egyesült Államok területén volt elérhető.

A felhasználók nagyobb ellenőrzést kaptak készülékük felett a tárhely, akkumulátor, és a WiFi segéd alkalmazások segítségével. A fejlesztők számára is új API-k állnak rendelkezésre, így elérhetővé vált a videó szerkesztő alkalmazások készítése is.

Változás történt a *hub*ok terén, a Zenék és Videók *hub*ot kettéválasztották, és bolti alkalmazásként kerültek az operációs rendszerbe. Ennek egyik előnye az volt, hogy könnyebben frissíthetővé váltak, nem kellett a felhasználóknak megvárniuk az egyes frissítéseket az operációs rendszerhez.

Windows Phone 8.1 elhozta a legkívántabb funkciókat a Windows Phone-ok számára, javította az egész rendszer használatát és kinézetét. Az operációs rendszer ezen verziója minden Windows Phone 8-as készülék számára is elérhető és ingyenesen letölthető.

A Microsoft 2013 szeptemberében bejelentette, hogy felvásárolja a Nokia készülék és szolgáltatások ágát. Az egyesülés 2014 áprilisában véglegesült, Nokia készülék ága azóta a Microsoft irányítása alatt ál és Microsoft Mobile nevet viseli.

Bemutatásra kerültek az első Windows Phone 8.1-el érkező készülékek is, a Nokia Lumia 930 és a Nokia Lumia 630. De nem csak Nokia, hanem a HTC is megjelentette első Windows Phone 8.1-es mobiltelefonját a HTC One M8-at. Rajtuk kívül a következő gyártók képviseltetik magukat jelenleg a Windows Phone 8.1-es készülékek piacán: Gionee, JSR, Karbonn, LG, Lenovo, Longcheer, Micromax, Samsung, Alcatel, Lava, Blu és a ZTE.

### Windows Phone 8.1-es frissítések

8.1 megjelenése után nem sokkal már érkezett is a GDR 1-es frissítés, mely elérhetővé tette Cortana-át az Egyesült Királyságban, Kanadában, Ausztráliában, Kínában, és Indiában. Ezen kívül az elő csempéket gyűjtő mappákba lehetett szervezni.

2014. október 30-án Microsoft bejelentette Windows operációs rendszerének legújabb tagját a Windows 10-et. Állításuk szerinte ez a Windows 10 fog futni minden Microsoft eszközön, beleértve a Windows Phone-okat is. Microsoft megerősítette, hogy minden Windows Phone 8-as, illetve 8.1-es készüléket fel lehet majd frissíteni 10-re [15].

## Fejlesztés Windows Phone-ra

Ebben a fejezetben arról lesz szó, hogy mik azok a feltételek, amikre szükségünk van ahhoz, hogy elkezdhessünk Windows Phone alkalmazásokat fejleszteni és az alkalmazás boltba feltölteni. Le fogom írni, hogy milyen alkalmazás fajtákból lehet választani, és milyen sablonok állnak rendelkezésre a fejlesztés elkezdéséhez.

### Előfeltételek

Hogy elkezdhessünk alkalmazásokat fejleszteni Windows Phone-ra, bizonyos feltételeknek meg kell felelnünk. Az első ilyen feltétel, hogy rendelkeznünk kell egy Microsoft fiókkal. Ezt ingyenesen bárki regisztrálhat magának a Microsoft web oldalán. Ez azért szükséges, hogy beregisztrálhassunk fejlesztőként és meg kapjuk a fejlesztői licencet.

Már fizetés nélkül is elkezdhetünk alkalmazásokat fejleszteni Windows Phone-ra. Hozzáférhetünk ingyenes fejlesztői eszközökhöz, köztük a Windows Phone emulátorhoz, amin tesztelhetjük az alkalmazást. Ingyenesen rátölthetjük a készülő alkalmazást saját Windows Phone készülékünkre további tesztelés céljából. Ezeken felül rengeteg forráshoz és tudásanyaghoz juthatunk hozzá ingyenesen a Microsoft jóvoltából.

Azonban nem tölthetjük fel az elkészült alkalmazást a Windows Phone Store-ba. Ehhez regisztrálnunk kell magunkat fejlesztőként a Windows Developer Centerben a dev.windows.com weboldalon. A Microsoft két féle fejlesztői fiókot különböztet meg, az egyéni fiókot és a vállalati fiókot. A fiókok regisztrálásáért egyszer kell fizetni, az egyéni fiók 19 $-ba, míg a vállalati fiók 99 $-ba kerül. A regisztráció során meg kell adni a szükséges kapcsolati információkat, például név, cím, telefonszám, e-mail cím stb., ezután pedig azt a nevet, amin az alkalmazás publikálásra kerül (Publisher display name). A fejlesztői fiókért bankkártyával lehet fizetni, de van lehetőség promóciós kód beváltására is.

Regisztráció befejezése után a fiók összegző oldalon további beállításokra van mód. Ahhoz, hogy a felhasználók megvásárolhassák az alkalmazást, egy fizetési és egy adózási profil kitöltése szükséges.

A fejlesztői fiókon kívül bizonyos hardveres és szoftveres követelményeknek is meg kell felelni. Windows Phone alkalmazásokat csak is Windows 8.1 operációs rendszert futtató számítógépeken lehet fejleszteni. A szükséges fejlesztő környezet pedig a Visual Studio 2013. Ennek ingyenes változata a Visual Studio Express 2013 for Windows.

### Alkalmazás fajták

Microsoft több féle alkalmazás típust vezetett be az évek során, melyek különböznek platform, illetve felhasznált keretrendszerek terén. Windows Phone 8.1-re szánt alkalmazások többsége Windows Runtime alapú. Windows Runtime alkalmazásoknak két fajtája van, Windows Store alkalmazások és a Windows Phone Store alkalmazások. Windows Store alkalmazásokat PC-ékre, tabletekre és laptopokra készülnek. Windows Phone Store alkalmazások pedig Windows Phone-ra. Windows Phone 8.1-re fejleszthetünk Silverlight UI keretrendszer alapú alkalmazásokat is, ez az opció főleg a meglévő Windows Phone 8-as Silverlight alkalmazásoknak kedvez.

### Alkalmazás sablonok

Új Windows Phone alkalmazás fejlesztésekor fontos kiválasztani a megfelelő alkalmazás sablont. Visual Studio több féle alkalmazás sablont kínál a fejlesztőnek.

Elsőként az univerzális alkalmazás sablont említeném. Két változata van, az első a Blank, azaz üres, a második a Hub alkalmazás sablon. Univerzális alkalmazások fejlesztése során két alkalmazás készül, egy telefonra és egy Windows 8.1-es készülékekre. A két alkalmazás megosztott kódot használ, ami által a fejlesztési idő nagymértékben lecsökkenhet. Blank sablon használatával az egész alkalmazást az alapjaitól nekünk kell felépíteni, egy üres oldalt foglal csak magában. A Hub sablonban már sokkal több mindent találhatunk. A Hub vezérlőt használja, ami egy legalább három oldalból álló felület, amik között balra és jobbra pöccintés segítségével lehet navigálni. A sablonhoz több segédosztály is tartozik, amik az oldalak közötti navigációt és az alkalmazás állapotának mentését segítik. Ezek az osztályok nem képezik részét a Windows Phone SDK-nak.

Windows Phone Runtime alkalmazás fejlesztésére négy féle alkalmazás sablon áll rendelkezésre. Itt is megtalálható a Blank és a Hub sablon, de a lista kibővül a Pivot és a Webview sablonokkal. A Pivot sablon a Hubhoz hasonló, de ez a Pivot vezérlőt használja és kinézetben is eltér a Hubtól. A Pivotban található oldalak között a felhasználó itt is balra és jobbra pöccintéssel navigálhat. A WebView sablonban egy WebView vezérlőt tartalmazó oldal van, ami weboldalak megjelenítésére képes. A Hub és a Pivot sablonokhoz is tartoznak segédosztályok, mint az Univerzális Hub sablon esetében.

### Támogatott programozási nyelvek

A fejlesztő több támogatott program nyelv közül is választhat Windows Phone alkalmazás készítésekor. Támogatott programozás nyelvek: C++, C#, Microsoft Visual Basic, és JavaScript. JavaScript használatakor a felhasználói felületet HTML5 használatával lehet elkészíteni, míg a többi nyelv használatakor Extensible Application Markup Language (XAML) segítségével lehet.

# Feladat specifikáció

Szakdolgozatom elkészítése során, egy olyan Windows Phone 8.1-es kliens alkalmazás tervezése és implementálása volt, aminek segítségével az orvosok nyomon tudják követni a betegeiknek állapotát. A betegek rögzíthetik vérnyomás és vércukor értékeiket, amit a rendszer eltárol. Az itt eltárolt információk megkönnyítik az orvos munkáját. Az alkalmazás főcélja, hogy elősegítse a páciensek gyógyításhoz és gyógyuláshoz szükséges diagnosztikai és terápiás információk gyors célba jutását. Az alkalmazás használatával csökkenthető és optimalizálható a munkaterhelés, aminek köszönhetően nagymértékben elősegítheti a kezelések eredményességét, csökkentheti az ellátás költségét és javíthatja az orvosi munka minőségét.

A kliensalkalmazás egy nagyobb rendszerhez csatlakozik, ahol az adatok egy adatbázisban vannak tárolva. A kliens a háttérrendszerben (backend) található web szerveren keresztül képes kommunikálni az adatbázissal. Ebben a fejezetben leírom, hogy milyen funkciókat kell tartalmaznia a készterméknek és milyen elvárásoknak kell megfelelnie.



## Rendszer követelmények

Az alkalmazásnak futnia kell minden Windows Phone 8.1 operációs rendszerrel rendelkező készüléken. Az 512 MB RAM-mal rendelkező Windows Phone 8.1-es telefonokon is működnie kell. Az alkalmazás használatához élő internet kapcsolat szükséges.

## Funkcionális követelmények

Az alkalmazás igénybevételéhez bejelentkezés szükséges. Bejelentkezéskor a felhasználó megadja felhasználó nevét, ami egy e-mail cím, és jelszavát. Legyen lehetőség a felhasználó név elmentésére a bejelentkezés megkönnyítéséhez. Sikertelen bejelentkezés esetét hibaüzenettel kell jelezni a felhasználónak.

Lehessen új fiókot regisztrálni. Az orvosok és betegek külön felhasználói típusba tartoznak, ezért a regisztrációkor ennek megfelelően kell eljárni. Új orvos regisztrálása során meg kell a felhasználónak adnia egy e-mail címet, jelszót, a jelszó megerősítését, teljes nevét, praxisát, telefon számát, és az orvosi rendelő címét. Új páciens regisztrálásakor meg kell adni egy e-mail címet, jelszót, a jelszó megerősítését, teljes nevet, születési dátumot, és a nemet. A telefonszám megadása opcionális. Helytelenül megadott adatok esetén visszajelzést kell biztosítani a felhasználó felé. Sikeres regisztrációt közölni kell a felhasználóval és továbbítani a bejelentkező felülethez.

Sikeres bejelentkezés után továbbítsa a felhasználót az alkalmazás a saját felületéhez. Orvosoknak és pácienseknek külön felületük van. Az orvos legyen képes megtekinteni a saját betegeit, azoknak adatait és rögzített értékeit. Az alkalmazás jelenítse meg grafikonon is az adott páciens rögzített értékeit. Legyen képes megnézni a függőben levő pácienseket, ahol eldöntheti, hogy kit fogad pácienséül és kit utasít el.

Jelezze az alkalmazás valamilyen módon az esetleg veszélyben levő betegeket. Azokat a betegeket tekintjük veszélyben, akik a normális érték tartományon kívüli értéket rögzítettek. A vérnyomás értékek besorolását az alábbi táblázat alapján kell megtenni.

|  |  |  |
| --- | --- | --- |
| **Kategória** | **Szisztolés (Hgmm)** | **Diasztolés (Hgmm)** |
| **Normál** | 90-119 | 60-79 |
| **Prehipertónia** | 120-139 | 80-89 |
| **1. fokozató hipertónia** | 140-159 | 90-99 |
| **2. fokozatú hipertónia** | 160-179 | 100-109 |
| **Hipertóniás krízis** |  |  |

4.1. táblázat Vérnyomás értékek kategorizálása

Diabéteszes betegek esetén a vércukor mérés eredménye étkezés előtt nem lehet nagyobb, mint 7 mmol/L. Étkezés utáni mérés nem lehet magasabb, mint 8 mmol/L. Ha mégis ezeknél az értékeknél nagyobbat mért, akkor a veszélyben levő betegek közé kell sorolni.

Az orvosok képesek űrlapokat küldeni a kiválasztott betegének. Az űrlap sablonokat az alkalmazás a szerverről kéri le, ezekből választhat az orvos. A visszakapott űrlapokat is meg kell tudni jeleníteni, az orvos az űrlap megnézése után törölheti azt. Az űrlapoknak az egyszerű szöveges megjelenítésen és beviteli mezőkön kívül támogatnia kell a legördülő listákat, rádió gombokat és jelölőnégyzeteket is.

A páciens felületen a beteg tekinthesse meg az orvosa adatait. Ha még nincsen kiválasztott orvosa, akkor az alkalmazás ajánlja fel az orvos választás lehetőségét. A beteg a regisztrált orvosok listájából választhat magának orvost. Ekkor a kiválasztott orvos értesítést kap arról, hogy új felkérése van. A beteg értesítést kap arról, hogy az orvos elfogadta vagy visszautasította a kérést.

A páciens rögzíthesse vérnyomását, vércukrát és tekinthesse meg a rögzített értékeket. Az alkalmazás jelenítse meg grafikonon is ezeket az értékeket. Szintén jelenítse meg az orvostól kapott űrlapokat, az űrlapok kitöltése után lehessen visszaküldeni őket.

A páciensek elhelyezhetnek egy pánik gombot második csempeként a kezdőképernyőn az alkalmazásból. Ez egy pánik gombként funkcionál, ami egy olyan felületre viszi a beteget, ahonnan gyorsan indíthat segélyhívást.

Az orvosok és páciensek számára is biztosítani kell, hogy megnézhessék és módosíthassák személyes adataikat. Lehessen cserélni a felhasználó jelszavát is.

Az alkalmazásnak legyen magyar és angol nyelvű változata is. Lehessen az alkalmazáson belül nyelvet váltani.

A kliensalkalmazás HTTP kapcsolaton keresztül kommunikáljon a szerverrel GET és POST kérésekkel. Az átvitt adatok JSON formátumban legyenek.

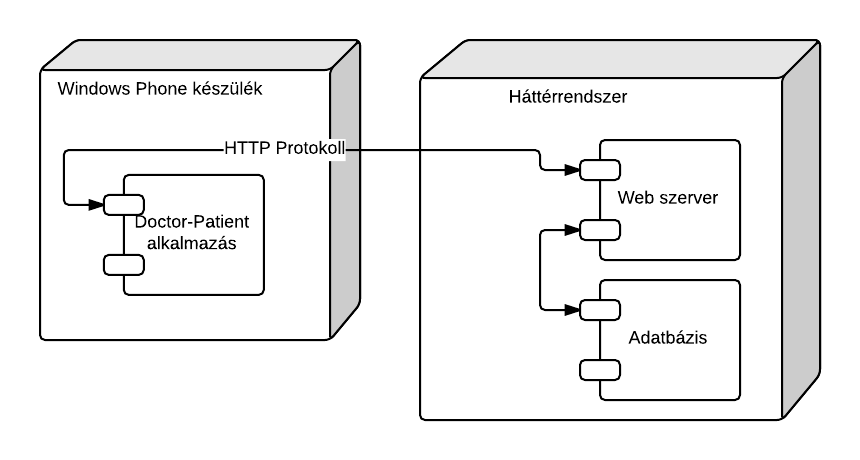
# Tervezés

Ebben a fejezetben az alkalmazás tervezését fogom részletezni. Szó lesz az alkalmazás mögött levő szerverről, be fogom mutatni, hogyan terveztem meg az alkalmazás felépítését és kiemelek néhány felület megtervezését.



## Teszt szerver

A feladatspecifikációban vázolt feltételek alapján az alkalmazás megfelelő működéséhez szükség volt egy szerverre és egy ahhoz kapcsolódó adatbázisra. Az alkalmazás tervezésekor ezt figyelembe kellett vennem. Azért el kellett készítenem egy teszt szervert, aminek segítségével működhetnek a szerver specifikus funkciók az alkalmazásban. Az alkalmazás és a szerver felépítését és a köztük levő kapcsolatot az 5.1-es ábrán látható komponens diagram mutatja be.



5.1. ábra Rendszer komponens diagramja

A teszt szerver felé vezető első lépés a szerver technológia kiválasztása volt. Az elsődleges szempont az egyszerűség és könnyen elkészíthetőség volt, mivel a szakdolgozatom témája magáról az alkalmazás elkészítéséről szól, nem pedig a mögötte levő szerverről. De fontosnak tartom, hogy szót ejtsek a teszt szerverről. A szerver oldal programozási nyelve PHP lett. Ez mellé választottam a Slim nevezetű mikro keretrendszert, aminek segítségével gyorsan lehet egyszerű, de ugyanakkor erőteljes web alkalmazásokat fejleszteni. A Slim keretrendszer fontos szerepet játszik a HTTP REST kérések feldolgozásában.

A technológia kiválasztása utána a következő lépes az alkalmazás működéséhez szükséges kérések és a ráadott válaszok meghatározása volt. Hogy ezt a feladatot teljesíteni lehessen egy erre alkalmas adatbázist kellett felvázolnom.

## Adatbázis

A PHP szerver mellé a legkézenfekvőbb megoldás a MySQL adatbázis kezelő szerver választása volt. A MySQL Workbench program segítségével terveztem meg a táblákat és a köztük levő kapcsolatokat. Majd innen SQL kódként exportáltam ki az adatbázist, amit a teszt szerver phpMyAdmin felületén keresztül importáltam.

### Az adatbázis felépítése

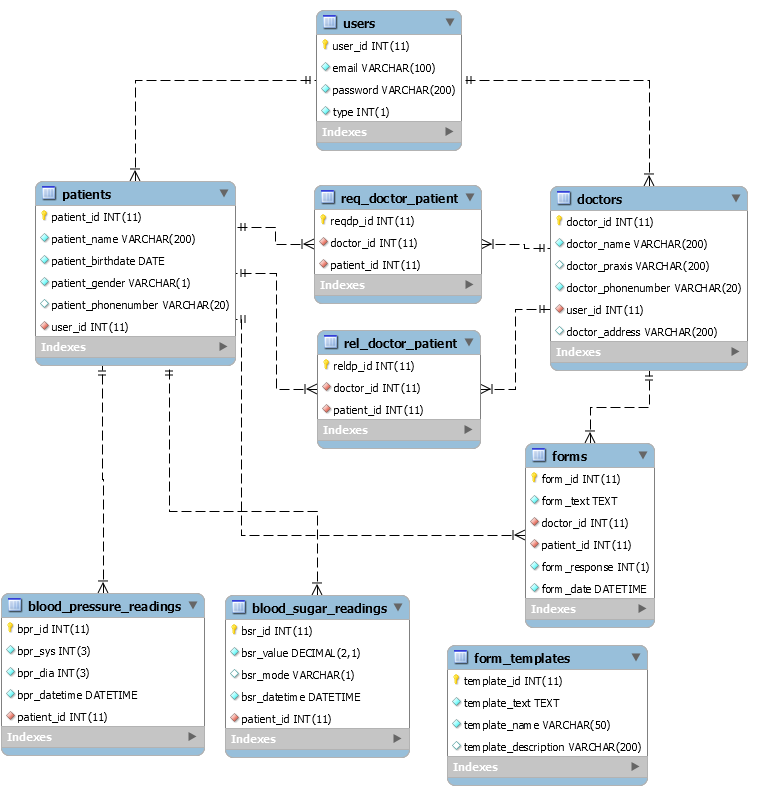
Az adatbázisban kilenc tábla lett definiálva. Az 5.2-es ábrán láthatók a táblák és a köztük fent álló kapcsolatok. Egy éles rendszernek ennél bonyolultabb és összetettebb adatbázisra lenne szüksége, de az én célomnak ez teljességgel megfelelt.

A *users* táblát hoztam létre elsőként. Ez a tábla tartalmazza a felhasználói adatokat, az e-mail címet, jelszót és a fiók típusát. A fiók típusa orvos esetén a 0, beteg esetén az 1-es értéket kapja.

A két felhasználói típust a *doctors* és a *patients* táblákban fejtettem ki. A *doctors* táblában tárolom az orvos teljes nevét, praxisát, telefon számát, és a rendelőjének címét. A *patients* táblában a páciens teljes nevét, születési idejét, nemét, és telefon számát tárolom.

Az orvosok és betegek egymáshoz rendelése a *req\_doctor\_patient* és a *rel\_doctor\_patient* tábla feladata. Felépítésben identikusak, de feladatuk eltérő. A *req\_doctor\_patient* az orvos felé intézett felkérést tárolja, míg a *rel\_doctor\_patient* a már létrejött kapcsolat tárolásáért felelős.

A felhasználók által rögzített értékeket is tárolnom kellett, a vércukor és vérnyomás értékek tárolására két külön táblát készítettem. A vércukor értékeket a *blood\_sugar\_readings* táblába kerülnek. Magán az értéken kívül elmentem a rögzítés pontos időpontját és természetesen, hogy melyik páciensé a mérés. A *blood\_pressure\_readings* táblában kaptak helyet a vérnyomásmérések. Felépítése hasonló az előző táblához, de itt két értéket kell menteni, a szisztolés és diasztolés vérnyomás értékeket.



5.2. ábra Adatbázis terv

Az űrlapok általi kommunikáció alapját a *forms* és a *form\_templates* táblák adják. A *form\_templates* táblában előre elkészített űrlap sablonok vannak, amiket az orvos böngészhet, majd ezekből választhat egyet elküldésre egy betegének. Az elküldött űrlapot tárolom a *form* táblában. Rögzítem, hogy melyik orvos és melyik beteg között van a kommunikáció. Továbbá küldés időpontját is elmentem benne. A *form\_response* oszlop értéke határozza meg a küldés irányát. Ha az értéke 0, akkor az orvostól a beteg felé, ha 1, akkor fordítva. Ez az oszlop azért kell, hogy lehessen szűrni kimenő és bejövő űrlapokat.

## Alkalmazás

A Microsoft Windows Phone alkalmazás tervező csapatának tervezési folyamata öt elkülöníthető szakaszból áll: **koncepció**, **struktúra**, **interakció**, **vizualitás**, és **prototípus**. Ezek a szakaszok ösztönzik nagyszerű alkalmazások elkészítését. Az alkalmazásom tervezése során próbáltam követni Microsoft által kijelölt utat. A feladatspecifikáció elkészítésével a koncepció fázis nagy részét teljesítettem. Az elkövetkező fejezetekben leírom, hogyan mentem végig ezeken a tervezési szakaszokon.

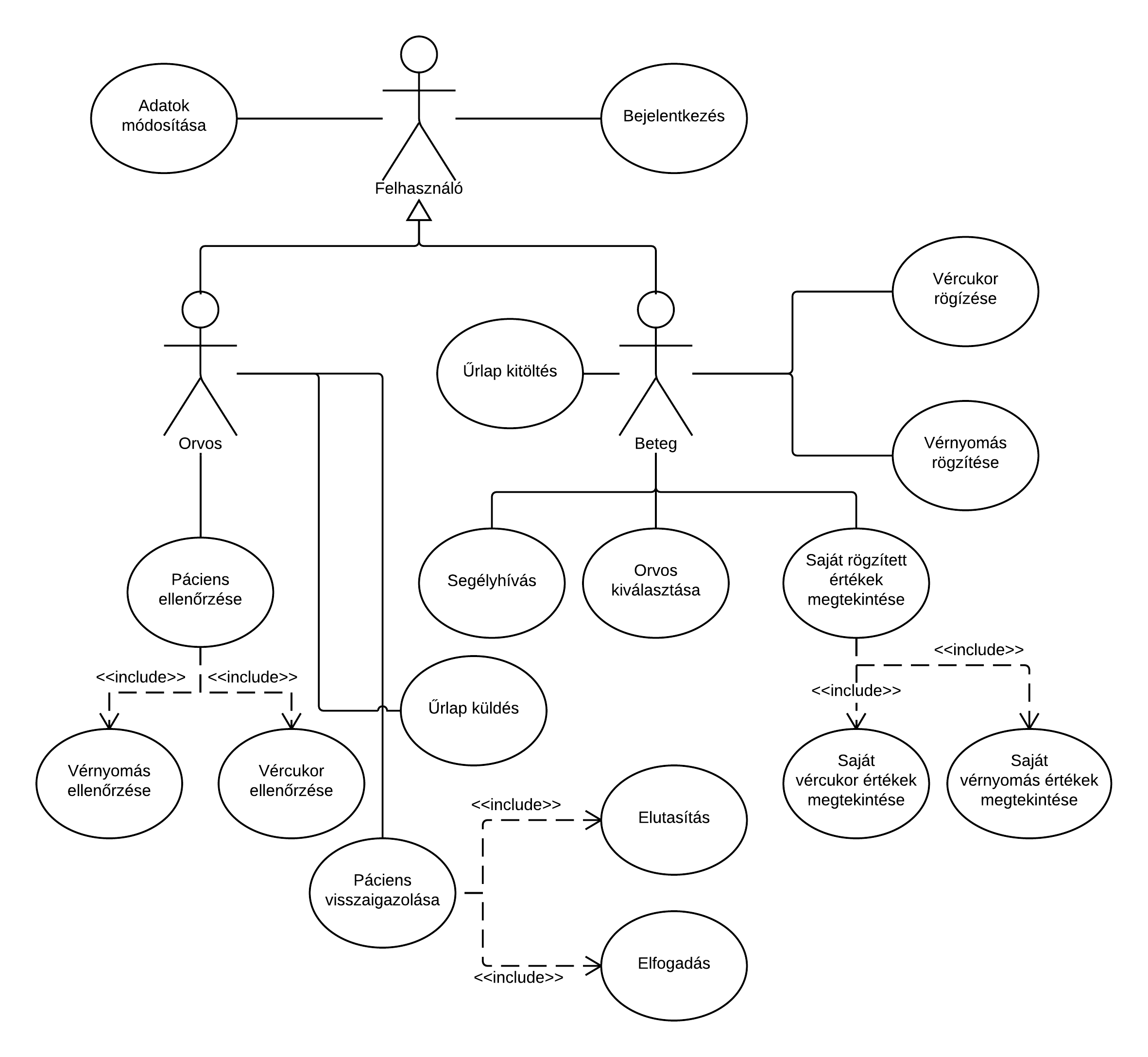
### Használati esetek

A koncepció fázisban definiálnom kellett, hogy pontosan milyen használati esetekre van szüksége az alkalmazásomnak. A tervezés kezdetén rögzítettem a felhasználók által végrehajtható legfontosabb funkciókat. Ezek a használati esetek egy biztos alapot szolgáltattak a tervezés további szakaszai és az implementálás során. A legfontosabb használati eseteket összesíti az 5.3-as ábra.

A felhasználók orvosokra és páciensekre vannak osztva. Az ábrán látható, hogy mindkét csoport rendelkezik a bejelentkezés és adat módosítás képességével. Adatmódosításkor azokat a személyes adatokat módosíthatja a felhasználó, amikkel regisztrálta magát. A bejelentkezéshez használatos jelszó cseréjére is van lehetőség.

Az orvos legfőbb képessége, hogy ellenőrizheti saját betegeit, azok adatait, mint a például a vérnyomás és vércukor értékeket. Válaszolhat betegek felkéréseire, amiket vagy elfogad, vagy elutasít. Ezeken felül űrlapokat küldhet a betegeinek.

A páciens elsődleges feladata, hogy rögzítse vérnyomás vagy vércukor értékét, esetleg mindkettőt. Saját maga által rögzített értékeit megtekintheti és ő választhatja ki, hogy ki legyen az orvosa. Az orvosától kapott űrlapokat kitöltheti és visszaküldheti. Ezeken felül a pánik gomb segítségével képes segélyhívást kezdeményezni.



5.3. ábra Használtai eset diagram

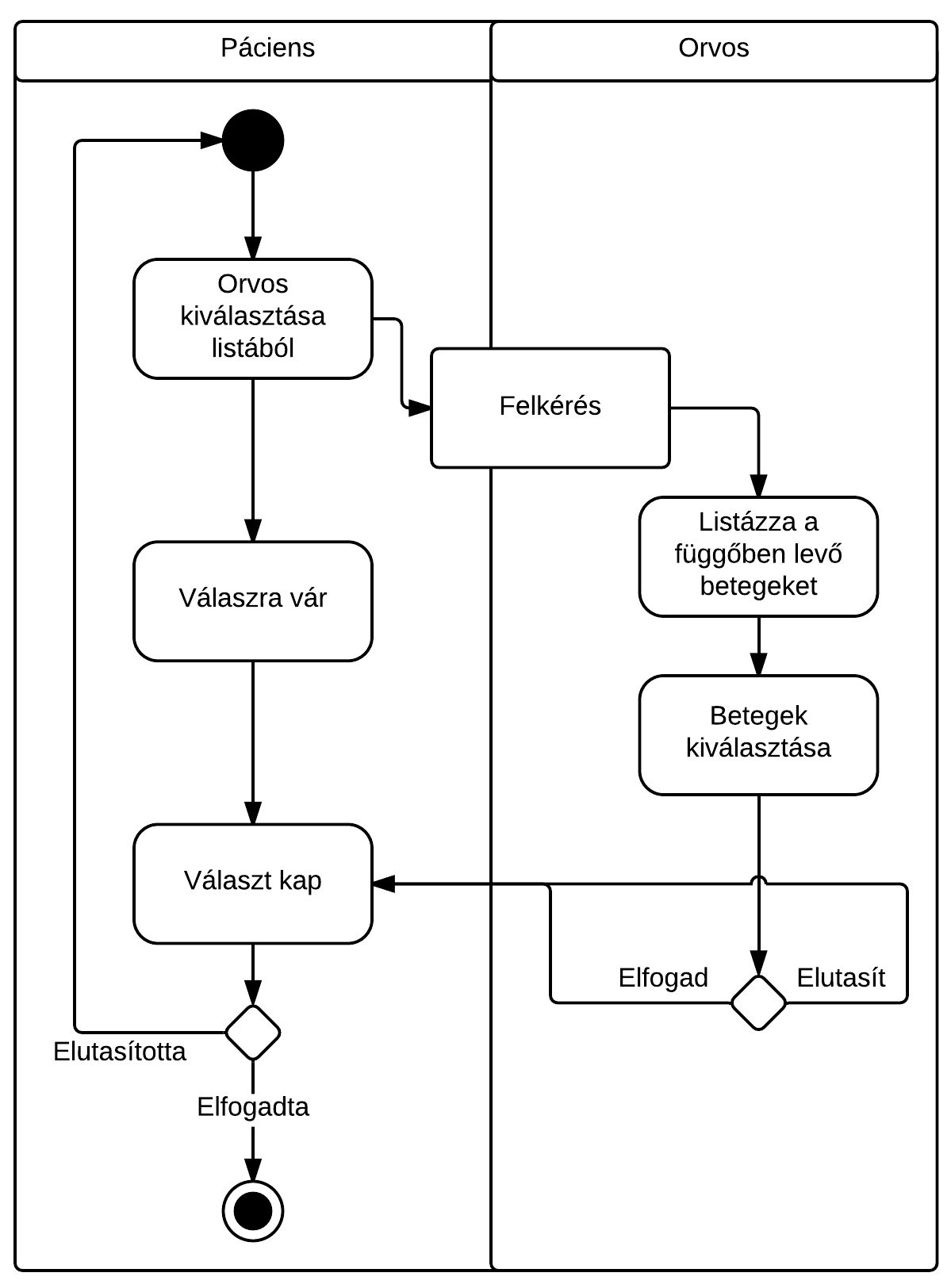
### Folyamatok

Az alkalmazásban található kritikusabb folyamatokat érdemes előre átgondolni és meghatározni a részleteiket. Ebben az alfejezetben bemutatok pár folyamatot, amivel az alkalmazás használata során találkozhat a felhasználó. Elsőként a bejelentkezés lépéseiről lesz szó.

A bejelentkezés folyamata a felhasználó név és jelszó beírásával kezdődik. A felhasználónak lehetősége van megjegyeztetni a felhasználó nevét, hogy a következő bejelentkezéskor ne kelljen ismét beírnia. A felhasználó egy gomb megnyomásával megpróbálhat bejelentkezni. A gomb lenyomása után az alkalmazás ellenőrzi, hogy a két mező megfelelően van-e kitöltve, ha valamelyik mező üres, akkor arról hiba üzenetben értesül a felhasználó. Hibaesetén az üzenetnek megfelelően kell eljárnia a felhasználónak, azaz ki kell tölteni-e a hiányzó mezőt. Ha minden rendben van, akkor az alkalmazás elküldi a felhasználó nevet és jelszót a szervernek, ahol ellenőrzésre kerül helyességük. A szerver fogadja a felhasználó nevet és jelszót az alkalmazásból és ellenőrzi, hogy van-e ilyen nevű felhasználó regisztrálva az adott jelszó párossal. Jó felhasználó név és jelszó esetén visszaküldi az alkalmazás a felhasználó azonosítóját és típusát. Ekkor az alkalmazás tovább engedi a felhasználót, és ha a felhasználó úgy döntött, hogy megjegyezteti a felhasználó nevét, akkor az alkalmazás elmenti azt. Ha rossz felhasználó nevet vagy jelszót adott meg a felhasználó, akkor a szerver válaszban közli az alkalmazással, hogy a felhasználó rossz adatokat adott meg. Ezt az alkalmazás hiba üzenetként jelzi a felhasználó felé.

Az orvos választás folyamatát az 5.4-es ábra mutatja be. A bejelentkezett pácienssel közli az alkalmazás, hogy még nem választott magának orvost és felajánlja neki a választás lehetőségét. Ha a felhasználó él a lehetőséggel, akkor megjelenik egy lista, ahol fel vannak sorolva a regisztrált orvosok. Ebből a listából választhat egyet. A választás után egy felkérés kerül az orvoshoz. Az orvost értesíti az alkalmazás, hogy van egy új függőben levő betege. Az orvos megnyitja a függőben levő betegek listáját, itt kiválaszthat egy vagy több pácienst is és eldöntheti, hogy elfogadja vagy visszautasítja a felkérésüket. A választ megkapja a páciens, aki eddig a válaszra várt. Válasz tartalma dönti el, hogy van-e új orvosa, vagy kezdheti a folyamatot elölről.

Utolsó folyamatként az űrlapküldést részletezném. A folyamat első lépéseként az orvos megnyit egy űrlap sablont a sablonok listájából. Ekkor megjelenik a kiválasztott űrlap. Ha úgy dönt, hogy szeretné elküldeni valamelyik betegének, akkor egy gomb lenyomásával előhoz egy listát, ami tartalmazza a betegeit. Ebből a listából kiválaszthat egy darab beteget. A kiválasztás után az alkalmazás rákérdez, hogy biztosan ennek a betegnek szeretné elküldeni az űrlapot. Ha az orvos megerősíti szándékát, akkor az alkalmazás ezt a kérést elküldi a szerver felé, ahol a szerver megteszi a megfelelő lépéseket az űrlap kézbesítéséhez. Ha mégsem ennek a páciensnek szeretné elküldeni az űrlapot az orvos, akkor az alkalmazás megszakítja a küldés folyamatát és az orvos másik pácienst választhat. A pácienst értesíti az alkalmazás, hogy új űrlapot kapott az orvosától. A páciens megtekintheti a kapott űrlapot és kitöltheti annak megfelelően. Miután kitöltötte visszaküldheti az orvosának az űrlapot. Szerver fogadja a kérést és értesíti az orvost, hogy a páciense visszaküldte az űrlapot. Ekkor az orvos megtekintheti a kitöltött űrlapot, amit a megtekintés után törölhet.



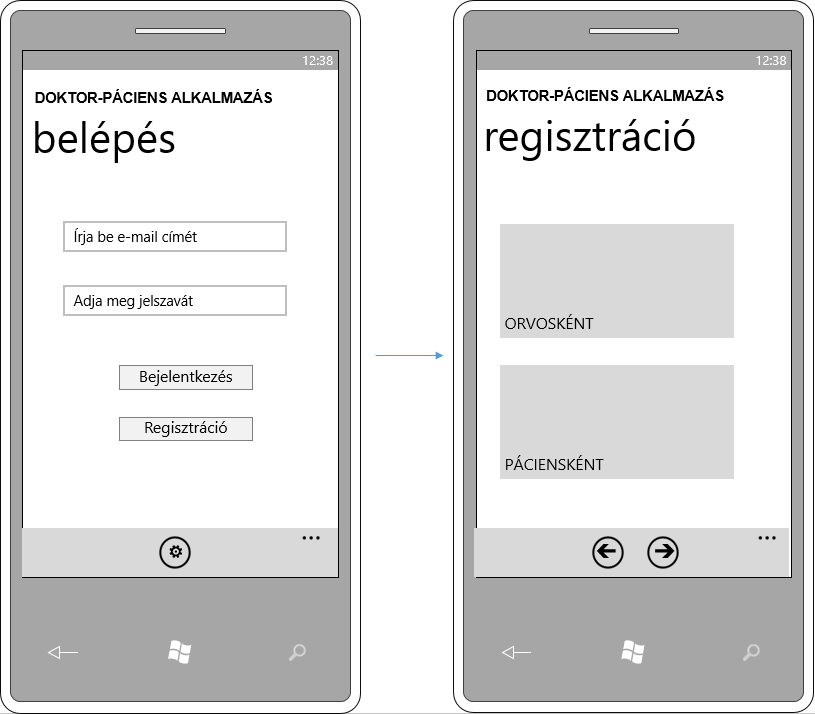
5.4. ábra Orvos választás aktivitás diagramja

### Struktúra

Miután összeállt az alkalmazás koncepciója a tervezés következő fázisába léphetett. Az alkalmazás tervrajzaként elkészítettem az alkalmazás Információs Architektúráját (IA), ami strukturálja az alkalmazás tartalmát. Segít definiálni az alkalmazás navigációs modelljét és az alkalmazás identitását. Az IA elkészítésével rendszereztem az alkalmazásban található tartalmakat és jobban átláthattam, hogy a felhasználó milyen élményben fog részesülni. Az alkalmazás IA-ja három részből áll: STORE (alkalmazás bolt), START (indítás pontja), APPLICATION (alkalmazás). Az alkalmazás részben már az egyes képernyők is körvonalazódnak. Ezek a képernyők egyelőre csak a legfontosabb szerepüket tartalmazzák, a tényleges felhasználói felület megtervezése a következő fázisban van. Az elkészült IA ábra a szakdolgozat mellékletei közt megtalálható.

### Interakció

A koncepció fázisban meghatároztam az alkalmazás célját, ebben a szakaszban ennek a célnak a végrehajtási módjáról lesz szó. Az alkalmazás és a felhasználó kommunikációja a felhasználói felület (user interface, UI) segítségével megy végbe. Ennek a tervezési fázisnak fontos része a felhasználói felületek megtervezése és a felületek között levő navigáció definiálása.

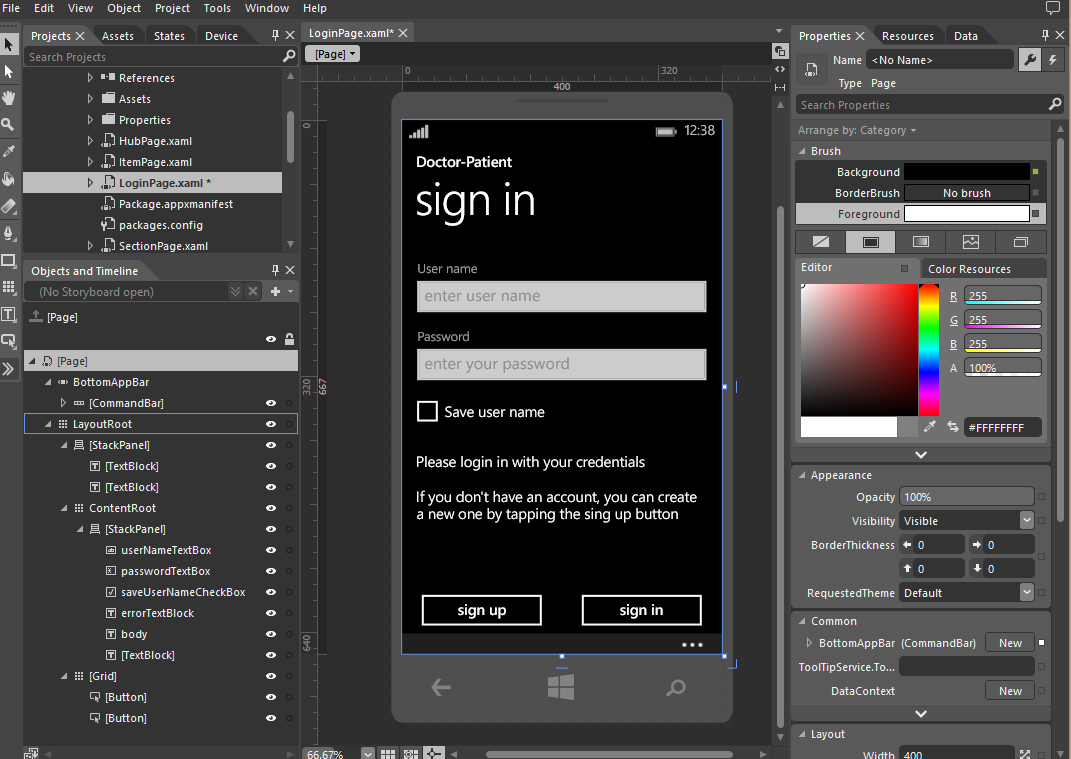


5.5. ábra Bejelentkezési és regisztrációs felületek vázlata

Felületek tervezésénél mindig két kérdés szoktam feltenni magamban. Milyen funkciókat kell tudnia a felületnek? Hogyan lehet ezeket a funkciókat a minél felhasználóbarátabb módon megvalósítani? A legegyszerűbb módja a tervezés elkezdésének vázlatok készítése papíron. Ilyenkor elég csak nagyjából betájolni az egyes elemeit a felhasználói felületnek. Az egyes elemekhez megjegyzéseket írtam, hogy emlékeztessem magamat az adott vezérlő feladatáról. Ezek a rajzok rendkívül hasznosak voltak számomra, segítségükkel sikerült jobban átlátnom az alkalmazást és megfelelő alapot szolgáltattak a tervezés későbbi szakaszában. A vázlatokat a Microsoft Office PowerPoint Storyboarding funkciójának segítségével fejlesztettem tovább. Az 5.5-ös ábrán látható a bejelentkezési és regisztrációs oldalak kezdetleges vázlata.

Windows Phone alkalmazás felületek tervezésénél fontos észben tartani a Microsoft által előírt irányelveket. Ezeknek az irányelvek betartására én is törekedtem, ezáltal az alkalmazás nem töri meg a Windows Phone operációs rendszer képi világának egységét.

A felületek tervezése során az egyik legnehezebb feladat volt az orvosok és a páciensek oldalának kialakítása. A Windows Phone-okban jól ismert hub vezérlőt vettem alapul a felületek elkészítésnél. Az orvosoknak és a pácienseknek külön hub-ot terveztem. Mindkét hub három darab oldalból áll. Ez az orvosoknál egy üdvözlő képernyőből, saját betegek oldalból, és egy űrlapok oldalból áll. Betegek hub oldala szintén egy üdvözlő képernyővel kezdődik, ezt követi a vérnyomás oldal, amit a vércukor oldal zár. Ezen oldalak között könnyedén lehet majd váltani a képernyő jobbra vagy balra pöccintésével.



5.6. ábra Bejelentkezési felület terve Blend for Visual Studio 2013-ban

A következő lépés ezeknek a vázlatoknak tovább fejlesztése volt. Ehhez a Blend for Visual Studio 2013-as programot használtam. Ennek egyik nagy előnye, hogy a felület összerakásával magát a kódot is elkezdi készíteni a szoftver és az implementálás folytathatom a Visual Studioban ennek a projektnek a felhasználásával. A bejelentkezési felület tovább fejlesztett változata látható az 5.6. ábrán.

## JSON

Az alkalmazásban több helyen is JSON formátumban levő adatokat használok. Ezek az JSON adatok egy előre definiált séma alapján épülnek fel. Két féle JSON sémát készítettem el, egyet a hálózaton keresztül küldött adatokhoz és egyet az űrlapok tárolásához.

A szerver felé küldött adatok kéréstől függően változhatnak, de minden POST kérésben van egy *params* JSON objektum. A *params* objektumon belül kaptak helyet a küldendő adatok. Új vércukor adatok rögzítésekor a következő JSON-t küldi el az alkalmazás: *{"params":{"bsr\_value”: " ", "bsr\_mode": " ", "bsr\_datetime": " ", "patient\_id": " "}}*

Az űrlapokat JSON formátumban tárolom a szerver adatbázisában. Az űrlapok kiolvasásához és eltárolásához megfelelő szerkezetet kellett terveznem, hogy az alkalmazásban azt fel tudjam dolgozni. Ez a JSON séma egy *document* objektumból áll, ami az egész dokumentumot reprezentálja. A dokumentumon belül szekciók találhatóak, ezeket a JSON-ban *section* tömbben tárolom. Egy szekciónak három adattagja van, egy egyedi azonosító (*id*), a szekció címe (*title*), és egy tömb, amiben a mezők vannak (*fields*). Egy mező képviseli az űrlapon megjelenített beviteli mezőket. Ez lehet akár egyszerű szöveges beviteli mező (*text*, *email*¸ *number*), jelölő négyzet (*checkbox*), legördülő menü (*spinner*), vagy rádió gomb (*radiobuttons*) is. A mező *type* adattagjában lehet megadni a mező fajtáját. Egy mező ezen kívül rendelkezik egy egyedi azonosítóval (*id*), egy címkével (*label*), egy értékkel (*value*), egy szerkeszthetőségi változóval (*editable*) és egy tulajdonságok tömbbel (*properties*). A *properties*t használom a legördülő menü és a rádió gomb elemeinek tárolására. Az űrlap JSON-ra a mellékletben helyeztem el egy példát.

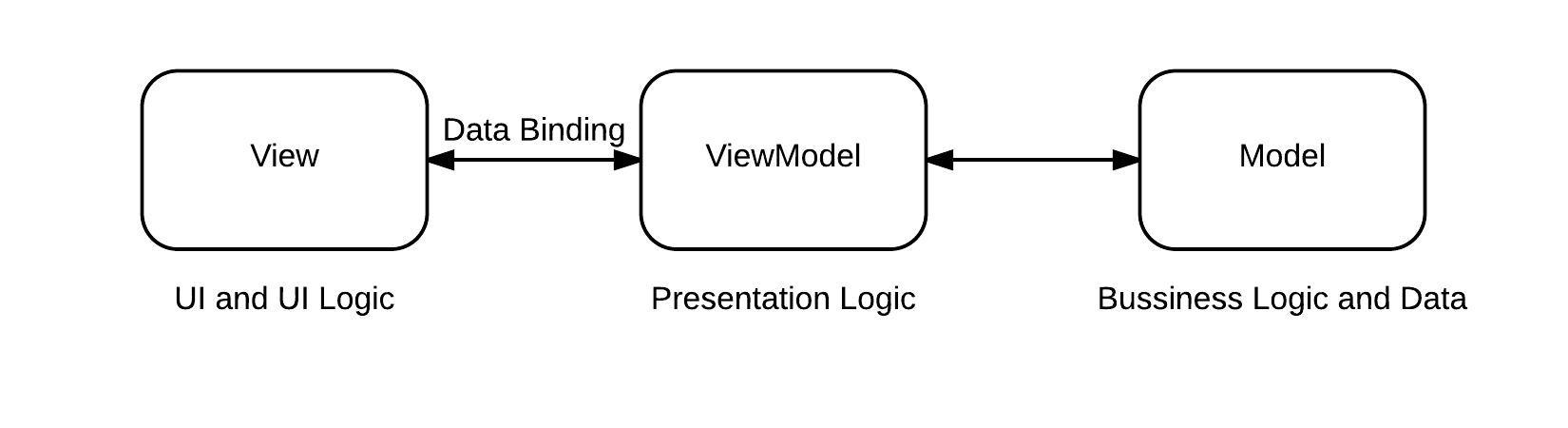
# Implementálás

Ebben a fejezetben a fejlesztés kerül részletezésre, bemutatom a felmerült problémákat és ismertetem, hogy milyen megoldásokat használtam az implementálás során. Az 5.3. fejezetben említett tervezési folyamat utolsó két szakasza egybe olvadt az implementációval. A tervezett felületekből prototípusokat készítettem, amiket iteratívan fejlesztettem tovább.



## Model View ViewModel

Szeretném röviden ismertetni a Model View ViewModel fogalmát, mivel a fejezet további részeiben hivatkozni fogok rá. A Model View ViewModel (MVVM) egy architektúra minta, amit a szoftverfejlesztésben használnak. A minta lényege, hogy elkülöníti a megjelenítést az alkalmazás logikai részétől. Ennek használatával a designerek és a programozók párhuzamosan tudnak dolgozni. Másik előnye, hogy sok kódot újra lehet használni. A lenti ábrán látható felépítése, három részre tagolódik. A Viewban kapot helyet a megjelenítés és az azért felelős logika. A Modelben definiáljuk az adatokat leíró osztályokat és a hozzájuk kapcsolód logikát. A ViewModel felelős azért, hogy a Viewt és a Modelt összekösse. A megfelelő adatokat hozzárendeli a megjelenítési felülethez, ezt hívják data bindingnak. Ez a réteg értesíti a View-t, ha megváltozott a hozzárendelt adat, ekkor a View frissíteni tudja a megjelenített adatot.



6.1. ábra Model View ViewModel

Az MVVM-nek sok féle implementációja létezik, fejlesztésem során próbáltam követni a meglévő trendeket, de mivel egyedül dolgoztam az alkalmazáson ezért az én implementációmban a View és a ViewModel közötti határok elmosódtak.

## Univerzális Hub alkalmazás sablon

A programozás megkezdése előtt ki kellett választanom a megfelelő sablont az alkalmazásom számára. A választás az univerzális Hub sablonra esett. Az univerzális sablon három projektet készít a létrehozott megoldásban (Solution). Egy Windows, és Windows Phone projektet és egy harmadikat, ami a megosztott kódot tartalmazza (Shared). Ennek a sablonnak a legnagyobb előnye, hogy időt és energiát lehet vele megtakarítani, mivel egyszerre két platformra lehet fejleszteni az alkalmazást, ami nagyrészt megosztott kódot használ. A sablon Hub változatában egy minta Hub oldal található, aminek a segítségével könnyebben elindulhat a fejlesztő, mint ha nulláról kellene felépítenie az oldalt. A sablon ezek mellett létrehoz pár segéd osztályt is, amiket a *Common* mappában helyez el.

Az első ilyen osztály az *ObservableDictionary*, egy rendkívül hasznos eszközt ad a fejlesztő kezébe, aminek a segítségével valamilyen objektumot tárolhatunk egy string kulcsszóval. Az „observable” szó onnan ered, hogy ez az osztály értesíti a Viewt, ha a tárolt objektum megváltozott. Ezt az osztályt a Viewban használtam data bindinghoz.

A *NavigationHelper* osztály az oldalak közötti navigációt segíti. Használatával előre és hátra navigálhatunk az oldalak között és kezeli a különböző billentyű leütéseket és a telefon vissza gombjának leütését.

A *SuspensionManager* osztály megkönnyíti az alkalmazás élet ciklusának kezelését. Segítségével elmenthetjük az alkalmazás állapotát és visszaállíthatjuk az elmentett állapotot.

A fejlesztés során kitöröltem a Windows-os projektet a megoldásból, mivel a szakdolgozat keretén belül nem készítettem el az alkalmazásnak ezt a változatát. De a kód nagy része így is a megosztott projektben van, így a lehetőség ugyanúgy fenn áll, hogy később elkészüljön az alkalmazás Windows-os portja.

## Felhasznált programkönyvtárak

Az alkalmazásom elkészítéséhez több programkönyvtárat is felhasználtam, amik segítségével könnyebben implementálhattam egy adott funkciót az alkalmazásomban. A programkönyvtárakat a Visual Studioban a NuGet[[1]](#footnote-1) segítségével töltöttem le. Ebben az alfejezetben ezekről a könyvtárakról ejtek pár szót.

### Json.NET

Az egyik legfontosabb programkönyvtár az alkalmazásom számára a Json.NET, ami egy nagy teljesítményű JSON keretrendszer .NET-hez. A csomag 6.0.6-os verzióját használtam. Segítségével JSON formátumban levő karaktersorozatból lehet objektumokat készíteni és fordítva. Ezeket a funkciókat alkalmazom, amikor a szervertől kapott üzenetet feldolgozom, illetve az űrlap generálásánál is felhasználom.

### QKit

A QKit egy olyan osztály könyvtár, ami nehezen implementálható vagy gyakran használt eszközöket tartalmaz. A QKit 0.4.1-es változatát használtam. Egyik hasznos eszköze a *ListPlaceholder*, amivel egy alapértelmezett szöveget helyezhetünk el a listáknál, jelezve, hogy a lista üres. Másik hasznos eleme ennek a könyvtárnak az *AlphaJumpList*. Ezzel könnyedén átalakíthatjuk a már meglevő *ListView*-t egy ugró listává. Ajánlott az ugró listák használata hosszú listáknál, mivel így a felhasználónak nem kell végig görgetni-e az egész listát, hogy elérjen az utolsó elemig. A lista elemeinek kezdőbetűjével ugró pontok jelennek meg a képernyőn, amiket megérintve a lista az adott elemhez ugrik.

### NotificationsExtensions

Ezt a hordozható könyvtárat a Microsoft készítette. A csempékről és értesítésekről szóló minta kódból lehet beszerezni. A könyvtár segítségével XML Document Object Model (DOM) használata nélkül készíthetünk csempéket, jelvényeket (badge) és toast értesítéseket. Ezáltal sokkal egyszerűbben és könnyebben készíthetjük el az alkalmazás értesítéseit. A könyvtár használata a Microsoft Limited Public License[[2]](#footnote-2) (MS-LPL) alatt áll. Ezt jelenti, hogy saját felelősségre, szabadon felhasználható a program kód.

### NodeTime és DaveSmits.ThemeManager

A NodaTime egy dátum és idő alkalmazásprogramozási interfész, a 1.3.0-ás verzióját használtam. A páciensek életkorának kiszámítására használtam fel.

A DaveSmits.ThemeManager segítségével felülírhatjuk készüléken beállított témaszínt (accent color), így egységesebbé téve az alkalmazás kinézetét. Az 1.1.1-es verziót használtam az alkalmazásban.

## Megosztott kód

A kódnak egy jelentős része a *DoctorPatientApp.Shared* projektben található. Ez a projekt tartalmazza a megosztott kódot, így ami ebben a projektben van, azt tudja használni a Windows Phone és a Windows projekt is. Itt helyeztem el a különböző segéd osztályokat, az adat modellt, az átalakító osztályokat és a nyelvi támogatáshoz szükséges fájlokat is. Itt kapttak helyet az *App.xaml* és az *App.xaml.cs* fájlok.

A Visual Studio automatikusan létrehozza ezt a két fájlt új projekt készítésekor. Az *Application* osztály működését egészíti ki az *App* osztály az alkalmazás működéséhez szükséges specifikus dolgokkal. Itt kell például lekezelni az alkalmazás felfüggesztését és folytatását. A következő kódsorokkal definiálom, hogy melyik oldalt nyissa meg az alkalmazás indításakor.

if (!rootFrame.Navigate(typeof(LoginPage), e.Arguments))

{

throw new Exception("Failed to create initial page");

}

### Segédosztályok

A *Common* névtérben találhatók a segédosztályok, amik kiszolgálják az alkalmazás Viewját és ViewModeljét.

A *HttpHelper.cs* fájlban található a statikus *HttpHelper* osztály. Ez az osztály felelős a HTTP kérések lebonyolításáért. A *baseUri* adattagjában tárolom a szerver elérhetőségét. Kívülről egy publikus metódussal érhető el.

private static string baseUri = "http://cika.rozsenich.hu/";

public static string BaseUri{

get { return baseUri; }

}

Erre azért volt szükség, mert a grafikonok megjelenítése WebViewban történik és a betöltendő URL alapját ez a *baseUri* képzi. A *HttpHelper* osztály többi részét a különböző kéréseket lebonyolító metódusok adják. Minden ilyen metódus először megvizsgálja, hogy van-e élő internet kapcsolata a készüléknek. Csak azután engedélyezi lefutni a függvény lényegi részét, ha az *IsInternetAvailable* adattag *get* metódusa igazzal tér vissza. Ezzel kiküszöbölhetők a nem várt hibák és értesíteni lehet a felhasználót, ha nincs internet kapcsolata.

public static bool IsInternetAvailable

{

get

{

var internetProfile = NetworkInformation.GetInternetConnectionProfile();

return (internetProfile != null && internetProfile.GetNetworkConnectivityLevel() == NetworkConnectivityLevel.InternetAccess);

}

}

Két HTTP kérés metódusát emelném ki. Az egyik lenne a vérnyomás értékét rögzítő POST kérés, aminek *AddNewBloodpressureReading* a neve. A másik ennek párja, a *GetPatientsBloodpressureReadings* metódus, ami egy GET kérést küld a szerver felé, hogy lekérje az adott páciens vérnyomás értékeit. Először a vérnyomás hozzáadásával kezdeném.

public static async Task<bool> AddNewBloodpressureReading(string sys, string dia)

Két string paramétert igényel, a vérnyomás mérés során kapott szisztolés és diasztolés értéket. A metódus hívása aszinkron és *Task<bool>* a visszatérési értéke. A *Task* osztály típus azért szükséges, hogy a metódust awaitable legyen, azaz megvárható. Ezáltal a felhasználói felület ugyanúgy válaszképes marad, amíg a HTTP kérés végbemegy a háttérben.

Az élő internet kapcsolat esetén egy try-catch blokkba lép a program, ahol megtörténik a tényleges HTTP kérés. A *Windows.Web.Http.HttpClient* osztály segítségével bonyolítom le a kérést. Először létrehozok egy új példányt a *HttpClient* osztályból és beállítom a megfelelő URI-t. Előkészítem az üzenet tartalmát a *msg* változóba.

HttpClient httpClient = new HttpClient();

Uri uri = new Uri(baseUri + "bloodpressure");

string msg = string.Format("{{\"params\":{{\"bpr\_sys\": \"{0}\" ,\"bpr\_dia\": \"{1}\", \"bpr\_datetime\": \"{2}\", \"patient\_id\": \"{3}\" }} }}", sys, dia, DateTime.Now.ToString("yyyy.MM.dd. HH:mm:ss"), SettingsHelper.GetUserTypeId());

IHttpContent jsonContent = new HttpJsonContent(JsonValue.Parse(msg));

HttpResponseMessage response = await httpClient.PostAsync(uri, jsonContent);

Ezek után a  *msg* változó tartalmát átadom a *Windows.Web.Http.IHttpContent*

osztály egy példányának. A *HttpJsonContent* szintén egy segédosztály, de ez nem része a hivatalos API-nak, hanem ezt a Microsoft készítette HTTP kéréseket bemutató minta kódjához. Ez az osztály szintén az MS-LPL alatt áll. A *HttpJsonContent* osztály megfelelően beállítja a HTTP kérés tartalmát. A *Windows.Web.Http.HttpResponseMessage* osztály tárolja le a *PostAsync* metódustól visszaérkező választ.

IHttpContent jsonContent = new HttpJsonContent(JsonValue.Parse(msg));

HttpResponseMessage response = await httpClient.PostAsync(uri, jsonContent).AsTask(cts.Token);

Ezek után fel kell dolgozni a választ, amit kaptunk. Ellenőrzöm a státusz kód helyességét és átalakítom karaktersorozattá a válasz tartalmát.

response.EnsureSuccessStatusCode();

var responseText = await response.Content.ReadAsStringAsync();

A *responseText* változó tartalma JSON formátumban van, ezt át kell alakítanom objektumokká, hogy dolgozhassak vele. Itt hívom segítségül a Json.NET keretrendszert. A *ResponseMessage* egy az adat modellben található osztály, ez alapján dolgozza fel *responseText* változót a *JsonConvert.DeserializeObject* metódus, aminek eredménye egy *ResponseMessage* típusú objektum. Így már elérhetjük az üzenet adattagjait. Az *AddNewBloodpressure* függvény esetén csak azt kell tudnom, hogy sikerült-e a rögzítés vagy sem. Ezt a *Params.Succes* logikai változó tartalmazza. Ha ez igaz, akkor minden rendben volt és igazzal tér vissza a függvény is. Ha hamis értéket kaptunk vissza, akkor a metódus is hamis értéket ad vissza. Mindkét esetben eldobjuk az eddig használt *httpClient* változót.

ResponseMessage responseMessage = JsonConvert.DeserializeObject<ResponseMessage>(responseText);

Params \_params = responseMessage.Params;

if (\_params.Success)

{

httpClient.Dispose();

return true;

}

else

{

httpClient.Dispose();

return false;}

A *GetPatientsBloodpressureReadings* metódus hasonló az előzőhöz, de pár dologban eltér. Visszatérési típusa *Task<ObservableCollection< BloodpressureReading>>*. A kérés GET metódust használja, ekkor a következőre módosul az URI és a kérés küldése.

Uri uri = new Uri(baseUri + "/bloodpressure?patient\_id=" + patientId + "&nocache=" + Guid.NewGuid().ToString());

HttpResponseMessage response = await httpClient.GetAsync(uri);

Ez esetben nem kell külön üzenetet összeállítani, hanem elég az URI-ban megadni a megfelelő adatokat, jelen esetben a páciens azonosítóját. A GET kérések esetén az URI-t kibővítem a "&nocache=" + Guid.NewGuid().ToString() kódsorral a következő okok miatt. Az operációs rendszer eltárolja a gyorsítótárában a GET kéréseket, amivel sok esetben nem a megfelelő hatást érjük el. Ennek kiküszöbölésére kibővítem egy egyedi azonosítóval, aminek hatására mindig más URI képződik és így mindig a legfrissebb adatokat kapjuk vissza.

Sikeres adatátvitel után, hasonlóképpen kapjuk meg az üzenet tartalmát. A tartalom paraméterei közül kiolvassuk a vérnyomás méréseket a következő módon.

if (\_params.Bloodpressure\_readings != null)

{

httpClient.Dispose();

ObservableCollection<BloodpressureReading> readings = new ObservableCollection<BloodpressureReading>();

foreach (var reading in \_params.Bloodpressure\_readings)

{

readings.Add(reading);

}

return readings;

}

Egy foreach segítségével végig haladunk a vérnyomás mérések listáján és hozzáadjuk az egyes elemeket egy *ObservableCollection*höz. Ez egy speciális lista típus, ami értesíteni tudja a Viewt, ha megváltozott a tartalma. Azért használom ezt a típust, mivel a kapott listát hozzá fogom később rendelni az adott oldal adat kontextusához (data context).

A *SettingsHelper.cs* fájlban található *SettingsHelper* osztály segítségével mentem el a bejelentkezett felhasználó adatait. Az adatok egy részét csak a memóriában tárolom, de egy részét el is mentem az alkalmazás számára fent tartott, a többi alkalmazástól elkülönített tárhelyen. Ezt a tárhelyet csak is ez az alkalmazás képes elérni, így az ide mentett adatok biztonságban vannak.

private static ApplicationDataContainer localSettings = ApplicationData.Current.LocalSettings;

public static void SetUserame(string username)

{

localSettings.Values["userName"] = username;

}

A fenti kódrészlet segítségével mentem el a bejelentkezett felhasználó e-mail címét, ha bejelentkezéskor kipipálta, hogy az alkalmazás jegyezze meg az e-mail címét. A memóriában tárolt adatok publikus és statikus változókként jelennek meg az osztályban. Azért választottam azt a megoldást, mert így bármilyen oldal ViewModeljéhez elérhetem a kellő adatokat, és az adatok így egy helyen vannak tárolva. Sok adatot elég egyszer lekérni a szerverről, így ebből az osztályból azt elérhetem, és nem kell fölöslegesen terhelnem a hálózatot.

A *TileHelper* osztály felelős az élő csempe frissen tartásához, amíg az alkalmazás fut. Az alkalmazás élő csempéjén jelenítem meg a függőben levő betegeket és azok számát. Ha az alkalmazásban az orvos elfogadott vagy elutasított betegeket, akkor ennek megfelelően kell frissíteni a csempét is, hogy az alkalmazás bezárásával az aktuális értéket mutassa. Ennek véghezviteléhez használom a *NotificationsExtensions* könyvtárt. Az osztály egy darab publikus, statikus függvényt tartalmaz, ami a függőben levő betegek listáját várja paraméteréül.

A *Resoucre.cs* fájlban található *Resoucre* osztályt segít betölteni a megfelelő nyelvhez tartozó szövegrészletet. Az osztálynak az alábbi két adattagja van:

private static ResourceLoader loader = new Windows.ApplicationModel.Resources.ResourceLoader();

public static ResourceLoader Loader{

get { return loader; }

}

Ezek használatával a következőképpen lehet betölteni a megfelelő szöveg forrást, ahol „MessageDoctorMissingTitle” az értékhez tartozó kulcs.

Resource.Loader.GetString("MessageDoctorMissingTitle")

Az alkalmazás során használt jelszavakat SHA512-es hash képző algoritmussal titkosítom. A *CryptographicHelper* osztályban található *HashPassword* metódus képzi a hasht. Mivel csak egy teszt szerver állt rendelkezésemre, további titkosítást nem implementáltam.

### Átalakítók

A *Converters* mappában található három darab átalakító osztály. Ezek az osztályok az *IValueConverter* interfészt implementálják. Az osztályok úgy lettel elnevezve, hogy a nevükből egyértelműen kiderüljön, hogy miből mivé alakítanak át változókat. Mindhárom osztálynak van egy *Convert* és egy *ConvertBack* metódusa, de csak a *Convert* lett implementálva, mivel csak erre volt szükségem. A *DateTimeToAgeConverter* osztály *DateTime* típusú születési dátumot alakít át életkorrá a NodaTime könyvtár segítségével. A *DateTimeToStringConverter* osztály *DateTime* típusú változóból készít stringet, ami csak a dátumot tartalmazza. A *GenderToStringConverter* osztály „f” (férfi) és „n” (nő) betűket alakítja át teljes szavas változatukká, figyelembe véve a nyelvi beállításokat.

### Adat modell

A *DataModel* mappában kaptak helyett a Modelhez és ViewModelhez tartozó osztályok. A *Doctor* és a *Patient* osztály tartalmazza az orvos és a beteg modelljét. Mindkét osztály implementálja az *INotifyPropertyChanged* interfészt, aminek használatával értesíthetjük a Viewt, ha az adott érték megváltozott az osztályban. Az alábbi kódrészlet bemutatja az *INotifyPropertyChanged* interfész megfelelő implementálását egy adattag példájával. Az adattag publikus része felett levő sorban közlöm a JSON szerializálóval, hogy mindig ezzel a névvel használja ezt az adattagot.

private string address;

[JsonProperty("doctor\_address")]

public string Address

{

get { return address; }

set

{

if (value != this.address)

{

this.address = value;

NotifyPropertyChanged("Address");

}

}

}

public event PropertyChangedEventHandler PropertyChanged;

private void NotifyPropertyChanged([CallerMemberName] String propertyName = "")

{

if (PropertyChanged != null)

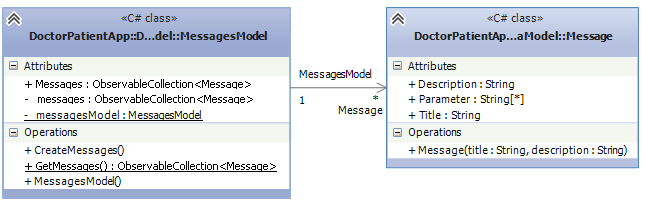
{

PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

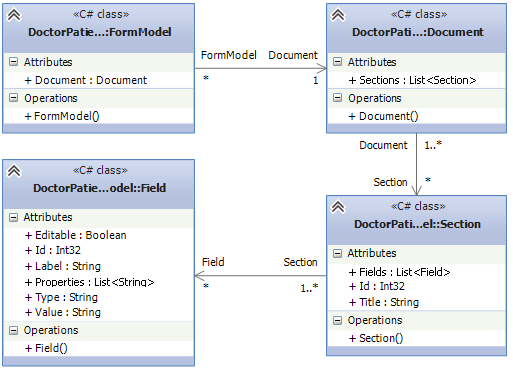
}

A fenti módszer szerint implementáltam minden olyan modellt, amit fontos frissen tartani a Viewban.



6.2. ábra MessagesModel.cs osztályainak osztálydiagramja

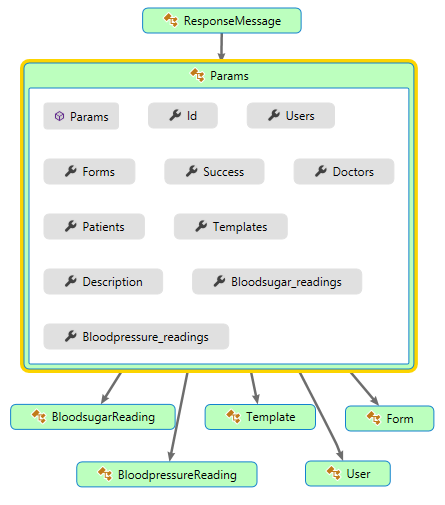
A *Messages.cs* fájl két osztályt tartalmaz, a *Message* osztályt és a *MessagesModel* osztályt. Az előbbi a Modelt míg az utóbbi a ViewModelt képviseli. A két osztály viszonya jól látható a 6.2-es ábrán. A *MessagesModel* osztályban levő *CreateMessages()* metódus készíti el az üzeneteket, amik megjelennek az orvos vagy páciens Hub oldalának üdvözlő képernyőjén.



6.3. ábra FormModel.cs osztályainak osztálydiagramja

A *FormModel.cs* fájlban találhatjuk az űrlap modelljét. Egy űrlap egy dokumentumból áll, amiben szekciók vannak. Egy szekciónak van egy címe és akár több mezője is. A mezőnek van címkéje, típusa (egyszerű beviteli mező, rádió gomb, jelölő négyzet), értéke, szerkeszthetősége, és tulajdonságai. Az osztályok kapcsolatát és adattagjait szemlélteti a 6.3-as ábra.

A *ResponseMessage.cs* fájlban található osztályok reprezentálják a HTTP kérésekben visszakapott üzenet modelljét. Ennek az osztálynak a segítségével deszerializálom a JSON karaktersorozatot *ResponseMessage* objektummá, hogy kinyerhessem az üzenet tartalmát. A *ResponseMessage* osztály legfontosabb része egy *Params* osztály típusú adattag. A *Params* osztályban sok féle adattag került, hogy lefedjem az üzenetben érkező adatokat. Ezeket az osztályokat és adattagokat szemlélteti a következő ábra.



6.4. ábra Params osztály szemléltetve a Code Map-en

### Nyelvi támogatás

Szintén a megosztott kódban kapott helyett a *Strings* mappa, amiben a nyelvi támogatáshoz szükséges mappákat és fájlokat találjuk. A *Strings* mappában két mappa található, amit az adott nyelv BCP-47 kódja alapján van elnevezve. Az alkalmazásom az amerikai angol és a magyar nyelvet támogatja azért *en-US* és *hu-HU* nevet kapták. Mindkét mappában van egy-egy *Resources.resw* fájl, ez egy speciális XML fájl, ami a Microsoft RexX Schema-át használja. A Visual Studio biztosít hozzá egy szerkesztőt, ami egy 3 oszlopos táblázat. Az első oszlop tartalmazza a kulcs nevét, a második az értéket, és a harmadik oszlopban pedig megjegyzést helyezhetünk el. Ebben a táblázatban kell elhelyezni a fordítás kulcs érték párjait. A 6.4.1-es fejezetben már említettem, hogy hogyan kell C#-ban lekérni az adott kulcs értékét. XAML fájlokban is meglehet ezt tenni, de más módszerrel. Az adott XAML vezérlőhöz hozzá kell rendelni egy azonosítót, amit az *x:Uid* adattagjában kell megadni. Ezt az azonosítót helyezzük el a *Resource.resw* fájlban, a kulcs oszlopban. Ha a vezérlőnek egy specifikus adattagjához kell hozzárendelni a szöveget, akkor a kulcs oszlopban egy pont után meg kell adni az adattag nevét. Lent látható kódrészlet esetén a kulcs értéke *LoginSignInBody.Text.*

<TextBlock Text="Please login in with your credentials"

x:Uid="LoginSignInBody"

Name="body"

Style="{ThemeResource BaseTextBlockStyle}"

TextWrapping="Wrap" />

## Háttérfolyamatok

A háttérfolyamatokat a *BackgroundTasks* hordozható projektben hoztam létre. Az egyik ilyen folyamatot az *UpdateTilesBackgroundTask.cs* fájlban definiáltam. Az *UpdateTilesBackgroundTask* a *TileHepler* osztályhoz hasonló, de akkor lép működésbe, amikor az alkalmazás nem fut. Windows Phone-on minimum 30 percenként futhat le egy ilyen háttérfolyamat. A háttérfolyamatok kezelésében a *BackgroundTaskHelper* osztály segít, ami a megosztott kód *Common* mappájában található a *BackgroundTaskHelper.cs* fájlban. Ezzel az osztállyal lehet új folyamatokat regisztrálni és meglévőket törölni. Az alábbi kódrészletben regisztrálom az *UpdateTilesBackgroundTask* háttérfolyamatot, hogy fusson le 30 percenként, ha van internet kapcsolat.

BackgroundTaskHelper.RegisterBackgroundTask("BackgroundTasks.UpdateTilesBackgroundTask", "UpdateTilesBackgroundTask", new TimeTrigger(30, false), new SystemCondition(SystemConditionType.InternetAvailable));

## Megjelenítés

A *View* mappában találhatóak az alkalmazás oldalainak fájljai. Minden oldal két fájlból áll, egy *xaml* és egy *cs* fájlból. A *xaml* fájl felelős a megjelenítésért a *cs* pedig a mögötte levő logikáért.

Szinte minden oldal rendelkezik egy *ObservableDictionary* típusú *defaultViewModel* változóval. Az *ObservableDictionary* osztályról már esett szó a 6.2-es fejezetben, most bemutatom, hogyan használtam fel az alkalmazásomban. A *defaultViewModel* feladata, hogy tárolja az adott oldalhoz tartozó ViewModeleket. Az oldal mögötti kódban tölthetjük fel tartalommal. Az egyes kulcsokhoz a következőképpen rendeltem hozzá a tartalmakat.

ObservableCollection<Message> messages = MessagesModel.GetMessages();

this.DefaultViewModel["Messages"] = messages;

A fenti példában az üzeneteket kérem le és rendelem hozzá az oldal ViewModeljéhez. A XAML fájlban a *Page* tagben a következő sorral definiálni kell az adat kontextust, hogy az oldal tudja, honnan vegye az adatokat.

DataContext="{Binding DefaultViewModel, RelativeSource={RelativeSource Self}}"

A fent sor az oldal mögötti kódban levő *DefaultViewModel* változóra állítja be az adat kontextust. Ekkor a XAML kódban már hozzáköthetjük a változókat a ViewModelhez.

### HubPage

Az orvosok és páciensek elsődleges felületei egy Hub oldalban kaptak helyet. Mindkét esetben hasonlóan épül fel a két oldal, így most csak az egyiküket emelném ki. Legyen ez a páciensek felülete. A *HubPagePatient.xaml* fájlban definiáltam egy *Hub* vezérlőt. Ez a *Hub* vezérlő három darab szekcióból áll, amit a XAML-ben *HubSection*nek hívnak. A *HubSection*ön belül egy *DataTemplate* tag található. Ezen belül kell definiálni az oldal tényleges tartalmát.

Sokszor nem elég a Data Binding használata az adatok megjelenítéséhez, hanem szeretnénk elérni a kódból is az adott vezérlőt, hogy módosításokat eszközöljünk rajta. A Hub megannyi előnye mellett azzal a hátránnyal jár, hogy a *DataTemplate*n belül definiált vezérlőket nem lehet elérni a kódból (*cs* fájlból). Ezt csak egy körülményes eljárással lehet megkerülni.

Futás közben kell megkeresni név alapján az adott *HubSection* vezérlő gyermekei közül a keresendő vezérlőt. Ennek véghezvitelét a *FindChildControl* rekurzív metódus végzi. A *Windows.UI.Xaml.Media.VisualTreeHelper* osztály segítségével megvizsgálja a paraméterben megadott vezérlő gyermekeit. A függvény paraméterében a szülő vezérlő mellett a keresendő vezérlő nevét kell megadni. Megvizsgálja, hogy a vezérlő típusa és neve megegyezik-e a paraméterben megadott vezérlővel. Ha nem találta meg, akkor meghívja önmagát az előbb vizsgált gyermek vezérlővel. Ha megtalálta, akkor visszatér az adott vezérlővel, így használatba vehetjük a kódban. Ha a keresés végére sem találja meg akkor null értékkel tér vissza.

private DependencyObject FindChildControl<T>(DependencyObject control, string controlName)

{

for (int i = 0; i < VisualTreeHelper.GetChildrenCount(control); i++)

{

DependencyObject child = VisualTreeHelper.GetChild(control, i);

FrameworkElement frameworkElement = child as FrameworkElement;

if (frameworkElement == null) return null;

if (child is T && frameworkElement.Name == controlName)

{

return child;

}

else

{

DependencyObject nextLevel = FindChildControl<T>(child, controlName);

if (nextLevel != null)

return nextLevel;

}

}

return null;

}

A következő módon lehet használni.

TextBlock tb = FindChildControl<TextBlock>(HubSectionWelcome, "TextBlockPraxis") as TextBlock;

### Űralap generálás

Az űrlapok megjelenítéséért a *FormPage.xaml.cs* fájlban található *FormPage* osztály felelős. Ezen az oldalon jelenik meg az orvos és a beteg számára is az űrlap. Az oldal a navigációs paraméterekben kapott változók alapján dönti el, hogy melyik megjelenítést készítse elő. Három esetet különböztettem meg: sablon megjelenítés, űrlap megjelenítése páciens számára, és űrlap megjelenítése orvos számára. Ezek az esetek az alkalmazás sáv és a felső címsáv megjelenítésének módjában térnek el.

Miután az oldal előkészítette a felületet a három eset alapján, meghívódik a *BuildForm* metódus. Paraméteréül a JSON-ban kódolt űrlapot várja. Ez a függvény végzi a felület generálását a JSON-ból képzett objektumból. A paraméterből először elkészítem a *FormModel* objektumot.

formModel = JsonConvert.DeserializeObject<FormModel>(formText);

Ezek után egy for ciklus segítségével minden egyes szekcióban leképzem a megfelelő elemeket. Első lépésként a szekció címét készítem el, amit az alábbi kód részlet mutat be.

var section = formModel.Document.Sections[i];

TextBlock sectionTitle = new TextBlock();

sectionTitle.TextAlignment = TextAlignment.Left;

sectionTitle.TextWrapping = TextWrapping.WrapWholeWords;

sectionTitle.Style = Resources["GroupHeaderTextBlockStyle"] as Style;

sectionTitle.Text = section.Title;

ContentRoot.Children.Add(sectionTitle);

A fenti kódban létrehozok egy új *TextBlock* vezérlőt, aminek adattagjait beállítom, és végül hozzárendelem a *ContentRoot* *Grid*hez. Ez után ismét egy for ciklus következik, ami a szekcióban található mezőket iterálja. A mező típusa szerint hozzáadásra kerülnek a megfelelő elemek. Ha a mező típusa „text”, „email”, vagy „number”, akkor *TextBox*ot adok hozzá a *ContentRoot*hoz. Ha a típus „spinner”, akkor egy *ComboBox*ot, ha „checkbox”, akkor egy *CheckBox*ot, ha pedig „radiobuttons”, akkor *RadioButton*okat adok hozzá a felülethez. *TextBox* esetén, a típus alapján beállítom az odaillő billentyűzetet is. Például e-mail címhez egy e-mail cím bevitel megkönnyítésére szolgáló billentyűzetet állítok be. Ehhez a *TextBox InputScope* adattagját kell módosítanom.

Az űrlap visszaküldéséhez el kell menteni a beírt értékeket a ViewModelbe. Ehhez a vezérlők esemény kezelő függvényében (event handler) azonosítanom kell, hogy pontosan melyik mezőhöz tartozik. Ezt úgy oldottam meg, hogy a szekción belül található vezérlők *Tag* adattagjában eltárolom a szekció és a mező egyedi azonosítóját. Egy *ComboBox* esetén a következőképpen csinálom.

int[] array = new int[2];

array[0] = section.Id;

array[1] = field.Id;

tempComboBox.Tag = array;

A *ComboBox\_SelectionChanged* eseményben kiolvasom a küldő *ComboBox* változó *Tag* értékét, ami alapján már betudom azonosítani a megfelelő elemet. Ezt LINQ segítségével hajtom végre az alábbi módon.

int[] tag = cb.Tag as int[];

int section\_id = tag[0];

int field\_id = tag[1];

var tempSection = FormModel.Document.Sections.FirstOrDefault(section => section.Id == section\_id);

var tempField = tempSection.Fields.FirstOrDefault(field => field.Id == field\_id);

tempField.Value = cb.SelectedIndex.ToString();

Miután hozzáadásra került az összes szekció és a szekciókhoz tartózó mezők, az űrlap készen áll a megjelenítésre. Az űrlap elküldéséhez az oldal a ViewModelből készíti el a JSON-t, így az orvos által beleírt dolgok is megjelennek a páciensnél.

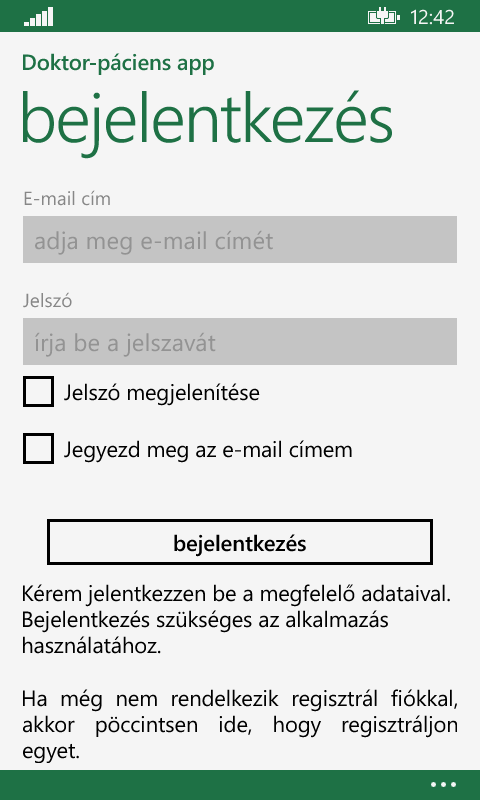
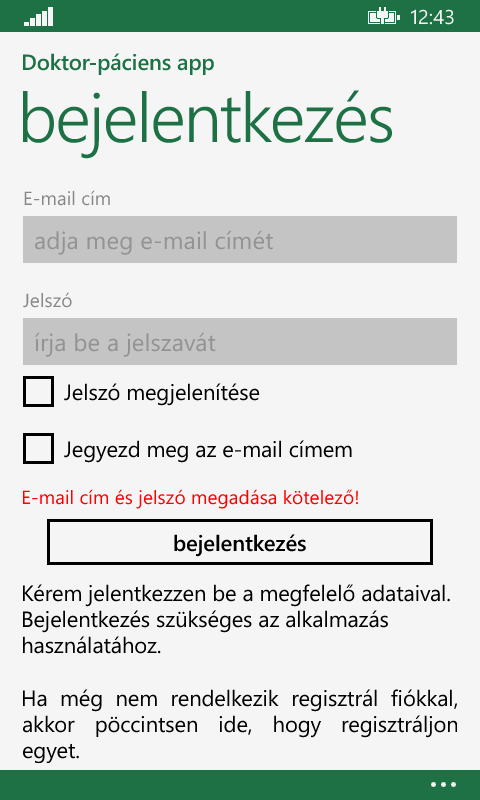
# Alkalmazás bemutatása

Ebben a fejezetben szeretném bemutatni az elkészített alkalmazást működés közben. Képernyőmentésekkel illusztrálva ismertetem a felhasználói felületeket és a rajtuk végrehajtható funkciókat.



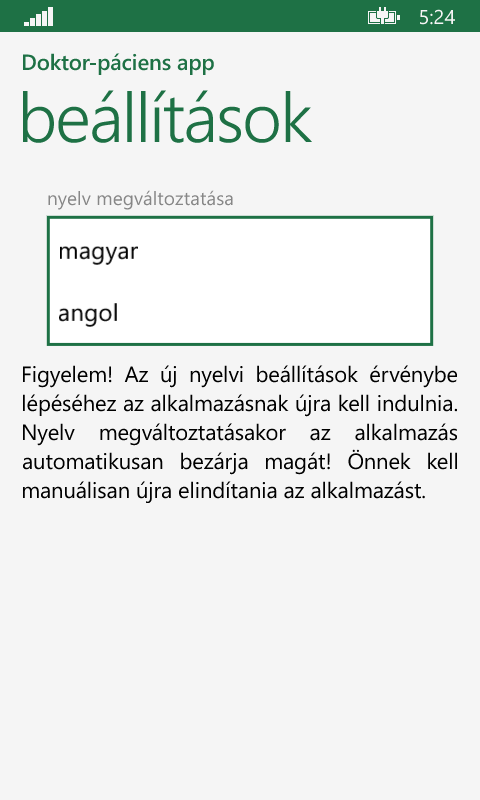
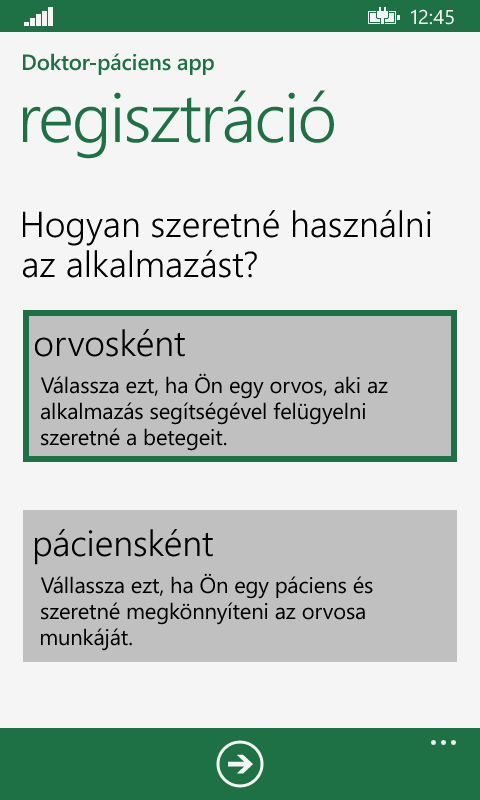
## Bejelentkezési és regisztrációs felület

Az alkalmazás indítása után a bejelentkezési felület fogadja a felhasználót. Az e-mail címe és jelszava megadása után a bejelentkezés gomb megnyomásával belépteti az alkalmazás. A felhasználó megjegyeztetheti a belejelentkezéshez használt e-mail címet, hogy a következő bejelentkezéskor ne kelljen újra beírnia. A jelszó beviteli mező alatt van egy jelölő négyzet, ami a beírt jelszó karaktereit megjeleníti. A bejelentkezés gombot megnyomva indul el a beléptetési folyamat. Ha rossz e-mail címet vagy jelszót ad meg a felhasználó, vagy valamelyik, vagy mindkét mező hiányos, akkor annak megfelelő hibaüzenet jelenik meg a bejelentkezés gomb felett. A 7.1-es ábrán két kép látható a bejelentkezési oldalról, a jobb oldali képen hibaüzenet is megfigyelhető.

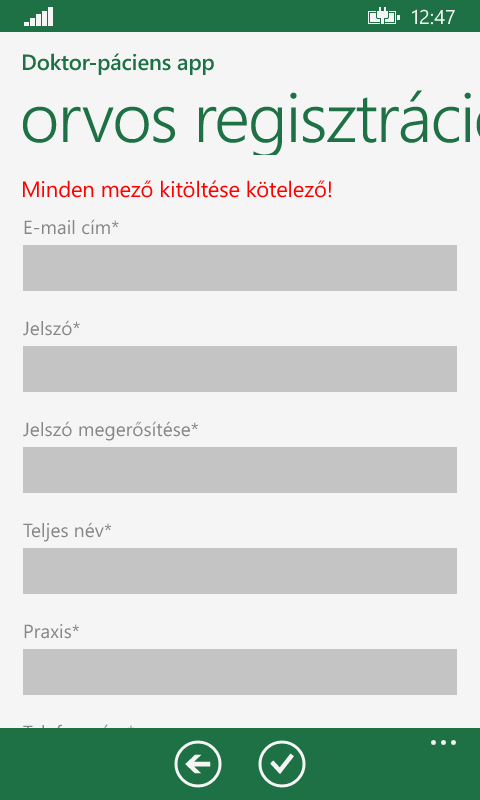
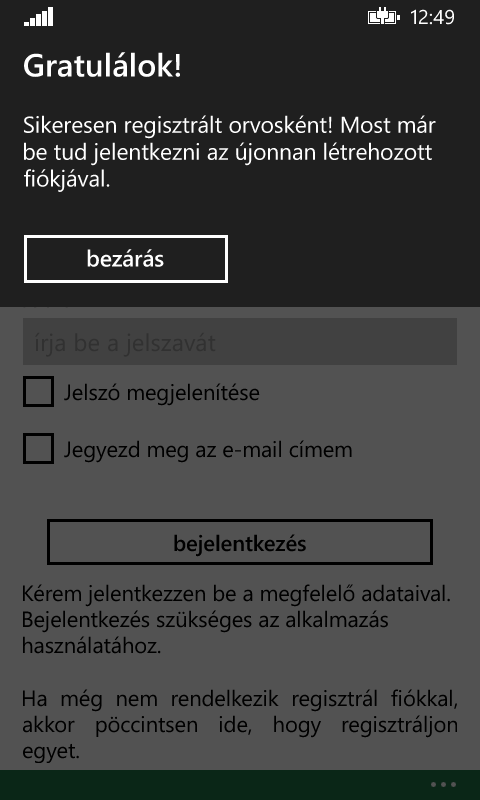
7.1. ábra Bejelentkezési oldal és bejelentkezési oldal hiba üzenettel

A felület alján helyezkedik el egy alkalmazás sáv (Application bar), amit megnyitva láthatóvá válik egy menü. Ebből a menüből lehet eljutni a beállítások felületre, amit a 7.2-es ábra bal oldali képén látszik. Ezen a felületen átlehet állítani az alkalmazás nyelvét. Az alkalmazás az angol és magyar nyelvet támogatja. Hogy az új nyelvi beállítások érvénybe lépjenek az alkalmazás bezárja magát és a felhasználónak kell manuálisan elindítania. Alapértelmezetten applikáció nyelve a készülék nyelvéhez igazodik, ha a készülék nyelve nem tartozik a támogatott nyelvek közé, akkor angol nyelv lesz az alapértelmezett nyelv.

7.2. ábra Beállítások oldal és regisztrációs folyamat első oldala

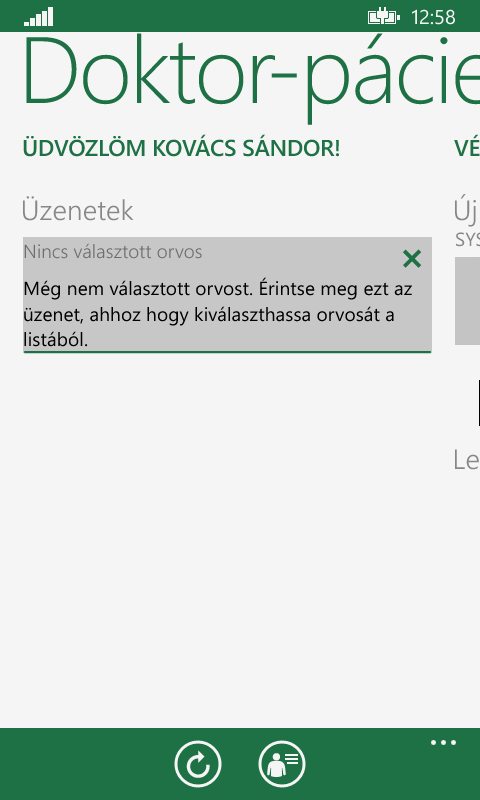
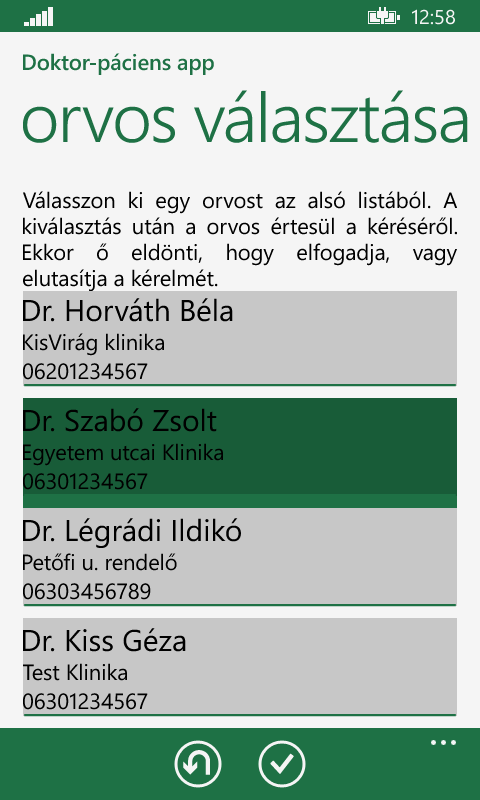
Ha a felhasználó még nem rendelkezik regisztrált fiókkal, akkor a bejelentkezési oldal alján található szövegre pöccintve kezdheti el a regisztráció lépéseit. Ekkor az alkalmazás elnavigál a 7.2-es ábra jobb oldalán látható felületre. Itt a felhasználónak ki kell választania, hogy orvosként vagy páciensként szeretne regisztrálni. Az opció kiválasztása után a tovább nyílra nyomva léphet a következő oldalra. A választástól függően megjelenik az orvos vagy a páciens regisztrációs felület. Az orvos regisztrációs felület látszik a 7.3-as ábra bal oldali képén. A csillaggal jelölt mezőket kötelező kitölteni a sikeres regisztrációhoz. Ha a felhasználó kihagy egy ilyen mezőt, akkor hibaüzenet jelenik meg az oldal tetején. A felhasználó addig nem teljesítheti a regisztrációt, amíg minden kötelező mezőt ki nem tölt. Miután megadta a kellő adatokat a kész gombra pöccintve véglegesítheti a regisztrációt. Ha minden rendben történt, akkor az alkalmazás visszanavigál a bejelentkezési oldalra és egy üzenetben jelzi a felhasználónak a sikeres regisztrációt a 7.3-as ábra jobb oldali képén látható módon. Az újonnan regisztrált felhasználó bejelentkezhet az alkalmazásba.

7.3. ábra Orvosi regisztációs oldal és sikeres orvos regisztráció üzenet

## Páciens felületei

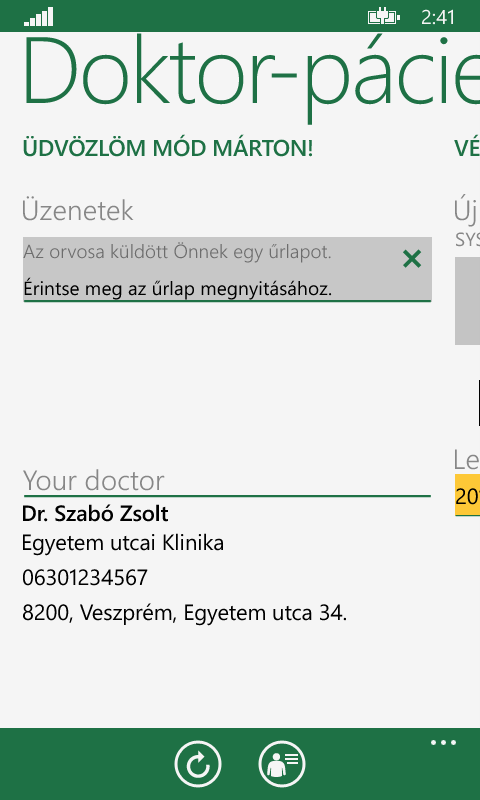
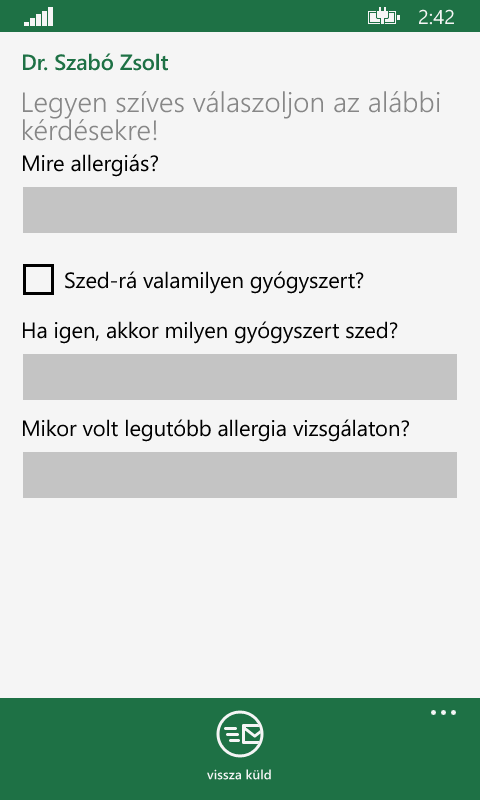
A páciens sikeres beléptetése után az alkalmazás a páciens Hub oldalára navigál. Ez az oldal három darab képernyőből áll, amik között jobbra és ballra pöccintéssel lehet navigálni. A Hub oldal első oldala egy üdvözlő képernyőként funkcionál. Az Üzenetek felirat alatt jelennek meg az alkalmazás által generált üzenetek. Betegek esetén három féle üzenet jelenhet meg. Az első ilyen üzenet akkor jelenik meg, ha a betegnek nincsen választott orvosa. Ezt az esetet szemlélteti a 7.4-es ábra bal oldali képe. Az üzenetet megérintve a megjelenik az orvos választó oldal, ami a 7.4-es ábra jobb oldalán látszik, ahol a regisztrált orvosok listájából választhat a beteg. A választás után az applikáció visszanavigál a Hub oldalra, ahol egy új üzenet várja a beteget, amiben az áll, hogy a választott orvos még nem reagált a kérésére.

7.4. ábra Páciens kezdőképernyő üzenettel és az orvos választási oldal

Ha az orvos elfogadta a páciens felkérését, akkor az Üzenetek alatt megjelenik egy új részleg, ami a beteg orvosának adatait tartalmazza, ami a 7.5-ös ábra bal oldali képén látható. A telefonszám megérintésével felhívhatja a beteg az orvosát. A rendelő címét megérintve az alkalmazás elindítja a készülék alapértelmezett navigációs szoftverét (például Here Drive), ami a jelenlegi koordinátáktól indít egy navigációt a rendelőhöz.

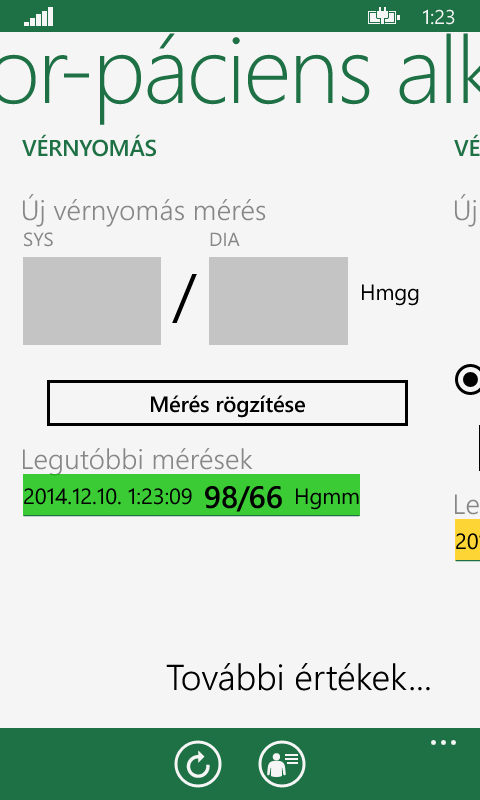
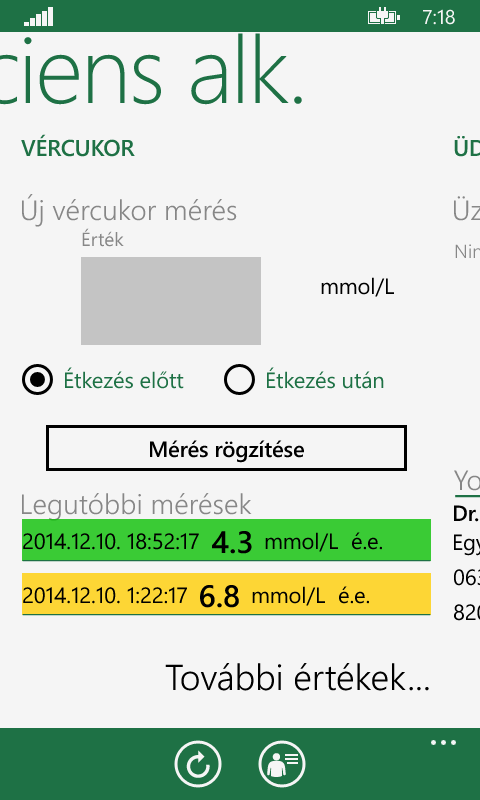
Az üzenetek harmadik fajtája az orvostól kapott űrlapokat hivatottak jelezni, ezt szintén a 7.5-ös ábra bal oldalán látható. Az üzenetre pöccintve meg lehet jeleníteni a kapott űrlapot, ami a 7.5-ös ábra jobb oldalán tekinthető meg. A páciens szerkeszthető mezőket kitöltheti és a képernyő alján található gombbal visszaküldheti az orvosának.

7.5. ábra Páciens kezdőképernyője orvos adataival és új űrlap üzenettel (bal), orvostól kapott űrlap megjelenítve (jobb)

A páciens Hub oldal második képernyőéjét mutatja be a 7.6-os ábra bal oldali képe. Ezen a felületen rögzítheti mért vérnyomás értékét a páciens és innen érheti el az eddig rögzített értékeit is. Új vérnyomás érték rögzítéséhez meg kell adni a szisztolés és diasztolés vérnyomás értékeket Hmgg mértékegységben. A két mezőbe csak számot lehet bevinni, amit a külön erre való billentyűzet segítségével érvényesít az alkalmazás. A beviteli mezők alatt a legutóbbi mérések vannak listázva. Az egyes mérések szín kódolva vannak, attól függően, hogy az érték milyen tartományba esnek. A normális értékek zöld színnel, míg az egyre rosszabb értékek piros szín árnyalataival vannak jelezve. A További értékek gomb megnyomásával meg lehet tekinteni az összes eddig rögzített értéket.

A páciens vércukor oldala a 7.6-os ábra jobb oldali képén látszik. Itt rögzítheti új mért vércukor értékét és megtekintheti a legutóbbi méréseinek eredményeit. Új vércukor mérés rögzítésekor a páciensnek meg kell adnia a vércukrának értékét mmol/L-ben és, hogy étkezés előtt vagy étkezés után mérte az értéket. Ezt megteheti a beviteli mező alatt elhelyezkedő rádió gombok segítségével. A megjelenített mérések itt is színkódolva vannak. A zölddel jelölt értékek a normális tartományon belülre esnek, ettől az intervallumtól távolodva az értékeket piros szín árnyalataival jelöli az alkalmazás.

7.6. ábra Páciens vérnyomás és vércukor oldala

A vérnyomás és vércukor oldalak alján található További értékek gomb, aminek megnyomásával megjelennek az eddig rögzített értékek. Ennek az oldalnak az alján található az alkalmazás sávon egy diagram gomb. Ennek megnyomásával egy új oldalra navigál az applikáció, ahol megjelenek az adatok grafikonon ábrázolva. Ez az oldal érzékeli a készülék elforgatását, hogy jobban látható legyen a grafikon. A grafikon egy beágyazott weboldalon jelenik meg így szabadon lehet nagyítani a tartalmát. A vérnyomás grafikonra látható példa a 7.7-es ábrán. A grafikon oldal alkalmazás sávjában állítható a mutatott értékek időbeli tartománya. Alapértelmezetten az egy hét távlatában rögzített értékeket tekintheti meg a páciens. Ezen kívül lehet szűrni egy hónapon, három hónapon és egy éven belül mért értékekre is. A 7.7-es ábra jobb oldalán látható a megnyitott alkalmazás sáv benne a választható opciókkal.



7.7. ábra Vérnyomás grafikon ábrázolva

A páciens Hub oldalán található alkalmazás sávban két ikon található. Az első ikon segítségével frissíteni lehet az oldalon található adatokat. A másik gombbal pedig a felhasználói profil oldalra navigálhat a páciens. Itt megtekintheti és szerkesztheti az adatokat, amikkel regisztrált. Lehetősége van a bejelentkezéshez használt jelszó megváltoztatásához. Ehhez jelszó megváltoztatása gombra kell pöccintenie, ekkor az alkalmazás a jelszó megváltoztató oldalt hozza be. Itt meg kell adnia régi jelszavát és a rögzítendő új jelszót és annak megerősítését. Csak akkor engedi megváltoztatni a jelszót, ha helyesen adta meg ezeket az értékeket.

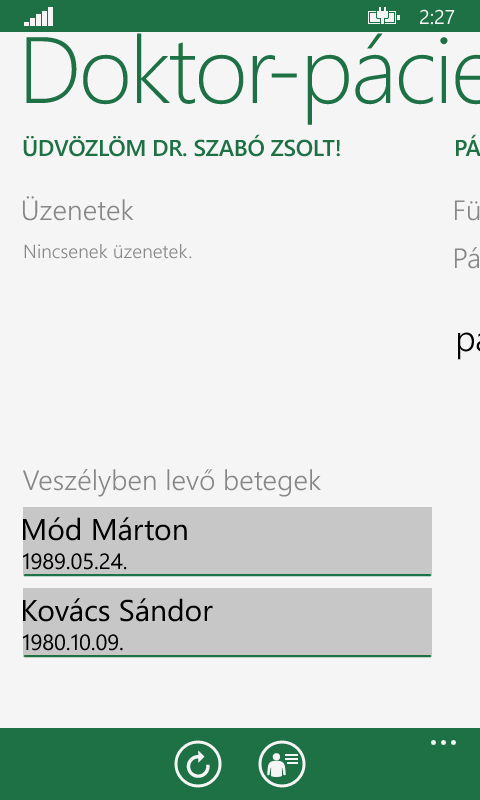
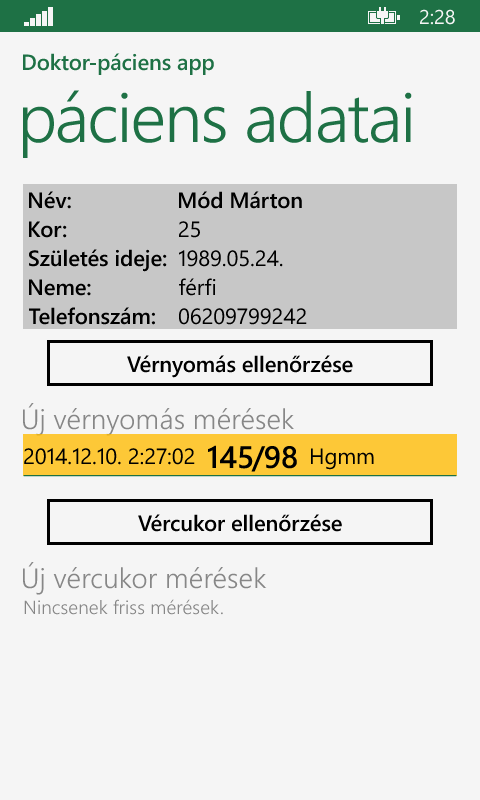
A páciens Hub oldal alkalmazás sávjának felugró menüjének három pontja van. A menü utolsó két pontja a beállítások és a kijelentkezés. A beállítások ugyan arra az oldalra visz, amit a bejelentkezési felületnél ismertettem. A kijelentkezés gombbal a felhasználó kijelentkezik és visszakerül a bejelentkezési felületre. A menü első pontja a segélyhívás. Innen egy olyan oldalra navigál a páciens, ahonnan könnyedén lehet segélyhívásokat indítani. Az oldalon megtalálhatók a magyarországi segélyhívó számok, a tűzoltóság, rendőrség, mentők és az általános segély hívószám. Ha a betegnek van választott orvosa, akkor annak a telefonszáma is megjelenik. A hívószámokat megérintve hívást lehet kezdeményezni az adott fél felé. Az operációs rendszer először megkérdezi a felhasználót, hogy biztosan tárcsázni akarja ezt a számot, ha a felhasználó megerősítette indítékát, csak utána indul a tárcsázás. Az oldal alján található gombbal lehetőség van egy másodlagos alkalmazás csempét a kezdőképernyőre tűzni. Ez a csempe egy pánik gombként funkcionál. A kezdőképernyőn elhelyezve könnyen elérhető, ezt megnyomva az alkalmazás elindul és a segélyhívás oldalt jeleníti meg. Az alábbi ábrán látható segélyhívás oldal felülete és a kezdőképernyőre kitűzött csempéje.

7.8. ábra Segélyhívás oldal és annak csempéje a kezdőoldalon

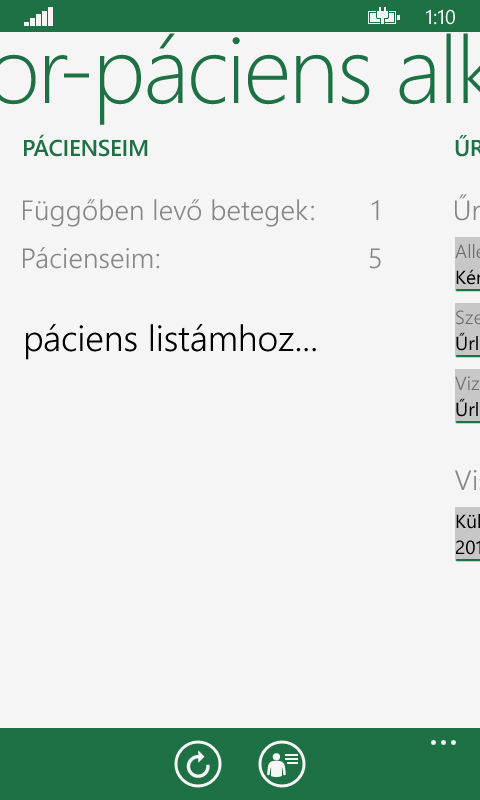
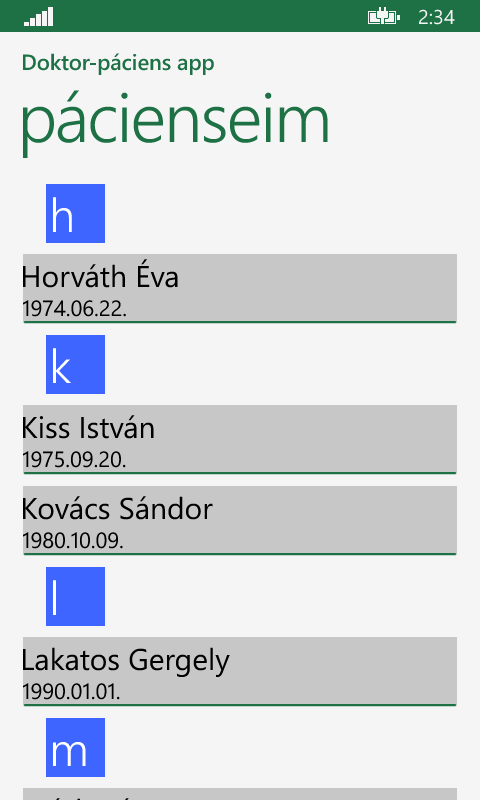
## Orvos felületei

Az orvos bejelentkezése után az orvos Hub oldalra kerül. Felépítésében hasonló a páciens Hub oldalához, de más tartalmakat érhet el róla az orvos. Az üdvözlő oldalon itt is megtalálható az Üzentek szekció. Az orvos üzenet kap arról, ha függőben levő betegei vannak, vagy ha még nincsenek betegei. Az üzenetek alatt a veszélyben levő betegek listája található. A szerver által veszélyben levőnek tekintett betegek kerülnek erre a listára. A 7.9-es ábra bal oldali képén látható ez a lista. Listában levő betegre pöccintve lehet megtekinteni a páciens adatait. Az alapvető adatai mellett a legutóbb rögzített mérései is megjelennek. A 7.9-es ábra jobb oldali képe mutatja be ezt a felületet. A legutóbbi mérések felett található gombokkal meg lehet tekinteni az összes rögzített eredményt. A Vérnyomás ellenőrzése gomb megnyomásával a vérnyomás mérések oldal jelenik meg. Időrendi sorrendbe rendezve és színkódolva jelennek meg a vérnyomás mérések, amiket a páciens rögzített. Az oldal alján, az alkalmazás sávon található gombbal jeleníthető meg a grafikonos ábrázolás. A Vércukor ellenőrzése a vércukor mérések oldalra visz, ahol a vércukor mérések jelennek meg hasonlóképpen. Ezek az értékek is megtekinthetőek grafikonon az alkalmazás sávon található gombbal.

7.9. ábra Orvos üdvözlő oldala veszélyben levő betegekkel (bal), Páciens adatainak megtekintése (jobb)

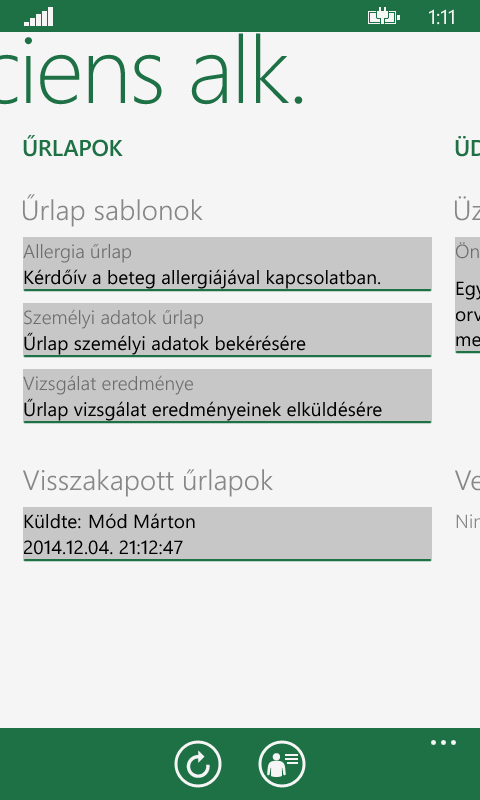
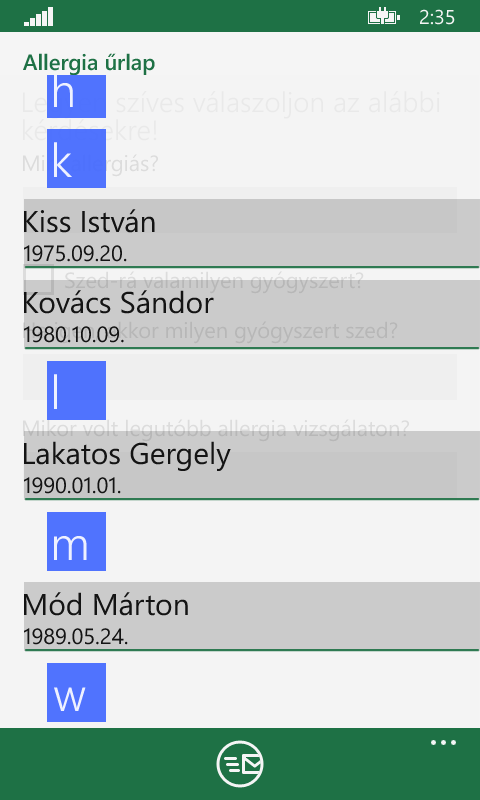
Az orvos Hub oldal második képernyőjén keresztül érhetőek el az orvos páciensei, amit a 7.10-es ábra baloldali képe mutat be. Megjelenítésre került a páciensek és a függőben levő betegek száma. A páciens listámhoz gombra pöccintve megjelenik a páciens lista, amit a 7.10-es ábra jobb oldalán tekinthető meg. Nagyszámú páciensek esetén megkönnyíti a navigálást a listában levő ugró pontok, amik a nevek kezdőbetűit tartalmazzák. Egy ugró pont megérintése után az ábécé betűit tartalmazó felület jelenik meg, ahol az adott kezdőbetűt megérintve a lista adott kezdőbetűvel kezdődő elemeihez ugrik. A listából kiválasztott beteg adatai a 7.9-es ábra jobb oldali képén látható felületen jelennek meg.

7.10. ábra Páciensem felület az orvos Hub oldalán és a páciens lista

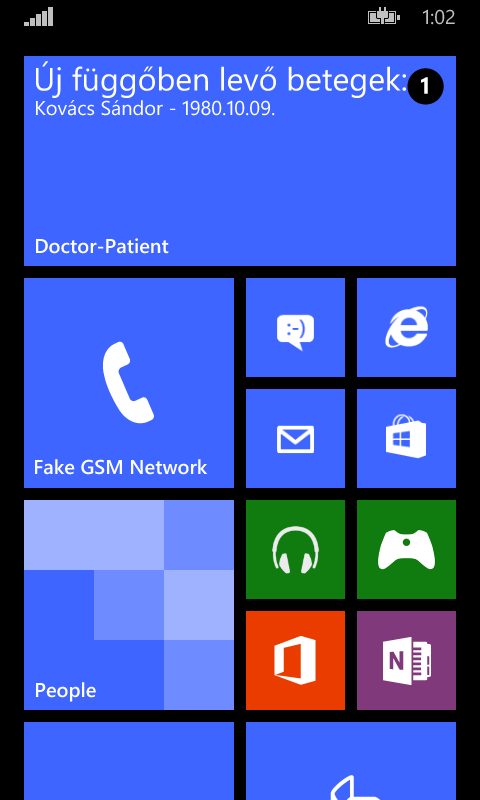
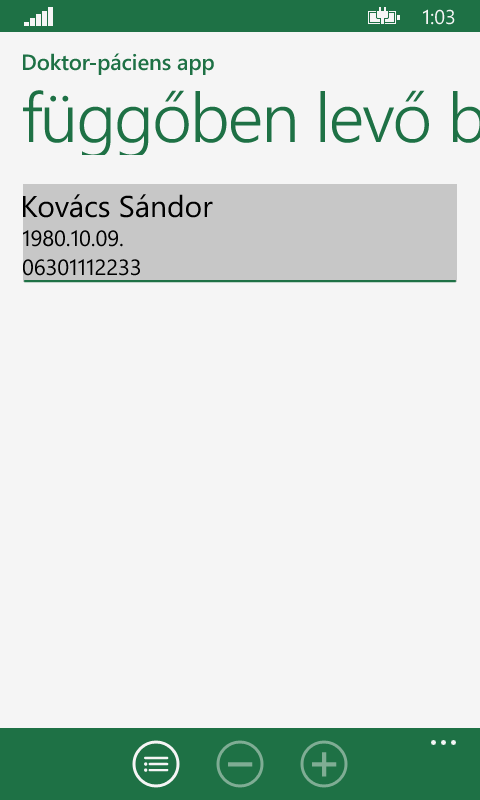
Az orvos Hub oldal harmadik képernyőjén az Űrlapok képernyő kapott helyet. Az oldal felépítését a 7.11-es ábra bal oldali képe szemlélteti. Ez a felület két részre oszlik, a felső részen találhatóak az űrlap sablonok. A sablonok alatt a páciensektől visszakapott űrlapok helyezkednek el. A sablonok a szerver adatbázisában tárolódnak és onnan tölti le az alkalmazás, így új sablon létrehozására nincs lehetőség az alkalmazásból. A sablonra pöccintve megjelenik az űrlap megjelenítő oldal, ahol megtekinthető a sablon. Az oldal alján található alkalmazás sávban van egy küldés gomb. A gombot megnyomva a 7.11-es ábra jobb oldali képén látható felület jelenik meg, ahol kiválaszthatja az orvos, hogy melyik páciensének szeretné elküldeni az űrlapot. Miután kiválasztotta a pácienst az orvos, az alkalmazás egy felugró üzenetben a küldés megerősítését kéri. Ha az igent választja, az űrlapot elküldi az alkalmazás. Mikor a páciens legközelebb bejelentkezik az alkalmazásba meg fog jelenni az űrlap az üdvözlő felületén.

A visszakapott űrlapok megjelenítése is hasonló módon történik. Az egyetlen különbség az, hogy az oldal alkalmazás sávjában küldés gomb helyett egy törlés gomb található. Ezt megnyomva az űrlap törlődik és a visszakapott űrlapok listából is eltávolításra kerül.

7.11. ábra Űrlapok oldal és űrlap címzett választó felület

Az alkalmazás több féleképpen is jelzi az orvosnak, hogy új beteg felkérése érkezett. Az egyik módját már ismertettem, amikor a Hub oldalának üdvözlő képernyőjén jelenik meg egy üzenet. A másik fajtáját a 7.12-es ábra bal oldalán látható. A készülék kezdőképernyőjére elhelyezett élő csempén jelenik meg az értesítés. Az élő csempe mind három méretén megjelenik az értesítés. A legkisebb változatában csak a függőben levő betegek száma jelenik meg. A közepes csempén egy szöveg is értesíti az orvost. A széles csempén maximum négy beteg neve és születési dátuma is megjelenik. A 7.12-es ábra bal oldali képén egy függőben levő betegről kap az orvos értesítést a széles csempén keresztül. A csempe félóránként frissíti magát.

7.12. ábra Függőben levő betegeket jelző csempe a kezdőképernyőn (bal), függőben levő betegek listája (jobb)

A függőben levő betegek listájához az üdvözlő képernyő üzenetén keresztül, vagy a pácienseim képernyőn keresztül is eljuthatunk. 7.12-es ábra jobb oldali képén egy darab függőben levő beteg látható. A listában levő elemeket ki lehet választani. Az alkalmazás sáv első gombjával az összes elem kiválasztásra kerül. Ha van kiválasztott lista elem, akkor a plusz és mínusz ikonú gombokkal lehet elfogadni vagy elutasítani a kiválasztott pácienseket.

A páciensek Hub oldalához hasonlóan az orvosok is frissíthetik az oldal tartalmát az alkalmazás sávon levő gombbal. Szintén megtekinthetik és módosíthatják a profiljukat az alkalmazás sáv második gombjával.

# Tesztelés

Ebben a fejezetben röviden bemutatom, hogyan teszteltem az alkalmazást a fejlesztés során.



## Fejlesztői környezet

A fejlesztést Windows 8.1-es operációs rendszeren, a Microsoft Visual Studio Ultimate 2013 fejlesztő környezetben végeztem. A fejlesztő környezet legfrissebb változatát használtam, ami a 12.0.31101.00 (Update 4) verzió számot viselte.

## Teszt szerver

Az alkalmazás funkcióinak megfelelő implementálásához és teszteléséhez szükségem volt egy szerverre, aminek segítségével tesztelni tudtam az alkalmazás adott funkcióit. Az alkalmazást helyi hálózaton futó szerveren és egy távoli elérésű szerveren is teszteltem.

A helyi hálózaton futó szerverhez a XAMP 1.8.3-as verziójú csomagját használtam, ami 5.5.15-ös verziójú PHP-t, 5.6.20-as verziójú MySQL szervert és 4.2.7.1-es verziójú phpMyAdmin felületet tartalmaz.

A teszt szerver HTTP kéréseit a Chrome böngésző Advanced REST Client bővítményével teszteltem.

## Alkalmazás teszt

A fejlesztés során az alkalmazást a Visual Studioban található Windows Phone 8.1-es emulátoron teszteltem. Az emulátor segítségével szimulálni tudtam különféle jel erősségű és sebességű mobil hálózatokat, hogy tesztelhessem a hálózati adatátvitelt.

A Visual Studioban elérhető teljesítmény és diagnosztikai eszközökkel is teszteltem az alkalmazást használat közben. Ezekkel az eszközökkel meg lehet vizsgálni az alkalmazás processzor, memória, energia felhasználását és felület válaszkészségét. Az alkalmazás minden ilyen teszten jól teljesített.

A Windows Phone Developer Power Tools eszközzel szintén lehet teljesítmény teszteket végezni, ezeket a teszteket rögzíteni is lehet. Az eszköznek van egy Application Verifier nevezetű része, ami a natív kódot figyeli futás közben, hibák, helytelen memória kezelés után kutatva.

Fizikai készüléken is teszteltem az applikációmat. A teszt készülék egy Nokia Lumia 620-as okostelefon volt, amin Windows Phone 8.1-nek a 8.10.14219.341-es verziója futott. Azért teszteltem egy valós készüléken is, mert itt előjöhetnek olyan hibák, amiket emulátorral nem lehet megtalálni.

Az elkészült alkalmazás csomagot a Windows App Certification Kittel is teszteltem. Ezzel a programmal érdemes letesztelni az alkalmazás mielőtt feltöltenénk az alkalmazás boltba. Az alkalmazásom sikeresen átment a tesztelőn.

# Összefoglalás

Szakdolgozatom elkészítése során létrehoztam egy Windows Phone 8.1-es alkalmazást, aminek segítségével a betegek rögzíthetik mért adataikat, amik alapján az orvosaik nyomon követhetik állapotukat.

A betegek rögzíthetik vérnyomás és vércukor értékeiket, amit a rendszer eltárol. Az eltárolt értékek visszanézhetők és grafikonon is megtekinthetőek. Az alkalmazás értesíti a beteg orvosát, ha a páciens veszélyes értéket rögzített. Vészhelyzet esetén az alkalmazásban található pánik gomb használatával a beteg gyorsan hívhat segítséget.

Az alkalmazás főcélja, hogy elősegítse a páciensek gyógyításhoz és gyógyuláshoz szükséges diagnosztikai és terápiás információk gyors célba jutását. Az alkalmazás használatával csökkenthető és optimalizálható a munkaterhelés, aminek köszönhetően nagymértékben elősegítheti a kezelések eredményességét, csökkentheti az ellátás költségét és javíthatja az orvosi munka minőségét.

Megvizsgáltam a Windows Phone platformon a jelenleg piacon kapható egészségügyi alkalmazásokat és azok hiányosságait. A vizsgálat eredménye alapján szükség lenne egy jó minőségű alkalmazásra, ami segítené a betegek gyógyulását és az orvosaik munkáját. Bízom benne, hogy az alkalmazásom hasznos segédeszközt biztosít a betegek és orvosaik számára.

# *Irodalomjegyzék*

1. AdDublex Windos Phone Status Report - October 2014.pdf Windows Phone 8 Manufacturers (5. oldal)
2. http://www.diabetes.hu/cikkek/hypertonia/0802/a-vercukor-merese TRIEBER L. (letöltés dátuma 2014. november 30.) *A vércukor mérése*
3. http://en.wikipedia.org/wiki/Windows\_Phone (letöltés dátuma 2014. november 24.) *Windows Phone*
4. http://en.wikipedia.org/wiki/Windows\_Phone\_8 (letöltés dátuma 2014. november 24.) *Windows Phone 8*
5. http://en.wikipedia.org/wiki/Windows\_Phone\_8.1 (letöltés dátuma 2014. november 24.) *Windows Phone 8.1*
6. http://hu.wikipedia.org/wiki/Magas\_v%C3%A9rnyom%C3%A1s (letöltés dátuma 2014. november 30.) *Magas vérnyomás*
7. http://www.idc.com/prodserv/smartphone-os-market-share.jsp IDC (letöltés dátuma 2014. november 23.) *Smartphone OS Market Share, Q2 2014*
8. http://james.newtonking.com/json/help/index.html Newton-King, J.

(letöltés dátuma 2014. november 21.). *Json.NET Documentation*

1. http://www.jmir.org/2014/4/e104/ ARNHOLD, M. (letöltés dátuma 2014. november 28). *Mobile Applications for Diabetics: A Systematic Review and Expert-Based Usability Evaluation Considering the Special Requirements of Diabetes Patients Age 50 Years or Older*
2. http://modernography.wordpress.com/2014/07/18/ready-to-use-list-controls-for-wp8-1-winrt/ Q. (letöltés dátuma 2014. november 10.). *Ready to Use List Controls for WP8.1*
3. http://msdn.microsoft.com/en-us/library/windows/apps/hh868184.aspx (letöltés dátuma 2014. november 25.) *Opening a developer account*
4. http://msdn.microsoft.com/en-us/library/windows/apps/jj863494.aspx (letöltés dátuma 2014. november 25.) *Account types, locations, and fees*
5. http://msdn.microsoft.com/library/windows/apps/dn726767.aspx (letöltés dátuma 2014. november 25.) *What's a Windows Runtime app?*
6. https://www.nuget.org/packages/NodaTime Skeet, J. (letöltés dátuma 2014. november 12.). *NodaTime 1.3.0d*
7. http://www.windowscentral.com/microsoft-claims-windows-10-upgrades-all-lumias (letöltés dátuma 2014. november 24.) *Microsoft claims Windows 10 upgrades for all Lumia Windows Phones*
8. http://wmpoweruser.com/a-history-of-windows-phone-the-road-to-threshold/ ALLISON, M. (letöltés dátuma 2014. november 24.) *A* *History of Windows Phone: The Road to Threshold*

# *Ábrajegyzék*

[3.1. ábra Windows Phone 7-es készülék 7](#_Toc406111966)

[3.2. ábra Windows Phone 8-as készülékek, balról jobbra: Nokia Lumia 920, Nokia Lumia 820, Nokia Lumia 620 10](#_Toc406111967)

[3.3. ábra Nokia Lumia 930 Windows Phone 8.1-el 13](#_Toc406111968)

[5.1. ábra Rendszer komponens diagramja 21](#_Toc406111969)

[5.2. ábra Adatbázis terv 23](#_Toc406111970)

[5.3. ábra Használtai eset diagram 25](#_Toc406111971)

[5.4. ábra Orvos választás aktivitás diagramja 27](#_Toc406111972)

[5.5. ábra Bejelentkezési és regisztrációs felületek vázlata 28](#_Toc406111973)

[5.6. ábra Bejelentkezési felület terve Blend for Visual Studio 2013-ban 29](#_Toc406111974)

[6.1. ábra Model View ViewModel 31](#_Toc406111975)

[6.2. ábra MessagesModel.cs osztályainak osztálydiagramja 41](#_Toc406111976)

[6.3. ábra FormModel.cs osztályainak osztálydiagramja 42](#_Toc406111977)

[6.4. ábra Params osztály szemléltetve a Code Map-en 43](#_Toc406111978)

[7.1. ábra Bejelentkezési oldal és bejelentkezési oldal hiba üzenettel 49](#_Toc406111979)

[7.2. ábra Beállítások oldal és regisztrációs folyamat első oldala 50](#_Toc406111980)

[7.3. ábra Orvosi regisztációs oldal és sikeres orvos regisztráció üzenet 51](#_Toc406111981)

[7.4. ábra Páciens kezdőképernyő üzenettel és az orvos választási oldal 52](#_Toc406111982)

[7.5. ábra Páciens kezdőképernyője orvos adataival és új űrlap üzenettel (bal), orvostól kapott űrlap megjelenítve (jobb) 53](#_Toc406111983)

[7.6. ábra Páciens vérnyomás és vércukor oldala 54](#_Toc406111984)

[7.7. ábra Vérnyomás grafikon ábrázolva 55](#_Toc406111985)

[7.8. ábra Segélyhívás oldal és annak csempéje a kezdőoldalon 56](#_Toc406111986)

[7.9. ábra Orvos üdvözlő oldala veszélyben levő betegekkel (bal), Páciens adatainak megtekintése (jobb) 57](#_Toc406111987)

[7.10. ábra Páciensem felület az orvos Hub oldalán és a páciens lista 58](#_Toc406111988)

[7.11. ábra Űrlapok oldal és űrlap címzett választó felület 59](#_Toc406111989)

[7.12. ábra Függőben levő betegeket jelző csempe a kezdőképernyőn (bal), függőben levő betegek listája (jobb) 60](#_Toc406111990)

# *Mellékletek*

A CD melléklet tartalma:

* Alkalmazás
  + Alkalmazás csomagok.rar
  + DoctorPatientApp.rar
* Teszt szerver
  + doctorpatient.sql
  + htdocs.rar
* Ábrák.rar
* Forrás.rar
* Doctor-Patient App Information Architecture.pdf
* Microsoft Limited Public License.rtf
* űrlap minta.json
* Szakdolgozat\_Mód\_Márton\_Z127H9.docx
* Szakdolgozat\_Mód\_Márton\_Z127H9.pdf

1. NuGet egy csomagkezelő Microsoft fejlesztői platformjai számára, www.nuget.org [↑](#footnote-ref-1)
2. További részletek a mellékletben [↑](#footnote-ref-2)