Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Urban Kocmut

Merjenje motoričnih znakov pri nevrodegenerativnih boleznih s pomočjo akcelerometrije

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Aleksander Sadikov
Somentor: doc. dr. Jure Žabkar

Ljubljana, 2018

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Kandidat naj zasnuje in implementira mobilno aplikacijo, ki z uporabo merilnika pospeška in žiroskopa meri motnje gibanja pri nevrodegenerativnih boleznih. Aplikacija mora omogočati prikaz, analizo in pošiljanje izmerjenih podatkov v strežniško aplikacijo za dolgoročno spremljanje bolnikov. Kandidat naj tudi primerja uporabnost pametnega telefona z drugimi napravami za merjenje gibanja.

Zahvaljujem se dr. Dejanu Georgievu za strokovno svetovanje in vsem drugim, ki so mi pomagali na poti do diplome.

Kazalo

Povzetek

Naslov: Merjenje motoričnih znakov pri nevrodegenerativnih boleznih s pomočjo akcelerometrije

Avtor: Urban Kocmut

Povzetek: Nevrologi trenutno merijo in štejejo tremorje pri obolelih osebah tako, da opazujejo osebo v različnih pozicijah. Opazujejo predvsem roke. Nimajo na voljo naprave, s katero bi lahko izmerili tresenje rok obolelih oseb, ki bi jim te meritve prikazala in naredila preprosto obdelavo podatkov za lažjo analizo. Narejena je bila mobilna aplikacija, ki z uporabo žiroskopa in senzorja pospeška izmeri, hrani in vizualizira podatke tremorjev obolelih oseb. Aplikacija omogoča nevrologom pridobiti natančne podatke o stanju obolele osebe, prav tako omogoča primerjavo s starimi meritvami in tako nudi vpogled v napredovanje bolezni. Aplikacija je preprosta za uporabo in dostopna osebam s slabim vidom in slabim nadzorom prstov.

Ključne besede: android, pametni telefon, parkinson, senzorji, merilnik pospeška, žiroskop.

Abstract

Title: Measuring motor signs of neurodegenerative diseases using accelerometery

Author: Urban Kocmut

Neurologists currently measure and count tremors of patients, by observing them in different positions. They usually observe the patients arms. They do not have a device which would allow them to measure tremors of patients, would visualize tremors and would process the measured data for easier analysis. The developed mobile application solves these problems. It uses a gyroscope and an accelerometer to measure tremors. The measured data is saved and visualised. The application allows neurologists to get accurate data about the state of their patients, compare measured data with previous measurements and so it also allows to track the progress of a patients disease. The application is simple to use and accessible to people with bad vision and bad finger control.

Keywords: android, smartphone, parkinson, sensors, accelerometer, gyroscope.

Poglavje 1

Uvod

Sodobni pametni telefoni imajo veliko število funkcionalnosti in uporablja jih velika večina ljudi. Že leta 2012 je 45% odraslih Američanov imelo pametni telefon. V starostni skupini 18 - 29 let kar 66%. [9] Pametni telefoni nam olajšajo številna opravila, zato so nepogrešljiv pripomoček v veliko panogah. Področje, na katerem pametni telefoni še niso razširjeni, je zdravstvo. Razlog nam ni znan. Smo se pa odločili, da naredimo mobilno aplikacijo na področju zdravstva, natančneje, mobilno aplikacijo za merjenje nevroloških simptomov. Razlog za to je, da trenutno nevrologi nimajo dobrega načina za pogosto in objektivno merjenje stanja pacientov s Parkinsonovo boleznijo. Naša aplikacija omogoča merjenje tremorjev, vizualizacijo izmerjenih podatkov in pošiljanje teh podatkov na zunanjo bazo.

Raziskava, pri kateri so uporabili aplikacijo na pametnem telefonu za pridobivanje podatkov obolelih, je že bila opravljena. Njen namen je bil razviti objektivno meritev stanja Parkinsonove bolezni in ugotoviti veljavnost meritve z merjenjem dnevnih nihanj sprememb bolezenskih znakov, preveriti korelacijo s standardnimi merami za Parkinsonovo bolezen in odziv na dopaminsko terapijo. V raziskavi so sodelovale osebe, ki imajo Parkinsonovo bolezen in osebe, ki je nimajo. Sodelujoči so morali opraviti pet testov (glas, tapkanje prstov, hoja, ravnotežje in odzivni čas). Iz pridobljenih podatkov so nato s pomočjo strojnega učenja pridobili oceno mPDS (mobile Parkinson

disease score) o napredovanju bolezni. mPDS objektivno ovrednoti rezultate posamičnih testov in je ocenjena z lestvico 0 - 100 (višja ocena pomeni večji napredek bolezni). Raziskava je trajala šest mesecev. Ocena mPDS je za dnevno nihanje bolezenskih znakov zaznala povprečno spremembo 13,9 na lestvici 0 - 100. Meritve korelirajo z Movement Disorder Society Unified Parkinson Disease's Rating Scale ($r = 0.81; P < .001$). Ocena mPDS se je pri odzivu na dopaminsko terapijo v povprečju izboljšala za 16,3. Raziskava je prišla do zaključka, da ocena mPDS komplementira standardnim meram za Parkinsonovo bolezen z dodajanjem pogostih in objektivnih meritev, ki lahko dopolnijo klinično oskrbo in ocenjevanje novih terapij.[1]
Parkinsonova bolezen (PB) je nevrodegenerativna bolezen, pri kateri propadajo nevroni substance nigre (pars compacta), ki proizvajajo živčni prenašalec dopamin. Z napredovanjem bolezni propadejo tudi številni drugi nevroni v možganih. Vzrok bolezni ni znan. [14] Za PB so najbolj značilne motnje gibanja. Poznamo jih več vrst:

- **Ataksija** je motena usklajenost mišičnih gibov.[4]
- **Distonija** je nevrološka motnja, pri kateri podaljšana nehotena mišična krčenja povzročijo nesmotrno zvijanje udov in telesa, ponavljanje gibov in nenormalne položaje telesa.[4]
- **Horea** je bolezen bazalnih ganglijev, ki se kaže s sunkovitimi nekoordiniranimi zgibki telesa in grimasiranjem.[4]
- **Mioklonus** so sunkoviti in asinhroni zgibki udov ali trupa zaradi krčenja ene mišice ali več mišic.[4]
- **Tremor** je nenamerno, ritmično, nihajoče gibanje dela telesa.[3] Večina tremorjev se pojavi v rokah, vendar lahko vplivajo tudi na ude, glavo, obraz, glas, trup. Vzrok za tremor je pogosto težava v globjih predelih možgan odgovornih za gibanje.[8] Tremorji se ločijo na tremorje, ki se pojavijo pri mirovanju in tremorje, ki se pojavijo pri gibanju.[3]

- **Tourettov sindrom** je nevropsihološka motnja znana po trzajih (nehoten, ponavljajoč se nagel gib ali zaporedje gibov v področju določene mišične skupine, ki jim tudi pacient ne vidi smisla).[4, 10]
- **Rigidnost** je zmanjšana in počasnejša gibljivost zaradi organskih sprememb. [4]
- **Bradikinezija** je upočasnjenost gibov.[4]

Te gibalne motnje močno spremenijo življenje obolelih. Zgodaj v poteku bolezni se spremeni pisava. Postane manjša (mikrografija). Tudi glas postane tišji in monoton. Bolniki težko opravljajo fina opravila, kot je npr. zapenjanje drobnih gumbov, šivanje. [14] Z napredovanjem bolezni lahko samostojno opravljajo vedno manj opravil, dokler ne postanejo popolnoma odvisni od tuje oskrbe.

Pri PB so pomembni redni pregledi, da se lahko preveri stanje bolezni in ustrezno prilagodi zdravljenje. Trenutno potekajo pregledi v ambulantah, zato nevrologi nimajo možnosti spremljanja stanja bolnikov izven teh rednih ambulantnih pregledov. Obstaja tudi težava objektivnega merjenja motenj gibanja. Nevrologi nimajo pripomočka s katerim bi lahko natančno izmerili bolezenske znake pri bolniku s PB. Narejena aplikacija ponuja rešitev obeh problemov. Omogoča natančno merjenje in analizo simptomov v ambulanti in izven nje ter tako pomaga specialistu pri ocenjevanju stanja bolnika.

Večina motenj gibanja se lahko izmeri z merilniki pospeška in žiroskopi, ki so prisotni na skoraj vsakem pametnem telefonu. Meritev se najlažje opravi tako, da pacient v roki drži telefon in opravlja različne gibe. Pri tem merilnik pospeška izmeri pospeške in žiroskop izmeri rotacijo, ki deluje na telefon. S pridobljenimi podatki se lahko že na preprostem grafu (vrednost senzorja skozi čas meritve) vizualno opazi anomalijo v gibanju.

Pametni telefoni imajo veliko možnosti za povezovanje z drugimi napravami, zato se lahko izmerjeni podatki pošljejo v nadaljno analizo na poljubno napravo. Mi smo se odločili za pošiljanje podatkov na podatkovno bazo. Možno je tudi priključiti dodatne senzorje na pametni telefon in tako izboljšati način

ali natančnost merjenja. Mi smo testirali dodatne senzorje predvsem z vidika uporabniške izkušnje, če uporabnik lažje drži drugo napravo namesto telefona.

Poleg testiranja uporabniške izkušnje držanja pametnega telefona v primerjavi z drugačnimi napravami, smo testirali uporabniško izkušnjo pri uporabi uporabniškega vmesnika. Ker je povprečna starost obolelih s PB 58-60 let [14], nam je bilo pomembno, da je uporabniški vmesnik preprost za uporabo ljudem s slabšim znanjem uporabe mobilnih aplikacij in ljudem s slabšim vidom.

Poglavje 2

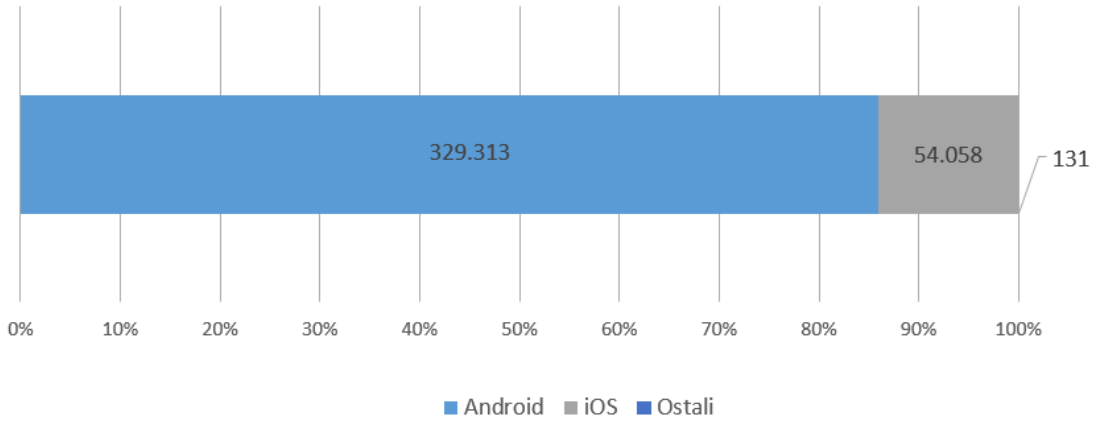
Mobilna aplikacija

Mobilna aplikacija se ukvarja z merjenje pospeška in rotacije. V tem poglavju predstavimo uporabljene tehnologije, opišemo, kako smo pristopili k razvoju aplikacije, predstavimo funkcionalnosti in uporabniški vmesnik mobilne aplikacije. Aplikacija je narejena za operacijski sistem Android 8.1. Na voljo je na GitHubu (https://github.com/UrbanKocmut/MeNS) v obliki Android Studio projekta. Za izgradnjo uporablja Gradle.

2.1 Uporabljene tehnologije

V tem podpoglavju predstavimo uporabljene tehnologije. Vsako tehnologijo na kratko opišemo in obrazložimo, zakaj smo se odločili za njeno uporabo. Napišemo tudi, kako smo tehnologijo uporabili in nove ugotovitve, ki smo jih pridobili med razvojem.

Android je s približno 86% tržnim deležem trenutno najpopularnejši mobilni operacijski sistem.[2] Ostali tržni deleži so razvidni na sliki 2.1. Poleg tega je zelo dostopen za razvoj aplikacij, saj je na voljo veliko literature za pomoč razvijalcem. Razvija se lahko na operacijskih sistemih maxOs, Windows in Linux v programskih jezikih Java in Kotlin ter knjižnice v C/C++. Zaradi teh lastnosti in predhodnih izkušenj z razvojem, smo se odločili, da



Slika 2.1: Tržni deleži mobilnih operacijskih sistemov. Številka na grafu predstavlja tisoč prodanih naprav v 1. četrtletju 2018. Kategorija OSTALO ni vidna zaradi malega deleža prodanih naprav (0,0%)

razvijemo aplikacijo za Android.
Odločitev za uporabo operacijskega sistema Android se nam je izplačala, saj smo med razvojem aplikacije vedno našli ustrezno literaturo in z uporabo programskega jezika Java smo lahko uporabili naše izkušnje z razvojem v tem programskem jeziku.

GraphView je knjižnica za Android, ki omogoča ustvarjanje grafov. Podpira več vrst grafov (stolpične, črtne, točkovne, ...) in izrisovanje grafov v realnem času. Ima tudi dobro dokumentacijo. Glavna razloga za uporabo te knjižnice sta dobra dokumentacija in podpora za izrisovanje grafov v realnem času. [6]
Za naše potrebe smo uporabili 4 strukture iz knjižnice GraphView. Za predstavitev grafa smo uporabili strukturo GraphView. Struktura predstavlja element na uporabniškem vmesniku. Temu elementu se dodajo podatki, ki jih želimo prikazati. Podatki se dodajo preko strukture LineGraphSeries. En GraphView ima lahko več LineGraphSeries, tako je prikazanih več skupin podatkov. LineGraphSeries je sestavljen iz struktur DataPoint. Te strukture

hranijo vrednost X in Y ter predstavljajo točko na grafu. Nadzor nad prikazom podatkov na grafu imamo preko strukture ViewPort. Ta struktura nam omogoča, da določimo maksimalne in minimalne prikazane vrednosti. Preko nje tudi omogočimo povečavo in premikanje po grafu preko zaslona na dotik. Predstavlja viden del grafa.
Med razvojem aplikacije smo odkrili omejitev knjižnice GraphView. Nismo našli načina, kako avtomatsko skrčiti graf, da lahko prikažemo vse podatke naenkrat. Zaradi te pomanjkljivosti je potrebno ročno, z uporabo zaslona na dotik, zmanjšati povečavo grafa. Ob ponovnem izbiranju bi najverjetneje izbrali drugo knjižnico.

SQLite je javno dostopna knjižnica, ki implementira SQL (structured query language) podatkovno bazo znotraj procesa. Baza je samozadostna, ne potrebuje strežnika in dodatne konfiguracije. Baza je v celoti shranjena v eni datoteki. Datoteka je prenosljiva med različnimi sistemi (32 in 64 bitni) in arhitekturami (veliki in mali endian). Zaradi teh lastnosti je popularna za shranjevanje podatkov aplikacij. Za uporabo SQLite v aplikaciji smo se odločili, ker se knjižnica že nahaja v Android SDK.[13]
V aplikaciji smo uporabili bazo SQLite samo preko knjižnice Room, ker je zadostovala vsem potrebam. Vendar lahko, če se v prihodnje pojavijo nove zahteve, dostopamo direktno do SQLite baze.

Room Persistence Library je knjižnica, ki ustvari abstraktni sloj čez knjižnico SQLite. Poenostavi ustvarjanje in dostopanje SQLite baze. Room omogoči, da razvijalec definira objekte in preko njih dostopa do baze. S tem se izogne morebitnim napakam pri pisanju kode za komunikacijo z bazo in si skrajša čas razvoja aplikacije. Za Room smo se odločili, ker smo si z njeno uporabo poenostavili shranjevanje izmerjenih podatkov.[11]
Med razvojem smo opazili dodatno prednost knjižnice Room. Če smo spremenili strukturo baze, nas je, zaradi povezanosti objektov in baze, razvojno okolje opozorilo na napako na vseh mestih, kjer komuniciramo s spremenjenimi objekti v bazi. Tako smo se izognili iskanju napak kasneje v razvoju.

SharedPreferences je vmesnik vključen v Android SDK (software development kit), ki omogoča shranjevanje parov ime in podatek. Shranjuje se lahko vse osnovne podatkovne tipe Jave in tip *String*. Glavna lastnost *SharedPreferences* je, da se podatki ohranijo po zaprtju aplikacije.
V aplikaciji smo *SharedPreferences* uporabljali za shranjevanje nastavitev povezave z zunanjo bazo in za shranjevanje identifikacijskih številk med prehajanjem med meniji, ker je njegova uporaba preprosta in ne zahteva veliko dodatne kode.
Za našo uporabo se je *SharedPreferences* izkazal za primernega in z njim nismo imeli težav.

2.2 Opis razvoja

Na začetku razvoja smo zožali možne načine izdelave aplikacije na 2 načina. Izdelava s programskim jezikom QML (Qt Modeling Language) ali z Javo in Android SDK. QML je deklerativni programski jezik, ki omogoča hitri razvoj aplikacij. Njegove prednosti so predvsem hitrost in preprostost razvoja ter prenosljivost med operacijskimi sistemi. Pomanjkljivosti pa so slabša podpora in težja uporaba nekaterih funkcij operacijskega sistema. Prednosti Android SDK so zmogljivost, število knjižnic in literatura ter podpora vseh funkcij operacijskega sistema. Njegova slabost je večja količina in kompleksnost napisane kode. Odločili smo se za razvoj v Android SDK, ker že imamo predhodnje izkušnje z Android SDK in smo tako lahko hitreje razvijali aplikacijo ter lažje rešili nepredvidene probleme.

Za razvojno okolje smo uporabili IDE (integrated development environment) Android Studio, ker nam omogoča preprost način zagona aplikacije na mobilnem telefonu ali na emulatorju, predloge za projekt, povezavo s sistemom za avtomatsko izgradnjo Gradle, ki poenostavi dodajanje knjižnic in izgradnjo aplikacije. Android Studio prav tako omogoča povezavo s sistemom za upravljanje verzij Git, ki smo ga uporabili pri razvoju tega projekta. Pro-

jekt gostuje na GitHub. Uporaba sistema za upravljanje verzij nam nudi pregled nad vsemi spremembami, varnostno kopijo in preprost način prenašanja projekta med računalniki.

2.2.1 Metoda razvoja

V tem podpoglavju opišemo pristop, ki smo ga uporabili pri izdelavi aplikacije. Predstavimo tudi druge metode razvoja, ki smo jih preučili pred začetkom razvoja aplikacije.
Obstaja več metod razvoja programske opreme. Poznamo Waterfallov model, Spiralni model in Agilne metode. Zaradi hitro spreminjajočih zahtev, majhnosti razvojne ekipe in stroge časovne omejitve smo se odločili, da ne sledimo nobenemu modelu in pri razvoju uporabimo samo pristope, kot so *Continuous integration*, *Prototyping* in *Incremental development*.

Continuous integration je pristop k razvoju programske opreme, pri katerem razvijalci pogosto združujejo svojo kodo projekta z glavno kodo projekta. Pri tem se napake hitro odkrijejo in se lahko tudi hitro popravijo. Na ta način se izognemo veliki količini napak, ki so posledica nezdružljivosti kode razvijalcev z glavno kodo projekta. Pri tem projektu smo implementirali ta pristop tako, da smo vsaj enkrat na dan združili novo napisano kodo z glavno kodo projekta. Pri tem smo uporabili GitHub, kjer smo gostili glavno kodo projekta in Git, ki smo ga uporabili za združevanje razvite kode z glavno kodo. Ta pristop nam je tudi zmanjšal tveganje za prekoračitev roka izdelave projekta v primeru tehnične okvare strojne opreme ali razvojnega okolja, saj smo imeli manj kot en dan staro verzijo projekta na GitHubu. Potrebno je opozoriti, da od pristopa *Continuous integration* nismo uporabili testiranja *UnitTest*, zaradi časovnega stroška implementacije tega tipa testiranja. [5]

Prototyping je pristop k razvoju programske opreme, pri katerem se izdela nedokončano programsko opremo z minimalno funkcionalnostjo. Ti prototipi omogočajo testiranje in demonstracijo posameznih funkcij programa. Na ta način lahko stranka poda svoje mnenje in predloge za spremembe ter izboljšave določene funkcije programske opreme, še preden je ta dokončana. Za pristop prototipiranja smo se odločili z namenom, da lahko hitro predstavimo nove funkcionalnosti in dobimo povratno informacijo o njih.[12] Poznamo več vrst prototipov:

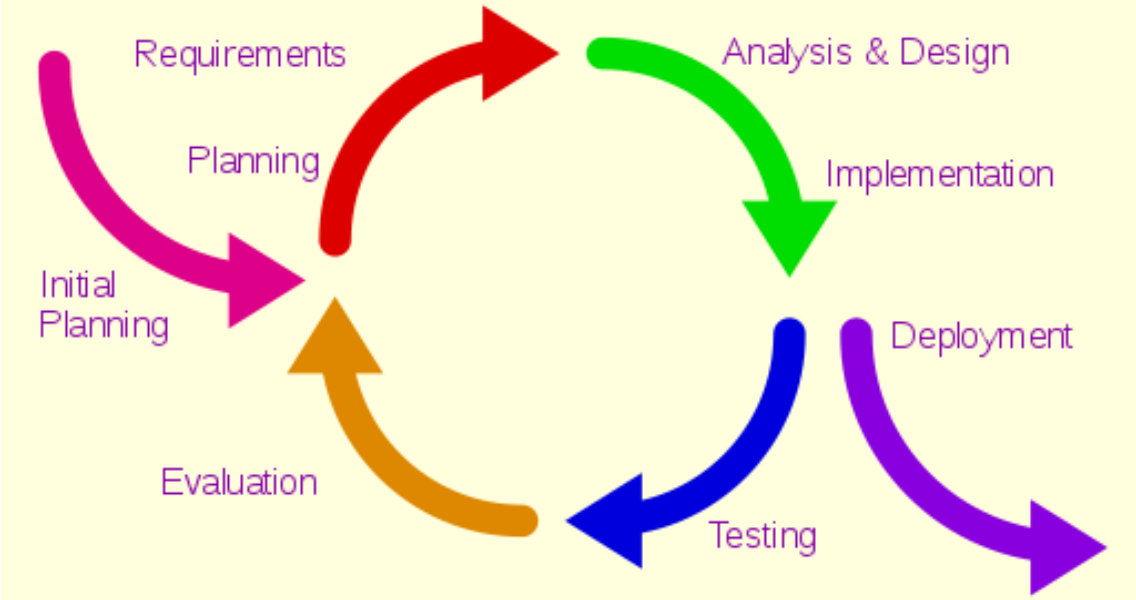
- ***Throwaway prototyping:*** Pri tej vrsti prototipe zavržemo, ko jih naredimo. Njihov namen je hiter preizkus določene funkcije programske opreme. Pomagajo nam določiti končne funkcije programske opreme.

- ***Evolutionary prototyping:*** Pri tej vrsti prototipe nadgrajujemo z novimi funkcijami in jih ne zavržemo. To nam omogoča, da z razvojem programske opreme dodajamo nove zahteve, na katere pri začetku razvoja nismo pomislili.

- ***Incremental prototyping:*** Tukaj izdelamo več prototipov, ki jih na koncu združimo.

Pri tem projektu se je uporabilo pristop evolucijskega prototipiranja, zaradi možnosti dodajanja nepredvidenih zahtev.

Incremental and iterative development je pristop k razvoju programske opreme, ki temelji na iteriranju skozi stopnjo načrtovanja, stopnjo razvoja in stopnjo ocenjevanja. Iteriranje se ponavlja, dokler projekt ni končan. Pristop omogoča učenje iz napak prejšnjih iteracij in dodajanje novih funkcionalnosti v odziv prejšnji iteraciji. Večkratno ocenjevanje funkcionalnosti med razvojem omogoča sprotno odpravljanje napak. Ta pristop je bil izbran, ker omogoča večkratno ocenjevanje in spreminjanje funkcionalnosti.[7] Primer na sliki 2.2



Slika 2.2: Diagram procesa iterativnega razvoja.

Uporabljena je bila mešanica vseh pristopov. Iz vsakega pristopa so bile uporabljene najprimernejše in najbolj praktične lastnosti. Od *Continuous integration* je bilo uporabljeno sprotno združevanje kode, od *Prototyping* je bilo uporabljeno postopno dodajanje funkcionalnosti in od *Incremental development* je bilo uporabljeno iteriranje med načrtovanjem, razvojem in ocenjevanjem. Za kombinacijo pristopov smo se odločili zaradi mnenja, da bi strogo sledenje določenega postopka upočasnjevalo razvoj aplikacije.

2.3 Funkcionalnosti

V tem podpoglavju opišemo funkcionalnosti, ki jih ima razvita aplikacija. Za vsako aplikacijo predstavimo kako deluje, kako smo jo implementirali in čemu služi v aplikaciji. Opišemo tudi naše ugotovitve od razvoja funkcionalnosti.

Merjenje ja najpomembnejša fukcionalnost naše aplikacije. Omogoča nam pridobivanje podatkov s senzorjev na mobilni napravi. Pri naši aplikaciji smo

se odločili samo za uporabo merilnika pospeška in žiroskopa, vendar se lahko funkcionalnost merjenja razširi na dodatne senzorje. Za uporabo teh dveh senzorjev smo se odločili, ker smo prepričani, da nam podata dovolj potrebnih podatkov za analizo tremorjev. Ostale senzorje (senzor magnetnega polja, senzor svetlosti, barometer, ...) nismo uporabili, ker se nam po našem mnenju časovni vložek implementacije ne bi povrnil. Pridobljeni podatki ne bi bistveno pripomogli k izboljšavi naše analize tremorjev.
Merjenje smo implementirali tako, da smo ustvarili nov razred, ki razširi vmesnik *SensorEventListener*. V tem razredu smo ustvarili metodo *onSensorChanged*, ki za parameter sprejme objekt tipa *SensorEvent*. Ta objekt nam pove tip senzorja in izmerjene vrednosti dogodka. V naši implementaciji smo uporabili en razred za zajemanje podatkov od obeh senzorjev, ker smo na ta način prihranili čas in zmanjšali število vrstic kode. Posledica naše odločitve je to, da moramo v metodi *onSensorChanged* napisati kodo za shranjevanje vsakega senzorja posebej, prav tako moramo z vejitvenim stavkom preveriti tip senzorja izmerjenih vrednosti. Med izvajanjem merjenja se podatki zapisujejo v seznam tipa *ArrayDeque*. Po končani meritvi se izmerjeni podatki zapišejo v SQLite bazo na napravi. Za takšen način shranjevanja meritev smo se odločili, ker pisanje v bazo vzame veliko časa v primerjavi s shranjevanjem v seznam. To nam je bilo pomembno, saj smo želeli, da proces merjenja poteka nemoteno in čimbolj učinkovito.
Naša implementacija se je izkazala za učinkovito, vendar smo imeli težave z napakami zaradi ponavljajoče kode. Boljša implementacija bi bila, da bi ustvarili razred, ki bi se mu podal tip senzorja in bi meril podatke samo tega tipa senzorja.

Žiroskop: Aplikacija meri rotacijo z zajemanjem podatkov žiroskopa mobilne naprave. Pri tem uporablja knjižnice Sensor, SensorEvent, SensorEventListener in SensorManager. Od žiroskopa pri vsaki meritvi pridobi 3 vrednosti. Rotacije v radianih na sekundo za osi X, Y in Z. Izmerjeni podatki so tipa *float*. Pri tipu *float* se lahko pojavi napaka pri zaokroževanju, vendar za

naše potrebe ne potrebujemo takšne natančnosti, da bi nam zaokrožitvena napaka predstavljala problem. Zakasnitve med meritvami so nastavljene na minimalno vrednost. Natančna vrednost zakasnitve je odvisna od operacijskega sistema in strojne opreme mobilne naprabe. Na naši testni napravi meri žiroskop s frekvenco 200Hz.
Rotacija je zelo koristen podatek v stuaciji, ko je pacientova roka sproščenem položaju (roka počiva na mizi), vendar se vseeno trese tako, da se telefon vrti okoli določene osi. V takšni situaciji telefon z vidika merilnika pospeška miruje, ker se senzor pospeška ne premika. Pri testiranju, kjer pacient opravlja določene gibe, nam podatki z žiroskopa pokažejo zelo drugačen prikaz gibanja v primerjavi z merilnikom pospeška. Težave pri gibih, kjer je prisotna rotacija, so veliko bolj razvidne.

Merilnik pospeška: Tako kot pri žiroskopu, uporablja aplikacija za merjenje pospeška senzor na napravi. Pri tem uporablja iste knjižnice kot žiroskop. Merilnik pospeška pri vsaki meritvi pridobi tri meritve. Pospešek v m/s^2 v smereh X, Y in Z. Enako kot pri žiroskopu so podatki pridobljeni v tipu *float*. Zakasnitve med meritvami so nastavljene na minimalno vrednost. Natančna vrednost zakasnitve je odvisna od operacijskega sistema in strojne opreme mobilne naprave. Na naši testni napravi meri merilnik pospeška s frekvenco 100Hz.
Merilnik pospeška zelo dobro izmeri tremorje roke, ko pacient drži telefon v roki. Pri vizualizaciji izmerjenih podatkov so tresljaji hitro vidni. Merilnik pospeška v večini primerov bolje izmeri tremorje pri pacientu kot žiroskop, razen v primeru, ko je roka naslonjena na mizi.

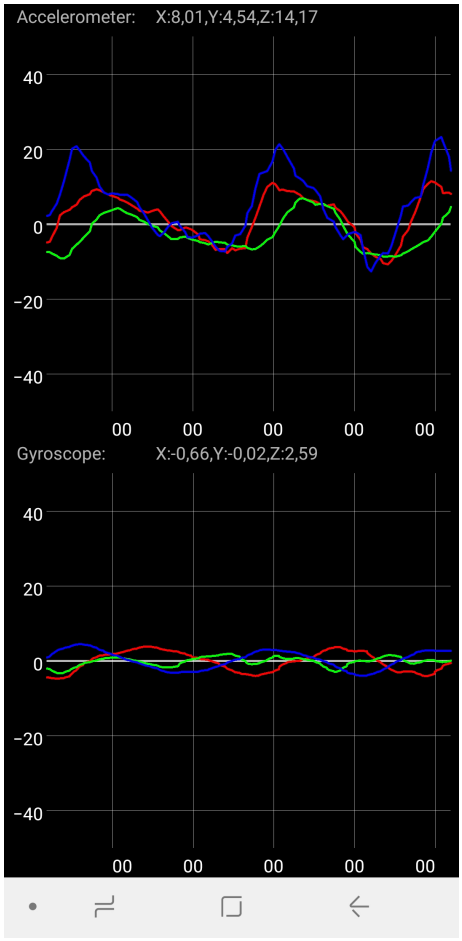
Vizualizacija meritve: Med izvajanjem meritve se izmerjeni podatki v realnem času prikazujejo na dveh grafih. Na abscisni osi je čas, na ordinatni osi pa vrednost meritve. Zgornji graf prikazuje meritve merilnika pospeška, spodnji meritve žiroskopa. Na vsakem grafu so barvno ločeno (rdeča, modra in zelena) prikazane meritve za vse tri osi, ki jih senzor meri.
Med razvojem vizualizacije meritve smo prišli do težave, ker se je skala av-

tomatsko prilagajala trenutno prikazanim podatkom na grafu. Problem smo rešili tako, da smo nastavili skalo grafa na maksimalno vrednost 40 in minimalno vrednost -40. Tako je prikaz vrednosti na grafu jasen in uporabnik se ne zmede zaradi spreminjanja skale. Enako nastavitev skale smo uporabili za oba grafa, saj menimo, da tako uporabnik lažje primerja podatke obeh grafov. Potrebno pa je opozoriti, da bi graf žiroskopa lahko imel manjšo skalo, saj amplituda izmerjenih vrednosti ni tako visoka, kot pri merilniku pospeška.

Podoben problem smo imeli s skalo abscisne osi grafa. Na njej so bile vrednosti prevelike in so se med seboj prekrivale. To smo rešili tako, da na osi prikazujemo samo dve števki. Natančne vrednosti sicer ni možno razbrati, vendar to tudi prej, zaradi prekrivanja, ni bilo možno. Rešitev ni optimalna, vendar reši problem nepregledne abscisne osi.

Poleg grafov se med merjenjem prikazujejo tudi izmerjene vrednosti. Vrednosti za merilnik pospeška so prikazane nad grafom merilnika pospeška za vsako os posebej. Enako kot podatki za merilnik pospeška, so prikazani podatki žiroskopa nad grafom žiroskopa. Prikazane vrednosti imajo omejeno natančnost na dve decimalni mesti, saj menimo, da je to zadostna natančnost za sprotno spremljanje meritve.

Vizualizacija meritve je razvidna na sliki 2.3.



Slika 2.3: Posnetek zaslona aplikacije med merjenjem. Razvidna sta grafa merilnika pospeška in žiroskopa.

Različni tipi meritev: Aplikacija loči med šestimi tipi merjenja, po tri za vsako roko. V mirovanju, z iztegnjeno roko in v gibanju. Vsak tip meritve traja 20s. Tip meritve se izbere v meniju za izbiro tipa meritve. Te tipe meritev smo izbrali po posvetu z nevrologom, ki nam je opisal trenutni postopek izvajanja testov.
Merjenje v mirovanju poteka tako, da pacient prime v roko telefon in začne merjenje. Med merjenjem pacient ne premika roke v kateri ima telefon in jo poskuša imeti v sproščenem položaju.

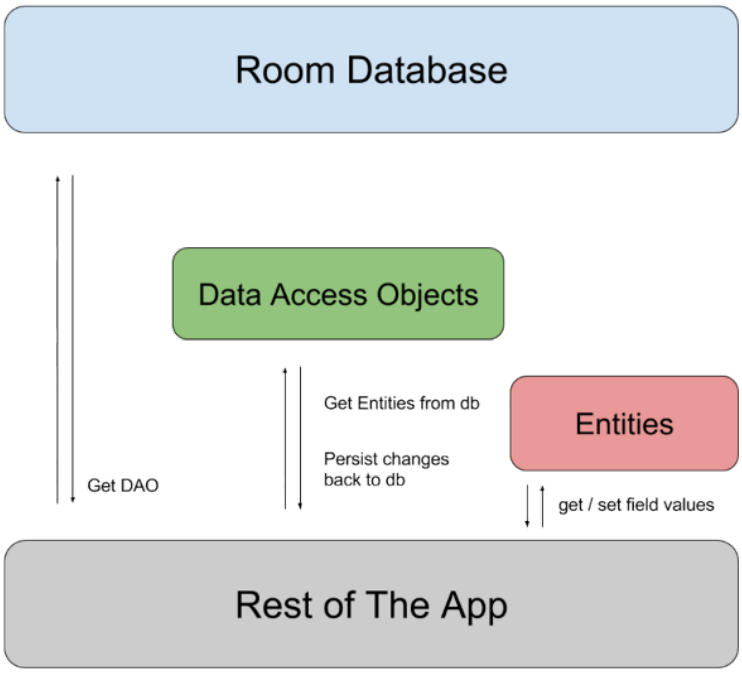
Merjenje z iztegnjeno roko se začne enako kot merjenje v mirovanju. Med merjenjem pacient iztegne roko v kateri drži telefon in jo poskuša čimbolj mirno držati v tem položaju do konca meritve.
Merjenje v gibanju se začne enako ko prejšnja merjenja. Med merjenjem se pacient z roko, v kateri drži telefon, premika iz stanja, ko ima roko iztegnjeno pred seboj, v stanje, ko se njegova roka dotika njegovega nosu, in nazaj v začetno stanje. Pacient do konca merjenja ponavlja ta gib.
Pri vseh tipih meritev je pomembno, da pacient s telesom čim bolj miruje in ne izvaja nepotrebnih gibov, ker so senzorji na mobilni napravi dovolj občutljivi, da zaznajo tudi majhne gibe telesa.

Hranjenje nastavitev: Aplikacija omogoča hranjenje nastavitev povezave baze z uporabo vmesnika SharedPreferences. Nastavitve se vnesejo v meniju za nastavitve. Shranjene nastavitve ob ponovnem zagonu ni potrebno ponovno vnašati, saj ostanejo shranjene med zagoni. Nastavitve se lahko vnaša ali spreminja samo, ko uporabnik ob zagonu aplikacije vpiše posebno številko. Takrat se v glavnem meniju prikaže dostop do menija za nastavitve.
Zaradi časovne omejitve nam ni uspelo preveriti varnosti implementacije hranjenja nastavitev. Obstaja možnost, da nepooblaščena oseba dostopa do nastavitev.

Pošiljanje podatkov v zunanjo bazo: Aplikacija omogoča pošiljanje podatkov v zunanjo bazo. To poteka preko direktne povezave na zunanjo bazo. Za povezavo se uporablja knjižnica Sql, ki je del Jave. Pred pošiljanjem podatkov v zunanjo bazo, se iz shranjenih nastavitev preberejo IP (internet protocol) naslov in vrata baze, ime sheme, uporabniško ime ter geslo. Iz teh podatkov se sestavi naslov za povezavo na zunanjo bazo. Ta naslov se poda knjižnici Sql, ki ustvari povezavo. Ko je povezava ustvarjena, se začne pošiljanje podatkov v skupinah stavkov SQL. Pošiljanje poteka tako, da se podatki preberejo iz tabele na lokalni SQLite bazi in se vnesejo v skupino stavkov SQL. Potem se skupina stavkov pošlje na bazo in enak postopek se ponovi za drugo tabelo. Pošiljajo se samo podatki iz dveh tabel (measure-

ments in readings), saj se predvideva, da ostale tabele na zunanji bazi že vsebujejo vse potrebne podatke. Po končanem vstavljanju v zunanjo bazo, se povezava zapre.

Naša implementacija pošiljanja podatkov v zunanjo bazo ni optimalna, saj je potrebna direktna povezava na zunanjo bazo. Direktna povezava potrebuje zanesljivo povezavo z omrežjem, ki pa ni vedno na voljo na mobilni napravi. Boljša rešitev bi bila uporaba *Representational State Transfer* (REST). To bi odpravilo pomanjkljivosti direktnega dostopa do baze, vendar bi bila implementacija zahtevnejša. Zaradi zahtevnejše implementacije in časovnih omejitev razvoja, smo se vseeno odločili za uporabo direktne povezave z zunanjo bazo.



Slika 2.4: Diagram knjižnice Room

Shranjevanje podatkov v bazi: Izmerjeni podatki se vpišejo v SQLite bazo preko knjižnice Room. Za upravljanje baze se uporabljajo posebni javanski razredi. Ti razredi definirajo strukture v bazi preko anotacij. Anota-

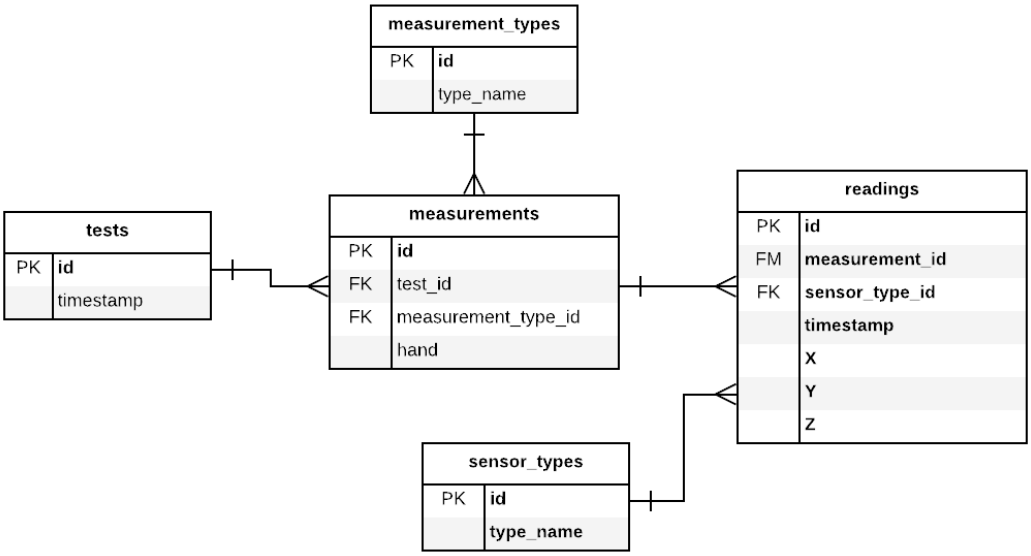
cije definirajo tip objekta v bazi, ki ga javanski razred predstavlja in njegove lastnosti. Uporabljeni so bili 3 tipi razredov. *Database*, *DAO* (data access object) in *Entity*. Struktura knjižnice Room je predstavljena na sliki 2.4.
Database razred definira, katere entitete ima baza, hrani instanco baze in z bazo poveže razred *DAO*.
Razred *DAO* definira vse možne transakcije z bazo in je v osnovi tipa *Interface*. Pri operacijah brisanja in vstavljanja se definira samo parameter metode in tip, ki ga metoda vrne. Pri poizvedbah se v anotacijo zapiše še stavek SQL, ki se uporabi za poizvedbo.
Zadnji uporabljeni tip razreda je tip *Entity*. Ta tip definira entiteto v bazi. Poleg tipa hranjenih podatkov so tu definirane metode, ki vrnejo privzete vrednosti potrebne pri postavitvi baze. Diagram podatkovne baze je predstavljen na sliki 2.5.
Lokalno shranjevanje podatkov nam omogoča, da aplikacija ni odvisna od konstantne povezave z zunanjo bazo. Uporaba podatkovne baze za lokalno shranjevanje podatkov nam: zmanjša možnosti za napake v shranjenih podatkih v primerjavi s shranjevanjem podatkov direktno v datoteko, omogoči uporabo jezika SQL za dostopanje do shranjenih podatkov in poenostavi pošiljanje podatkov v zunanjo bazo, saj sta si strukturi zunanje in lokalne baze zelo podobni.
Spodnja izseka prikazujeta uporabo anotacij za definiranje strukture baze in operacij z njo.

Izsek iz razreda tipa *Entity*

```
@Entity(tableName = "measurements",
        foreignKeys = {@ForeignKey(entity = Test.class,
                                   parentColumns = "id",
                                   childColumns = "test_id",
                                   onDelete = ForeignKey.CASCADE),
                @ForeignKey(entity = MeasurementType.class,
                            parentColumns = "id",
                            childColumns = "measurement_type_id")})
public class Measurement {

    @PrimaryKey(autoGenerate = true)
    @ColumnInfo(name = "id")
    public int id;

}
```

Izsek iz razreda tipa *DAO*

```
   @Insert(onConflict = OnConflictStrategy.REPLACE)
    public void insertReadings(Reading... readings);

    @Delete
    public void deleteMeasurements(Collection<Measurement>
        measurements);

    @Query("SELECT * FROM tests")
    public Test[] loadAllTests();
```



Slika 2.5: Diagram podatkovne baze.

Analiza podatkov: Aplikacija poleg prikaza zajetih podatkov, te podatke še analizira. Naredi dve analizi: analizo frekvenc in analizo spremembe povprečne amplitude v odsekih 100ms. Analizi se izvedeta neodvisno za vsako izmerjeno os vsakega senzorja.

Pri analizi frekvence se naredi Fourierova transformacija zajetih podatkov. Nato se transformirane podatke kvadrira. Analiza prikaže amplitude nihanja po frekvencah.

Amplituda se izračuna iz vsote absolutne vrednosti meritev v 100ms. Izračuna se vse odseke 100ms od začetka do konca merjenja. Amplituda se za vsak odsek izračuna po naslednji formuli:

$$\frac{s}{n}$$

Pri čemer je n število meritev v odseku in s vsota absolutne vrednosti meritev v odseku.

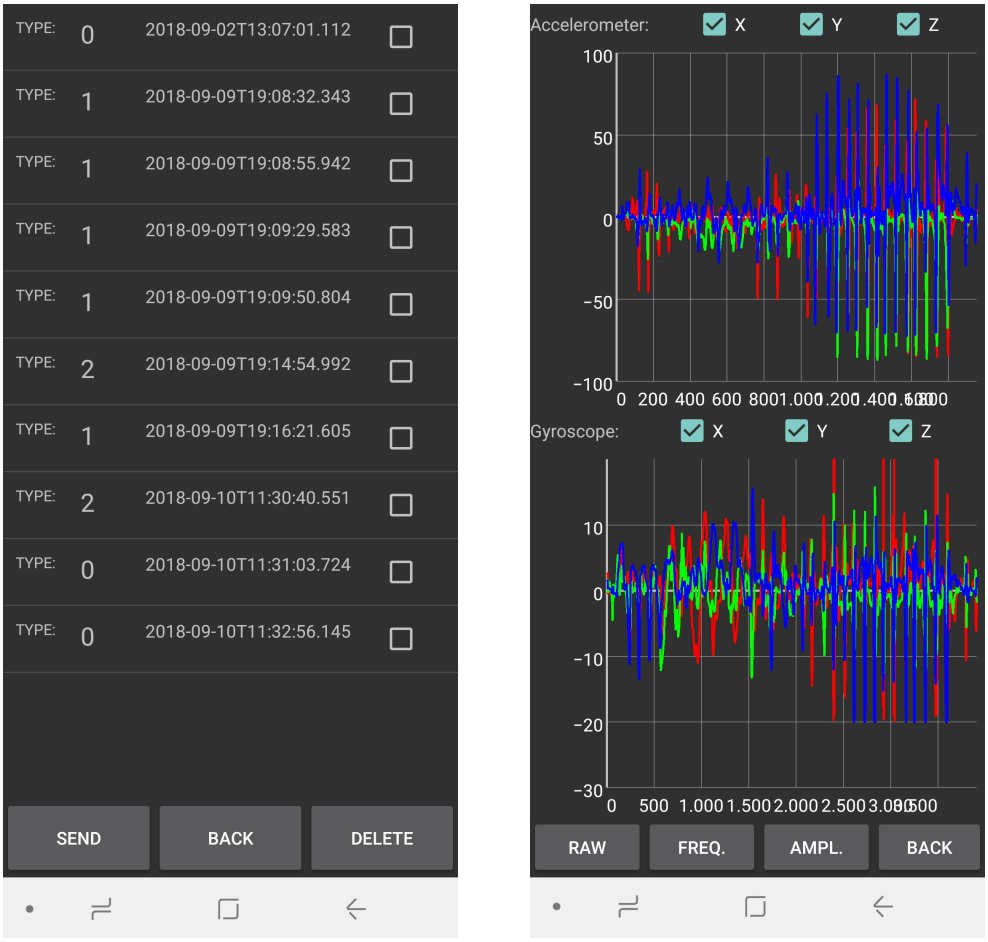
Lokalna analiza podatkov omogoča, da uporabnik dobi dodatne podatke o meritvah brez pošiljanja meritev v zunanjo bazo.

2.4 Uporabniški vmesnik

V tem podpoglavju je opisan uporabniški vmesnik mobilne aplikacije in postopek njegove predvidene uporabe. Poleg tega so predstavljene zahteve, ki smo si jih zadali pri izdelavi uporabniškega vmesnika.

Najpomembnejša lastnost uporabniškega vmesnika je preprosta uporaba. Uporabniški vmesnik je zasnovan z velikimi gumbi, velikim tekstom in preprosto navigacijo med meniji. Tekst (oznake gumbov in podatkov) v uporabniškem vmesniku je shranjen v samostojni datoteki. To omogoča preprosto prevajanje v druge jezike.
Po zagonu aplikacije se uporabniku prikaže meni za vnos osebne identifikacijske številke. Vnos identifikacijske številke je potreben, da lahko uporabnik dostopa samo do svojih meritev.
Po uspešnem vnosu identifikacijske številke se prikaže glavni meni. Ta meni je osrednja točka aplikacije. S tega menija lahko uporabnik izbere pregled že opravljenih meritev ali zajemanje novih meritev.
Meni za izbiro zajemanja novih meritev ima gumbe za vseh šest možnih tipov meritev in gumb za vrnitev na glavni meni. Ob pritisku na enega od gumbov za meritev, se prične nova meritev.
Prikaže se zaslon za merjenje (prikazan na sliki 2.3), na katerem se v realnem času prikazujeta graf merilnika pospeška in graf žiroskopa. Po končanem merjenju se ponovno prikaže meni za izbiro tipa meritve.
Drugi meni dostopen iz glavnega menija, je meni za prikaz obstoječih meritev (prikazan na sliki 2.6). S tega menija lahko uporabnik izbriše, pošlje in pogleda shranjene meritve. Izbira za brisanje in pošiljanje poteka preko elementa *CheckBox*, operacija nad izbranimi elementi se izvede s pritiskom na ustrezen gumb (*SEND* ali *DELETE*). Za prikaz vizualizacije meritve je

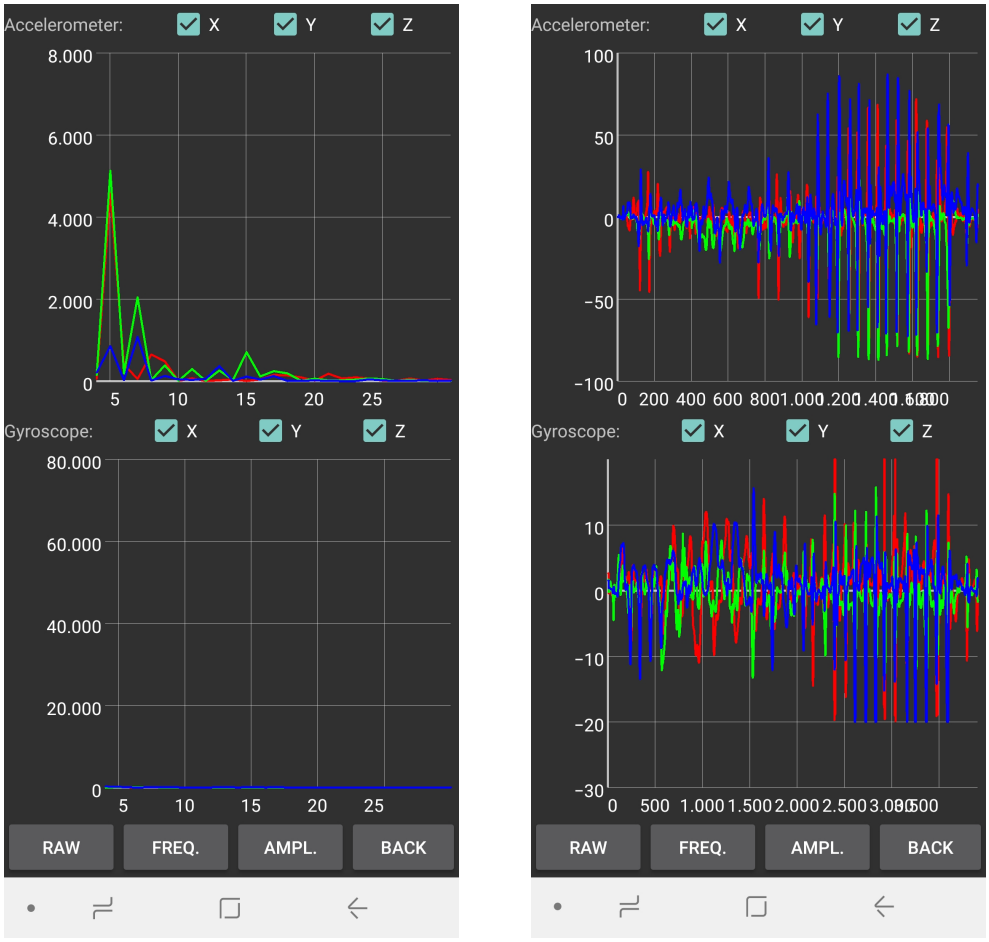
potreben pritisk na vrstico meritve. V vrstici je napisan tip meritve in čas zajema meritve. V vsaki vrstici je element *CheckBox*, ki se uporablja za izbris in pošiljanje.
Zatem se prikaže zaslon za prikaz vizualizacije (slika 2.6). Na tem zaslonu so tri možne vizualizacije: neobdelani podatki, analiza frekvence (razvidno na sliki 2.7) in analiza amplitude (razvidno na sliki 2.7). Prikaz posamezne vizualizacije se sproži s pritiskom na ustrezen gumb ob spodnjem robu zaslona.
Začetni prikaz je vizualizacija neobdelanih podatkov. Možna je tudi izbira prikazanih osi, ki pa, zaradi pomankljivosti izbrane knjižnice GraphView, ne deluje optimalno. Os X mora biti vedno izbrana, medtem ko os Y in os Z nista obvezni. Po izbiri prikazanih osi je potreben ponovni pritisk na gumb za izbiro vizualizacije. Prikazana grafa podpirata funkcijo *Pinch to zoom* in premikanje prikazanih podatkov z drsanjem prsta po grafu. Vizualizacija analize frekvence ne podpira premikanja vidnega polje, saj nas zanima samo določeno območje frekvenc.
Pri vizualizaciji neobdelanih podatkov in vizualizaciji analize amplitude je na x-osi čas in na y-osi amplituda. Pri vizalizaciji analize frekvence je na x-osi frekvenca in na y-osi amplitude nihanja.
Aplikacija ima tudi skriti del uporabniškega vmesnika. To je meni za nastavitev povezave z zunanjo bazo. V njem se vnese IP, omrežna vrata, ime sheme, uporabniško ime in geslo. Podatki morajo biti vnešeni v celoti. Uporabniku se prikaže obvestilo, če poskuša shraniti nepopolne podatke. Podatke je potrebno nastaviti preden se poskuša pošiljati meritve v zunanjo bazo. Gumb za ta meni se prikaže v glavnem meniju, če uporabnik vnese **2390** na začetnem zaslonu. Možno je, da se uporabniku nehote prikaže meni za nastavitve, vendar je verjetnost majhna.



(a) Posnetek zaslona menija za izbiro shranjenih meritev.

(b) Posnetek zaslona vizualizacije neobdelanih podatkov meritve.

Slika 2.6: Meni za izbiro meritve in vizualizacija neobdelane meritve.



(a) Posnetek zaslona vizualizacije spremembe frekvence meritve.

(b) Posnetek zaslona vizualizacije spremembe amplitude meritve.

Slika 2.7: Primer vizualizacije frekvence in amplitude

Poglavje 3

Testiranje

V tem poglavju predstavimo postopek testiranja aplikacije. Opišemo populacijo uporabnikov, na katerih smo izvajali testiranje, postopek testiranja in odzive uporabnikov. Opišemo spremembe, ki smo jih implementirali glede na odziv uporabnikov. Na koncu še opišemo postopek testiranja funkcionalnosti.

3.1 Odziv uporabnikov

Uporabniški vmesnik je bil testiran že med razvojem aplikacije. Dodatno testiranje je bilo opravljeno pred zaključkom razvoja. Testiranje je bilo opravljeno na desetih uporabnikih različnih starosti, z različno spretnostjo uporabe mobilnih naprav (spretni in nespretni) in z različno kakovostjo vida (dober in slab). Štirje uporabniki so bili spretni z mobilnimi napravami in so imeli dober vid. Trije uporabniki so bili spretni z mobilnimi napravami in so imeli slab vid. Trije uporabniki so bili nespretni z mobilnimi napravami in imeli slab vid. Razporeditev uporabnikov je prikazana na spodnji tabeli.

Spretnost z mobilno napravo \vid	**dober**	**slab**
dobra	4	3
slaba	0	3

Zaradi časovne omejitve nam ni uspelo testirati aplikacijo na uporabniku z dobrim vidom in slabo spretnostjo uporabe mobilne naprave. Pri našem iskanju uporabnikov nismo našli mlade osebe, ki bi bila nespretna pri uporabi mobilne naprave. Vsi uporabniki, ki so nespretni pri uporabni mobilne naprave, so starejše osebe, ki zaradi starosti nimajo več dobrega vida.

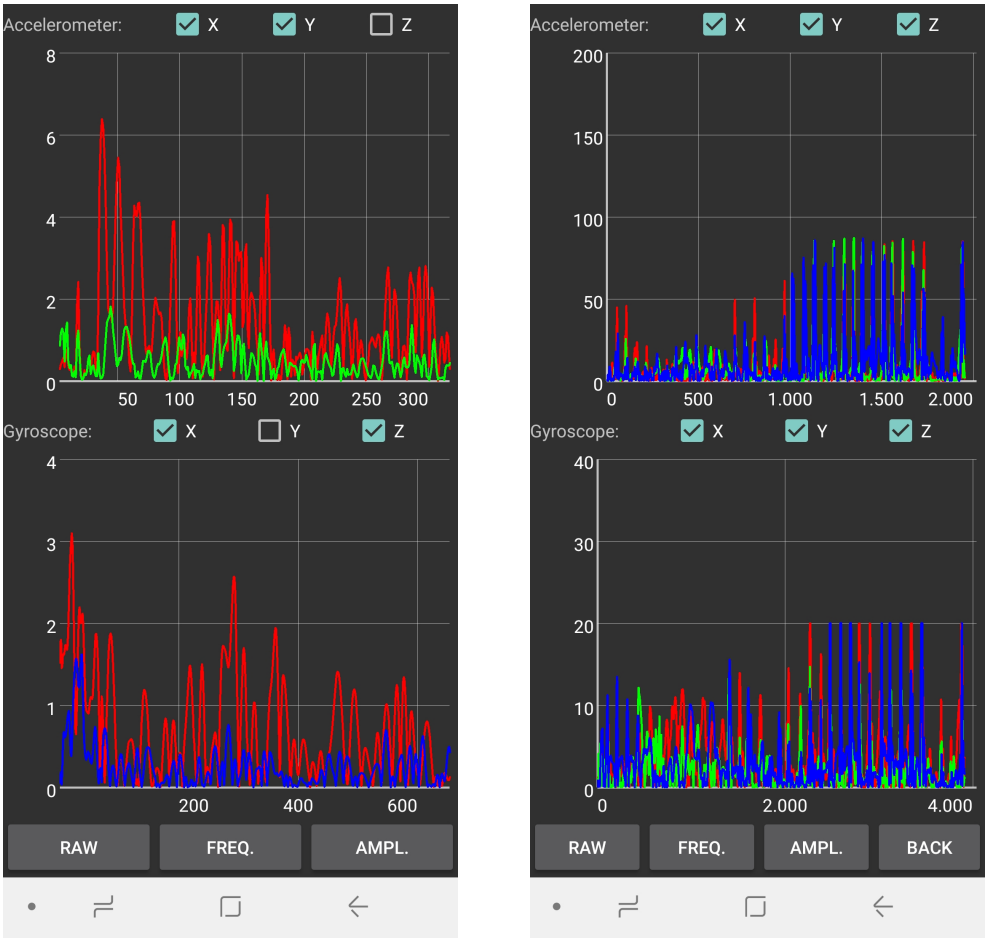
Testiranje je za vsakega uporabnika potekalo po naslednjem postopku:

1. Uporabniku je bila dana mobilna naprava z odprto aplikacijo.
2. Uporabnik je bil pozvan, da vnese poljubno številko, da lahko začne uporabljati aplikacijo.
3. Uporabnik je bil pozvan, da začne test.
4. Uporabnik je bil pozvan, da si ogleda izmerjene podatke.
5. Uporabnik je bil pozvan, da pogleda različne vizualizacije.

Če so imeli uporabniki težave pri uporabi aplikacije, jim je bila nudena pomoč. Na koncu testiranja so uporabniki podali svoje opažene pomankljivosti in svoje mnenje za izboljšavo.
Odkrita pomankljivost je bila, da uporabniki s slabim vidom težko vidijo gumb operacijskega sistema za prehod na prejšnji meni. Predlog za izboljšavo je bil prevod aplikacije v slovenščino.

3.2 Opis sprememb

Po testiranju z uporabniki in zajemanju pomankljivosti in predlogov, je bila aplikacija izboljšana. Na zaslon s prikazom vizualizacije je bil dodan gumb za nazaj. Primerjava je prikazana na sliki 3.1. Vsi gumbi za nazaj so bili poimenovani enako. Prevod aplikacije se zaradi časovne omejitve razvoja ni izvedel, vendar je prevod preprost, saj so vsa besedila prebrana iz ene datoteke. Tako je za prevod aplikacije potrebno spremeniti samo eno datoteko.



(a) Pred testiranjem

(b) Po testiranju

Slika 3.1: Primerjava vizualizacije pred in po spremembah na podlagi testiranja.

3.3 Testiranje funkcionalnosti

Celotno testiranje funkcionalnosti je potekalo med razvojem funkcionalnosti. Testiranje je potekalo tako, da je bil najprej testiran predviden način uporabe. Nato so bili testirani možni nepredvidljivi primeri uporabe. Testiranje je bilo opravljeno s strani razvijalcev in manjšega števila uporabnikov.

Edina netestirana funkcionalnost je pošiljanje podatkov v zunanjo bazo. Za-

radi nedostopnosti zunanje baze testiranje ni bilo možno.

3.4 Primerjava pametnega telefona z drugimi senzorji

Odločili smo se, da poleg mobilne aplikacije testiramo tudi strojno opremo. Primerjali smo različne oblike senzorjev. V naši primerjavi smo uporabili rokavico s senzorji, prikazano na sliki 3.2, industrijski paket senzorjev, prikazan na sliki 3.2 in pametni telefon, prikazan na sliki 3.3. Nismo primerjali natančnosti med napravami, ker nismo imeli testa, s katerim bi lahko z napravami opravili enake gibe. Preverjali smo, kako preprosto je držati napravo in z njo izvajati meritve. Testirali smo na treh uporabnikih. Vsi uporabniki so bili brez gibalnih težav. Testiranje je potekalo tako, da je uporabnik v roki držal senzor in izvedel vse teste iz mobilne aplikacije. Med testiranjem se podatki niso beležili. Spodaj so napisane naše ugotovitve za vsako napravo.

(a) Rokavica s senzorji (b) Industrijski paket senzorjev

Slika 3.2: Primeri alternativnih oblik senzorjev.

Slika 3.3: Pametni telefon Samsung A8.

Rokavica s senzorji: Rokavica se najlažje drži v roki, saj jo uporabnik natakne na roko. Problem pri rokavici je, da se težko natakne na roko pri uporabnikih z velikimi rokami. Rokavica potrebuje tudi kabel za napajanje in prenos podatkov, ki mora biti priključen med merjenjem. Kabel ovira uporabnika pri izvajanju testa.

Industrijski paket senzorjev: To je oranžen kvader narejen iz plastike in kovine. Preprosto ga je držati v roki in zaradi majhne velikosti uporabniki z malimi rokami nimajo težav z držanjem naprave. Pomanjkljivost je, da paket senzorjev, enako kot rokavica, potrebuje povezavo s kablom za prenos podatkov in napajanje. Kabel ovira uporabnika pri izvajanju testa.

Pametni telefon: Prednost telefona je, da med izvajanjem testa ne potrebuje povezave s kablom. Težavo imajo samo uporabniki z malimi rokami, saj telefon težko držijo med merjenjem. Potrebno je opozoriti, da obstajajo manjši pametni telefoni, ki bi odpravili težavo pri teh uporabnikih.

Po primerjavi pametnega telefona z drugimi senzorji smo ugotovili, da je

pametni telefon ustrezna naprava za izvajanje testiranja. Od vseh naprav je tudi cenovno najbolj dostopen.

Poglavje 4

Zaključek

Nevrologi trenutno nimajo možnosti merjenja tremorjev pri pacientih s senzorji za pospešek in žiroskopom. Narejena aplikacija reši ta problem z uporabo senzorjev v mobilni napravi. Poleg tega omogoča hranjene, pošiljanje in vizualizacijo izmerjenih podatkov. Aplikacija je bila razvita z združitvijo različnih pristopov razvoja programske opreme. Posebna pozornost pri razvoju aplikacije je bila namenjena primernosti uporabniškega vmesnika za uporabnike s slabšim vidom in slabšim nadzorom prstov. Aplikacija je bila testirana pri uporabnikih in odziv uporabnikov je bil upoštevan pri razvoju. Pri razvoju aplikacije sem se naučil pomembnosti predvidevanja in načrtovanja zapletov pri razvoju programske opreme. Spoznal sem pomembnost testiranja pri različnih uporabnikih. Naučil sem se uporabe novih knjižnic.

4.1 Možnosti za nadaljni razvoj

Med razvojem aplikacije smo opazili naslednje možnosti za nadaljni razvoj:

- Trenutno se uporabljajo samo senzorji na telefonu. S telefonom bi se lahko povezalo dodatne senzorje (žiroskope in merilnike pospeška) in se jih namestilo na druge dele telesa. Tako bi se lahko dobili podatki s celotnega telesa. V tem primeru bi se lahko dodalo tudi nove teste, ki zahtevajo gibanje celotnega telesa.

- Analizirali samo posamične meritve. Naredila bi se lahko analiza večih meritev. Primerjalo bi se lahko povprečne amplitude večih meritev. Lahko bi se naredilo frekvenčno analizo večih meritev in prikazalo najbolj izrazite frekvence vsake meritve. Z analiziranjem večih meritev, se lahko dobi pregled v napredovanje pacientove bolezni med pregledi.

Literatura

[1] Zhan A, Mohan S, Tarolli C, and et al. Using smartphones and machine learning to quantify parkinson disease severity: The mobile parkinson disease score. *JAMA Neurology*, 75(7):876–880, 2018.

[2] Gartner says worldwide sales of smartphones returned to growth in first quarter of 2018. Dosegljivo: `https://www.gartner.com/newsroom/id/3876865`. [Dostopano: 10.8.2018].

[3] Günther Deuschl, Peter Bain, Mitchell Brin, and Ad Hoc Scientific Committee. Consensus statement of the movement disorder society on tremor. *Movement Disorders*, 13(S3):2–23, 1998.

[4] Sodelavci Medicinske fakultete v Ljubljani in drugi. *Slovenski medicinski slovar*. Univerza v Ljubljani, Medicinska fakulteta, 2012.

[5] Martin Fowler and Matthew Foemmel. Continuous integration. *Thought-Works) http://www. thoughtworks. com/Continuous Integration. pdf*, 122:14, 2006.

[6] Graphview. Dosegljivo: `http://www.android-graphview.org/`. [Dostopano: 28.8.2018].

[7] Craig Larman and Victor R Basili. Iterative and incremental developments. a brief history. *Computer*, 36(6):47–56, 2003.

[8] NIH Neurological Institute tremor information page. Dosegljivo: https://www.ninds.nih.gov/Disorders/All-Disorders/Tremor-Information-Page#disorders-r1. [Dostopano: 4.8.2018].

[9] Lee Rainie. Two-thirds of young adults and those with higher income are smartphone owners. *Race/ethnicity*, 65(830):11, 2012.

[10] Hugh Rickards. Tourette's syndrome and other tic disorders. *Practical neurology*, 10(5):252–259, 2010.

[11] Room persistence library. Dosegljivo: https://developer.android.com/topic/libraries/architecture/room. [Dostopano: 29.8.2018].

[12] Michael F Smith. *Software prototyping: adoption, practice and management*. McGraw-Hill, Inc., 1991.

[13] About sqlite. Dosegljivo: https://www.sqlite.org/about.html. [Dostopano: 29.8.2018].

[14] Maja Trošt. Parkinsonova bolezen. *Farmacevtski vestnik*, 59(2):60–63, 2008.