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, igy bonyolult feladatok elvégzését is jásznyi könnyedséggel hajtsák 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 kiszáradás. Mivel ezek az eszközök óriási körben elterjedtek, szinte minden ember nap mint nap magával viszi mindenhova, ezt kihasználva akár orvosi, megfigyelői célokra is fel lehet használni a szenzorok által mért adatokat, megkönnyítve az orvosok munkáját, növelve az élet színvonalt.

A betegségek sokszínűsége miatt, egyre többen szorulnak folyamatos megfigyelésre, amely korházi 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 pácienst mindennapi tevékenységei alatt. Természetesen ezek az eszközök nem érnek fel a korházakban talált eszközökhöz, azonban pontosságuk nem nagy arányban tér el egy profi eszköztől. Az ilyen rendszerek képesek akár több típusú betegség felfedezésére. Néhány esetben az idő, ami alatt a páciens eljut a rosszullét helyétől a korházig igen kritikus, sokszor perceken múlik az élet vagy a halál, ezért is fontos, hogy a veszélyeztettet 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 tekintetébe, illetve az egészségi állapotára kiterjedő kérdőívet tölthet ki. Az 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 pulzus száma, amelyet a telefonos applikáció percenként ellenőriz, valós időben elérve a tároló egységről. Összehasonlítja egy szakértők által kijelentett küszöb értékkel. 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 mükődése (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 összefügésben áll más stroke-ot okozó rizikófaktorokkal. Tehát kijelenthetjük, hogy egymást kiváltó és fenttartó folyamatokról beszélünk [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 kilalakulásának 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
- éltekor ≥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].

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. Orvos tudományban 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 korházon 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 funkcionalitásai:

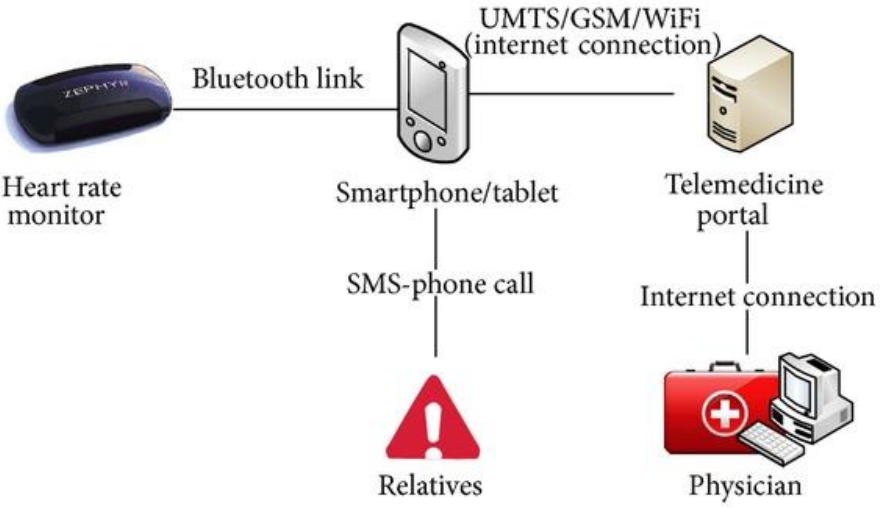
- részletes jelentések a felhasználó egészségügyi állapotáról
- adattárolás
- segélyhívások generálása
- távorvoslás adatok megosztásával

A legújabb tudományos fejlesztések lehetővé teszik, az orvosi eszközök kimozdítását a megszokott korházi környezetből, mivel az okostelefonok és a biometrikus érzékelők már széles körben elérhetőek. Fontos cél a szívbetegek monitorizálása s az esetleges betegségek megelőzése. Az ilyen betegséges 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

Számos hasonló rendszert fejlesztettek már [5], de nagyrészük csupán a nyers adatok megjelenítésével foglalkozik. 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 stressztesztben 20 egészséges egyén vett részt (6 nő és 14 férfi, 23-27 év között). 10 percig elemezte őket az algoritmus. Ki töltöttek 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.



Rendszer architektúra [5] cikk

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) különbféle teszteken esett át, egyidejűleg három gyorsulásmérőt használtak, ezeket egyidejűleg fűtatták egy Android vagy iOS eszközön, közben korházban 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. Igy 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 jeleníthetőek meg. Az Android based health care monitoring system [7] cikkben leírt rendszer ezen dolgok kiküszöbölésén dolgozik új módszerekkel. A rendszer a beteg testének különféle biológiai paramétereinek, például pulzusszám, vér oxigén telítettség, hőmérséklet folyamatos figyelését, teszi lehetővé. Az adatok feldolgozásra kerülnek egy webszerver és Android alkalmazás segítségével, ahol az orvos okos telefonján 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 biztosítania, 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 be állítható 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 egyes 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 forradalmasítják 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.

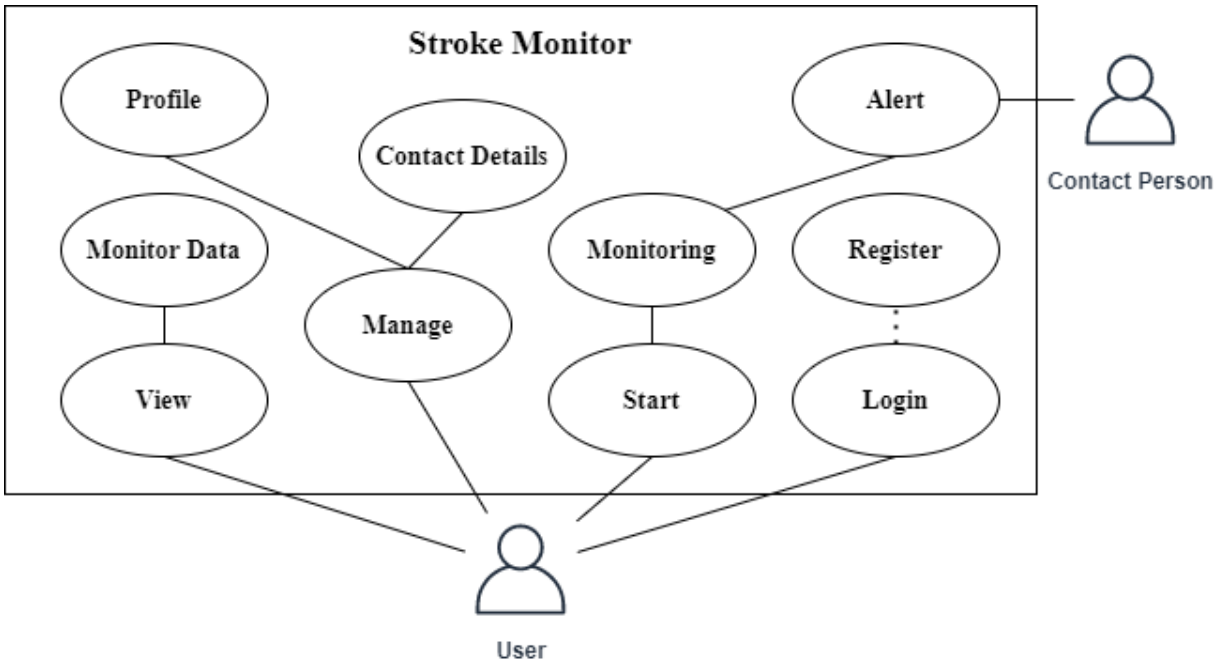
4. Követelmény specifikáció

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

4.1. Felhasználói követelmények

A 2. ábrán található a használati eset diagram, amely a rendszer modellezésére szolgál a megrendelő szemszögéből. Megjelenik, hogy kik és mire használják a rendszert. 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 viszonnyal rendelkező felhasználó érhet el. A felhasználó elindíthatja a megfigyelés funkciót monitorizálva percenként pulzusát. A monitorizálás során, ha az eszköz úgy dönt, hogy a felhasználó veszélybe lehet, akkor értesíti az előzetesen megadott kontakt személyt. A felhasználó továbbá menedzselheti a saját és a kontakt személy adatait. Megnézheti a monitorizálás során mentet adatokat különböző formátumba.



Ábra 2. Use case diagram

4.2. Rendszer követelmények

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

4.2.1. Funkcionális követelmények

- Regisztráció
- Bejelentkezés
- Megfigyelő funkció elindítása
- Mellék adatok megadása
- Grafikonok megtekintése
- Okosóra applikáció beüzemelése

A regisztráció során a felhasználó megkell adjon különböző adatok, mint email cím, jelszó, fizikai jellemzők (testsúly, magasság, nem). Sikeres regisztráció esetén az applikáció át navigál a fő oldalra. A bejelentkezéshez szükséges megadni egy olyan jelszó és email cím kombinációt, ami szerepel a regisztrált felhasználok 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. A mellék adatok megadása során ki kell tölteni egy kérdőívet, amelyben egészségügyi állapotra reflektáló kérdések vannak.

Kitöltés után kiderül, a bevitt adatok alapján mekkora rizikóval rendelkezik a felhasználó. Továbbá meg kell adni a vészhelyzetben értesítendő személy nevét és telefon számát. A felhasználó grafikonok formájában megtekintheti különböző leosztásban milyen pulzus értékkel rendelkezett az elmúlt napokba, hónapokba. A felhasználó az okosóra applikációt le kell töltse készülékére, és a megfelelő módon üzembe kell helyezze a 6. fejezeteben 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

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.

4.2.2. Külső nem funkcionális követelmények

- Jogi háttér
- GDPR
- Fejlesztés
- Tesztelés

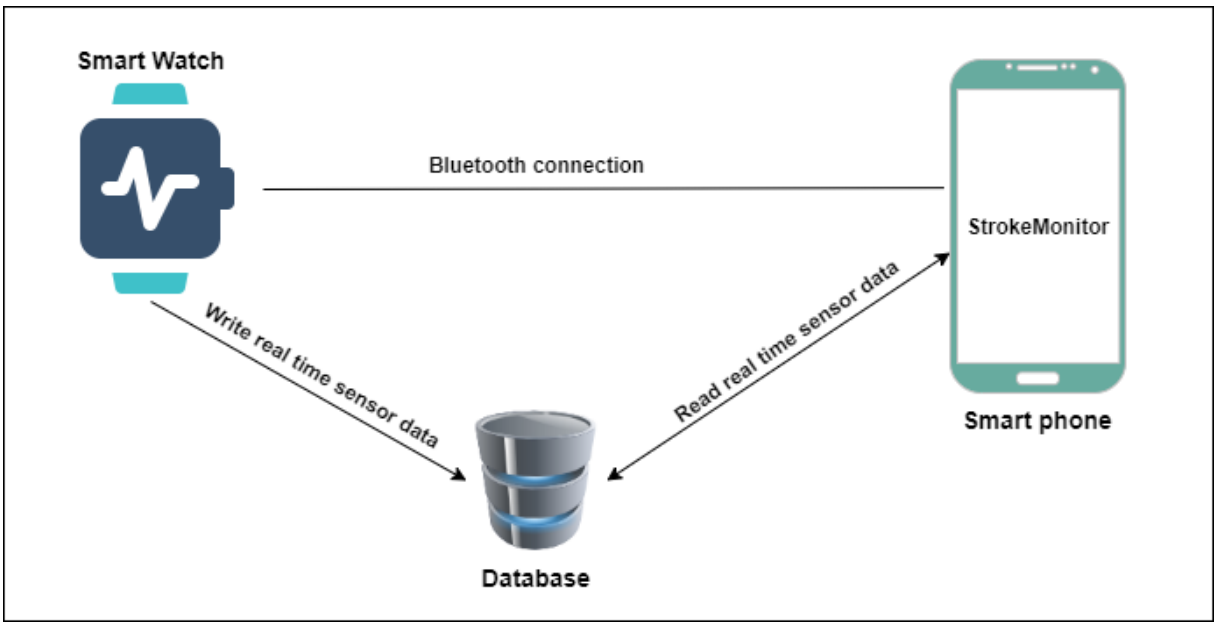
Mivel a rendszer orvosi használatra szánt, ezért be kell tartania az orvoslás egyik alapelvét, azaz az orvosi titoktartást, amely biztosítja, hogy a páciens tudta, s akarata nélkül nem kerülhet nyilvánosságra egyetlen adat se. Éppen ezért az adatok tárolása biztonságos módon kell legyen megvalósítva, úgy, hogy a felhasználók csak saját adataikhoz tudjanak hozzáférni. A fejlesztés és tesztelés során az adatok nem kerülhetnek ki a csapat keretein kívülre. Bármely ilyen

adatvesztés, szivárogtatás jogi következményeket vonhat maga után. Továbbá fontos a GDPR szerződés betartása mind a szoftvert birtokló fél, mind a felhasználó részéről. Igy 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 feltételeke mellett adhatók ki, akár a felhasználó beleegyezése nélkül [9].

5. A rendszer leírása

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

5.1. Architektúra

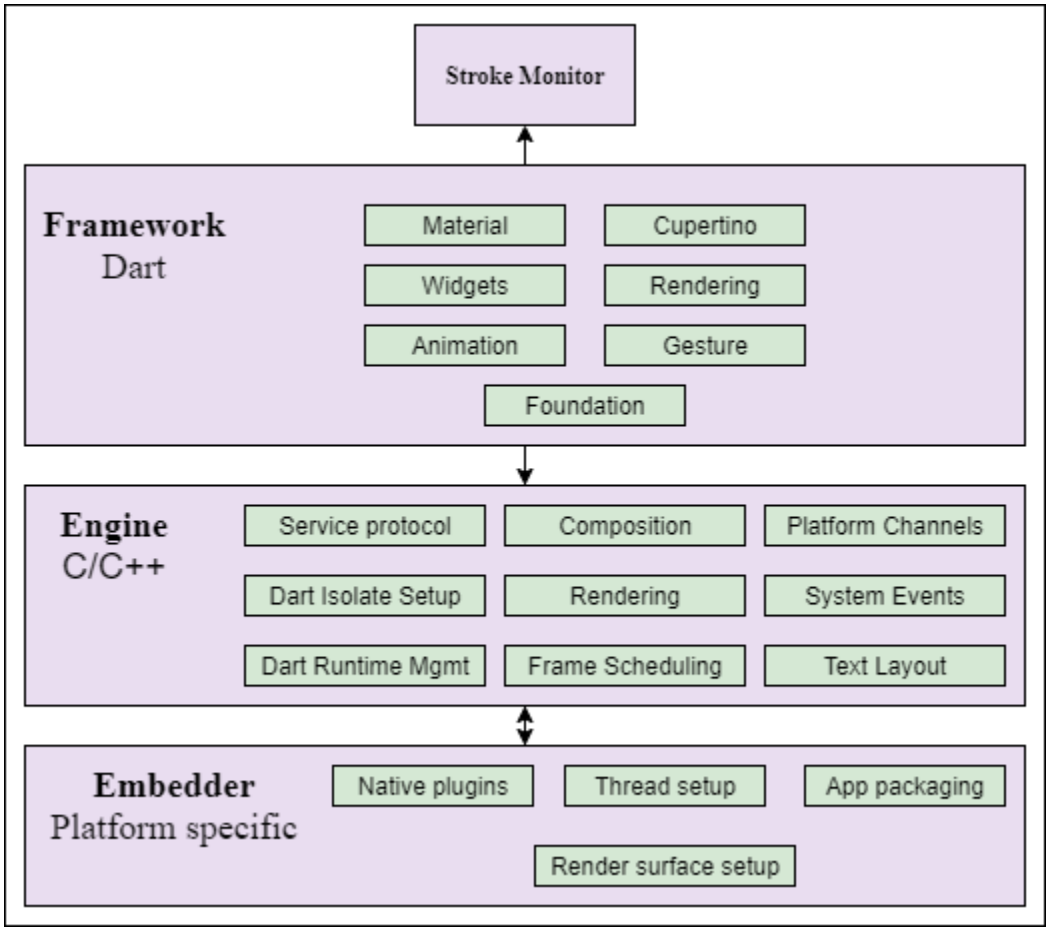


Ábra 3. Rendszer architektúra

A 3. 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 okos óra 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 4. 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].



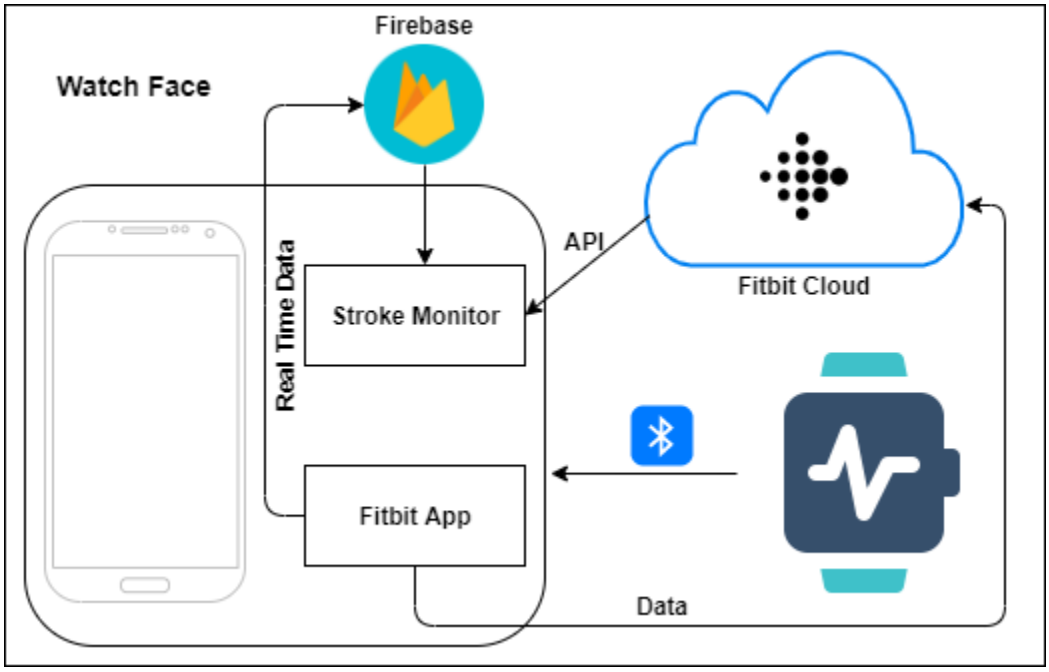
Ábra 4. Mobil alkalmazás architektúra

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, szövegmegjelenítő, 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 eszköz tarát 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 5. Óra applikáció architektúra

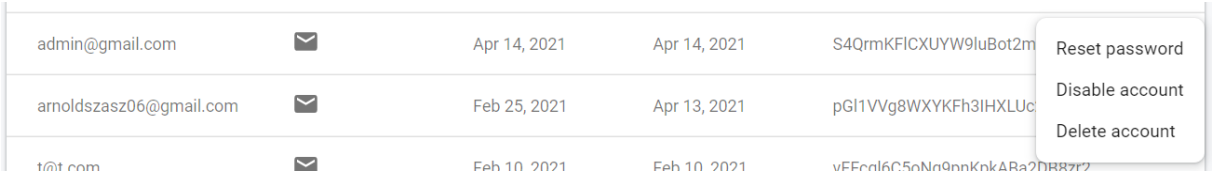
Az 5. ábrán az okosóra applikáció architektúrája figyelhető meg. Az órán található óra számlap begyűjti a szenzorok által mért adatokat és Bluetooth kapcsolat segítségével továbbítja a telefonon lévő Fitbit Applikáció felé. Ez a művelet után, a mért pulzus szám valós időben az adatbázisba kerül, míg a Fitbit által strukturált összes mért érték elküldésre kerül a Fitbit felhő alapú szolgáltatásába, ahonnan a Stroke Monitor a Fitbit APIn keresztül hozzáfér. A strukturált adatban napi szinten elmentésre kerül az átlagos égetett kalória, a minimum, maximum pulzus szám, továbbá a pulzus értéke perecenként a nap 24 órájában.

5.4. Adatbázis

A rendszer által használt adatbázis a Firebase, 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,

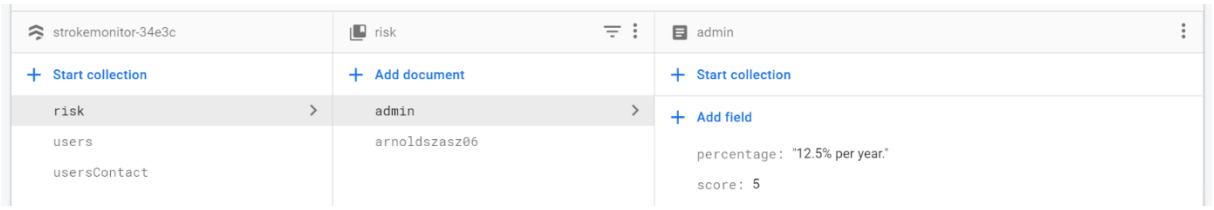
tulajdonképpen egy hídként szolgál az okosóra és a telefon közt, a valós idejű adatcserét lehetővé téve.

A regisztráláshoz és belépéshez szükséges metódusokat a Firebase Authentication biztosítja, a projekt esetében a jelszó emailcím páros van beállítva. Adatbiztonság szempontjából a jelszók az adminisztrátor számára se nyilvánosak, csupán az email címek és a hozzá fűzhető műveletek publikusak.



Ábra 6. Firebase Authentication

Ezen felül használatban van a Cloud Firestore, 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.



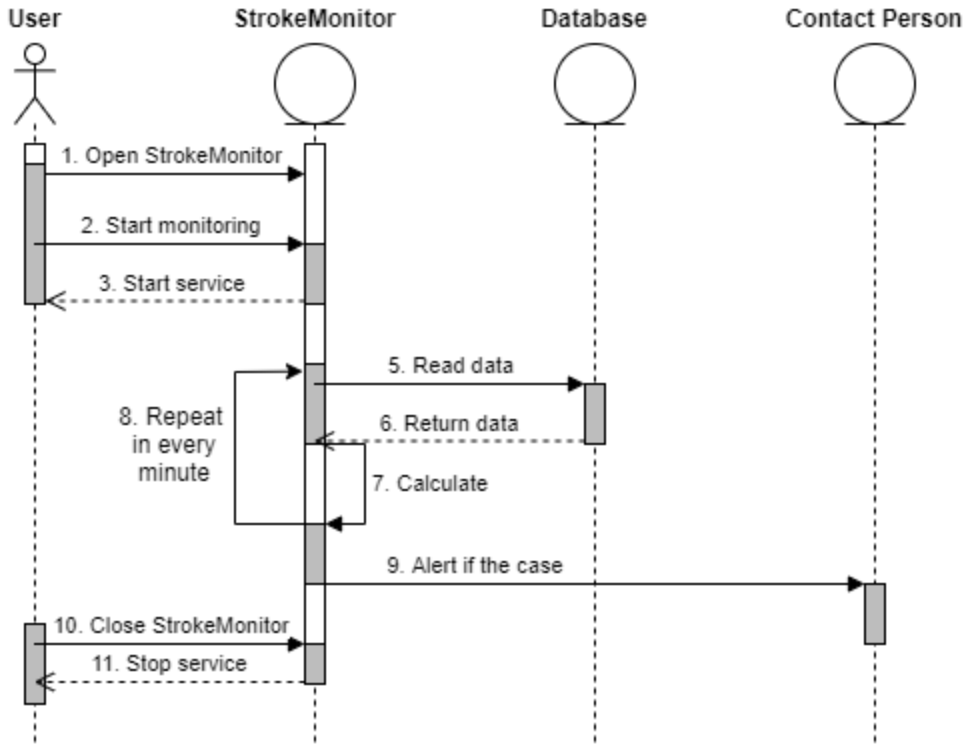
Ábra 7. Cloud Firestore szerkezet

A 7. 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. Minden egyes kollekcióba egyedi névvel ellátott documentumok szerepelnek. Úgy jön létre ez a dokumentumok, hogy a felhasználó által megadott email címből a @ előtti részt tekinti 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.

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.

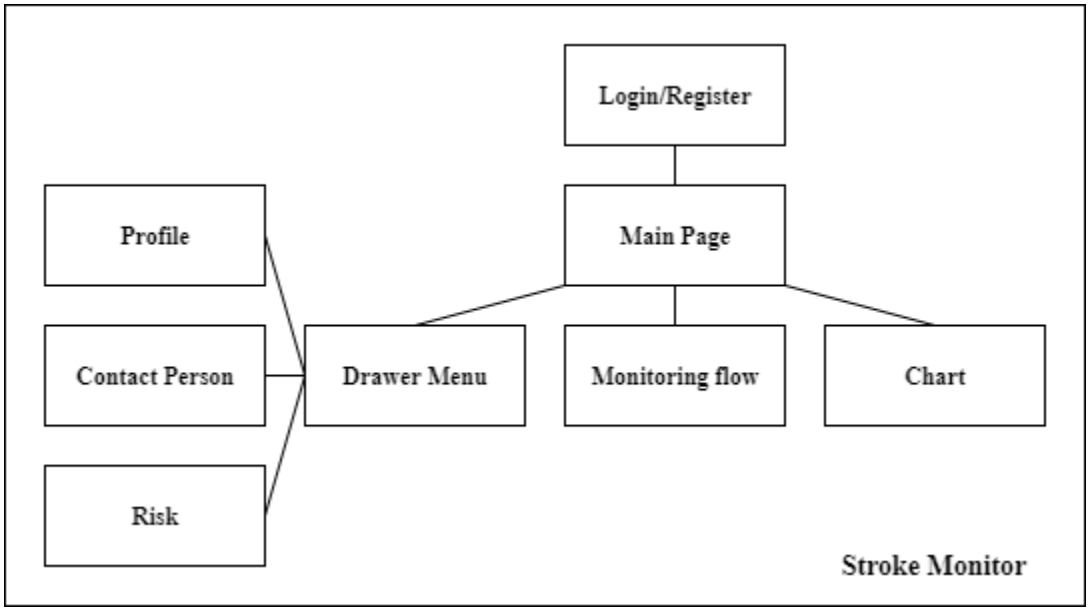
5.5. Szekvenciális diagram

A 6. ábrán a rendszer szekvenciális 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. Az alkalmazás minden percben egy előre meghatározott feladat soron megy végig. Adatot olvas az adatbázisból ezzel megkapja a valós időben mért szenzor értéket, amint megérkezik az adat, különböző számításokat hajt végre. Megnézi, hogy a felhasználó milyen rizikó számmal rendelkezik, illetve összehasonlítja a küszöb értékkel, amely az orvosi cikkek tanulmányozása során definiált küszöbszám. Ha a felhasználónak a valós idejű pulzus száma nagyobb, mint a küszöbszám és rizikó száma is nagyobb az átlagosnál, akkor értesíti a kontakt személyt, automatikusan tárcsázva a megadott telefonszámot. Végül, ha a felhasználó úgy dönt, hogy a megfigyelés már nem releváns, akkor leállíthatja az applikációt, ilyenkor is visszajelzést kap a művelet sikerességéről.



Ábra 6. Szekvenciális diagram

5.6. UI diagram



Ábra 7. Telefonos alkalmazás UI diagramja

A 7. ábrán a blokk telefonos applikáció blokk diagramja látható feltüntetve a fő alkotó komponenseket. A kezdő komponens a belépésre vagy regisztrációra szolgáló komponens, ezt követi a fő oldal, amely elágazik több irányba igy 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.

```
      initialRoute: '/',
      routes: {
        '/': (context) => StreamBuilder(
              stream: FirebaseAuth.instance.authStateChanges(),
              builder: (ctx, userSnapshot) {
                if (userSnapshot.hasData) {
                  return MainScreen();
                } else {
                  return AuthScreen();
                }
              },
            ),
        StrokeRiskScreen.routeName: (contex) => StrokeRiskScreen(),
        ProfileDataScreen.routeName: (context) => ProfileDataScreen(),
        ContactPersonScreen.routeName: (context) =>
ContactPersonScreen(),
      },
```

A kódrészletből kiderül, hogy az alapértelmezett útvonal a „/” jelölt, ami két irányba vezethet, egyik a bejelentkezés/regisztrációs felület, másik a fő képernyő, ami csak akkor látható, ha a felhasználó előtte már bejelentkezett. Lentebb a 13, 14, 15 sorokba útvonalak vannak, más képernyők irányába, például a rizikó képernyő, profil, illetve kontakt személy képernyő. Ezen útvonalak hívásánál definiálva van, melyik widget osztály fog megjelenni.

Ha a felhasználó nem rendelkezik felhasználói fiókkal, akkor regisztrálnia kell, vagy ha kijelentkezett akkor szükséges a már létező adatok megadása az újboni bejelentkezéshez.

Kezdetben két írható mező jelenik meg, egyik az email cím másik a jelszó beírásának a helye, ennek a widgetenek a struktúrája a következőképpen néz ki:

```
                    TextFormField(
                      key: ValueKey('password'),
                      validator: (value) {
                        if (value.isEmpty || value.length < 8) {
                          return 'Password must be at least 8
characters.';
                        }
                        return null;
                      },
                      decoration: InputDecoration(
                        labelText: 'Password',
                      ),
                      obscureText: true,
                      onSaved: (value) {
                        _userPassword = value;
                      },
                    ),
```

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.

```
                      FlatButton(
                        textColor: Theme.of(context).primaryColor,
                        child: Text(_isLogin
                            ? 'Create new account'
                            : 'I already have an account'),
                        onPressed: () {
                          setState(() {
                            _isLogin = !_isLogin;
                          });
                        },
                      ),
```

A 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 rejti el az adott elemeket.

```
void _trySubmit() {
  final isValid = _formKey.currentState.validate();
  FocusScope.of(context).unfocus();
  if (isValid) {
    _formKey.currentState.save();
    widget.submitFunction(
      _userEmail,
      _userPassword,
      _userBirthDay,
      _userGender,
      _userHeight,
      _userWeight,
      _isLogin,
    );
  }
}
```

Ha a felhasználó megérinti a bejelentkezés vagy regisztráció gombot, a kérdőív jelenlegi állapota mentésre kerül. 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.

```
    if (isLogin) {
      authResult = await _auth.signInWithEmailAndPassword(
        email: email,
        password: password,
      );
    } else {
      authResult = await _auth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );
    }
```

Továbbá létrehozásra kerülnek a 7. ábrán szemléltetett kollekciók. Kezdetben a felhasználó személyes adatai kitöltésre kerülnek a megadott adatokkal. A rizikó szám és a kontakt személy kollekcióba a mezők alapértelmezett értékkel jönnek létre. Ezt a felhasználó bármikor felül írhatja.

```
    FirebaseFirestore.instance
        .collection('users')
        .doc(_auth.currentUser.email
            .substring(0, _auth.currentUser.email.lastIndexOf('@')))
        .set({
      'birthday': DateFormat.yMd().format(birthday),
      'gender': gender,
      'height': height,
```

```
        'weight': weight,
      });
```

Először egy példányt kell létrehozni a Firebaseről, majd az építő programtervezési minta segítségével megkell 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 fő képernyőre navigál, ahol több lehetőség közül választhat a felhasználó. Két típusú menü lelhető fel. Egyik az alsó navigációs sáv, amely alapértelmezett képernyője a Home, másik lehetőség a Charts, amely a grafikonok megjelenítésére szolgál.

```
void initState() {
  _pages = [
    {
      'page': HomeScreen(),
      'title': 'Home',
    },
    {
      'page': ChartsScreen(),
      'title': 'Charts',
    },
  ];
  super.initState();
}
```

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

```
    appBar: AppBar(
      title: Text(_pages[_selectedPageIndex]['title']),
    ),
    drawer: MainDrawer(),
    body: _pages[_selectedPageIndex]['page'],
    bottomNavigationBar: BottomNavigationBar(
      onTap: _selectPage,
      backgroundColor: Theme.of(context).primaryColor,
      unselectedItemColor: Colors.white,
      selectedItemColor: Colors.black,
      currentIndex: _selectedPageIndex,
      items: [
        BottomNavigationBarItem(
          backgroundColor: Theme.of(context).primaryColor,
          icon: Icon(Icons.home),
          title: Text('Home'),
        ),
        BottomNavigationBarItem(
          backgroundColor: Theme.of(context).primaryColor,
          icon: Icon(Icons.bar_chart),
          title: Text('Charts'),
        ),
      ],
    ),
```

Az oldal címe ki választásra kerül az inicializált 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 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 lenyitásakor 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 érdekébe egy általános, többször felhasználható függvény van létrehozva a projektbe, így csak különböző paramétereket kell át adni.

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

Ez a függvény 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, amely alapján az útvonal választó navigátor hitelesen beazonosítja az adott képernyőt.

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

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,
7 _auth.currentUser.email.lastIndexOf('@')))
8                        .snapshots(),
9         builder: (context, snapshot) {
10          if (snapshot.connectionState == ConnectionState.waiting) {
11            return Center(
12              child: CircularProgressIndicator(),
13                          );
14                        }
15          final documents = snapshot.data;
16          return Column(
17            mainAxisSize: MainAxisSize.min,
18            children: <Widget>[
19              Text('Risk: ${documents['percentage']}',
20                style: TextStyle(
21                  fontSize: 28,
22                  color: Theme.of(context).primaryColor,
23                  ontWeight: FontWeight.bold,
24                ),
25              ),
        ...
```

Egy StreamBuilder widget kerül meghívásra. A stream paraméterénél egy konkrét Firebase snapshot letöltése történik. Hasonló képen működik, mint a regisztrációs rész. Megkell adni a kollekciót, a dokumentumot és hogy egy snapshotról van szó. A builder résznél amig a kapcsolat státusza várakozó egy körkörösen 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 érdekébe, hogy könnyen lehessen dolgozni a meglévő adatokkal. Mivel a Firebase JSON formátumba menti el az adatok, a kliens feladata az adatok beazonosítása és típusának megjelelőlése. A 15. sortól kezdődően az adat 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 tesztelés ként 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');
```

Mindkét értesítési mechanizmus egy plugin segítségével történik. A 16. sorba a telefonhíváshoz szükséges függvényhívás látható, a currentPhone változóba az adatbázisból lekért telefonszám található. A 15. sorba a fentebb definiált függvény hívása történik. Bizonyos adatokat megkell adni az SmsSender pluginnak 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, kijelentések. 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.

```
 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});
```

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

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