
**UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,
TÎRGU-MUREȘ
SPECIALIZAREA CALCULATOARE**

Stroke Monitor

PROIECT DE DIPLOMĂ

Coordonator științific:
Dr. Szántó Zoltán

Absolvent:
Szász Arnold-Levente

2021

LUCRARE DE DIPLOMĂ

Coordonator științific:
ș.l. dr. ing. Szántó Zoltán

Candidat: **Szász Arnold-Levente**
Anul absolvirii: **2021**

a) Tema lucrării de licență:

Dezvoltarea unui sistem pentru monitorizarea pulsului cu ajutorul unei aplicații android și un dispozitiv tip fitbit

b) Problemele principale tratate:

- Studiu bibliografic privind comunicarea între telefon și fitbit
- Studiu bibliografic privind serviciile Android
- Studiu bibliografic privind realizarea accidentul vascular cerebral, din punct de vedere medical
- Crearea unei interfețe prin care datele capturate de fitbit pot fi citite
- Dezvoltarea unei aplicații pe smartphone care prezintă și gestionează datele capturate
- Testarea sistemului în condiții reale

c) Desene obligatorii:

- Schema bloc al aplicației
- Diagrame UML privind software-ul realizat.

d) Softuri obligatorii:

- Aplicație în flutter
- Aplicație smartwatch
- Colectare și stocare a valorilor ritmului cardiac într-o bază de date
- Configurarea și cererea acces la Fitbit Web API

e) Bibliografia recomandată:

- [1] Services overview <https://developer.android.com/guide/components/services>
- [2] Send and sync data on Wear <https://developer.android.com/training/wearables/data-layer>
- [3] Tison, Geoffrey H., et al. "Passive detection of atrial fibrillation using a commercially available smartwatch." *JAMA cardiology* 3.5 (2018): 409-416.
- [4] Lawrie, Sophie, et al. "Evaluation of a smartwatch-based intervention providing feedback of daily activity within a research-naïve stroke ward: a pilot randomised controlled trial." *Pilot and feasibility studies* 4.1 (2018): 157.
- [5] Wasserlauf, Jeremiah, et al. "Accurate Detection and Quantification of Atrial Fibrillation Using a Smartwatch With ECG Watchband." *Circulation* 138.Suppl_1 (2018): A15366-A15366
- [6] Wasserlauf, Jeremiah, et al. "Smartwatch performance for the detection and quantification of atrial fibrillation." *Circulation: Arrhythmia and Electrophysiology* 12.6 (2019): e006834.
- [7] Micallef, Nicholas, Lynne Baillie, and Stephen Uzor. "Time to exercise! An aide-memoire stroke app for post-stroke arm rehabilitation." *Proceedings of the 18th international conference on Human-computer interaction with mobile devices and services*. 2016.
- [8] Android Development with Kotlin by M. Moskala, I. Wojda, 2017

f) Termene obligatorii de consultații: săptămânal

g) Locul și durata practicii: Universitatea Sapiientia,
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș

Primit tema la data de: 12.05.2020

Termen de predare: 28.06.2021

Semnătura Director Departament

Semnătura coordonatorului

**Semnătura responsabilului
programului de studiu**

Semnătura candidatului

Declarație

Subsemnatul Szász Arnold-Levente, absolvent a specializării Calculatoare, promoția 2021 cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență de diplomă se bazează pe activitatea personală, proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea,

Data:

Absolvent

Semnătura.....

Declarație

Subsemnatul Szántó Zoltán, funcția șef lucrări, titlul științific dr. ing. declar pe propria răspundere că Szász Arnold-Levente, absolvent al specializării Calculatoare a întocmit prezenta lucrare sub îndrumarea mea.

În urma verificării formei finale constat că lucrarea de licență corespunde cerințelor de formă și conținut aprobate de Consiliul Facultății de Științe Tehnice și Umaniste din Târgu Mureș în baza reglementărilor Universității Sapientia. Luând în considerare și Raportul generat din aplicația antiplagiat „Turnitin” consider că sunt îndeplinite cerințele referitoare la originalitatea lucrării impuse de Legea educației naționale nr. 1/2011 și de Codul de etică și deontologie profesională a Universității Sapientia, și ca atare sunt de acord cu prezentarea și susținerea lucrării în fața comisiei de examen de licență.

Localitatea,

Data:

Semnătura îndrumătorului

Ide kerül a Turnitin similarity report

Stroke Monitor

Extras

În zilele noastre, datorită descoperirilor tehnologice, telefoanele mobile au o capacitate de calcul neimaginat în trecut. De lângă apeluri telefonice, și mesagerie aceste dispozitive au multe funcții noi. Pentru atingerea experiența utilizatorului la cel mai înalt nivel, sunt necesare o mulțime de senzori și module de comunicație. Pe lângă telefoane pot fi asociate diverse accesorii dotate cu senzori noi, astfel este posibil monitorizarea continuă ale semnelor vitale.

Scopul acestei lucrări de diplomă este dezvoltarea unui sistem, care să beneficieze de posibilitățile oferite de senzori. Aplicația StrokeMonitor include două software diferite, care constă dintr-o aplicație telefonică și una de smartwatch, interacțiunea lor permițând utilizatorului să își monitorizeze ritmul cardiac chiar și în afara spitalului. Scopul monitorizării este informarea utilizatorului și a persoanei de contact în cazul unui risc posibil de accident vascular cerebral care este implementat pe baza datelor furnizate de smartwatch și procesate de aplicația telefonică. Are un rol important în cazul utilizatorilor vulnerabili, fiindcă măsurile de urgență medicale precoce reduce sechelele post accident vascular cerebral. De asemenea aplicația telefonică este capabilă să afișe diferite grafice, să determine scorul de risc, să ofere sfaturi zilnice pentru prevenirea riscului de accident vascular cerebral. Pe parcursul dezvoltării sistemului, am folosit tehnologia flutter, astfel încât aplicația este independentă de platformă.

Lucrarea de diploma include definiția accidentului vascular cerebral și recunoașterea acestuia pe baza ritmului cardiac, proiectarea, implementarea și dezvoltarea aplicațiilor telefonice și de smartwatch, precum și testarea sistemului în condiții reale.

Cuvinte cheie: Accident vascular cerebral, monitorizare, aplicație telefonică, aplicație smartwatch

**SAPIENTIA ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR
SZÁMÍTÁSTECHNIKA SZAK**

Stroke Monitor

DIPLOMADOLGOZAT

**Témavezető:
Dr. Szántó Zoltán**

**Végzős hallgató:
Szász Arnold-Levente**

2021

Kivonat

Napjainkban a technológiai áttöréseknek köszönhetően, a mobiltelefonok olyan számítási kapacitással rendelkeznek, amely régebb elképzelhetetlen volt. Rengeteg új funkciót látnak el ezek az eszközök a telefonhívás, üzenetküldés mellett köszönhetően a bennük található technológiai vívmányoknak. Rengeteg szenzor, különböző kommunikációért felelős modul dolgozik azon, hogy a felhasználói élmény a legmagasabb szinten legyen. Továbbá a telefonok mellé különböző kiegészítő tartozékok társíthatóak, amelyek újabb érzékelőkkel vannak ellátva, így képesek akár a felhasználó életjeleit is mérni a nap 24 órájában.

Ennek hatására a dolgozat célkitűzése egy olyan rendszer kialakítása, amely képes felhasználni a szenzorok által nyújtott lehetőségeket. A StrokeMonitor nevű alkalmazás két szoftvert foglal magába, egy telefonos és egy okosóra alkalmazást, ezek kölcsönös működése lehetővé teszi a felhasználó szívritmusának monitorizálását akár otthoni környezetben. A megfigyelés célja a felhasználó, illetve a kontakt személy értesítése esetleges stroke veszély esetén, amelyet az okosóra által szolgáltatott, majd a telefonos alkalmazás által feldolgozott adatok alapján valósít meg. Fontos szerepet tölthet be veszélyeztetett felhasználók esetén, mivel, ha fenáll a stroke lehetősége fontos a mihamarabbi orvosi ellátás, ezzel tovább növelve a túlélés esélyeit, illetve a gyors beavatkozás csökkentheti a maradandó károsodások esélyét. A telefonos alkalmazás továbbá képes különböző kimutatásokra, rizikó számok meghatározására, napi tanácsokkal látja el a felhasználót a stroke kockázatának csökkentése céljából. A rendszer fejlesztése során Flutter technológiát használtunk, így az alkalmazás platform független.

A dolgozatban szó lesz a strokeról illetve annak felismeréséről pulzus szám alapján, telefonos illetve okosóra alkalmazás tervezésről, kivitelezésről, fejlesztésről, továbbá a szoftver felhasználásával történő kísérletekről.

Kulcsszavak: Stroke, megfigyelés, telefonos alkalmazás, okosóra alkalmazás

Abstract

Today, thanks to technological breakthroughs, mobile phones have a computing capacity unimaginable in the past. In addition to phone calls and messaging, these devices have many new features. To achieve user experience at the highest level, they need a lot of sensors and communication modules. In addition, phones can be equipped with various accessories associated with new sensors, making it possible to continuously monitor vital signs.

The purpose of this diploma work is to develop a system that would benefit from the possibilities offered by sensors. The StrokeMonitor application includes two different software, which consists of a telephone application and a smartwatch application, their interaction allowing the user to monitor their heart rate even outside the hospital. The purpose of monitoring is to inform the user and the contact person in case of a possible risk of stroke which is implemented based on the data provided by the smartwatch and processed by the telephone application. It plays an important role for vulnerable users, because early medical emergency measures reduce the aftermath of stroke. The telephone application is also able to display various graphs, determine the risk score, provide daily advice to prevent the risk of stroke. During the development of the system, we used flutter technology, so the application is platform independent.

The diploma thesis includes the definition of stroke and its recognition based on heart rate, design, implementation and development of telephone and smartwatch applications, as well as testing the system in real conditions.

Keywords: stroke, monitoring, mobile app, smartwatch app

Tartalomjegyzék

Ábrák jegyzéke	12
Kódrészletek jegyzéke	13
1. Bevezető	14
2. Célkitűzések	16
3. Elméleti megalapozás és bibliográfiai tanulmány	17
3.1. Orvostudományi háttér.....	17
3.2. Orvostudományban napi szinten alkalmazott eszközök.....	18
4. Követelmény specifikáció	22
4.1. Felhasználói követelmények	22
4.2. Rendszer követelmények	23
4.2.1. Funkcionális követelmények	23
4.2.1. Termékkel kapcsolatos nem funkcionális követelmények	24
4.2.2. Külső nem funkcionális követelmények.....	24
5. A rendszer leírása.....	26
5.1. Architektúra	26
5.2. Okostelefon applikáció.....	26
5.3. Okosóra applikáció.....	28
5.4. Adatbázis	28
5.5. Elvi diagram	30
5.6. UI diagram	31
5.7. Fitbit applikáció	45
6. Üzembe helyezés	50
7. Kísérletek	54
7.1. Egyéni kísérletek	55
7.2. Csoportos kísérlet	57
8. Következtetések.....	59
8.1. Megvalósítások	59
8.2. További fejlesztési irányok	59
9. Irodalomjegyzék.....	61

Ábrák jegyzéke

Ábra 1. Rizikó pontok jelentése	18
Ábra 2. Rendszer architektúra [5]	20
Ábra 3. Használati eset diagram.....	23
Ábra 4. Rendszer architektúra	26
Ábra 5. Mobil alkalmazás architektúra	27
Ábra 6. Okosóra applikáció architektúra.....	28
Ábra 7. Firebase Authentication.....	29
Ábra 8. Cloud Firestore szerkezet	29
Ábra 9. Elvi diagram	30
Ábra 10. Monitorizálás elvi diagram.....	31
Ábra 11. Telefonos alkalmazás UI terv	31
Ábra 12. Fitbit engedély kérelem	41
Ábra 13. Fitbit Alkalmazás alegységei	46
Ábra 14. Fitbit óra hozzáadása	50
Ábra 15. Fitbit StrokeMonitor telepítés	51
Ábra 16. Fitbit óraszám lap beállítása	52
Ábra 17. StrokeMonitor kezdő lépések	52
Ábra 18. StrokeMonitor használat	53
Ábra 19. Okosóra összehasonlítása vérnyomásmérővel százalékos eltérés.....	54
Ábra 20. Okosóra összehasonlítása vérnyomásmérővel	54
Ábra 21. Gyors szaladás közben pulzus változás.....	55
Ábra 22. Kerékpározás közben pulzus változás.....	56
Ábra 23. Gimnasztika közben pulzus változás.....	57
Ábra 24. Csoportos kísérlet.....	58

Kódrészletek jegyzéke

Kódrészlet 1. Navigáció	32
Kódrészlet 2. Bemeneti mező.....	33
Kódrészlet 3. Bejelentkezés gomb	33
Kódrészlet 4. Bemeneti adatok validálása.....	33
Kódrészlet 5. Regisztráció vagy beléptetés	34
Kódrészlet 6. Regisztrációs adatok beszúrása.....	34
Kódrészlet 7. Menüsáv oldalak	35
Kódrészlet 8. Menüsáv inicializáció	35
Kódrészlet 9. Szendvics menü inicializáció	36
Kódrészlet 10. Útvonal.....	37
Kódrészlet 11. Adatok letöltése és megjelenítése	37
Kódrészlet 12. Üzenet küldés, telefon hívás	38
Kódrészlet 13. Rizikó szám frissítés	39
Kódrészlet 14. Kijelentkezés	39
Kódrészlet 15. Fitbit autentikáció	39
Kódrészlet 16. Android Manifest	40
Kódrészlet 17. Visszacsofolást végző weboldal.....	41
Kódrészlet 18. Fitbit Web API hívás	42
Kódrészlet 19. Grafikon stílus, adatfeltöltés	43
Kódrészlet 20. Háttér szolgáltatás	44
Kódrészlet 21. Monitorizálás	45
Kódrészlet 22. Fitbit számlap.....	46
Kódrészlet 23. Elemek beazonosítása	47
Kódrészlet 24. Pulzus esemény.....	47
Kódrészlet 25. Üzenet küldés	48
Kódrészlet 26. Fitbit beállítások.....	49
Kódrészlet 27. Fitbit egyedi azonosító meghatározása	49

1. Bevezető

A technológiák gyors mértékű fejlődésének köszönhetően rengeteget változott a világ száz év alatt. Széles körben, szinte a világ minden pontján elérhető az elektromos áram, az internet, a műholdas helymeghatározás, és ez olyan lehetőségeket tár fel, amely régebb elképzelhetetlen volt. A távolságok lecsökkentek, pár másodperc alatt tömérdek információt lehet elérni, ami addig csak óriási könyvtárakban, levéltárakban volt elérhető. A régen napokig, hetekig tartó üzenetküldés, ma már pár másodperces művelet, mivel rengeteget fejlődött a technológia és az egykor szoba méretű számítógépek, ma zsebben elférő kis eszközökké zsugorodtak.

Az utolsó generációs eszközök igen komoly számítási kapacitással rendelkeznek, így bonyolult feladatok elvégzését is jásnyi könnyedséggel hajtják végre. Rengeteg szenzor, érzékelő található meg az eszközökben, illetve ezekhez nagy eséllyel társulnak kiegészítők, amelyek még másabb és specifikusabb érzékelőkkel vannak ellátva, ilyen eszköz például az okosóra. A bennük rejlő szenzorok igen nagy pontossággal képesek mérni bizonyos értékeket, ilyen érték lehet, fényerő, hang, gravitáció vagy az okosórák többsége képes mérni az elégetett kalóriát, a pulzusszámot.

Rengeteg szoftver tartozik hozzájuk, melyek nyomon követik a felhasználó szokásait, megfigyelve életjeleit alvás közben, figyelmeztet a mozgáshiányra, a kiszáradásra. Mivel ezek az eszközök óriási körben elterjedtek, szinte minden ember nap mint nap magával viszi mindenhol, ezt kihasználva akár orvosi, megfigyelői célokra is fel lehet használni a szenzorok által mért adatokat, megkönnyítve az orvosok munkáját, növelve az életszínvonalat.

A betegségek sokszínűsége miatt, egyre többen szorulnak folyamatos megfigyelésre, amely kórházi körülmények közt megoldható feladat, mert ott rendelkezésre állnak igen drága megfigyelő eszközök, azonban a páciens nem mindig engedheti meg magának, hogy saját eszközt vásároljon. Itt jöhet számításba a mobiltelefonok és kiegészítői által nyújtott lehetőségek. Elég pontos adatokkal szolgálnak és akár 24 órás megfigyelést képes ellátni a páciens mindennapi tevékenységei alatt. Természetesen ezek az eszközök nem érnek fel a kórházakban talált eszközökhöz, azonban pontosságuk nem nagy arányban tér el egy profi eszköztől. Az ilyen rendszerek képesek akár több típusú betegség felismerésére. Néhány esetben az idő, ami alatt a páciens eljut a rosszullét helyétől a kórházig igen kritikus, sokszor percek múlásával az élet vagy a halál, ezért is fontos, hogy a veszélyeztetett páciensek folyamatos megfigyelés alatt legyenek, hogy vész esetén minél előbb megfelelő ellátásban részesüljenek, ezzel is növelve a túlélési

esélyeiket, csökkentve a tartós károsodás kockázatát. Az általam létrehozott szoftver is ezt a technológiát próbálja kihasználni, monitorizálva a felhasználót.

2. Célkitűzések

Célunk egy olyan rendszer megtervezése és kivitelezése, amely képes a felhasználót napi 24 órában megfigyelni, figyelmeztetve az esetleges stroke bekövetkezésére. Ehhez szükséges egy telefonos applikáció, amely a feldolgozó egység szerepét tölti be. A felhasználó képes bejelentkezni, ezzel elérve a saját, eddig mért adatait, ezt megtekintheti több idő leosztásba grafikonok segítségével, beállíthatja a vész esetén tárcsázandó személy számát, napi tanácsokkal látja el a megelőzést tekintetbe, illetve az egészségi állapotára kiterjedő kérdőívet tölthet ki. A telefonos applikáció folyamatos kapcsolatban áll egy okos órával. Az okos órára írt applikáció feladata az eszköz által mért adatokat leolvasni, illetve elmenteni egy erre a célra használt tároló egységre. Az adat jelen esetben a felhasználó pulzusszáma, amelyet a telefonos applikáció percenként ellenőriz, valós időben elérve a tároló egységről. Összehasonlítja egy szakértők által kijelentett küszöbértékkel. Ha a mért érték meghaladja a kritikus szintet, akkor figyelmezteti a felhasználót, illetve automatikusan értesíti a kontakt személyt, amelyet előzetesen beállított a felhasználó. Az applikáció páros folyamatos működés mellett védelmet nyújthat a szélütéssel szemben, felismerve annak korai tüneteit, jeleit.

3. Elméleti megalapozás és bibliográfiai tanulmány

Az irodalomkutatás több részre osztható. Első sorban meg kell vizsgálni az orvostudományi szempontokat, miként ismeri fel az applikáció a betegséget, milyen időközönként kell az adatfeldolgozást elvégezni, másrészt hasonló rendszereket kell tanulmányozni a téma jobb megértése érdekében.

3.1. Orvostudományi háttér

A stroke (szélütés) vérkeringési zavar miatt hirtelen kialakuló agyi károsodás, amelynek következtében az érintett agyterület nem kap elégséges vért a létfontosságú funkciók fenttartásához. Leggyakoribb formája, az esetek mintegy 80%-ban az iszkémiás (vértelen) stroke, amely tulajdonképpen érelzáródással járó agyi infarktus. Az iszkémiás stroke [1] kialakulásának két oka ismert: az egyik az agy ereiben lerakódott ateroszklerotikus plakkok (trombus) miatt történő elzáródás, a másik pedig a bal pitvar gyenge és szabálytalan működése (pitvarfibrilláció) következtében kialakult vérrögök sodródása majd fentakadása az agy kis méretű ereiben (embólia), amely szintén elzáródáshoz vezet. Számos tanulmányban olvashatunk a pitvar fibrilláció és stroke közötti összefüggésről, amelynek során arra a következtetésre jutunk, hogy a pitvarfibrilláció stroke-ot okozhat, de a stroke is okozhat pitvarfibrillációt, valamint a szív ezen ritmuszavara szoros összefüggésben áll más stroke-ot okozó rizikófaktorokkal. Tehát kijelenthetjük, hogy egymást kiváltó és fenttartó folyamatokról beszélünk [1] [1]. Pitvarfibrilláció során sérül a sinus-csomó regulációs funkciója, megszűnik a pitvarok falának szabályos kontrakciója, amit egy gyors, szabálytalan remegő mozgás vált fel, ennek következményeként a kamrák fala sem tudja fenttartani szabályos ritmusú összehúzódásait. A pitvarok rendezetlen és erőtelen összehúzódásának következtében a véráramlás lelassul és örvénylővé válik, amely igencsak kedvez a trombusok (vérrögök) kialakulásához. A bal pitvar falán kialakult vérrögök a szapora, rendszertelen remegő mozgás miatt leszakadhatnak, így a vérárammal az agyba jutva érelzáródást (embóliát) okozhatnak. Sok esetben a pitvarfibrilláció tünetmentes, de jelentkezhet palpitáció (heves, rendszertelen szívdobogás), enyhe mellkasi diszkomfort érzés szorító-feszítő jelleggel, légszomj, fáradékonyság, szédülés, ájulásérzés. A nyaki ütőér vagy a csukló radiális artériája szintjén tappintható nyugalmi pulzus gyors és szabálytalan, ritmusa meghaladja a 140-160 ütés / percet (normál nyugalmi szívverés = 60-100 ütés/perc) [2]. Mint minden más betegségnek a stroke-nak is számos rizikófaktora van, amelyek jelentősen növelik a kialakulásának kockázatát. Némely kockázati tényező befolyásolható, ilyen például a magasvérnyomás, dohányzás, ritmuszavarok

(főleg a pitvarfibrilláció), cukorbetegség, magas vérzsírszint (főleg triglicerid), mozgásszegény életmód, elhízás, de vannak olyan rizikófaktorok is, amelyeket sajnos nem tudunk befolyásolni, mint az életkor (10%-a a 80 év feletti embereknek érintett), nem (sokkal gyakoribb a férfiaknál) és genetika (gyakran előfordul családi halmozódás). Ezen rizikófaktorok monitorizálása segít a stroke megelőzésében és az évi kockázati ráta megbecslésében. Az orvostudományban erre szolgál a CHA2DS2-VASC-Score rizikó osztályozás [4], amely a következőképpen néz ki:

- pangásos szívelégtelenség/ balszívfél elégtelenség: 1 pont
- magasvérnyomás: 1 pont
- cukorbetegség: 1 pont
- kórelőzményben átmeneti keringészavar (TIA), stroke vagy tromboembólia: 2 pont
- érbetegség (kórelőzményben szívinfarktus, perifériás érbetegség, aorta ateroszklerózis): 1 pont
- életkor 65-74 év: 1 pont
- életkor ≥ 75 év: 2 pont
- nem (nő): 1 pont

Minden pontszám összeadásával kapjuk meg a végső értéket, amely minél nagyobb annál magasabb az évi kockázata a stroke kialakulásának [3,4]. A legmagasabb pontnál 18.2 % az esély arra, hogy stroke alakuljon ki egy év alatt.

CHADS2 Score	Adjusted Stroke Risk (%) [4]
0	1.9
1	2.8
2	4
3	5.9
4	8.5
5	12.5
6	18.2

Ábra 1. Rizikó pontok jelentése

3.2. Orvostudományban napi szinten alkalmazott eszközök

Az irodalomkutatás során több hasonló rendszer dokumentációja, leírása került tanulmányozásra. Ezen rendszerek okos eszközök felhasználását mutatják be orvosi célokra,

felhasználva a bennük rejlő potenciált, hiszen a technológiai innovációknak hála, olcsón, pontos szenzorokkal ellátott eszközöket lehet vásárolni, mely kiváló kórházon kívüli megfigyelésre.

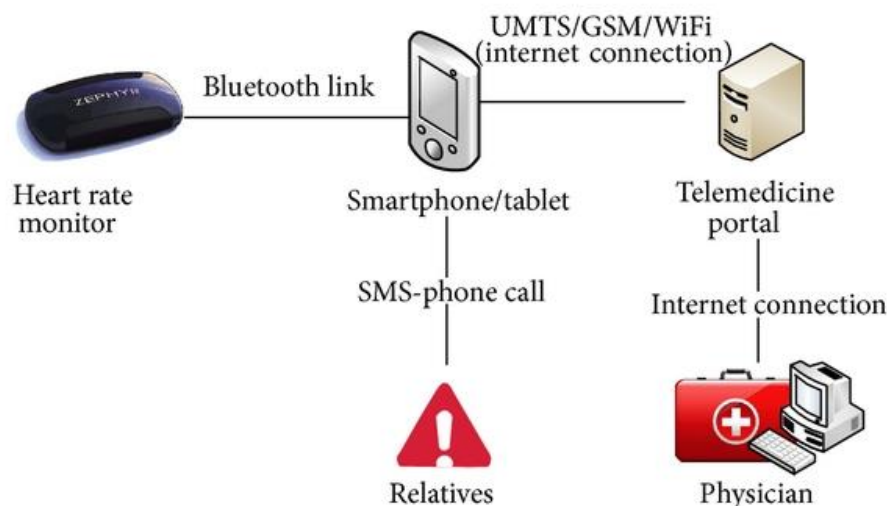
Az [5] tanulmány alapján folyamatos fejlődés mutatkozik az orvosi eszközök nem csak kórházban történő használatára. Így a sportolók által használt különböző szenzorokkal felszerelt okos eszközök teret nyernek az orvosi és diagnosztikai szegmensekben. A cikkben leírt rendszer fő szempontja az olcsón kivitelezhető rendszer, ami a pulzus adatok feldolgozását hajtja végre. A rendszer főbb funkcionálisai:

- részletes jelentések a felhasználó egészségügyi állapotáról
- adattárolás
- segélyhívások generálása
- távorvoslás adatok megosztásával

A legújabb tudományos fejlesztések lehetővé teszik, az orvosi eszközök kimozdítását a megszokott kórházi környezetből, mivel az okostelefonok és a biometrikus érzékelők már széles körben elérhetőek. Fontos cél a szívbetegsége monitorizálása s az esetleges betegségek megelőzése. Az ilyen betegségek felfedezésére nagyrészt EKG rendszereket használnak, de azt sok esetben nem lehet használni kórházi környezet nélkül. Ezért a leírt rendszer a sokkal egyszerűbb és olcsóbb pulzusmérőt (HR = heart rate) használja és a következőkre képes:

- stressz teszt, a pulzus változékonyság és a pillanatnyi pulzus alapján
- aritmia osztályozása, RR csúcs intervallum alapján
- energiafogyasztás pillanatnyi és nyugalmi pulzus alapján

[1]Az alkalmazás fő tevékenysége: adatfeldolgozás, megjelenítés, vészhelyzeti automatikus segélyhívások, illetve a különböző vezérlők is ez alapján vannak megvalósítva. Adatgyűjtés okostelefon vagy táblagép segítségével történik mely Bluetooth kapcsolat révén begyűjti az információkat. A stresszteszten 20 egészséges egyén vett részt (6 nő és 14 férfi, 23-27 év között). 10 percig elemezte őket az algoritmus. Kitöltöttek egy validált 10 kérdésből álló stressz tesztet, ez alapján a csoport pontszáma szignifikáns. Az aritmia teszt során 25 egészséges egyén vett részt 23-29 év között. Melyből 18-at normál, 5 alacsony és 2 magas pulzussal rendelkezett. A teszt EKG jelenlétében volt, ami igazolta ezeket az eredményeket.



Ábra 2. Rendszer architektúra [5]

A [6] tanulmány arra próbált rámutatni, hogy egyes eszközök mennyire precízek orvosi viszonylatba a pulzus mérésénél. A [6] cikk szerzői összesen 17 eszközt hasonlítottak, mérve a pontosságot és teljesítményt összehasonlítva pontos orvosi eszköz adataival. A tanulmány minden résztvevője (mind férfi, átlagéletkor 26.5 év) különféle teszteken esett át, egyidejűleg három gyorsulásmérőt használtak, ezeket egyidejűleg futtatták egy Android vagy iOS eszközön, közben kórházban is használt eszközök mérték és feldolgozták az adatokat. A teszt nagy volumenű volt, hiszen minden résztvevő a tesztet 40x hajtotta végre, 200, 500 és 1000 lépéssel. Minden teszt végén a lépések, illetve a mért pulzus és véroxigén telítettség mentésre került. A kontroll eszköz az Onyx Vantage 9590 profi klinikai pulzoximéter volt. A vizsgált termékek pontossága 79,8% és 99,1% között volt, míg a variációs együttható (pontosság) 4% és 17,5% között volt. Így kijelenthető, hogy a legrosszabb eszköz is a valós mért értéktől maximum 15%-al tért el.

Kritikus esetekben a betegeket folyamatos SPO2 (véroxigén szint), pulzusszám és hőmérséklet megfigyelés alatt kell tartani, amihez szükséges az orvosi személyzet folyamatos jelenléte. Sok esetben a páciens kórtörténete, előzetesen mért adatai nem jeleníthetők meg. Az Android based health care monitoring system [7] munkában leírt rendszer ezen dolgok kiküszöbölésén dolgozik új módszerekkel. A rendszer a beteg testének különféle biológiai paramétereinek, például pulzusszám, véroxigén telítettség, hőmérséklet folyamatos figyelését, teszi lehetővé. Az adatok feldolgozásra kerülnek egy webszerver és Android alkalmazás segítségével, ahol az orvos okostelefonján folyamatosan megfigyelheti a beteg egészségügyi állapotát. A leírt rendszer egy valós idejű megfigyelő rendszer, amely képes mérni és rögzíteni a

következő adatokat: vérnyomás, EKG, véroxigén szint. Ehhez kapcsolódik egy Android alapú applikáció mely képes meghatározni a páciens helyzetét, képes vész esetén értesíteni a mentőszolgálatot, vagy a páciens jelzésére is reagál értesítve az orvost. A rendszer alacsony komplexitással, energiafogyasztással rendelkezik, de magas potenciál rejlik benne a páciensek egészségügyi megfigyelésénél, mivel az orvos könnyedén akár otthonról megfigyelheti a beteg állapotát.

Az okostelefon-alapú technológiák és a széleskörű hozzáférhetőségük megváltoztatják a modern kardiológia gyakorlatának módját. A folyamatosan fejlődő technológiának köszönhetően egyre nagyobb eséllyel fedezhető fel a szív- és érrendszeri betegségek sokasága, amely nagyban hozzájárul ezek megelőzésére. A *Use of smartphone technology in cardiology* [8] cikk az okostelefon technológiát használó kardiológiai alkalmazások átfogó megbeszélését tartalmazza. Folyamatos, több paraméteres egészségügyi monitorizálásra azért van szükség, mert a kórházakban azon páciensek nagy része elhalálozik, akiket nem figyel valamilyen élet funkciót követő rendszer. A szegényebb országokban, vagy akár sok sérült esetén nem mindig van lehetősége a kórháznak elegendő, nagyon drága eszközt biztosítani, ekkor jön előtérbe az okostelefonokban rejlő potenciál, hiszen az új eszközök rengeteg szenzorral vannak felszerelve, melyek fontos méréseket végezhetnek. A megelőzés terén is fontos szerepet játszhat, hiszen a szív- és érrendszeri betegségekhez tartozó rizikófaktorok 80%-át a helytelen életmód adja. Az okos eszközök segítségével beállítható vagy legalábbis nyomon követhető az emberek testsúlya, cukor és fehérje bevitele, illetve követhető az adott helyzetben lévő páciens vérnyomása, véroxigén szintje, illetve pulzusa. Ezen adatok tárolásával és feldolgozásával az orvosoknak könnyebb dolguk van a diagnózis felállításában és a megfelelő kezelés kiválasztásában. Minden esetben, ha felmerül, hogy egy eszköz képes lehet orvosi eszközök kiváltására, szükség van egy helyesség ellenőrzés eljárásra, illetve egy pontosság mérésre, hiszen nem minden eszköz képes a megfelelő határokon belül mérni. Az okostelefonoknak megvan minden esélyük arra, hogy forradalmasítsák a kardiológiai gondviselést. Ez a technológia szinten minden esetben elérhető, segíthet a felhasználók jó útra terelésében a rizikófaktorok csökkentése érdekében, illetve segítenek az esetleges betegségei mihamarabbi felfedezésében, hiszen ezeknél a legfontosabb a mihamarabbi észlelés és kezelés.

4. Követelmény specifikáció

A rendszer különböző specifikációknak kell megfeleljen, mind felhasználói, mint rendszer szinten annak érdekében, hogy a lehető legoptimálisabb módon működjön, ezzel biztosítva, hogy minden a megrendelő igényei szerint történjen.

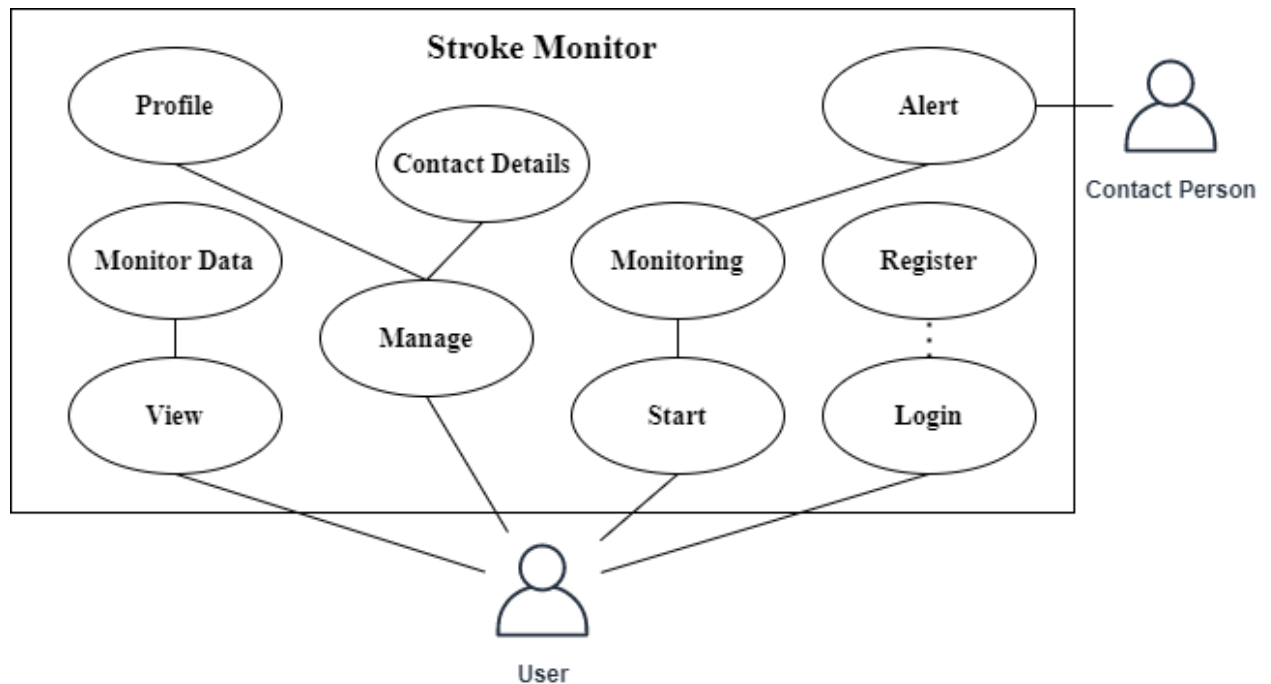
4.1. Felhasználói követelmények

A 3. ábrán található a használati eset diagram, amely a rendszer modellezésére szolgál a felhasználó szemszögéből. Megjelenik milyen szerepkörök kerülnek kontaktusba a rendszerrel, illetve ezek miként veszik igénybe a rendszer által nyújtott szolgáltatásokat. A diagram a rendszernek egy kiragadott részét írja le a felhasználó szemszögéből.

A felhasználó különböző műveleteket hajthat végre a rendszeren. Regisztrálhat, vagy meglévő fiókkal bejelentkezhet. Ezt követően olyan funkciók válnak elérhetővé, amelyet csak aktív felhasználói viszonyal rendelkező felhasználó érhet el. A felhasználó elindíthatja a megfigyelés funkciót monitorizálva percenként pulzusát. A monitorizálás során, ha az eszköz úgy véli, hogy a felhasználó veszélybe lehet, akkor értesíti az előzetesen megadott kontakt személyt akár üzenet küldés, akár telefonhívás formájában. A felhasználó továbbá menedzselheti a saját és a kontakt személy adatait. Megnézheti a monitorizálás alatt mentett pulzus adatokat különböző formátumba, többféle időbeosztásban.

Az összes ábra a draw.io¹ különböző ábrák létrehozására alkalmas weboldalas szerkesztő segítségével készült.

¹ <https://www.draw.io>



Ábra 3. Használati eset diagram

4.2. Rendszer követelmények

A rendszer követelményeket két alcsoportra lehet osztani. A funkcionális követelmények leírják, hogyan kellene működjön a rendszer, milyen funkcionalitásokkal kell rendelkezzen. A nem funkcionális követelmények a működtető rendszer specifikációit szögezi le.

4.2.1. Funkcionális követelmények

A regisztráció során a felhasználó meg kell adja különböző adatokat, mint email cím, jelszó, fizikai jellemzők (testsúly, magasság, nem), illetve el kell fogadja a szoftver felhasználási feltételeit. Sikeres regisztráció esetén az applikáció átnavigál a főoldalra. A bejelentkezéshez szükséges megadni egy olyan jelszó és email cím kombinációt, ami szerepel a regisztrált felhasználók adatbázisában. A megfigyelő funkció elindítása után az applikáció háttér szolgáltatásba kerül, ezzel egyidejűleg megjelenik az ikonja a telefon értesítésre szolgáló sávjában, ezzel jelezve, hogy működésbe lépett. Folyamatosan követhető, mikor történt meg az utolsó háttér szolgáltatás általi frissítés. A teljes körű adatlap meghatározás érdekében szükséges a regisztrációnál nem kért adatok megadása. Ezek alapértelmezett módon jönnek létre regisztráció során, csupán ezek módosítása szükséges. Ilyen adat a kontakt személy adatainak a megadása. Ez az adatpár a név és az elérésre szolgáló telefonszám. Továbbá a felhasználónak célszerű kitöltenie a stroke rizikó értékének a meghatározására létrehozott kérdőívet, amely a 3.1.

fejezetben tárgyalt kérdőív alapján készült. A felhasználó grafikonok formájában megtekintheti különböző leosztásban milyen pulzus értékkel rendelkezett az elmúlt napokban, hónapokban a megfelelő engedélyek megadása után. Ezeket az engedélyeket a Fitbit fiókjába való belépés után teheti meg. Ennek részletezése az 5.6. fejezetben történik meg, illetve szemléltetve van a 12. ábrán. A felhasználó az okosóra applikációt le kell töltsse készülékére, és a megfelelő módon üzembe kell helyezze a 6. fejezetben megadott útmutató alapján.

4.2.1. Termékkel kapcsolatos nem funkcionális követelmények

- Minimum Android 6.0 Marshmallow
- RAM 1 GB
- Internet kapcsolat
- Szabad tárhely ~120 MB
- Bluetooth kapcsolat
- Fitbit 4.2 OS

Az applikációt futtató készülék minimum Android 6.0 operációs rendszerrel kell rendelkezzen, hogy az összes funkciót használni tudja, mivel egyes szolgáltatások ez alatt nem érhetőek el. Minimum 1 GB szabad RAM memória szükséges annak érdekében, hogy az applikáció a többi éppen futó applikációval egyidejűleg, zökkenőmentesen tudjon működni és ne lassítsa le a készüléket, így az eddig nyújtott felhasználói élmény megmarad. Az applikáció nagyjából 120 MB tárhelyet foglal. Folyamatos internet kapcsolat biztosítva kell legyen, mert a megfigyelő rendszer ennek a segítségével képes elérni a szenzor által mért adatokat. Az adatforgalom nem nagy, de folytonos kell legyen, továbbá Bluetooth kapcsolat is biztosítva kell legyen, mert az okosóra és a telefon ezen keresztül kommunikál. Az okosóra alkalmazás 4.2 Fitbit operációs rendszeren alapuló eszközökön érhető el, ami név szerint Fitbit Versa Lite, Versa, Versa 2, Ionic. Más eszközökkel jelenleg nem üzemeltethető az okosóra alkalmazás.

4.2.2. Külső nem funkcionális követelmények

- Jogi háttér
- GDPR
- Fejlesztés
- Tesztelés

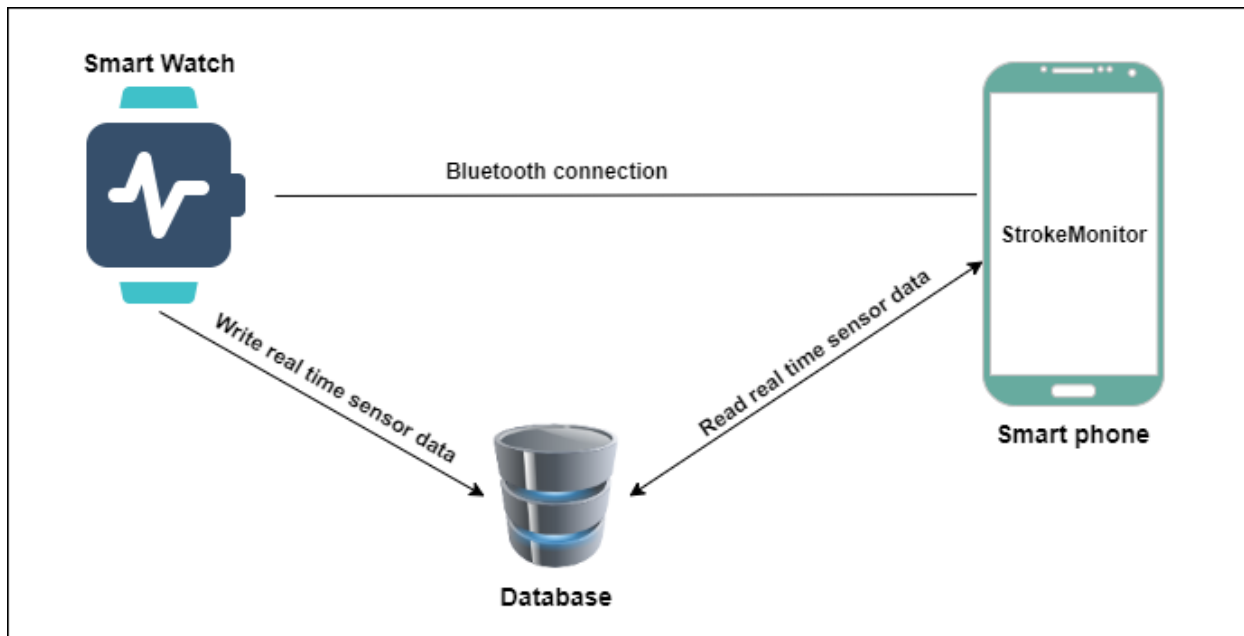
Mivel a rendszer orvosi használatra szánt, ezért be kell tartania az orvoslás egyik alapelvét, azaz az orvosi titoktartást, amely biztosítja, hogy a páciens tudta, s akarata nélkül nem kerülhet

nyilvánosságra egyetlen adat sem. Éppen ezért az adatok tárolása biztonságos módon kell legyen megvalósítva, úgy, hogy a felhasználók csak saját adataikhoz tudjanak hozzáférni. A fejlesztés és tesztelés során az adatok nem kerülhetnek ki a csapat keretein kívülre. Ennek érdekében az adatbázishoz csak a fejlesztő férhet hozzá, mivel az erre a célra létrehozott felhasználói fiók birtokába vannak az adatbázisba elmentett adatok, külső, engedéllyel nem rendelkező személy nem férhet hozzá ezekhez az adatokhoz. Bármely ilyen adatvesztés, szivárogtatás jogi következményeket vonhat maga után. Továbbá fontos a GDPR szerződés betartása mind a szoftvert birtokló fél, mind a felhasználó részéről. Így a felhasználó adatai két ponton is titkosak. Védi őket az orvosi titoktartás és a GDPR, amely az adatokat személyes tulajdonnak tekinti, ezen adatok kiadása csak bizonyos feltételek mellett adhatók ki, akár a felhasználó beleegyezése nélkül [9]. A tesztelés során a tesztben résztvevő alanyok személyazonossága nem nyilvános csupán az általuk használatra bocsájtott adatok érhetőek el. Ilyen adat például a teszt során mért pulzus, életkor, nem, fizikai állapot.

5. A rendszer leírása

A rendszer leírása során részletes bemutatásra kerül a teljes rendszer és alkotórészeinek architektúrája, majd következik a szoftver minden egyes elemének részletes leírása.

5.1. Architektúra

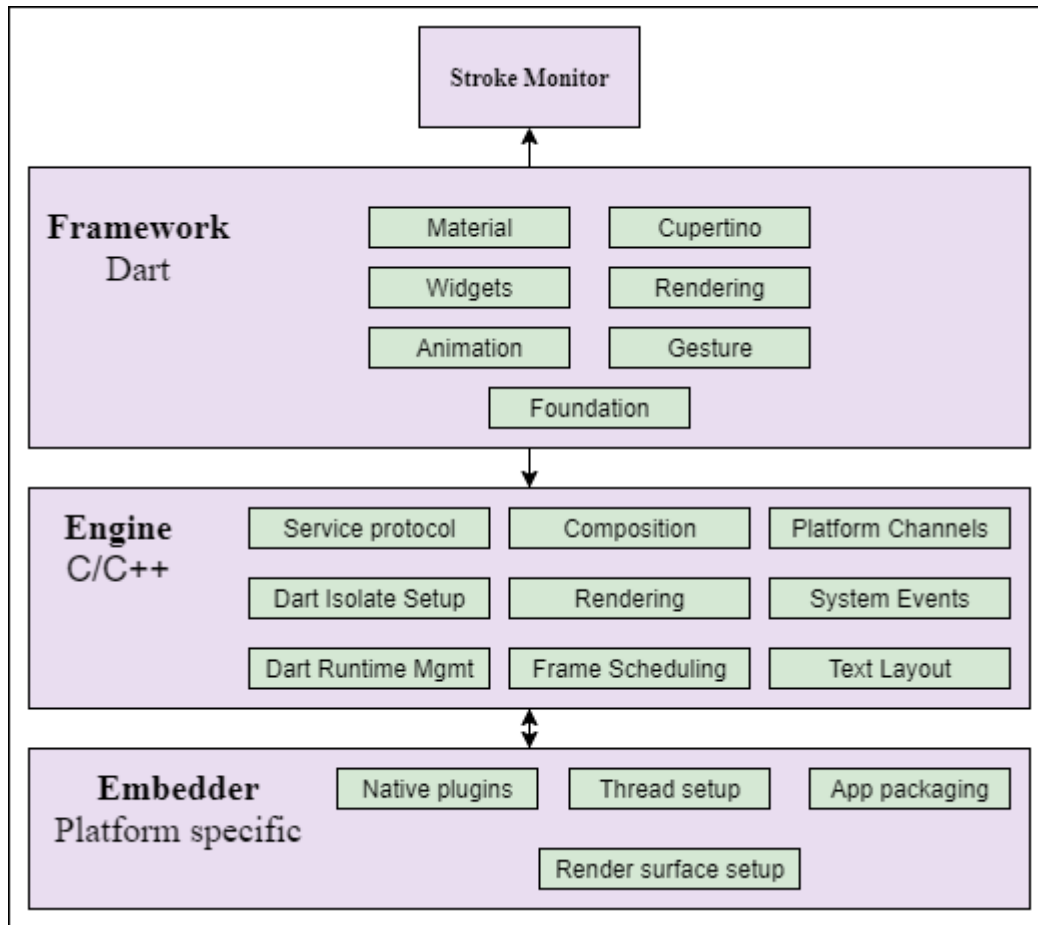


Ábra 4. Rendszer architektúra

A 4. ábrán a rendszer architektúrája látható. Az okos óra szerepe a rendszerben az adatgyűjtés és továbbítás a rendszer többi eleme felé. A mért pulzus értéket az okosóra elküldi az adatbázis felé, amelyből a telefonon futó Stroke Monitor alkalmazás lekéri és feldolgozza. Az adatbázis szerepe a rendszerbe a felhasználói adatok tárolása és mintegy csatornaként szolgál az óra és a telefon közt a valós idejű adat eléréshez, mivel ez más módon nem elérhető a projektben használt eszközök segítségével. A telefon pedig felhasználói felületként szolgál, amely a háttérbe adatfeldolgozás funkciót is betölt.

5.2. Okostelefon applikáció

A 5. ábrán a rendszer mobil alkalmazás architektúrája látható. Az egész architektúra úgy van tervezve, hogy minden egyes szint függ a másiktól, minden egyes elem cserélhető, opcionális, új elemek hozzáadása egyszerű, más komponensektől független [10].



Ábra 5. Mobil alkalmazás architektúra

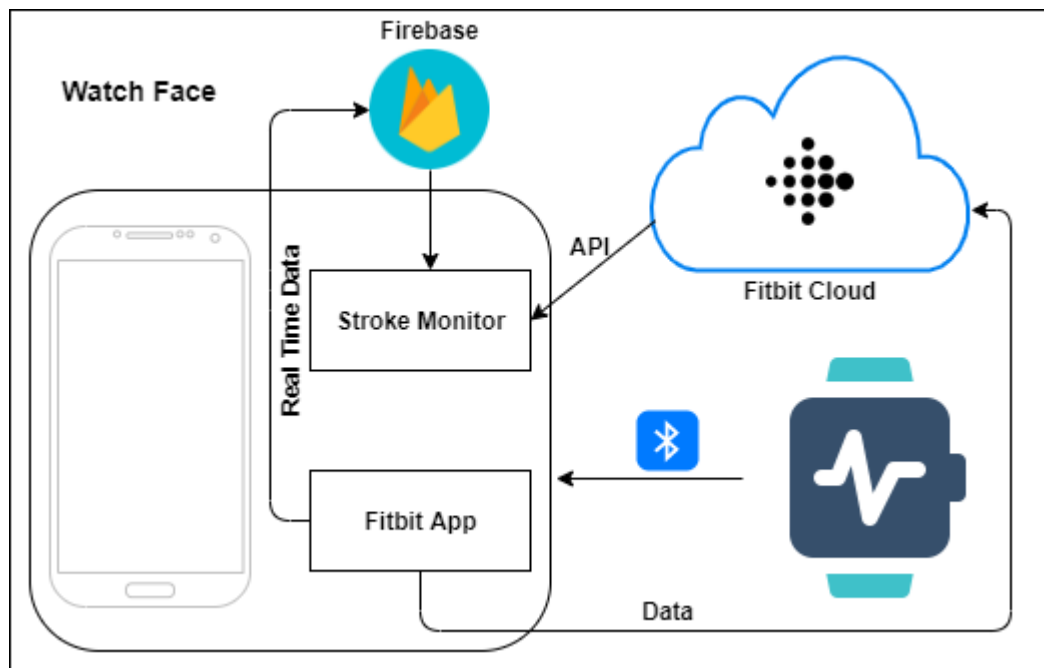
A Dart keretrendszer tartalmazza azokat a komponenseket, amelyek kombinálásával készül, a saját applikáció. Ilyen komponens például a **Material** ami tartalmazza az Androidra jellemző formákat, kinézeteket, míg a **Cupertino** tartalmazza az iOSra jellemzőket. A **widget** komponens tartalmazza a beépített alap widgeteket. Ilyen widget lehet nyomógomb, ikon, csúszka, szöveg megjelenítő, szövegmező stb. Az animáció, gesztus detektáló az ehhez tartozó komponensek elérését teszik lehetővé. A renderelés a tartalom megjelenítésére szolgál a képernyőn, létrehozva a widget fát átalakítva pixelekké.

Az engine tartalmazza azokat a primitíveket, amelyek nélkülözhetetlenek a keretrendszerben található komponensek működéséhez. Biztosítja az alacsony szintű megvalósításokat, mint például I/O művelet, szövegelrendezést, hálózati kommunikációt, grafikai megjelenítést [10]. Tartalmazza az eszköztarát a Dart kód fordítására és futtatására.

A legalsó szinten található a rendszer beágyazás, az itt található komponensek operációs rendszer specifikusak és tartalmazzák a megfelelő platformfüggő megoldásokat. Tartalmaz

Android, iOS, Windows, Linux natív könyvtárakat, ezek mind a platformnak megfelelő nyelven íródtak.

5.3. Okosóra applikáció



Ábra 6. Okosóra applikáció architektúra

Az 6. ábrán az okosóra applikáció architektúrája figyelhető meg. Az órán található óra számlap begyűjti a szenzorok által mért adatokat és Bluetooth kapcsolat segítségével továbbítja a telefonon lévő Fitbit Applikáció felé. Ez a művelet után, a mért pulzus szám valós időben az adatbázisba kerül, míg a Fitbit által strukturált összes mért érték elküldésre kerül a Fitbit felhő alapú szolgáltatásába, ahonnan a Stroke Monitor a Fitbit API²n keresztül hozzáfér. A strukturált adatban napi szinten elmentésre kerül az átlagosan égetett kalória, a minimum, maximum pulzus szám, továbbá a pulzus értéke percenként a nap 24 órájában, de természetesen ezek az értékek állíthatóak.

5.4. Adatbázis

A rendszer által használt adatbázis a Firebase³, ami egy NoSQL típusú adatbázis. Két típusa is jelen van a rendszerben, a valós idejű adatbázis, amelybe az okosóra menti percenként a felhasználó pulzus értékét. Onnan kérdezi le a telefonos alkalmazás az aktuális értéket,

² <https://dev.fitbit.com/build/reference/web-api/>

³ <https://firebase.google.com/docs/database>

tulajdonképpen egy hídként szolgál az okosóra és a telefon közt, a valós idejű adatcserét lehetővé téve.

A regisztráláshoz és belépéshez szükséges metódusokat a Firebase Authentication⁴ biztosítja, a projekt esetében a jelszó email cím páros van beállítva. Adatbiztonság szempontjából a jelszók az adminisztrátor számára se nyilvánosak, csupán az email címek és a hozzá fűzhető műveletek publikusak, ahogy a 7. ábrán is láthatóak.

proba@gmail.com	✉	Feb 18, 2021	Feb 18, 2021	Ri42254NMCORjmKhTUwp	Reset password
admin@gmail.com	✉	Apr 14, 2021	Apr 14, 2021	S4QrmKFICXUYW9luBot2m	Disable account Delete account

Ábra 7. Firebase Authentication

Ezen felül használatban van a Cloud Firestore⁵, azaz felhő alapú adatbázis, amely a nem valós időben változó adatok tárolását teszi lehetővé. Ilyen adat például a nem, életkor, magasság, súly, rizikó szám, illetve a felhasználó kontakt személyének az adatai.

+ Start collection	+ Add document	+ Start collection
risk >	admin >	+ Add field
userImg	arnoldszasz06	percentage: "12.5% per year."
users	testuser	score: 5
usersContact		

Ábra 8. Cloud Firestore szerkezet

A 8. ábrán látható milyen struktúrába szerepelnek az adatok az adatbázisba. Különböző kollekciók vannak létrehozva, amelyben egy típusú adat van tárolva. Ilyen kollekció a risk, users, usersContact, userImg. Minden egyes kollekcióba egyedi névvel ellátott documentumok szerepelnek. Úgy jönnek létre ezek a dokumentumok, hogy a felhasználó által megadott email címből a @ előtti részt tekintí egyedi azonosítónak. Mivel olyan email címmel nem lehet regisztrálni ami már használatban van, így biztosítva van az, hogy a felhasználók adatai nem keverednek egymással, és egy felhasználó csak a saját adataihoz férhet hozzá. A dokumentumokban különböző mezők szerepelnek, attól függően, hogy éppen melyik kollekcióba tartozik.

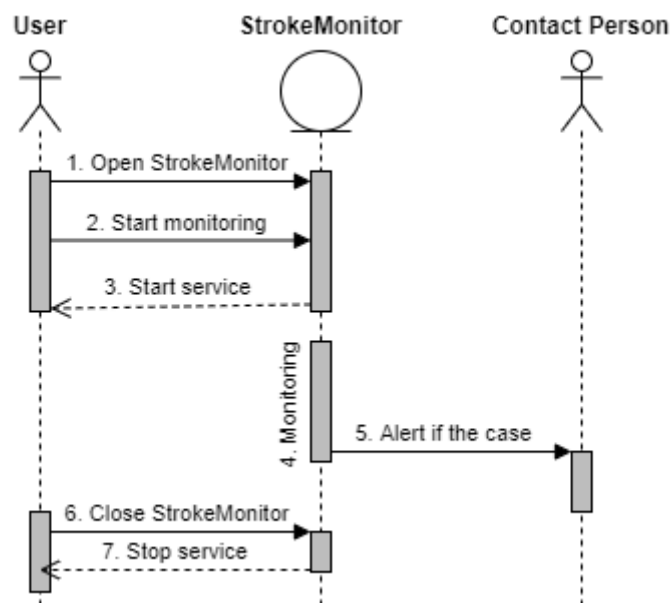
⁴ <https://firebase.google.com/docs/auth>

⁵ <https://firebase.google.com/docs/firestore>

Az adatbázisnak olyan védelmi elvek vannak beállítva, hogy csak egy bejelentkezett felhasználó képes írni és olvasni az adatokat, így a felhasználók adatait teljes mértékben biztonságba vannak, így az esetleges adatlopás csak a felhasználói fiókon keresztül lehetséges.

5.5. Elvi diagram

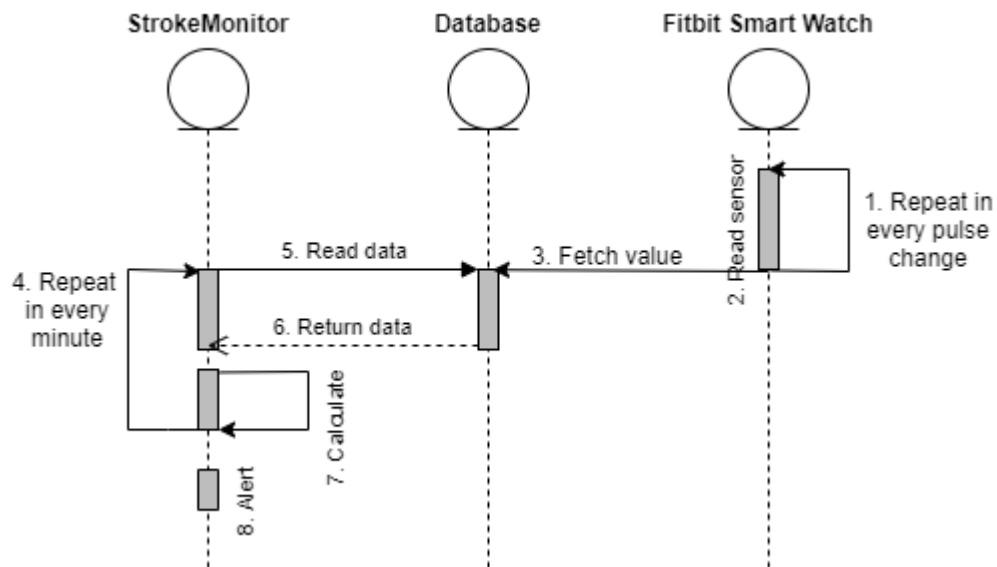
A 9. ábrán a rendszer elvi diagramja található, kiragadva a legfontosabb metódust, amit az alkalmazás végrehajt a felhasználó közreműködésével. A felhasználó elindítja az applikációt, ezzel megkezdődik a megfigyelés az alkalmazás részéről. Ha sikeres a művelet a felhasználó visszajelzést kap. A monitorizálás szekvencia diagramja a 10. ábrán látható. Ha a monitorizálás eredménye azt téríti vissza, hogy a felhasználó veszélybe lehet, értesíti automatikusan az előzetesen megadott kontakt személyt üzenet, illetve tárcsázás formájában. Végül, ha a felhasználó úgy dönt, hogy a megfigyelés már nem releváns, akkor leállíthatja az applikációt, ilyenkor is visszajelzést kap a művelet sikerességéről.



Ábra 9. Elvi diagram

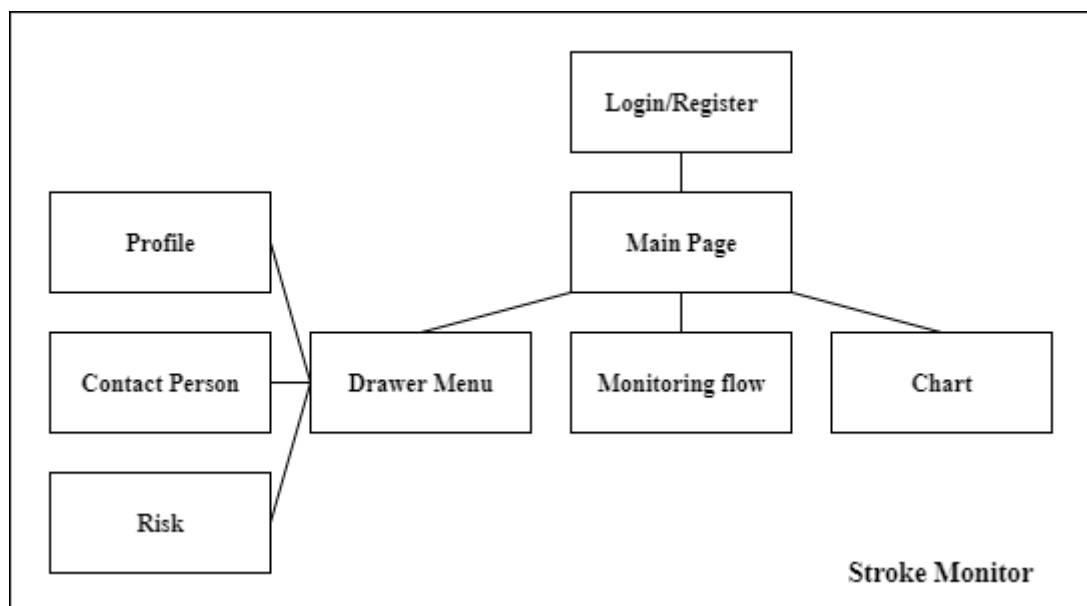
A 10. ábrán a monitorizálás elvi diagramja látható. Első lépésben az okosórán futó óra számlap leolvassa a szenzor értékét, amely felelős a pulzus méréséért, ezt minden egyes változásnál megteszi. A kapott értéket továbbítja az adatbázis felé. Az adatbázisban a frissen kapott érték felülírja a már meglévő értéket, ezzel biztosítva, hogy a telefonos alkalmazás mindig a legfrissebb értéket kapja. Ezzel párhuzamosan a Stroke Monitor telefonos alkalmazás adatot olvas az adatbázisból. Ezt a műveletet minden percben megismétli, mivel a pulzus értékét is egy perc alatt számolt szívverés ismétlődés határozza meg. Ha megvan az adat összehasonlítja, egy

küszöbértékkel, ami tudományos alapokon fekvő érték. Amint meghaladja a szenzor által mért pulzus a küszöbértéket, elindul a kontakt személyt riasztó folyamat.



Ábra 10. Monitorizálás elvi diagram

5.6. UI diagram



Ábra 11. Telefonos alkalmazás UI terv

A 11. ábrán a telefonos applikáció felhasználói interfész terve látható feltüntetve a fő alkotó komponenseket. A kezdő komponens a belépésre vagy regisztrációra szolgáló komponens, ezt követi a főoldal, amely elágazik több irányba így több komponens köthető hozzá. Ilyen a grafikonok megjelenítésére szolgáló komponens, a monitorizálásért felelős komponens, illetve a

szendvics menü, amelyhez több komponens csatlakozik. A profil adatok megjelenítésére szolgáló komponens, a kontakt személy menedzselésére szolgáló komponens és a rizikó faktor megadására szolgáló komponens. Az applikáció architektúráját figyelembe véve, minden komponens külön widgetként szolgál, minden widget kisebb, alkomponensekből épül fel.

A navigáció logikája a main osztályban van megírva, különböző útvonalak vannak definiálva, minden egyes útvonal egy adott komponensre mutat, egyedi azonosítóval ellátva.

```
1  initialRoute: '/',
2  routes: {
3    '/': (context) => StreamBuilder(
4      stream: FirebaseAuth.instance.authStateChanges(),
5      builder: (ctx, userSnapshot) {
6        if (userSnapshot.hasData) {
7          return MainScreen();
8        } else {
9          return AuthScreen();
10       }
11     },
12   ),
13   StrokeRiskScreen.routeName: (context) => StrokeRiskScreen(),
14   ProfileDataScreen.routeName: (context) => ProfileDataScreen(),
15   ContactPersonScreen.routeName: (context) => ContactPersonScreen(),
16 },
```

Kódrészlet 1. Navigáció

Az 1. kódrészletből kiderül, hogy az alapértelmezett útvonal (angolul route⁶) a „/” jelölt, ami két irányba vezethet, egyik a bejelentkezés/regisztrációs felület, másik a főképernyő, ami csak akkor látható, ha a felhasználó előtte már bejelentkezett. Az 1. kódrészlet 13, 14, 15 sorába útvonalak vannak, más képernyők irányába, például a rizikó képernyő, profil, illetve kontakt személy képernyő. Ezen útvonalak hívásánál definiálva van, melyik widget osztály fog megjeleníteni.

Kezdetben két írható mező jelenik meg, egyik az email cím másik a jelszó beírásának a helye, ennek a widgetnek a struktúrája a 2. kódrészletben látható.

```
1  TextFormField(
2    key: ValueKey('password'),
3    validator: (value) {
4      if (value.isEmpty || value.length < 8) {
5        return 'Password must be at least 8 characters.';
6      }
7      return null;
8    },
9    decoration: InputDecoration(
10     labelText: 'Password',
11   ),
```

⁶ <https://flutter.dev/docs/cookbook/navigation/named-routes>


```

12     obscureText: true,
13     onSave: (value) {
14         _userPassword = value;
15     },
16 },

```

Kódrészlet 2. Bemeneti mező

Minden egyes komponensnek van egy egyedi azonosítója, egy validálási része, egy dekoráció, amely meghatározza, milyen lesz a billentyűzet típusa, azaz milyen szöveget vár el a mező, megadható, hogy csillagozva jelenjenek meg a beírt karakterek (jelszónál van csak ilyen), illetve a metódus, amely mentés lenyomása után elmenti az adott értéket egy ennek megfelelő változóba.

```

1  FlatButton(
2      textColor: Theme.of(context).primaryColor,
3      child: Text(_isLogin
4          ? 'Create new account'
5          : 'I already have an account'),
6      onPressed: () {
7          setState(() {
8              _isLogin = !_isLogin;
9          });
10     },
11 ),

```

Kódrészlet 3. Bejelentkezés gomb

A 3. kódrészlet a widget állapotának a változását mutatja be, figyelve, hogy az `_isLogin` (az alul vonás a változó előtt annyit jelent, hogy privát változó) változó éppen milyen értékkel rendelkezik. Ha a felhasználó megérinti a gombot, a widget állapota megváltozik és a változó szerint mutatja vagy rejtje el az adott elemeket.

```

1  void _trySubmit() {
2      final isValid = _formKey.currentState.validate();
3      FocusScope.of(context).unfocus();
4      if (isValid) {
5          _formKey.currentState.save();
6          widget.submitFunction(
7              _userEmail,
8              _userPassword,
9              _userBirthDay,
10             _userGender,
11             _userHeight,
12             _userWeight,
13             _isLogin,
14             _terms,
15         );
16     }
17 }

```

Kódrészlet 4. Bemeneti adatok validálása

Ha a felhasználó megnyomja a bejelentkezés vagy regisztráció gombot, a kérdőív jelenlegi állapota mentésre kerül a 4. kódrészletben definiált módon. Ha minden egyes elem valós adatokkal rendelkezik, akkor a függvényhívás eredményeként a változóba mentett adatok egy másik osztályhoz kerülnek, amely elvégzi a két lehetséges művelet egyikét, létrehoz vagy belépteti a felhasználót az 5. kódrészlet szerint. A regisztráció a Firebase Authentication⁷ email jelszó páros megoldását használja.

```
1  if (isLogin) {
2    authResult = await _auth.signInWithEmailAndPassword(
3      email: email,
4      password: password,
5    );
6  } else {
7    authResult = await _auth.createUserWithEmailAndPassword(
8      email: email,
9      password: password,
10   );
11  }
```

Kódrészlet 5. Regisztráció vagy beléptetés

Továbbá létrehozásra kerülnek a 8. ábrán szemléltetett kollekciók. Kezdetben a felhasználó személyes adatai kitöltésre kerülnek a megadott adatokkal, ahogy a 6. kódrészletben látható. A rizikó szám és a kontakt személy kollekcióba a mezők alapértelmezett értékkel jönnek létre. Ezt a felhasználó bármikor felülírhatja.

```
1  FirebaseFirestore.instance
2    .collection('users')
3    .doc(_auth.currentUser.email
4      .substring(0, _auth.currentUser.email.lastIndexOf('@')))
5    .set({
6      'birthday': DateFormat.yMd().format(birthday),
7      'gender': gender,
8      'height': height,
9      'weight': weight,
10   });
```

Kódrészlet 6. Regisztrációs adatok beszúrása

Először egy példányt kell létrehozni a Firebaseről⁸, majd az építő programtervezési minta segítségével meg kell adni különböző adatokat. Ilyen adat a kollekció, a dokumentum neve és konkrétan az adat, amely map formába van létrehozva, az adatbázisba való feltöltés után JSON formátummá konvertálódik.

A sikeres bejelentkezést/regisztrációt követően az alkalmazás a főképernyőre navigál, ahol több lehetőség közül választhat a felhasználó. Két típusú menü lehetséges fel. Egyik az alsó

⁷ <https://firebase.flutter.dev/docs/auth/usage>

⁸ <https://firebase.flutter.dev/docs/storage/usage#uploading-files>

navigációs sáv, amely alapértelmezett képernyője a Home, másik lehetőség a Charts, amely a grafikonok megjelenítésére szolgál. Ez egy `_pages` változóba van eltárolva, a 7. kódrészletben.

```
1 void initState() {
2   _pages = [
3     {
4       'page': HomeScreen(),
5       'title': 'Home',
6     },
7     {
8       'page': ChartsScreen(),
9       'title': 'Charts',
10    },
11  ];
12  super.initState();
13 }
```

Kódrészlet 7. Menüsáv oldalak

A widget állapot inicializálásánál meg kell adni, milyen oldalak fognak szerepelni a navigációs sávba, mi a megnevezésük és az adott megnevezés milyen widgetre mutat.

```
1 appBar: AppBar(
2   title: Text(_pages[_selectedPageIndex]['title']),
3 ),
4 drawer: MainDrawer(),
5 body: _pages[_selectedPageIndex]['page'],
6 bottomNavigationBar: BottomNavigationBar(
7   onTap: _selectPage,
8   backgroundColor: Theme.of(context).primaryColor,
9   unselectedItemColor: Colors.white,
10  selectedItemColor: Colors.black,
11  currentIndex: _selectedPageIndex,
12  items: [
13    BottomNavigationBarItem(
14      backgroundColor: Theme.of(context).primaryColor,
15      icon: Icon(Icons.home),
16      title: Text('Home'),
17    ),
18    BottomNavigationBarItem(
19      backgroundColor: Theme.of(context).primaryColor,
20      icon: Icon(Icons.bar_chart),
21      title: Text('Charts'),
22    ),
23  ],
24 ),
```

Kódrészlet 8. Menüsáv inicializáció

A 8. kódrészletben a menüsáv inicializáció⁹ látható. Az oldal címe kiválasztásra kerül az 7. kódrészletben látható listából, majd a body részbe megjelenítésre kerül a kiválasztott widget. A 6. és 23. sor között a menü stílusa van meghatározva. Milyen színt használjon, mi legyen a

⁹ <https://flutter.dev/docs/cookbook/design/drawer>

különböző oldalak ikonja s neve, hogyan tűnjön ki az előtérbe lévő elem. A 4. sorba kerül meghívásra a második menü típus, a szendvics menü. A menü lenyitáskor megjelenik egy üdvözlő szöveg a bejelentkezett felhasználó részére, illetve négy menüpont. Profil adatok megtekintése, rizikó faktor kérdőív kitöltése, kontakt személy menedzselése, végül a kijelentkezés a fiókból.

Ez a menü típus úgynevezett listTitle alap widgetet használ. A kód átláthatósága érdekében egy általános, többször felhasználható függvény van létrehozva a projektbe, ami megfelel a 9. kódrészletnek, így csak különböző paramétereket kell átadni.

```
1  Widget buildListTile(String title, IconData icon, Function tapHandler) {
2      return ListTile(
3          leading: Icon(
4              icon,
5              size: 26,
6              color: Color.fromRGBO(153, 42, 35, 1.0),
7          ),
8          title: Container(
9              decoration: const BoxDecoration(
10                 color: Color.fromRGBO(250, 232, 230, 1.0),
11             ),
12             child: Text(
13                 title,
14                 key: ValueKey('nav$title'),
15                 style: TextStyle(
16                     fontFamily: 'RobotoCondensed',
17                     fontSize: 24,
18                     fontWeight: FontWeight.bold,
19                 ),
20             ),
21         ),
22         onTap: tapHandler,
23     );
24 }
25 ...
26 buildListTile('Profile data', Icons.person, () {
27     Navigator.of(context).pushNamed('/profile-data');
28 },
```

Kódrészlet 9. Szendvics menü inicializáció

A buildListtile() visszatérítési értéke egy widget, paraméterül egy címet, egy ikont és egy handler függvényt kér. A widget három fő részből tevődik össze. Kettő dizájn szerepet tölt be, amely meghatározza a cím és az ikon kinézetét, illetve a 22. sorba található onTap metódus, amely a fentebb említett útvonalak egyikére való navigálást teszi lehetővé. A 26.-28. sorba a függvény meghívása látható. Érdekesebb elem a handler függvény. Navigátor függvény megkeresi a main-be megadott útvonalak közül a megfelelőt és megnyitja a találat függvényébe az adott képernyőt. A szendvics menübe található elemek két osztályból állnak, egy képernyőből

és egy widget fa osztályból. A képernyő osztályba van megadva statikusan az az adattag a 10. kódrészletben látott módon, amely alapján az útvonal választó navigátor hitelesen beazonosítja az adott képernyőt.

```
1 static const routeName = '/profile-data';
```

Kódrészlet 10. Útvonal

A profil adatok menedzselésére szolgáló widgetbe betöltésre kerül a Firebase Firestoreba tárolt adatok összesége, amely az adott felhasználóra vonatkozik.

```
1 ...
2 child: StreamBuilder(
3   stream: FirebaseFirestore.instance
4     .collection('risk')
5     .doc(_auth.currentUser.email.substring(
6       0, _auth.currentUser.email.lastIndexOf('@'))
7     .snapshots(),
8   builder: (context, snapshot) {
9     if (snapshot.connectionState == ConnectionState.waiting) {
10      return Center(
11        child: CircularProgressIndicator(),
12      );
13    }
14    final documents = snapshot.data;
15    return Column(
16      mainAxisAlignment: MainAxisAlignment.min,
17      children: <Widget>[
18        Text('Risk: ${documents['percentage']}',
19          style: TextStyle(
20            fontSize: 28,
21            color: Theme.of(context).primaryColor,
22            fontWeight: FontWeight.bold,
23          ),
24      ),
25    ...
```

Kódrészlet 11. Adatok letöltése és megjelenítése

A 11. kódrészletben egy StreamBuilder widget kerül meghívásra. A stream paraméterénél egy konkrét Firebase snapshot¹⁰ letöltése történik. Hasonlóképpen működik, mint a regisztrációs rész. Megkell adni a kollekciót, a dokumentumot és hogy egy snapshotról van szó. A builder résznél amíg a kapcsolat státusza várakozó egy körkörösén forgó nyíl jelenik meg, hogy a felhasználó érzékelhesse, hogy várni kell az adatok megjelenítésére. A kódrészlet 14. sorába snapshotnak az adat része van változóba kimentve, annak érdekében, hogy könnyen lehessen dolgozni a meglévő adatokkal. Mivel a Firebase JSON formátumba menti el az adatok, a kliens feladata az adatok beazonosítása és típusának megjelölése. A 15. sortól kezdődően az adat

¹⁰ <https://firebase.flutter.dev/docs/firestore/usage>

felhasználói felületre való megjelenítési módjának a definiálása történik, mivel a StreamBuilder egy widgetet térít vissza. A kimentett adatba könnyedén lehet kulcsszó alapján keresni, ahogy a 18. sorba is látható.

A kontakt személy menedzselésére szolgáló osztályok is hasonló módon működnek, mint a profil adatok menedzselésére használatos. Kezdetben betöltődik egy erre a célra létrehozott mezőbe az adatbázisba tárolt személy neve s telefonszáma. Lehetőség van változtatni is, ekkor az adatbázisba lévő mező felül lesz írva. A felhasználó tesztelésként gombnyomásra kipróbálhatja, milyen módon lesz értesítve a kontakt személy vészhelyzet esetén. A kontakt személy egy smst kap egy előre meghatározott szöveggel, illetve tárcsázva is lesz az adott telefonszám, mindez automatikusan, felhasználói közbeavatkozás nélkül történik.

```
1 void _sendSMS(String address) {
2   SmsSender sender = SmsSender();
3   SmsMessage message = SmsMessage(address, 'Hello flutter!');
4   message.onStateChanged.listen((state) {
5     if (state == SmsMessageState.Sent) {
6       print("SMS is sent!");
7     } else if (state == SmsMessageState.Delivered) {
8       print("SMS is delivered!");
9     }
10  });
11  sender.sendSms(message);
12 }
13 ...
14
15 _sendSMS(currentPhone);
16 FlutterPhoneDirectCaller.callNumber('$currentPhone');
```

Kódrészlet 12. Üzenet küldés, telefon hívás

Mindkét értesítési mechanizmus egy plugin segítségével történik ahogy a 12. kódrészletben látható. A 16. sorba a telefonhíváshoz¹¹ szükséges függvényhívás látható, a currentPhone változóba az adatbázisból lekért telefonszám található. A 15. sorba a fentebb definiált függvény hívása történik. Bizonyos adatokat meg kell adni az SmsSender¹² függvénynek és a háttérbe elküldi a címre az üzenetet.

A felhasználó ki kell töltsön egy kérdőívet, amely alapja a 3.1 fejezetben lévő kérdőív, annak érdekébe, hogy meghatározza mekkora rizikóval éli mindennapjait. Ez a felület kijelölő négyzetekből áll. A felhasználó kiválasztja a rá leginkább jellemző kérdéseket, kijelentéseket. Ha befejezte a kérdőív kitöltését az adatbázisba való írás előtt az alkalmazás kiszámítja mennyi pontot érnek a válaszai és az ennek megfelelő értékek kerülnek feltöltésre a megfelelő kollekció és dokumentum alá, ahogy az a 12. - 16. sorban látható a 13. kódrészletben.

¹¹ https://pub.dev/packages/flutter_phone_direct_caller

¹² <https://pub.dev/packages/sms>

```

1 void _trySubmit() async {
2   var totalScore = 0;
3   totalScore = ageScore + genderScore + riskScore;
4   var str = "0% per year.";
5   if (totalScore == 0) str = "1.9% per year.";
6   if (totalScore == 1) str = "2.8% per year.";
7   if (totalScore == 2) str = "4% per year.";
8   if (totalScore == 3) str = "5.9% per year.";
9   if (totalScore == 4) str = "8.5% per year.";
10  if (totalScore == 5) str = "12.5% per year.";
11  if (totalScore == 6) str = "18.2% per year.";
12  FirebaseFirestore.instance
13    .collection('risk')
14    .doc(_auth.currentUser.email
15      .substring(0, _auth.currentUser.email.lastIndexOf('@')))
16    .set({'score': totalScore, 'percentage': str});

```

Kódrészlet 13. Rizikó szám frissítés

A negyedik menüpont a szendvics menübe az egyszerű kijelentkezési metódus meghívása, ami a 14. kódrészlet, hatására az alkalmazás vissza navigál a bejelentkezési oldalra.

```

1 onTap: () {
2   FirebaseAuth.instance.signOut();
3 },

```

Kódrészlet 14. Kijelentkezés

```

1 Future authenticate() async {
2   await _loadToken();
3   if (_token == '') {
4     final url =
5       'https://www.fitbit.com/oauth2/authorize;
6     final callbackUrlScheme = 'token';
7
8     try {
9       final result = await FlutterWebAuth.authenticate(
10         url: url, callbackUrlScheme: callbackUrlScheme);
11       SharedPreferences prefs = await SharedPreferences.getInstance();
12       prefs.setString('token',
13         "Bearer ${result.substring(result.indexOf('=')
14           + 1, result.indexOf('&'))}");
15       setState(() {
16         _status =
17           result.substring(result.indexOf('=')
18             + 1, result.indexOf('&'));
19       });
20     } on PlatformException catch (e) {
21       setState(() {
22         _status = 'Got error: $e';
23       });
24     }
25   }
26 }

```

Kódrészlet 15. Fitbit autentikáció

A Fitbit felhőszolgáltatásban tárolt adatok elérése érdekében a felhasználó be kell jelentkezzen a fitbit fiókjába, és különböző engedélyeket kell megadjon, adatainak elérése érdekében. Amint a felhasználó a grafikon képernyőre navigál, automatikusan lefut az authenticate függvény a 15. kódrészletből. Ez egy aszinkron függvény, mivel egy Chrome ablakot nyit meg. Kezdetben megvárja amíg betöltődik a token, amely a shared preferencesbe¹³ van tárolva. Ha a felhasználó először lépett erre a képernyőre, természetesen nincs token elmentve, ezért lefut a kód számottevő része. Egy URL megnyitásra kerül sor egy új ablakban. A web API használata szigorú feltételekhez kötött, előzetes regisztráció szükséges, ennek hatására különböző egyedi azonosítóval lehet ellátni az alkalmazást. Az 5. sorban látható az URL alap része, ez az alkalmazás forráskódjában jóval hosszabb, mivel tartalmazza a titkos azonosítókat. Az automatikusan megnyíló Chrome ablakban kezdetben egy bejelentkezési felület látható, ahol a felhasználó meg kell adja a Fitbit felhasználóját és a jelszót, majd egy lista tárul elé, amelyben meg kell adja milyen adatokra ad engedélyt az alkalmazás számára, ahogy a 8. ábrán is látható. A Chrome ablak megnyitásához szükséges Android részen konfigurálni a Manifest.xml fájlt, ahogy a 16. kódrészletben látható. A 6. sorban meg kell adni, milyen séma szerint történik a visszacsatolás, ilyen módon kell küldje a weboldal az adatot a telefonos alkalmazás felé. A Flutter azonosítás a flutter_web_auth¹⁴ nevezetű csomag segítségével történt. A navigáció a Chrome ablak és a telefonos alkalmazás közt egy átirányító weboldal segítségével történik, aminek a forráskód részlete a 17. kódrészletben látható. Ez a weboldal átküldi a 16. kódrészletben látott sémán keresztül a Chrome ablakban található URL címet a telefonos alkalmazás részére, ami tartalmazza a felhasználó tokenjét. Ennek a megvalósítása a 17. kódrészlet 2. – 11. sorai közt látható. A visszacsatolást végző weboldal hostja a Firebase Hosting¹⁵ biztosítja. Az alkalmazás megpróbálja feldolgozni a kapott URL címet, ahogy a 15. kódrészlet, 9. – 19. sorok közt látható. Siker esetén a shared preferencesbe mentésre kerül a felhasználó egyedi tokenje. Hiba esetén a felhasználó az ennek megfelelő üzenettel találkozhat.

```
1 <activity android:name="com.linusu.flutter_web_auth.CallbackActivity">
2   <intent-filter android:label="flutter_web_auth">
3     <action android:name="android.intent.action.VIEW" />
4     <category android:name="android.intent.category.DEFAULT" />
5     <category android:name="android.intent.category.BROWSABLE" />
6     <data android:scheme="token" />
7   </intent-filter>
8 </activity>
```

Kódrészlet 16. Android Manifest

¹³ https://pub.dev/packages/shared_preferences

¹⁴ https://pub.dev/packages/flutter_web_auth

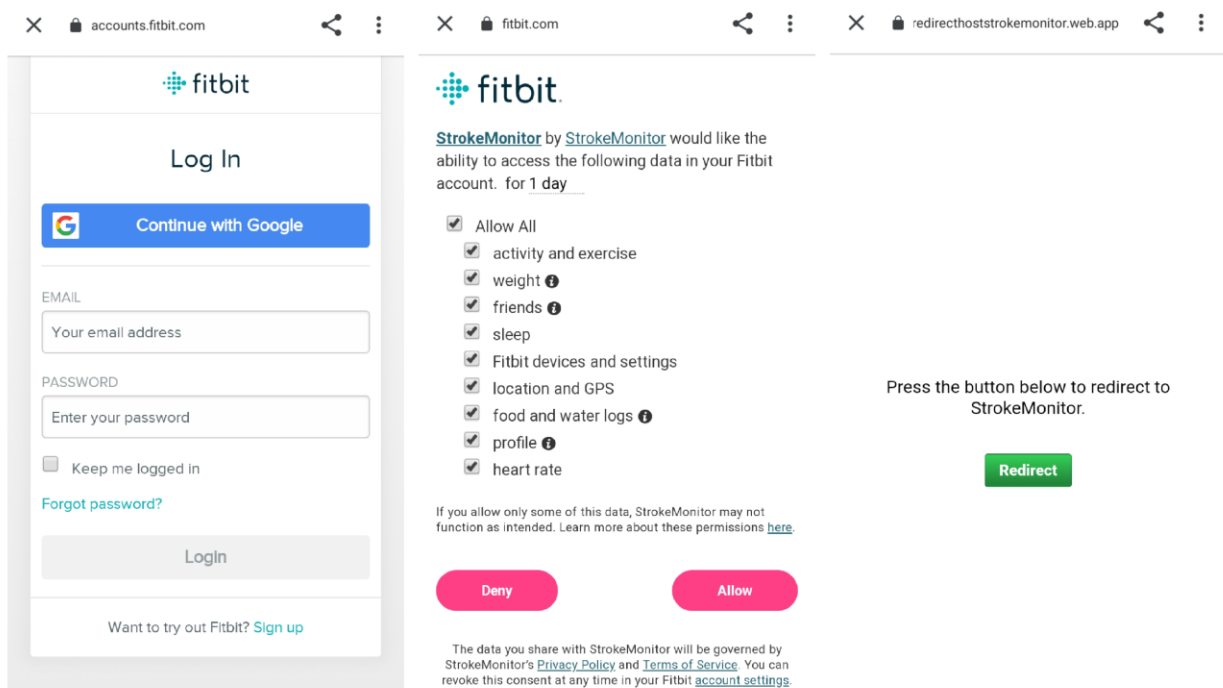
¹⁵ <https://firebase.google.com/docs/hosting>


```

1  <script>
2      function getHref() {
3          document.getElementById("token").href =
4              "token://" + window.location.href;
5      }
6  </script>
7  <main>
8      <div id="text">
9          Press the button below to redirect to StrokeMonitor.</div>
10     <div id="button" onclick="getHref()">
11         <a id="token" href="">Redirect</a></div>
12 </main>

```

Kódrészlet 17. Visszacsolást végző weboldal



Ábra 12. Fitbit engedély kérelem

Miután végbement a 12. ábrán látható művelet, a felhasználónak lehetősége van saját adatainak a megtekintésére. Választhat az utolsó nap választott óra, utolsó nap összes óra, utolsó hét nap lehetőségek közül. Az összes kép kollázs a Benfunky¹⁶ online kép szerkesztő segítségével készült.

```

1  Future _getFitbitToday() async {
2      type = 'day';
3      final response = await http.get(
4          'https://api.fitbit.com/1/user/-/$endpoint',
5          headers: {HttpHeaders.authorizationHeader: _token},
6      );
7      if (response.statusCode == 200) {

```

¹⁶ <https://www.befunky.com/create/collage/>

```

8      var jsonResponse = jsonDecode(response.body)
9      as Map<String, dynamic>;
10     fitbitData = [];
11     for (var i in jsonResponse['activities-heart-intraday']['dataset']) {
12         fitbitData.add(FitbitData(
13             DateTime.parse('1998-01-01 ' + "${i['time']}"), i['value']));
14     }
15     ScaffoldMessenger.of(context)
16         .showSnackBar(const SnackBar(content: Text('Succes!')));
17 } else {
18     if (response.statusCode == 401) {
19         ScaffoldMessenger.of(context).showSnackBar(
20             const SnackBar(content: Text('Authorization required!')));
21         _nullToken();
22     }
23     print('Request failed with status: ${response.statusCode}.');
24 }
25 }

```

Kódrészlet 18. Fitbit Web API hívás

A `_getFitbitToday` ami a 18. kódrészletben található egy aszinkron függvény, mivel egy API-val kommunikál. Először szükséges egy HTTP¹⁷ kérést indítani, ami jelen esetben egy GET metódus. A kérésnek szükséges megadni az URL címet, amely tartalmazza az alap URL címet és annak kiegészítéseit, mint például milyen dátumon, milyen pontossággal, milyen típusú adatra van szükség. Ezt követően fel kell tölteni az URL cím fejlécét az autórizációs tokennel, így az API tudni fogja, hogy melyik felhasználónak az adataira lesz éppen szükség. Ezt követően figyelve a válasz státuszát két forgatókönyv jöhet szóba. Siker, azaz 200-as kód esetén a válasz JSON formátumba megérkezve feldolgozásra kerül. A feldolgozás során az éppen fontos elemek hozzáadásra kerülnek egy előre létrehozott listához, ahogy a 12. - 13. sorok közt látható. A lista egy modell osztály példányait tárolja, amelyben egy `DateTime` típusú adattag található, ez a pulzus mérésének időpontját tárolja, és egy `Int` típusú adattag van a pulzus értékének tárolása érdekében. Mivel a JSON csak időt térít vissza azonban a modell osztály teljes dátumot kér, ezért kiegészítve egy évszámmal, hogy teljes legyen. Ez a kiegészítés leginkább azért szükséges, mert a más http kérés eredménye egyes esetben teljes dátumot térít vissza, így nem szükséges két logikájában egyforma modell osztályt létrehozni. Másik esetben a hibakód kiírásra kerül, illetve, ha a kód 401, ami egy specifikus hibakód¹⁸, a token nullázásra kerül. Azért van szükség erre, mert a 401-es hibakód arra utal, hogy a felhasználónak nincs érvényes tokenje, így szükséges megismételni a 12. Ábrán látható lépéseket, mivel ezek a tokenek csak egy bizonyos ideig érvényesek.

¹⁷ <https://pub.dev/packages/http>

¹⁸ <https://dev.fitbit.com/build/reference/web-api/troubleshooting-guide/error-handling/>

Az előző lépésben feldolgozásra került adatot a grafikon¹⁹ widget segítségével lehet megjeleníteni, ami a 19. kódrészlet mutat be. A megjelenítés módja függ a felhasználó választásától, mivel a különböző lehetőségek esetén más módon jelenik meg a grafikon.

```
1  Container(  
2    height: MediaQuery.of(context).size.height / 1.5,  
3    child: SfCartesianChart(  
4      title: ChartTitle(text: 'Resting heart rate'),  
5      enableAxisAnimation: true,  
6      primaryXAxis: CategoryAxis(),  
7      tooltipBehavior: TooltipBehavior(enable: true),  
8      enableSideBySideSeriesPlacement: true,  
9      legend: Legend(  
10       isVisible: true,  
11     ),  
12     series: <LineSeries<FitbitData, String>>[  
13       LineSeries<FitbitData, String>(  
14         enableTooltip: true,  
15         markerSettings: MarkerSettings(  
16           isVisible: true,  
17           height: 5,  
18           width: 5,  
19           shape: DataMarkerType.circle,  
20           borderWidth: 3,  
21           borderColor: Theme.of(context).primaryColor),  
22         dataSource: fitbitData,  
23         color: Theme.of(context).primaryColor,  
24         xAxisName: "Date",  
25         name: 'BPM',  
26         yAxisName: 'BPM',  
27         xValueMapper: (FitbitData data, _) =>  
28           data.date.hour.toString(),  
29         yValueMapper: (FitbitData data, _) => data.value)  
30       ],  
31     ),  
32   )
```

Kódrészlet 19. Grafikon stílus, adatfeltöltés

Első lépésben meghatározásra kerülnek a grafikon tulajdonságai, ahogy a 2. – 12. sorok közt látható. Ilyen tulajdonságok a méret, a cím a különböző tengelyek beállításai. Ezt követi az adat átadás a grafikon számára. Mivel egy vonal grafikont jelenít meg a widget szükség van két értékre, az OX tengelyen a dátum kerül megjelenítésre, míg az OY tengelyen a pulzus értéke. A konkrét érték feldolgozás a tengelyek számára a 27. – 29. sorok közt láthatóak. Az előző kódsorok ezen adatok megjelenítésének a konfigurálására szolgálnak. Ilyen konfiguráció például a szín, a megjelenés, vagyis az értékek ábrázolása és jelölése a grafikonon.

¹⁹ https://pub.dev/packages/syncfusion_flutter_charts

Annak érdekében, hogy az alkalmazás folyamatosan tudjon működni a háttérbe szükséges létrehozni egy háttér szolgáltatást, hogy a monitorizálás funkció ne csak előtérbe helyezett alkalmazás futtatás módon, illetve feloldott képernyővel működjön.

A háttér szolgáltatás megvalósításához egy külön csomag hozzáadása szükséges, segítségével a Flutter keretein belül meglehetősen valósítani a szolgáltatásokat, nem szükséges, külön specifikus Android vagy iOS kódokat implementálni a hasonló eredmény elérése érdekében. A csomag neve `flutter_background_service`²⁰.

```
1  FlutterBackgroundService.initialize(onStart);
2  ...
3  void onStart() {
4    WidgetsFlutterBinding.ensureInitialized();
5    final service = FlutterBackgroundService();
6    service.onDataReceived.listen((event) {
7      if (event["action"] == "setAsForeground") {
8        service.setForegroundMode(true);
9        return;
10     }
11   });
12   service.setForegroundMode(true);
13   Timer.periodic(Duration(seconds: 30), (timer) async {
14     if (!(await service.isServiceRunning()))
15       timer.cancel();
16     service.setNotificationInfo(
17       title: "Monitoring...",
18       content:
19         "Last updated at ${DateTime.now().hour}
20         :${DateTime.now().minute}",
21     );
22     service.sendData(
23       {"data": DateTime.now().toString()},
24     );
25   });
26 }
```

Kódrészlet 20. Háttér szolgáltatás

Ahogy a 20. kódrészletben is látható az alkalmazás fő metódusába inicializálni kell az `onStart` metódust, amely tartalmazza a szolgáltatás működéséhez szükséges kódsorokat. A 6. – 11. sorba látható, hogy a szolgáltatás képes adatok fogadására, így különböző parancsokat lehet közvetíteni a UI és a szervíz közt. A 13. – 25. sorok közt a szervíz előtér módba van állítva és egy értesítésként jelenik meg. Minden 30 másodpercbe frissül az értesítés. Amit a felhasználó lát, a 17. – 20. sor között van implementálva. Megjelenik, hogy mikor volt utoljára frissítve a szolgáltatás, ezzel egyidejűleg adat (lokális idő) továbbítódik a szervíz felé, amelyre azért van

²⁰ https://pub.dev/packages/flutter_background_service

szükség, hogy a UI részen található figyelő elvégezze azokat a kódsorokat, amely a 21. kódrészletben látható.

```
1  StreamBuilder<Map<String, dynamic>>(  
2    stream: FlutterBackgroundService().onDataReceived,  
3    builder: (context, snapshot) {  
4      return FutureBuilder<String>(  
5        future: _future,  
6        builder: (context, snapshot) {  
7          databaseReference.child('$uid').once().then(  
8            (DataSnapshot snapshot) {  
9              String currentValue = snapshot.value['data'];  
10             setState(  
11               () {  
12                 value = '${snapshot.value['data']}';  
13               },  
14             );  
15           });  
16           if (int.parse(value) >= 160) {  
17             HapticFeedback.heavyImpact();  
18             _sendSMS(currentPhone);  
19             FlutterPhoneDirectCaller.callNumber('$currentPhone');  
20           },  
21         );  
22       }),
```

Kódrészlet 21. Monitorizálás

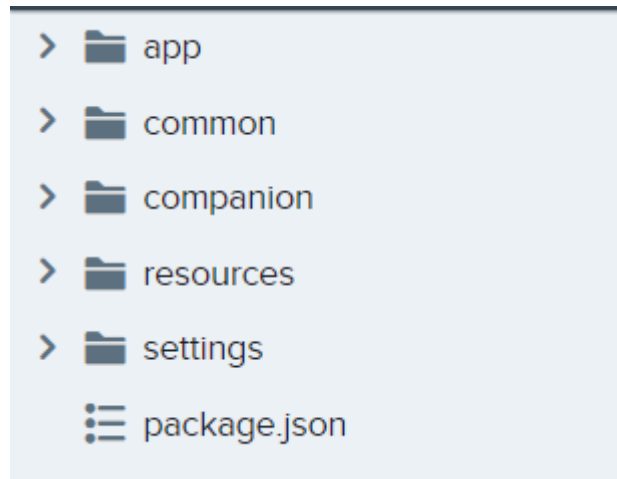
A 21. Kódrészlet a monitorizálást végzi, kezdetben egy StreamBuilder widget figyel, hogy a háttérszolgáltatástól érkezett adat vagy sem, ez az adat a lokális idő, azonban ez nem fontos, hogy milyen adat érkezik, mivel csak folyamat elindító szerepet tölt be. Ha adat érkezett, akkor megtörténik egy érték olvasás a Firebase valós idejű adatbázis megfelelő mezőjéről, ahogy a 7. sorban látható. Úgy történik a megfelelő mező kiválasztása, hogy egy a 18. kódrészlethez hasonló https kérés során meghatározásra kerül a Fitbit felhasználó egyedi azonosítója, ahova az okosóra küldi az adatokat, ez a későbbiekben kifejtésre kerül. Ha sikerült a pulzus értékét az adatbázisból leolvasni, és ha nagyobb ez az érték, mint a küszöbszám, akkor erős rezgést kezdeményez a telefon, illetve üzenet és telefonhívás kezdeményeződik, ugyanolyan módon, ahogy a 12. kódrészletben is látható.

5.7. Fitbit applikáció

A Fitbit applikációk tervezése során²¹ el kell dönteni, hogy milyen típusú alkalmazás lesz, mivel különbséget lehet tenni a facewatch, azaz az óra számlap alkalmazás, illetve egy konkrét

²¹ <https://dev.fitbit.com/getting-started/>

telefon alkalmazáshoz hasonló közt. A projekt architektúra²² szempontjából létezik egy applikáció, amely konkrétan az okosórán található, illetve egy companion-nak nevezett applikáció, amely a felhasználó telefonján fut. Azért van szükség a companionra, mert az órának nincs saját modulja, amely képes lenne kommunikálni interneten keresztül más eszközzel, jelen esetben a Firebase adatbázissal, ezért szükséges, hogy a telefont használja erre a célra. Az adat, amelyet az óra applikáció begyűjt, Bluetooth segítségével továbbítódik a telefonra.



Ábra 13. Fitbit Alkalmazás aleggységei

A 13. ábrán látható egy Fitbit óralap alkalmazás aleggységei. A package.json tartalmazza azokat a részleteket, amelyek meghatározzák, hogy az alkalmazás milyen engedélyekkel kell rendelkezzen, ahhoz, hogy megfelelő módon tudjon üzemelni, továbbá felsorolásra kerül a kompatibilis eszközök névsora is, illetve különböző nyelvi beállításokat is lehet eszközölni. Maga az alkalmazás több fájlból épül fel. Létezik egy html nyelven íródott fájl, amely a 22. kódrészletben látható. Meghatározza, milyen elemekből épül fel az óra számlapja²³, ehhez tartozik egy css fájl is, amely ezen elemek kinézetéért, elrendezéséért felel.

```
1 <svg class="background">
2   <use id="myAnimation" href="#frames" />
3   <div id="center">
4     <text id="timeLabel" />
5     <text id="dateLabel" />
6     <text id="accuLabel" />
7     <text id="stepsLabel" />
8     <text id="heartRateLabel" />
9   </div>
10 </svg>
```

Kódrészlet 22. Fitbit számlap

²² <https://dev.fitbit.com/build/guides/application/>

²³ <https://dev.fitbit.com/build/guides/clockfaces/>

Az app mappába található az index.js nevezetű fájl, amely JavaScript nyelven íródott. Ez a modul kérdezi le az okosóra szenzor adatait, illetve jeleníti meg a html kódban látott elemeken. Kezdetben minden egyes elem azonosítása történik id-k alapján, a 23. kódrészlet alapján, majd különböző értékek társulnak, annak függvényében, hogy az adott mező milyen adat megjelenítésére szolgál.

```
1  const accuLevel =
2      document.getElementById("accuLabel");
3  const stepsCounter =
4      document.getElementById("stepsLabel");
5  stepsCounter.text =
6      "Steps: " + today.adjusted.steps;
7  accuLevel.text =
8      "\u26A1 " + Math.floor(battery.chargeLevel) + "%";
```

Kódrészlet 23. Elemek beazonosítása

Az konkrét óra megjelenítése is hasonló módon történik, ahogy a 23. kódrészlet mutatja.

Lehetőség van különböző események figyelésére, amely megfelelő módon dokumentálva van a Fitbit Device API²⁴ dokumentációk közt. Az esemény a szenzor érték változása esetén végrehajtható.

```
1  if (HeartRateSensor &&
2      appbit.permissions.granted("access_heart_rate")) {
3      const hrm = new HeartRateSensor();
4      hrm.addEventListener("reading", () => {
5          heartRateLabel.text =
6              "\u2665 " + `${hrm.heartRate}`;
7          if (hrm.heartRate !== null &&
8              exercise.state == "stopped") {
9              sendMessage(hrm.heartRate);
10         }
11     });
12     hrm.start();
13 }
```

Kódrészlet 24. Pulzus esemény

Ha az óra alkalmazás rendelkezik pulzus mérő szenzorral, illetve, ha megkapta a felhasználótól az erre szükséges engedélyt, ami a 24. kódrészlet 1. – 2. sorában olvasható, akkor elindításra kerül a szenzor értékének az olvasása és megjelenítése, ahogy az a 4. – 6. sorban látható, ezt követi egy ellenőrzés, ha az érték nem null, azaz sikerült értéket visszakapni, akkor a sendMessage() függvény elküldi az adott értéket a companion alkalmazásnak, amely a felhasználó telefonján található. A sendMessage() a függvény tulajdonképpen egy socketet²⁵ nyit meg, és azon keresztül történik az adat átküldése a telefonra Bluetoothon keresztül. Az óra

²⁴ <https://dev.fitbit.com/build/reference/device-api/heart-rate/>

²⁵ <https://dev.fitbit.com/build/reference/device-api/messaging/>

alkalmazás háttérét egy mozgókép²⁶ biztosítja, amely tulajdonképpen 19 képből áll, ezek periodikus cseréjéből körvonalazódik ki a tulajdonképpeni mozgókép.

```
1  messaging.peerSocket.addEventListener("message", (evt) => {
2    var hdata=JSON.stringify(evt.data);
3    var data = hdata;
4    var url = '$firebaseURL';
5    fetch(url, {
6      method: 'PUT',
7      body: JSON.stringify(hdata),
8    })
9      .then(response => response.json())
10     .then(data => {
11       console.log('Success:', data);
12     })
13     .catch((error) => {
14       console.error('Error:', error);
15     });
16  });
```

Kódrészlet 25. Üzenet küldés

A companion rész is JavaScript nyelven íródott, amely tulajdonképpen eseményfigyelő részekből áll. Ha üzenet érkezik a nyitott socketen²⁷, akkor az adatot feldolgozza, ami a 25. kódrészlet alapján történik. Tulajdonképpen kiveszi a számára fontos részt, jelen esetben a pulzus adatot, majd egy fetch metódus segítségével a megfelelő Firebase linkre feltölti. Ez a művelet egy http PUT művelet, mivel nem célja az adat tárolás, csupán a jelenlegi legfrissebb adat frissítése. A tárolás funkcióját elvégzi a már említett felhő alapú szolgáltatás, a Fitbit Web API.

Mivel nem áll rendelkezésre olyan API metódus, amely meghatározná az eszközt használó személy egyedi azonosítóját, szükséges a Web API használata a companion részen belül. Az API használatához szükséges az óra alkalmazásnak is tartalmaznia hasonló, Fitbit fiókba beléptető rendszert, mint amivel a telefonos alkalmazás is rendelkezik és a 12. ábrán szemléltetve van. Ehhez a lépéshez szükséges a settings²⁸ mappán belül található jsx fájlba megadni az azonosításhoz szükséges adatokat, ami a 26. kódrészletben látható. Ilyen adatok az alkalmazásba megjelenő név, státusz, token, illetve azonosításhoz szükséges URL cím, előzetesen regisztrált alkalmazás kliens, illetve titkos azonosítója, továbbá az engedély típusa.

```
1  <OAuth
2    settingsKey="oauth"
3    title="Login"
4    label="Fitbit"
```

²⁶ <https://dev.fitbit.com/build/guides/user-interface/animations/>

²⁷ <https://dev.fitbit.com/build/reference/companion-api/messaging/>

²⁸ <https://dev.fitbit.com/build/reference/settings-api/>


```

5   status="Login"
6   authorizeUrl="https://www.fitbit.com/oauth2/authorize"
7   requestTokenUrl="https://api.fitbit.com/oauth2/token"
8   clientId="*****"
9   clientSecret="*****"
10  scope="profile"
11  />

```

Kódrészlet 26. Fitbit beállítások

Miután megtörtént a felhasználó beléptetése a Fitbit felhasználói fiókjába, megtörténhet az egyedi azonosító meghatározása, ahogy a 27. kódrészlet tárgyalja. Ezzel az azonosító segítségével az óra által mért pulzus adat a megfelelő mezőbe küldi az adatokat, ahonnan a telefonos alkalmazás leolvassa, ahogy a 21. kódrészletben látható.

```

1  settingsStorage.onchange = evt => {
2    if (evt.key === "oauth") {
3      let data = JSON.parse(evt.newValue);
4      fetch('https://api.fitbit.com/1/user/-/profile.json', {
5        method: 'GET',
6        headers: {
7          "Authorization": 'Bearer ${data.access_token}'
8        }
9      })
10     .then(response => response.json())
11     .then(data => {
12       console.log('Success:', data['user']['encodedId']);
13       id = data['user']['encodedId'];
14     })
15     .catch((error) => {
16       console.error('Error:', error);
17     });
18   }
19 };

```

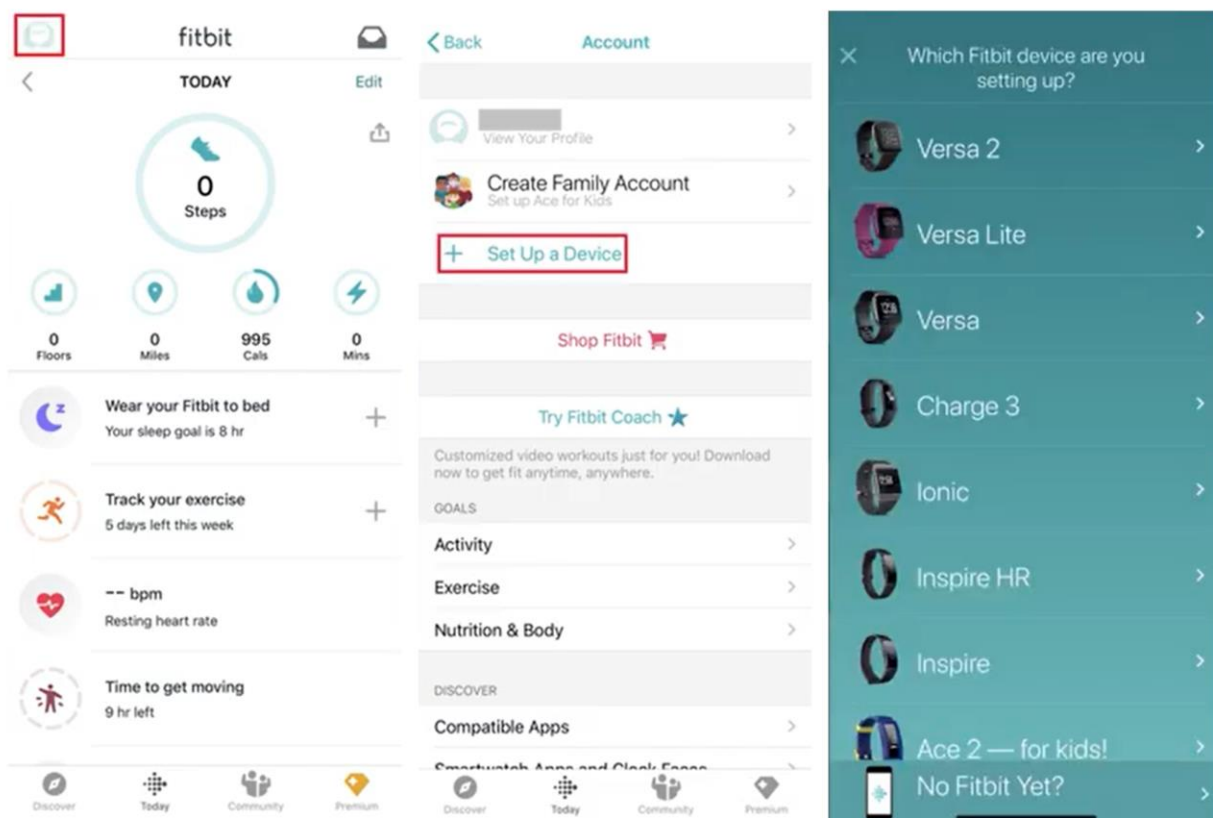
Kódrészlet 27. Fitbit egyedi azonosító meghatározása

A 27. kódrészlet egy fetch metódus segítségével lekérdezi az összes profilhoz köthető adatot, majd a megfelelő mezőbe található egyedi azonosítót egy változóba menti, amelyet a 25. kódrészlet felhasznál az adatbázissal történő kommunikáció létrehozása során.

6. Üzembe helyezés

Az üzembe helyezés során több kritikus pontot végre kell hajtani, annak érdekében, hogy az alkalmazás a vártnak megfelelő módon viselkedjen.

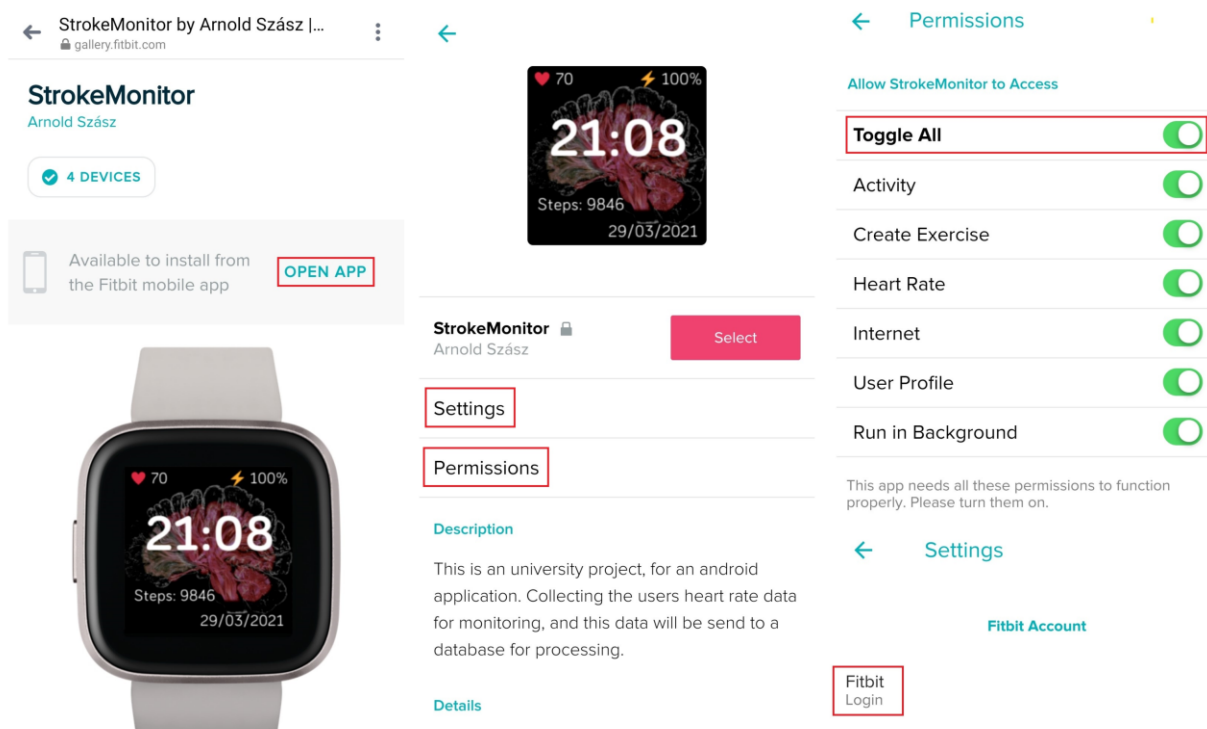
Kezdetben szükséges telepíteni a Fitbit²⁹ nevű alkalmazást. Az alkalmazás sikeres telepítése után szükséges regisztrálni, ezt követően a felhasználó beléphet és megkezdheti az óra hozzáadását a fiókjához.



Ábra 14. Fitbit óra hozzáadása

Ahogy a 14. ábrán látható a felhasználónak szükséges megérintenie a profil képét, ezzel elő nyílik egy új menüpont, itt el kell érintse a „Set Up a Device” mezőt, ezt követően elő ugranak a lehetőségek, itt ki kell válassza azt az órát amilyennel rendelkezik, figyelembe véve, hogy a StrokeMonitor csak a Fitbit Versa Lite, Versa, Versa 2, Ionic eszközökkel működik. Ezt követően meg kell adja az alkalmazás számára szükséges engedélyeket, mint például a helymeghatározás, Bluetooth kapcsolat keresése. Ha ezeket megadta a felhasználó a telefon megkeresi a közelében lévő eszközt, majd a felhasználási feltételek elfogadása után egy 4 számból álló pin kód jelenik meg az óra képernyőjén, ezt a felhasználó be kell írja a telefonos alkalmazásba, így jöhet létre a kapcsolat a telefon és az óra között.

²⁹ <https://play.google.com/store/apps/details?id=com.fitbit.FitbitMobile&hl=hu&gl=US>

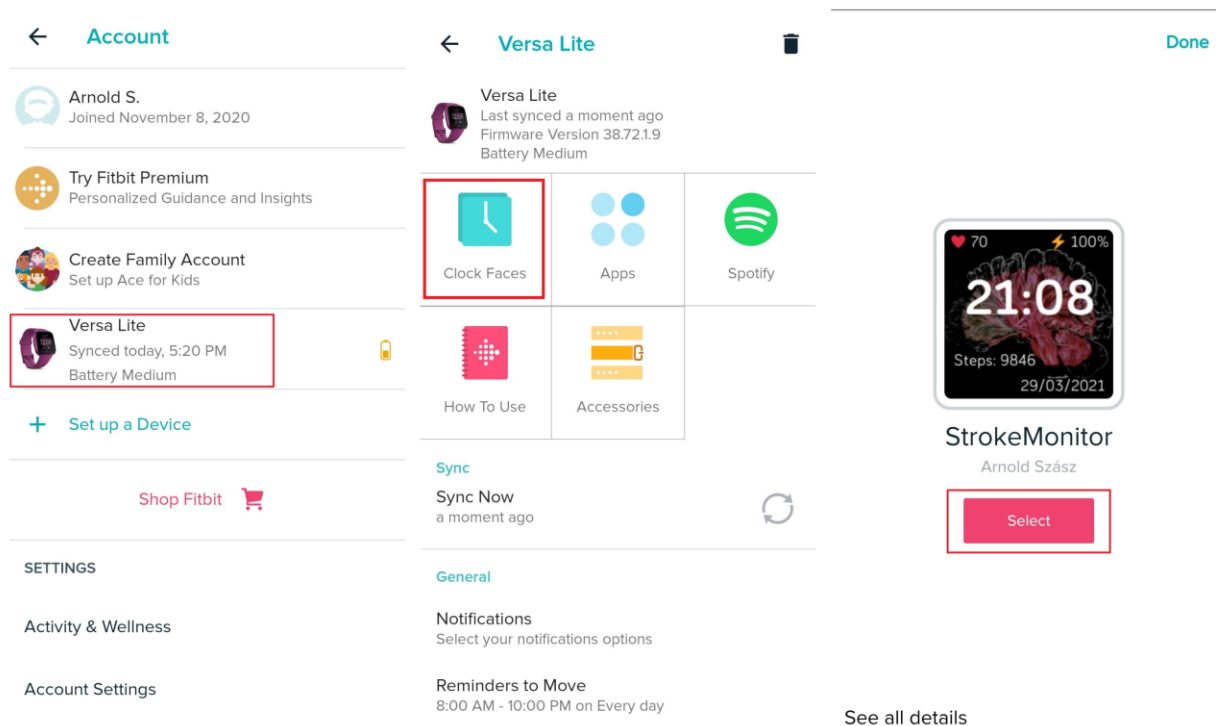


Ábra 15. Fitbit StrokeMonitor telepítés

A StrokeMonitor nevű óra számlap telepítéséhez szükséges ellátogatni a Fitbit Clock Face galériába³⁰ ahol le kell tölteni a számlapot. Ezt követően, ahogy a 15. ábrán is látható szükséges megadni az össze engedélyt annak érdekében, hogy az összes funkció elérhető legyen. Az engedélyek közül a legfontosabb a „Heart Rate” és a „Run in Background”. Az előbbi szükséges a pulzust mérő szenzor eléréséhez és lekérdezéséhez, utóbbi szükséges annak érdekében, hogy kikapcsolt képernyővel is folyamatosan lekérdezésre kerüljön az adat. A Settings menübe szükséges bejelentkezni a Fitbit fiókba ezzel engedélyt adva az óra számlap alkalmazásnak, hogy lekérdezze a felhasználó egyedi azonosítóját.

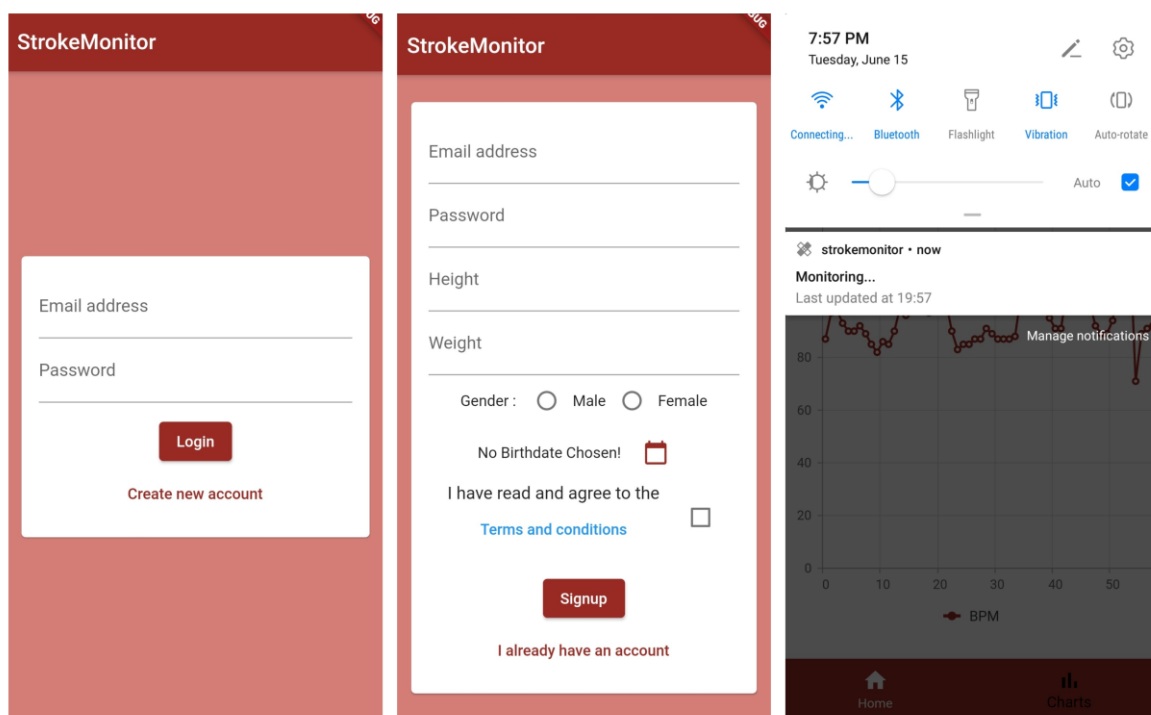
A következő lépésbe, alapértelmezettnek kell állítani az előbb telepített számlapot, ahogy a 16. ábrán szemléltetve van. A felhasználó el kell érintse az előbbi lépésben konfigurált órát, ezt követően megnyílik az óra konfigurálására szolgáló panel. Itt a Clock Faces menüpontra lépve, ki kell keresse a StrokeMonitor nevezetűt, ezt követően elkell érintse a Select mezőt. Így az óra számlapja megváltozik a StrokeMonitor számlapra. Pár másodperc kalibrálás után (leellenőrzi, hogy minden engedély megvan, ami a működéshez szükséges), amelyet automatikusan végez, elkezdi szolgáltatni az adatot a StrokeMonitor telefonos alkalmazás felé.

³⁰ <https://gallery.fitbit.com/details/109aaf9e-7d6e-402c-972d-d487a91f0c72>



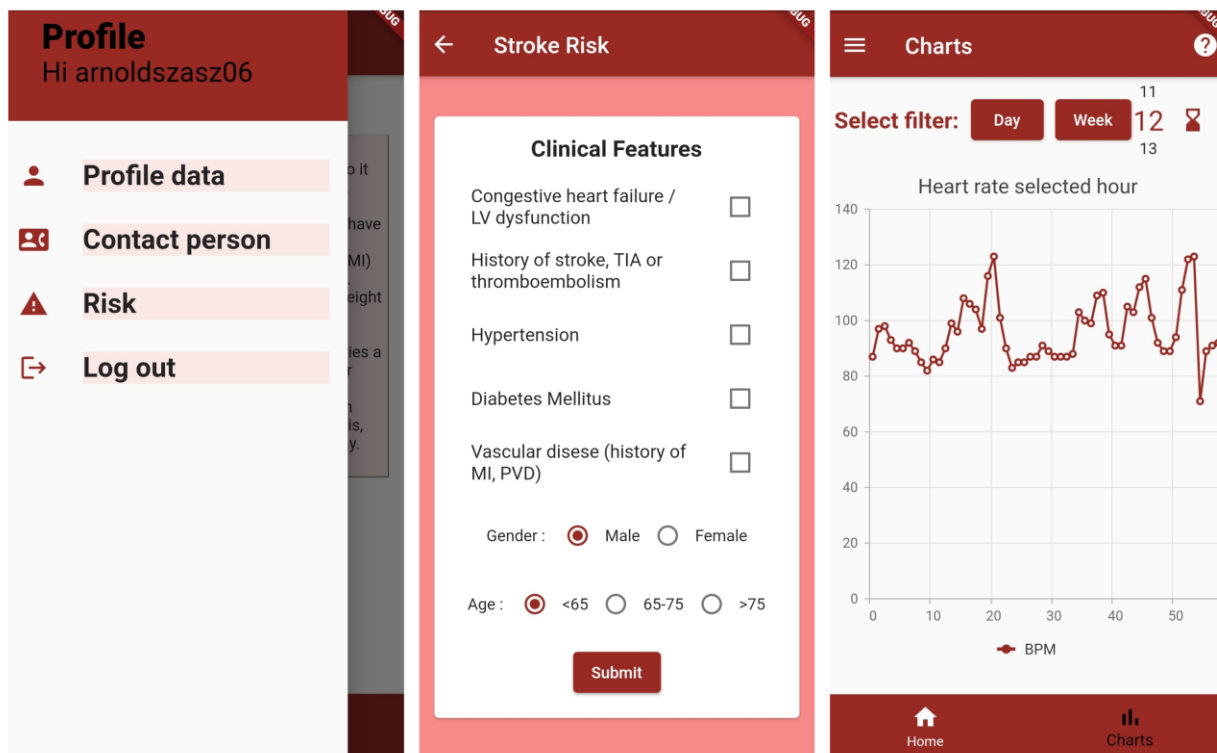
Ábra 16. Fitbit óraszámplap beállítása

Utolsó lépésként a felhasználónak szükséges telepítenie a StrokeMonitor telefonos applikációt, ahol egy regisztrációs vagy belépés folyamat után elkezdheti használni a különböző funkciókat, mint a monitorizálás, amely a 17. ábrán látható.



Ábra 17. StrokeMonitor kezdő lépések

A 18. ábrán a felhasználó által használható egyéb funkciók egy része látható. A menüből kiválaszthatja milyen funkciót szeretne elérni. Látható a rizikó pont meghatározására létrehozott kérdőív továbbá a grafikon funkció, amelybe az aznapi 12:00 - 12:59 közötti pulzus érték látható.



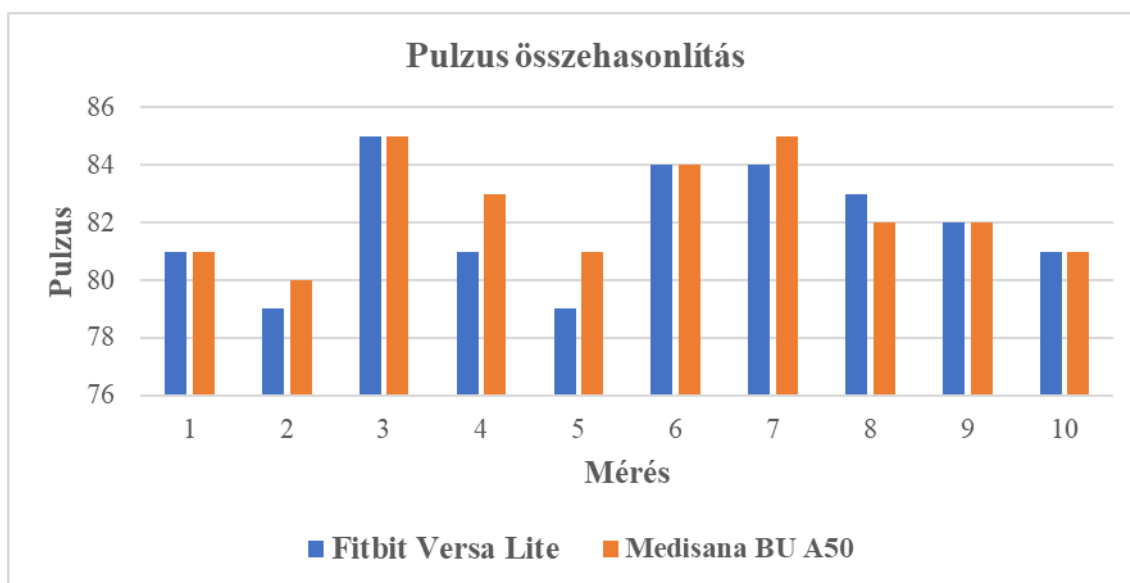
Ábra 18. StrokeMonitor használat

7. Kísérletek

Egy kísérlet során összehasonlításra került, milyen pontossággal méri a tesztre használt óra, azaz a Fitbit Versa Lite illetve egy Medisana BU A50 nevezetű vérnyomásmérő a pulzust. A vérnyomásmérő az adatlapja szerint $\pm 5\%$ -os pontossággal képes mérni, amelyet a Fitbit Versa Lite, mind a 10 mérési pontban tartani tudott ahogy az a 19. ábrán is látható. A grafikonok a kísérletek szemléltetésére a Microsoft Excel³¹ nevű programba készültek.

Fitbit Versa Lite	Medisana BU A50	Százalékos eltérés
81	81	0%
79	80	1%
85	85	0%
81	83	2%
79	81	2%
84	84	0%
84	85	1%
83	82	-1%
82	82	0%
81	81	0%

Ábra 19. Okosóra összehasonlítása vérnyomásmérővel százalékos eltérés



Ábra 20. Okosóra összehasonlítása vérnyomásmérővel

Ahogy a 19. és 20. ábrából kiderül, egészen hasonló értékeket mértek az eszközök, a legnagyobb eltérés az 5. mérési pontban látható, ahol eléri a 2%-os eltérést. Az alapérték a vérnyomásmérő által mért érték, ehhez viszonyítva van az okosóra által mért érték. A mérések

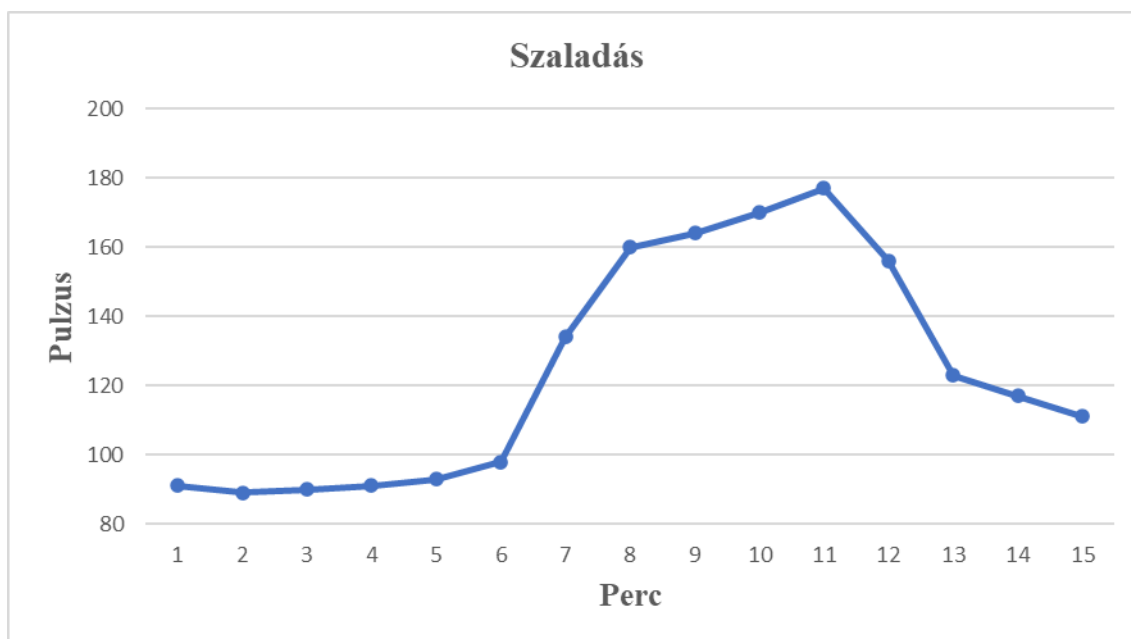
³¹ <https://support.microsoft.com/en-us/excel>

alapján kijelenthető, hogy egészen pontosan képes mérni a kísérletben használt okosóra. A kísérlet során a teszt alany nyugalmi pulzusa volt mérve.

7.1. Egyéni kísérletek

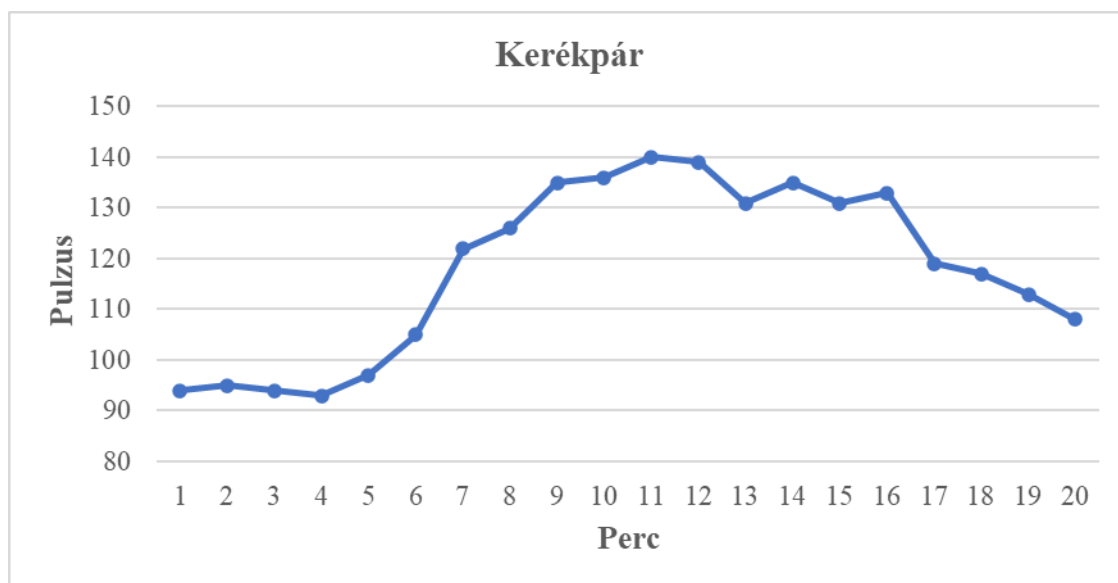
Eltérően az előbbi kísérlettel, a következő három kísérleti pontban a tesztalany pulzusa lesz mérve különböző terheléses próbák során. Az első kísérlet során a tesztben résztvevő alanynak a pulzusa Fitbit Versa Lite okosóra segítségével került monitorizálásra. A feladat összesen 15 percet vett igényben. Az első 5 perc nyugalmi idő, amikor tulajdonképpen az alany egyhelyben ül, ezt követi 5 perc szaladás, amelynek első percében fokozatosan gyorsuló mozgást kell végezni, ezt a sebességet tartania kell 3 percig, majd az utolsó 1 percben még erőteljesebben kell szaladnia, ezt követi 5 perc nyugalmi idő. Az első 2 perc lassú sétát jelent, mivel nem ajánlatos egyből megállni, erősebb, hosszabb ideig tartó terhelés esetén, végül 3 perc teljes leállás történik.

A 21. ábrán jól látható, hogy a kísérletben résztvevő személy nyugalmi állapotú pulzusa 90 BPM körül mozog. A szaladás megkezdése során, szépen fokozatosan növekvő tendenciát mutat a pulzusszám. A bemelegítő 1 perc után a pulzusszám 134 BPM-ről 160 BPM-re ugrik, ami lassan emelkedik a 3 perc folyamatos, azonos sebességű szaladás során. Az utolsó 1 perc feszített tempójú szaladás 177 BPM-nél csúcsosodik. A nyugalmi idő első percében, 156 BPM-re esik a pulzus, ezt követi a teljes leállás, ami 110 BPM-nél ér véget. A szaladás után 15 percre a teszt alany pulzusa megközelítőleg visszaállt a kísérlet előtt mért szintre, azaz 90 BPM.



Ábra 21. Gyors szaladás közben pulzus változás

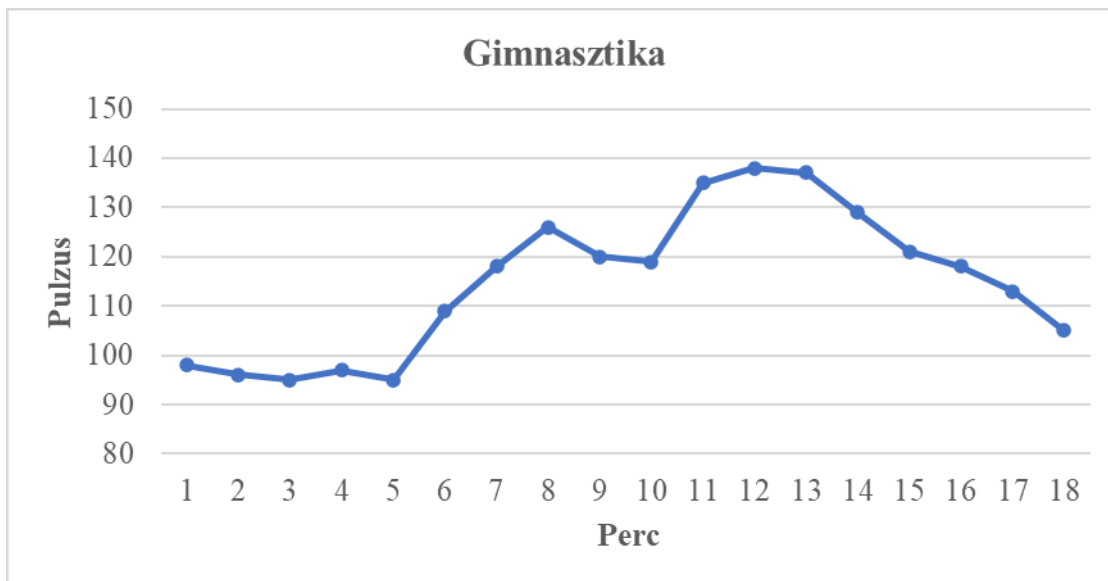
A második kísérlet során a teszt összesen 20 percet vett igénybe, ebből 5 - 5 perc pihenési idő, hasonló módon, mint az előbbi kísérletben. A konkrét terhelési idő 10 perc, amikor a kísérletben résztvevő, nagyjából 20 – 25 km/óra sebességgel kell kerékpározzon. Ez idő alatt nagyjából 4 km távolságot képes megtenni.



Ábra 22. Kerékpározás közben pulzus változás

Ahogy a 22. ábrán látható a pulzus változása hasonló ívet ír le, mint az első kísérletben. Az első 5 perc pihenési időszaka, nagyjából 95 BPM körül mozog. Az ezt követő 10 perc kerékpározás során a pulzus felkúszik 130 BPM fölé, s 130 – 140 BPM közt váltakozik, ez annak tudható be, hogy a kísérlet a szabadban zajlott, és néha erősebb széllesek nehezítették meg a tesztet, ilyenkor nagyobb erőfeszítés szükséges a sebesség tartáshoz. Az utolsó 5 perc, a pihenési szakasz során a pulzus szépen, fokozatosan csökkent, egész 98 BPM értékig. Hasonló módon, a teszt befejezése után 15 percre a tesztelő pulzusa visszaállt a normál értékre.

A harmadik kísérlet során egy tornaórakon megszokott feladatsort kellett végrehajtson a tesztben résztvevő. Az előbbi kísérletekhez hasonlóan, ennél a kísérletnél is érvényes volt az előtte, utána 5 - 5 perc pihenési idő. A konkrét erőfeszítés a következő tornagyakorlatokból állt: 2x15 fekvőtámasz, 2x20 guggolás, 2x30 felülés. A feladatok közt 30 másodperc extra pihenési idő állt rendelkezésre.



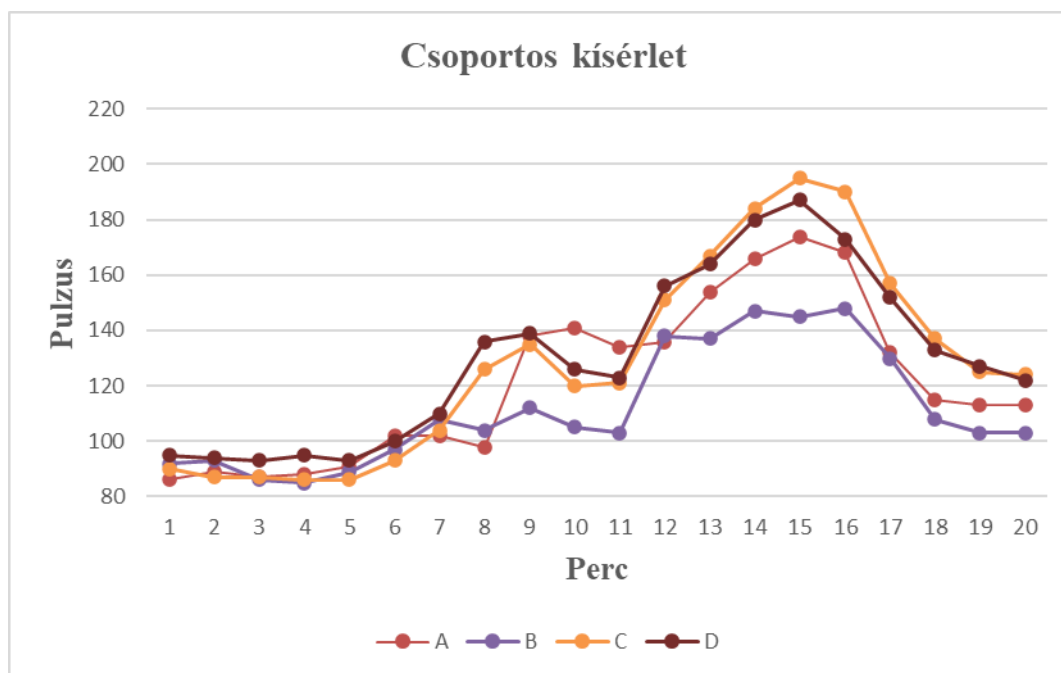
Ábra 23. Gimnasztika közben pulzus változás

A 23. ábrán látható, hogy a kezdeti, pihenés közbeni pulzus 95 BPM körül mozog, ezt követi 8 perc gimnasztika. Látható, hogy a pulzus szépen csúcsosodik, egész 126 BPM-ig, azonban a feladatok közt bevezetett pihenési idő megmutatkozik, mivel a maximális érték, nem éri el a 140 BPM értéket. A grafikon szerint a legkevesebb erőfeszítés a guggolás során történt, hiszen az előtte lévő fekvőtámasz feladathoz képest csökkent a pulzusszám. A legnehezebb feladat a felülés elvégzése volt, köszönhető annak, hogy a végén került rá sor, ennél a feladatnál a csúcs a 138 BPM. Az utolsó 5 perc pihenési idő során fokozatosan csökkent a pulzus értéke. Körülbelül 15 perc pihenés után állt vissza a megszokott pulzus értéke.

7.2. Csoportos kísérlet

Egy csoportos kísérlet során, előre meghatározott feladatsort kellett végrehajtaniuk a kísérletben résztvevő egyéneknek. A feladatok célja a minél magasabb pulzus szám elérése. A feladat összesen 20 percet vett igényben, ebből az első 5, illetve az utolsó 5 perc pihenési idő, annak érdekében, hogy meghatározásra kerüljön az alany nyugalmi pulzusa. A különböző feladatokat 10 percig végezték a tesztben résztvevő alanyok. A feladatok időre pontosan vannak elvégezve, hogy azonos mérési pontok keletkezzenek. A feladatsor 1 perc súlyemeléssel kezdődik 3 – 3 kg súllyal, ezt követi 30 másodperc pihenési idő, ami az első 5 perc minden feladata után megtörtént. A súlyemelés után következett 1 perc kitörés szintén súllyal, majd fél perc guggolás feltartott kézzel, súlyokat fogva. Elasztikus gumiszalag nyújtása a következő, aminek az ellenálló képessége 15 kg. A súlyokat használó gyakorlatok után szaladás következett különböző módokban, ennél a résznél nem volt a feladatok közt 30 másodperc szünet. Első mód

a sima szaladás közepes tempóval 1 percig, ezt követően 1 – 1 perc láb emelés elől, hátul majd végezetül 1.5 perc testsúly áthelyezés egyik lábról a másikra kis szökésekkel.



Ábra 24. Csoportos kísérlet

A 24. ábrán látható az összesítő grafikon, amely tartalmazza a négy tesztalany pulzusát percre lebontva. Két nő és két férfi vett részt a kísérletben. Az A és a B alany nő, míg a C és D férfi, mind a 4 alany egészséges, 20-as éveiben járó személy. Az A alany heti rendszerességgel, a B alany napi rendszerességgel, míg a C és D alany alkalmi rendszerességgel sportol.

Jól megfigyelhető, hogy nagyjából azonos eredmények születtek, csupán a pulzus mértéke változó, tekintettel arra, hogy az alany milyen fizikai állapotban van. A kezdeti nyugalmi pulzus 90 BPM körül mozog minden esetben. Az első részfeladat során a pulzus lassan emelkedik, de nem számottevően, majd ívesen felmegy egy szintre, ezt követően újra csökkenni kezd, ez betudható annak, hogy némely feladat elvégzése nem követelt nagy erőnlétet. A szaladás során az összes alanyunk egy felfelé ívelő görbét ír le a pulzusa, majd a második pihenési idő során hasonló íveken történik a pulzus normalizáció. Megfigyelhető, hogy a B alany pulzusa nem éri el a 150 BPM értéket, míg a C alany pulzusa szinte 200 BPM értéknél csúcsosodik, pontosabban 195 BPM. A grafikon íveinek az esetleges különbség adódhat abból a tényből, hogy bizonyos feladat típusokat egyes alanyok többször gyakorolták, esetleges rutinnal rendelkeznek, míg mások a teszt során találkoztak először a feladattal. A rutinnal rendelkező alany kisebb erőfeszítéssel tudja végrehajtani ugyanazt a feladatot, mint társai.

8. Következtetések

A határidő lejárta előtt sikerült megvalósítani a kitűzött célok jelentős részét, mivel elkészült az okosóra, illetve a telefonos szoftver. Teljes mértékben működő képesek és tesztelve voltak valós körülmények közt.

8.1. Megvalósítások

Az okosóra szoftver képes megjeleníteni az óra szenzorjainak az értékét. Ilyen érték az akkutöltöttség, aznap megtett lépésszám, illetve a friss pulzusszám, amelyet továbbítani képes Bluetooth kapcsolat segítségével a telefon irányába, ahol feltöltésre kerül a felhasználó egyedi azonosítójával létrehozott mezőbe. A továbbítás folyamatosan üzemel, még kikapcsolt képernyővel is. Mindemellett mutatja az időt és a napi dátumot.

A telefonos alkalmazás képes a folyamatos pulzusszám monitorizálására, illetve képes jelzést, riasztást küldeni üzenet formájában a felhasználó által beállított kontakt személynek. Az alkalmazás menedzselni tudja a felhasználó, illetve a kontakt személy adatait regisztráció vagy bejelentkezést követően. Képes kiszámítani a felhasználó stroke rizikó számát, amelyet egy kérdőív alapján valósít meg. Továbbá a felhasználó megtekintheti összesítő grafikonok segítségével a heti vagy napi pulzus számát különböző leosztásba. Az alkalmazás napi tanácsokkal látja el a felhasználót a stroke megelőzéssel kapcsolatba.

Mindkét alkalmazás cross platformos, témához illő, modern dizájnnal rendelkezik. Használatuk egyszerű, így alkalmasak a napi szintű használatra, hiszen csak akkor tudják elvégezni a nekik szánt feladatokat, ha az óra alkalmazás használatba van, illetve a telefonos alkalmazás el van indítva háttér szolgáltatásként.

8.2. További fejlesztési irányok

Fejlesztési lehetőségként fent áll a használható okosórák számának bővítése, mivel jelenleg csak a Fitbit által gyártott órákkal működik a rendszer, azon belül se az összes verzióval. Próbálkozások voltak más okosóra gyártó céggel is, mint például a Huawei, azonban nem lehetett megoldani a cég által gyártott eszközökkel a platform függetlenséget, illetve az adatokhoz való hozzáférés is nehézkes, továbbá nem lehetett megvalósítani a valós idejű szenzor adat leolvasást. Fejleszteni lehet az energiatakarékosság részét is az alkalmazásoknak, mivel a folyamatos internet, Bluetooth kapcsolat megrövidíti a készülékek üzemidejét. Az adatok biztonságosabb tárolása, kriptálása is fejlesztési lehetőségként szerepel, mivel jelenleg a fejlesztés gördülékenyebb végrehajtása végett, csak a Firebase által biztosított titkosítások,

adatvédelmek érvényesülnek. További fejlesztési lehetőségként szolgálna, az alkalmazás publikálása a Google Play áruházba, illetve Apple App Store-ba, ezt megelőzően a Firebase adatbázis lecserélése egy erre alkalmasabb, fix fizetési igénnyel rendelkező adatbázisra, esetleges saját API használata az adatok mentésére.

9. Irodalomjegyzék

- [1] Atrial Fibrillation and Mechanisms of Stroke: Hooman Kamel, Peter M. Okin, Mitchell S.V. Elkind, and Costantino Iadecola, 19 Jan 2016, Stroke. AHA journals 2016;47:895–900
- [2] Atrial Fibrillation: L. Brent Mitchell , MD, Libin Cardiovascular Institute of Alberta, MSD Manual, University of Calgary, Last revision Jan 2021
- [3] Belgyógyászat, Dr. Med. Gerd Herold, Medicina Könyvkiadó Zrt., Budapest 2015
- [4] CHADS2 Score for Stroke Risk Assessment in Atrial Fibrillation: Tarek Ajam, MD, MS Fellow in Cardiovascular Medicine, Feb 27, 2020
- [5] Paola Pierleoni, Luca Pernini, Alberto Belli, Lorenzo Palma, "An Android-Based Heart Monitoring System for the Elderly and for Patients with Heart Disease", International Journal of Telemedicine and Applications, vol. 2014, Article ID 625156, 11 pages, 2014. <https://doi.org/10.1155/2014/625156>
- [6] El-Amrawy, Fatema, and Mohamed Ismail Nounou. "Are currently available wearable devices for activity tracking and heart rate monitoring accurate, precise, and medically beneficial?." Healthcare informatics research 21.4 (2015): 315.
- [7] Kumar, Maradugu Anil, and Y. Ravi Sekhar. "Android based health care monitoring system." 2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS). IEEE, 2015.
- [8] Nguyen, Hoang H., and Jennifer NA Silva. "Use of smartphone technology in cardiology." Trends in cardiovascular medicine 26.4 (2016): 376-386.
- [9] AZ EURÓPAI PARLAMENT ÉS A TANÁCS (EU) 2016/679 RENDELETE, (2016. április 27.)
- [10] Flutter architectural overview, <https://flutter.dev/docs/resources/architectural-overview>

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREȘ
SPECIALIZAREA CALCULATOARE

Vizat decan

Conf. dr. ing. Domokos József

Vizat director departament

ș.l. dr. ing. Szabó László Zsolt