Saavutettavuus Android-sovelluskehityksessä

Marja Hyvärinen

Haaga-Helia ammattikorkeakoulu

Amk-opinnäytetyö

2021

Tradenomin tutkinto

Tiivistelmä

Tekijä(t)
Marja Hyvärinen

Tutkinto
Tradenomi

Raportin/Opinnäytetyön nimi
Saavutettavuus Android-sovelluskehityksessä

Sivu- ja liitesivumäärä
49 + 12

Opinnäytetyössä perehdyttiin Android-sovelluskehitykseen saavutettavuuden näkökulmasta. Työssä havainnoitiin saavutettavuusongelmia Google Play -sovelluskaupasta löytyvistä lääkityslista/lääkitysmuistuttajasovelluksista. Saavutettavuusanalyyseissä hyödynnettiin Web Content Accessibility Guidelines eli WCAG-ohjeistuksen kriteereitä, joista valikoitiin erityisesti mobiilisovelluksien kannalta olennaisia kohtia. Lisäksi sovelluksia testattiin Accessibility Scanner -sovelluksen, TalkBack-ruudunlukijan sekä Androidin Switch Access -kytkintoiminnallisuuden avulla.

Työssä toteutettiin Kotlin-ohjelmointikielellä lääkityslistamobiilisovellus, jossa on myös lääkkeenoton muistutustoiminnallisuus. Sovelluksen saavutettavuutta analysoitiin samoilla kriteereillä kuin kaupallisia sovelluksia. Saavutettavuusanalyysit suoritettiin iteratiivisissa Scrum-viitekehyksen mukaisissa sykleissä uusien ominaisuuksien valmistuttua. Todetut saavutettavuuspuutteet raportoitiin ja korjattiin.

Yleisimmät Android-sovelluskehitykseen liittyvät saavutettavuusongelmat, jotka tässä opinnäytetyössä tulivat esille, olivat riittämättömät kontrastisuhteet, puutteelliset ja virheelliset käyttöliittymäelementtien kuvaukset sekä liian pienet kosketusalueet. Myös tekstikoon suurentaminen aiheutti saavutettavuusongelmia, kuten käyttöliittymäelementtien peittymistä. Saavutettavuusongelmien yleisyys oli yllättävää tutkituissa sovelluksissa. Saavutettavuusongelmien syynä voivat olla kiire, saavutettavuuskriteerien huono tunteminen sekä riittämätön testaus. Saavutettavuus näyttäisi myös olevan vähemmän tärkeä ominaisuus, jonka toteuttamiseen ei ole varattu resursseja. Panostamalla saavutettavuuteen kehittäjä pystyy kuitenkin erottumaan edukseen mobiilisovellusmarkkinoilla, ja sovellus saa enemmän potentiaalisia käyttäjiä.

Opinnäytetyön analyysit tehtiin helmi-huhtikuun välisenä aikana ja lääkityslistasovellus toteutettiin maalis-syyskuun aikana 2021.

Asiasanat
saavutettavuus, WCAG, mobiiliohjelmointi, Android, mobiilisovellus, Kotlin

Sisällys

1 Johdanto

Saavutettava sovellus pyrkii huomioimaan, että sitä voidaan käyttää erilaisista vammoista tai rajoitteista huolimatta (W3C 2018). Saavutettavuuteen panostaminen sovelluskehityksessä lisää siis sovelluksen potentiaalisten käyttäjien määrää. Tekemällä sovelluksista saavutettavia edistetään myös vammaisten henkilöiden oikeuksien toteutumista ja heidän mahdollisuuttaan käyttää digitaalisia palveluita yhdenvertaisesti. Viranomaisilla ja julkishallinnon organisaatioilla on lakisääteinen velvollisuus toteuttaa omat digitaaliset palvelunsa saavutettavassa muodossa lain vaatimusten mukaisesti. (Laki digitaalisten palvelujen tarjoamisesta 306/2019.) Kun olin työharjoittelussa sovelluskehittäjänä julkishallinnon organisaatiossa, eräs työtehtäväni liittyi saavutettavuusvaatimuksiin ja digitaalisten palveluiden saavutettavuuden parantamiseen. Tätä kautta tutustuin hankaluuksiin, joihin esimerkiksi näppäimistön ja ruudunlukijan avulla digitaalisia palveluita käyttävät henkilöt saattavat törmätä.

Sovellukset toteutetaan usein valmiiksi asti, ennen kuin niille tehdään saavutettavuusanalyysi käyttäjätestausvaiheessa. Saavutettavuus tulisi kuitenkin huomioida jo kehitysprosessin aikana. Useimmiten saavutettavuusongelmat liittyvät koodissa oleviin ongelmiin sekä suunnitteluratkaisuihin, joita voi olla todella työlästä tai jopa mahdotonta muuttaa ilman projektin viivästymistä. (Horton & Sloan 2014.) Jos saavutettavuusanalyysia suoritetaan myös kehitystyön aiemmissa vaiheissa, ongelmalliset komponentit, puuttuvat saavutettavuuspiirteet tai huonot suunnitteluratkaisut tunnistetaan välittömästi, jolloin niiden korjaaminen on helpompaa. Pientenkin saavutettavuuskorjausten, kuten kuvaavan tekstin lisäämisen kuvaelementteihin ruudunlukijaa varten, tekeminen on vaivattomampaa kehitystyön aikana.

Tämän opinnäytetyön tarkoituksena oli perehtyä Android-mobiilisovellusten saavutettavuuteen liittyviin teknisiin piirteisiin sekä hyödyntää iteratiivista lähestymistapaa saavutettavuuden huomioimisessa kehitysprosessin aikana. Opinnäytetyön tuotoksena laadin saavutettavuusanalyysit kolmesta eri lääkemuistuttaja/lääkityslistasovelluksesta, ja toteutin oman lääkityslistamobiilisovelluksen. Valitsin aiheeksi lääkityslistasovelluksen, sillä tällä hetkellä ei ole olemassa tietoturvallista, digitaalista ja ajan tasalla pidettävää lääkityslistaa, vaikka tällaiselle on akuutti tarve. Kansallinen lääkityslista on vasta kehitteillä Kelassa, ja käyttöönotto ajoittuu optimitilanteessa arvion mukaan vuosille 2023–2024 (Kela 2021; Virkkunen 2020). Lääkityslistassa tulisi olla listattuna ainakin käytössä olevat lääkkeet sekä ravintolisät ja niiden annostus sekä ottoajankohdat. Lääkityslistan tärkeys korostuu monilääkittyjen henkilöiden kohdalla, mutta se on myös tarpeellinen tilanteissa, joissa joku toinen kuin lääkkeiden käyttäjä jakaa lääkkeet. (Suomen Apteekkariliitto 2021.)

Digitaalisen lääkityslistasovelluksen etuna on, että siihen voidaan lisätä myös lääkkeenoton muistuttamistoiminto. Myös muillakin kuin monilääkityillä ihmisillä voi olla vaikeuksia muistaa ottaa lääkkeitään. Vaikka lääkkeitä ei olisi paljon käytössä, esimerkiksi erityisannostelut, kuten harvemmin kuin kerran päivässä otettavat lääkkeet, voivat unohtua helpommin ilman muistutusta. Lääkehoidosta on kuitenkin hyötyä vain silloin, jos oikea lääke otetaan oikeaan aikaan ja oikealla tavalla. Lisäksi mukana kulkeva ja helposti muokattavissa oleva digitaalinen lääkityslista helpottaa kommunikaatiota esimerkiksi terveydenhuollon ammattilaisten kanssa asioidessa. Vaikka reseptit ovat sähköisessä muodossa Reseptikeskuksessa, ja käyttäjä voi tarkastella niitä sekä esimerkiksi tulostaa niistä yhteenvedon Omakanta-palvelun kautta, tämä ei tarkoita sitä, että kaikki Reseptikeskuksesta löytyvät lääkkeet olisivat käytössä (Kela 2019; Kela 2020). Jos esimerkiksi käytöstä poistunutta lääkettä ei muisteta mitätöidä, samasta lääkkeestä voi löytyä Reseptikeskuksesta reseptit kahdelle eri vahvuudelle, mikä voi aiheuttaa sekaannuksia ostettaessa lääkettä apteekista tai reseptiä uudistettaessa.

Halusin selvittää opinnäytetyössä:

- Miten voin soveltaa saavutettavuusvaatimuksia Android-sovelluskehityksessä?
- Millainen on hyvä saavutettavuustarkistuslista Android-sovelluskehityksessä?
- Mitkä ovat tyypillisiä Android-sovelluskehitykseen liittyviä saavutettavuusongelmia?

Tavoitteenani oli lisätä omaa ymmärrystäni sekä teknistä osaamistani mobiilisovellusten saavutettavuuteen liittyen. Lisäksi halusin haastaa itseni opettelemalla Android-sovelluskehitystä itselleni vieraalla ohjelmointikielellä Kotlinilla, joka on tällä hetkellä Android-sovelluskehityksen virallinen kieli (Haase 2019). Ensiksi tutustuin kansainväliseen Web Content Accessibility Guidelines (WCAG) -saavutettavuusohjeistukseen ja työkaluihin, joita hyödynsin saavutettavuusanalyysissäni. Laadin tarkistuslistan, jonka avulla pystyin käymään läpi tärkeimpiä mobiilisovellusten saavutettavuuteen liittyviä huomioita. Seuraavaksi vertailin jo olemassa olevia toteutuksia, joihin voi tallentaa oman muokattavan lääkityslistan ja joissa on lääkkeenottomuistutustoiminto. Tätä kautta pyrin lisäämään ymmärrystä tekniseen toteutukseen liittyvistä erilaisista ongelmista sekä ratkaisuista, joita hyödynsin omassa toteutuksessani.

Otin tarkasteluun kolme Google Play -sovelluskaupasta löytyvää lääkityssovellusta. Analysoin niitä saavutettavuuden näkökulmasta tarkistuslistallani olevien WCAG-kriteerien, saavutettavuustyökalujen sekä käyttäjätestauksen avulla ja laadin raportit valittujen mobiilisovellusten mahdollisista saavutettavuusongelmista. Lopuksi toteutin oman sovellukseni Scrum-viitekehystä soveltaen. Analysoin omaa sovellustani iteratiivisesti tarkistuslistani

sekä saavutettavuustyökalujen avulla kehitystyön aikana. Pyrin pitämään sovelluksen sisällön suoraviivaisena, jotta pystyn keskittymään enemmän tekniseen saavutettavuuteen. Laadin tuotteen kehitysjonon käyttäjätarinamuodossa. Tavoitteenani oli myös havainnollistaa etuja, joita toistuvalla saavutettavuusanalyysillä voidaan saavuttaa.

Työhön liittyvät keskeiset käsitteet:

Android Monelle erilaiselle laitteelle, kuten älypuhelimille ja tableteille, sopiva avoimeen lähdekoodiin perustuva ohjelmistopino, joka rakentuu käyttöjärjestelmästä, väliohjelmistosta, kirjastoista sekä järjestelmäsovelluksista (Meier 2012).

digitaalinen palvelu Tietoverkon välityksellä erilaisten päätelaitteiden avulla käytettävissä oleva verkkosivusto tai mobiililaitteelle suunniteltu mobiilisovellus (Laki digitaalisten palvelujen tarjoamisesta 306/2019).

Kotlin Yleiskäyttöinen ja staattisesti tyypitetty ohjelmointikieli, joka on tällä hetkellä Googlen tukemana Android-sovelluskehityksen ensisijainen ohjelmointikieli (Haase 2019; Kotlin 2021).

mobiilisovellus Yleiseen käyttöön kehitetty ja tarkoitettu sovellus, jota ajetaan ja älypuhelimessa tai tabletissa (Laki digitaalisten palvelujen tarjoamisesta 306/2019).

MVVM Arkkitehtuurimalli, jossa sovelluksen käyttöliittymä, esitys- ja sovelluslogiikka sekä tietomalli on erotettu selkeiksi ja paremmin hallittaviksi kokonaisuuksiksi (Ghoda & Ferracchiati 2012).

ruudunlukija Ohjelma, joka muuntaa ruudulla olevan tekstin näkövammaisille ymmärrettävissä olevaan muotoon, kuten puheeksi tai pistekirjoitukseksi pistenäyttöä varten (Google Material Design 2021a).

Scrum Projektinhallinnan viitekehys, jossa kehitystyötä tehdään iteratiivisesti sekä inkrementaalisesti pyrkien siihen, että jokaisen kehityssyklin jälkeen on olemassa arvoa tuottava ja julkaisukelpoinen kokonaisuus, jota muokataan saadun arvioinnin mukaan seuraavassa syklissä (Schwaber & Sutherland 2020).

saavutettavuus Digitaaliseen palveluun liittyvä ominaisuus, jolla tarkoitetaan sitä, että palvelua voi käyttää kuka tahansa huolimatta mahdollisista vammoista tai rajoitteista (Celia 2021).

saavutettavuusvaatimukset

Kansainvälisen Web Content Accessibility Guidelines -ohjeistuksen version 2.1 A- ja AA-tasoihin kuuluvat kriteerit (Aluehallintovirasto 2021a).

sprintti Enintään kuukauden pituinen tai lyhyempi aikaväli, jonka aikana toteutetaan julkaisukelpoinen versio tuotteesta (Schwaber & Sutherland 2020).

TalkBack Googlen Android-laitteille kehittämä ruudunlukija, joka mahdollistaa Android-laitteen käytön näkövammaisille (Google Android Accessibility Help 2021a).

tuotteen kehitysjono Lista toteutettavan tuotteen ominaisuuksista, vaatimuksista ja toiminnallisuuksista (Schwaber & Sutherland 2020).

WCAG Kansainväliset verkkosisällön saavutettavuusohjeet (Web Content Accessibility Guidelines), jotka antavat suosituksia saavutettavan verkkosisällön kehittämisestä (W3C 2018).

2 Saavutettavuus ja laissa säädetyt kriteerit

Jotta sovellus olisi saavutettava, sen tulee olla havaittava, hallittava, ymmärrettävä ja toimintavarma. Havaittavuudella tarkoitetaan sovelluksen tietojen ja käyttöliittymän eri osien esittämistä käyttäjälle siten, että ne ovat käyttäjän havaittavissa. Hallittavuudella tarkoitetaan tässä yhteydessä sovelluksessa käyttöliittymän osien ja navigoinnin hallittavuutta käyttäjän kannalta. Ymmärrettävyys liittyy sovelluksen käyttöliittymän ja toiminnan käsitettävyyteen, ja toimintavarmuudella tarkoitetaan sovelluksen sisällön toimintavarmuutta eli erityisesti sitä, että eri laitteet, ohjelmat ja avustavat teknologiat voivat tulkita sovelluksen sisällön ja toiminnan luotettavasti käyttäjälle. (Euroopan parlamentin ja neuvoston direktiivi julkisen sektorin elinten verkkosivustojen ja mobiilisovellusten saavutettavuudesta (EU) N:o 2102/2016.)

Digitaalisten palvelujen saavutettavuuden yhteydessä puhutaan sekä teknisestä että sisällöllisestä saavutettavuudesta. Teknisellä saavutettavuudella tarkoitetaan sitä, sovellus on koodattu virheettömästi ja saavutettavuuskriteereitä noudattaen, jotta sitä voidaan käyttää teknisten apuvälineiden avulla. Sisällöllinen saavutettavuus liittyy kognitiiviseen saavutettavuuteen eli siihen, että sovelluksen sisältö on laadittu ymmärrettäväksi sekä helposti omaksuttavaksi käyttämällä esimerkiksi selkeää yleiskieltä ja jaksottamalla tekstisisältö selkeiksi kokonaisuuksiksi. (Celia 2021.) Saavutettavuuteen panostaminen hyödyttää lähtökohtaisesti kaikkia käyttäjiä. Lisäksi mitä paremmin saavutettavaksi mobiilisovellus laaditaan, sen enemmän potentiaalisia käyttäjiä ja latauksia se saa.

2.1 EU:n saavutettavuusdirektiivi ja digipalvelulaki

Vuonna 2016 voimaan tulleen EU:n saavutettavuusdirektiivin avulla pyritään parantamaan julkisen sektorin digitaalisten palvelujen saavutettavuutta yhteisten vaatimusten avulla. Saavutettavuusdirektiivi pohjaa YK:n yleissopimukseen vammaisten henkilöiden oikeuksista, jonka mukaan vammaisilla henkilöillä tulisi olla yhtäläinen mahdollisuus käyttää sähköisiä palveluita sekä hyödyntää tieto- ja viestintäteknologiaa. Direktiivin saavutettavuusvaatimukset ovat teknologiariippumattomia ja niissä kuvataan edellytykset, joilla käyttäjät voivat havaita, käyttää, tulkita sekä ymmärtää digitaalisia palveluita. (Euroopan parlamentin ja neuvoston direktiivi julkisen sektorin elinten verkkosivustojen ja mobiilisovellusten saavutettavuudesta (EU) N:o 2102/2016.)

Digipalvelulaki eli laki digitaalisten palvelujen tarjoamisesta, jolla Suomessa pannaan täytäntöön EU:n saavutettavuusdirektiivi, tuli voimaan huhtikuussa 2019. Lakia sovelletaan julkisoikeudellisten laitosten sekä viranomaisten digitaalisiin palveluihin, ja sen tarkoituk-

sena on parantaa kansalaisten mahdollisuutta käyttää digitaalisia palveluita yhdenvertaisesti. Lisäksi sellaiset digitaaliset palvelut, joiden kehitys- tai ylläpitokustannuksiin viranomainen osallistuu vähintään 50 % osuudella, kuuluvat lain piiriin. Lain saavutettavuusvaatimukset koskevat myös osaa yksityisen sektorin toimijoista, kuten esimerkiksi pankkeja, vakuutusyhtiöitä, liikenne- sekä postipalveluja tarjoavia yksiköitä. (Laki digitaalisten palvelujen tarjoamisesta 306/2019.)

Digitaalisten palveluiden, jotka kuuluvat digipalvelulain piiriin, tulee täyttää saavutettavuusvaatimukset. Nämä vaatimukset on määritelty kansainvälisessä WCAG 2.1 -ohjeistuksessa, josta noudatetaan A- ja AA-tason kriteerejä. Tästä poikkeuksena ovat suorat verkkolähetykset, joihin ei sovelleta näitä kriteereitä. (Aluehallintovirasto 2021b.) Lisäksi digitaalisille palveluille tulee laatia saavutettavuusseloste, josta selviää palvelun saavutettavuuden toteutuminen. Selosteessa tulisi olla vähintään maininta mahdollisista poikkeamista lain vaatimuksiin, poikkeamien perustelut, ohjeet palvelun käytöstä vaihtoehtoisella tavalla, jos se ei ole käyttäjälle saavutettavassa muodossa, ja linkki valvontaviranomaisen sivustolle, jossa voi tehdä saavutettavuuskantelun. Saavutettavuusdirektiivin mukainen saavutettavuusselosteen malli pitää olla saatavilla valvovan viranomaisen eli Etelä-Suomen aluehallintoviraston verkkosivustolla seuraavassa osoitteessa: www.saavutettavuusvaatimukset.fi. Lisäksi digitaalisen palvelun käyttäjällä tulee olla mahdollisuus lähettää palautetta tai vaatia tarkennuksia palveluntarjoajalta saavutettavuuteen liittyen. Saavutettavuuspalautteeseen on vastattava viimeistään 14 päivän kuluttua yhteydenoton saapumisesta. (Laki digitaalisten palvelujen tarjoamisesta 306/2019.)

Muutkin lait velvoittavat tai pyrkivät ohjaamaan saavutettavuuden parantamiseen. Perustuslain mukaan ketään ei saa syrjiä vammaisuuden tai muun henkilöön liittyvän syyn perusteella. Yhdenvertaisuuslaki puolestaan velvoittaa viranomaisia edistämään yhdenvertaisuuden toteutumista. Asianmukaisten ja kohtuullisten mukautusten avulla vammaisen henkilön tulisi voida asioida, saada koulutusta, työtä sekä palveluita yhdenvertaisesti muiden kanssa. (Aluehallintovirasto 2021c.)

2.2 WCAG:n hyödyntäminen mobiilisovellusten saavutettavuudessa

Julkisen hallinnon ylläpitämien mobiilisovellusten tulee täyttää saavutettavuusdirektiivin vaatimukset 23.6.2021 alkaen (Valtiovarainministeriö 2021). WCAG (Web Content Accessibility Guidelines, verkkosisällön saavutettavuusohjeet) on joukko suosituksia ja ohjeistuksia, joiden avulla verkkosisällön saavutettavuutta voidaan parantaa (Aluehallintovirasto 2021b). Lisäksi saavutettavuusohjeisiin kuuluu teknologiariippumattomia ja testattavia kriteerejä onnistumiselle. Ohjeistus rakentuu saavutettavuuden neljän periaatteen ympärille (havaittavuus, hallittavuus, ymmärrettävyys ja toimintavarmuus). Ohjeistuksen 13 ohjetta kertovat kehittäjille perustavoitteet, jotka tulisi huomioida saavutettavuutta toteutettaessa. Jokaiseen ohjeeseen kuuluvat eritasoiset kriteerit, joiden avulla voidaan testata tai todeta, läpäiseekö digitaalinen palvelu vaatimukset. Kriteerit on jaettu kolmeen tasoon riippuen eri ryhmien tai tilanteiden tarpeista: A- (matalin), AA- ja AAA-taso (korkein). (Papunet 2020; W3C 2018.)

WCAG-ohjeistus kehittyy jatkuvasti ja pyrkii huomioimaan monipuolisemmin eri käyttäjäryhmät, kuten esimerkiksi kognitiivisia vaikeuksia omaavat henkilöt, sekä kehittyvät teknologiat. Ohjeistus pyrkii myös paremmin vastaamaan kehittäjien tarpeisiin. (W3C 2021a.) Tällä hetkellä WCAG-ohjeistuksen versio 2.1, johon digipalvelulain vaatimukset perustuvat, antaa ohjeita sovelluksen saavutettavuudesta lähinnä tekniseltä kannalta, ja korkeimman AAA-tason vaatimukset täyttävä sovellus ei välttämättä ole vielä riittävän saavutettava tietylle kohdeyleisölle (Aluehallintovirasto 2021b; W3C 2018).

WCAG-ohjeistus keskittyy pääsääntöisesti verkkoteknologioihin, mutta sitä voidaan soveltaa myös mobiilisovellusten saavutettavuuden parantamiseen. Esimerkiksi mobiililaitteiden pieni ruutukoko aiheuttaa omat haasteensa havaittavuuden toteuttamiselle, joten ruudulla näkyvän sisällön tulisi olla järkevästi aseteltu ja rajattu. Kriteerin 1.4.4 (Tekstin koon muuttaminen) mukaan tekstin kokoa pitäisi myös pystyä suurentamaan siten, että sovelluksen toiminnallisuus tai sisällön tulkitseminen ei kärsi tästä. Mobiililaitteen käyttöjärjestelmän saavutettavuusasetuksissa voidaan esimerkiksi vaikuttaa ruudun suurennokseen tai laitteen oletustekstikokoon. Mobiilisovelluksen tulisi tukea järjestelmän fontteja, jotka seuraavat käyttäjän omia asetuksia. Hyvän kontrastisuhteen noudattaminen on jopa kriittisempää mobiililaitteilla kuin tietokoneen näytöllä johtuen siitä, että mobiililaitteita käytetään paikoissa, joissa valaistusolosuhteet vaihtelevat. Kriteeri 1.4.3 (Kontrasti, minimi) antaa suosituksen kontrastisuhteesta (4,5:1 tai 3:1 isommalle tekstille). (W3C 2015.) Kriteerin 1.4.10 (Responsiivisuus) mukaan sisältö pitäisi pystyä esittämään ilman kahdensuuntaista vieritystä pystysuunnassa 320 pikseliä leveällä näytöllä tai vaakasuunnassa 256

pikseliä korkealla näytöllä pois lukien esimerkiksi kartta- ja pelisovellukset, joissa kahdensuuntainen esitystapa on olennainen merkityksen vuoksi (Aluehallintovirasto 2021b; W3C 2020).

Vaikka mobiilisovelluksia käytetään kosketusnäytön kautta, mobiililaitteiden käyttöjärjestelmät tukevat laitteen käyttöä ulkoisen näppäimistön avulla. Tämä on tärkeä hallittavuutta parantava toiminnallisuus esimerkiksi liike- tai näkövammaisille, ja se tulisi huomioida myös mobiilisovelluksen suunnittelussa esimerkiksi kriteerien 2.1.1 (Näppäimistö), 2.1.2 (Ei näppäimistöansaa), 2.4.3 (Kohdistusjärjestys) ja 2.4.7 (Näkyvä kohdistus) avulla. Näytöllä näkyvät elementit tulisivat olla riittävän isoja ja erillään toisistaan, jotta käyttäjä voi aktivoida juuri oikean elementin. Interaktiivisten elementtien sijoitteluun ruudulla tulisi myös kiinnittää huomiota. (Aluehallintovirasto 2021b; W3C 2015.) Muita hallittavuutta parantavia ominaisuuksia ovat kriteerissä 2.5.4 (Käyttö liikkeen avulla) mainittu kosketusnäytön ja laitteen manipuloinnin (ravistus, kallistaminen) eleiden huomiointi siten, että näiden hyödyntäminen ei estä käyttäjää toteuttamasta tiettyä toiminnallisuutta. Kriteerin 2.5.1 (Osoitineleet) mukaan monimutkaisille monipiste- tai reittiin perustuville ohjauseleille tulisi olla saavutettavampi vaihtoehto, ellei tietty ohjaustapa ole jollain tapaa olennainen sovelluksen informaation tai toiminnallisuuden kannalta. Kriteeri 2.5.2 (Osoitineleellä tehdyn valinnan peruuttaminen) ohjeistaa ehkäisemään tietyn toiminnon laukaisemista vahingossa siten, että toimintoa ei laukaista heti kosketuksen alussa, jotta käyttäjä voisi perua toiminnon esimerkiksi viemällä kohdistuksen sormella muualle ruudulla. (Aluehallintavirasto 2021b; W3C 2020.)

Ymmärrettävyyteen vaikuttaa näytön orientaation vaihtuminen pysty- tai vaakatasoon, ja mobiilisovelluksen pitäisi tukea molempia orientaatioita (WC3 2015). Tämä liittyy myös havaittavuuteen kriteerin 1.3.4 mukaan, jonka mukaan sisältöä ei saisi rajoittaa yhteen orientaatioon poissulkien tapaukset, joissa tietty asento kuuluu esittää tietyllä tavalla (WC3 2018; W3C 2020). Näkymässä olevat tärkeimmät elementit tulisi sijoittaa siten, että vieritystä ei tarvitse tehdä, jotta ne ovat näkyvissä. Elementit, jotka laukaisevat toimintoja, tulisi olla tunnistettavissa muusta sisällöstä. Jos sovelluksessa käytetään ei-tavanomaisia laite- tai kosketuseleitä, pitäisi näistä kriteerin 3.2.2 mukaan olla helposti löydettävissä olevat ohjeet näiden käyttöä varten. (W3C 2015.)

Sovelluksen toimintavarmuus paranee, jos käyttäjän tarvitsee itse syöttää sovellukseen mahdollisimman vähän informaatiota. Datan syöttämistä voidaan helpottaa erilaisilla valintavalikoilla ja -napeilla sekä tunnetun datan, kuten päivämäärän tai sijainnin, automaattisella täyttämisellä. Lisäksi voidaan hyödyntää erilaisia virtuaalisia näppäimistöjä datan syöttämisessä riippuen datan tyypistä, mutta tämä saattaa hämmentää ruudunlukijan

avulla sovellusta käyttäviä henkilöitä. Sovelluksen tulisi tukea kunkin mobiililaitteen käyttöjärjestelmän omia saavutettavuutta parantavia toimintoja, kuten tekstin fontin koon muuttamista tai sisällön tarkennusta. (W3C 2015.) Jonkin toiminnan tilaan liittyvät viestit tulee kriteerin 4.1.3 (Tilasta kertovat viestit) mukaan toteuttaa niin, että viesti esitetään käyttäjälle avustavan teknologian kuten ruudunlukijan avulla siten, että kohdistusta ei tarvitse siirtää (Aluehallintavirasto 2021b; W3C 2020).

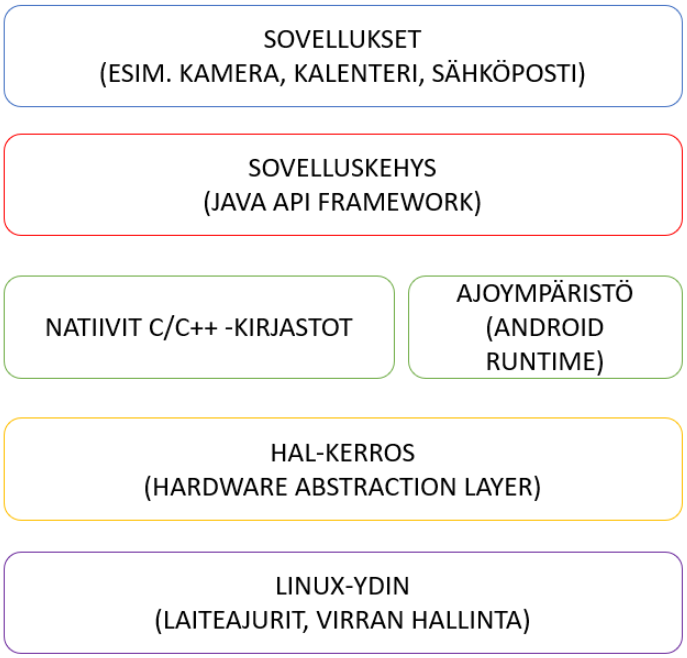
Tärkeitä WCAG-kriteerejä, jotka liittyvät nimenomaan mobiilisovellusten saavutettavuuteen on kerätty ja listattu liitteessä 1. Käytin näitä kriteerejä oman ja Google Play -kaupasta löytyvien sovellusten saavutettavuuden arvioinnissa.

3 Android-sovelluskehitys ja saavutettavuus

Android-sovelluskehittäjille on tällä hetkellä tarvetta, sillä vuoden 2021 alussa Androidin globaali markkinaosuus on noin 72 prosenttia myydyistä laitteista. Kilpailevan käyttöjärjestelmän iOS:n markkinaosuus on noin 27 prosenttia. (StatCounter 2021.) Tuoreen suomalaisen verkkokyselytutkimuksen mukaan, joka oli suunnattu pääsääntöisesti Näkövammaisten liittoon kuuluville jäsenille, iOS (70,5 %) on ylivoimaisesti Androidia (29 %) suositumpi käyttöjärjestelmä mobiiliympäristössä (Kallionpää 2021). Saavutettavuuden suhteen iOS-älypuhelimia onkin pidetty parempina Androidiin verrattuna, mutta Android on kehittänyt ja parantanut selkeästi saavutettavuusominaisuuksiaan alkuaikoihin verrattuna (Hill & Jansen 2021).

3.1 Android

Android-alusta on Linux-pohjainen, avoimeen lähdekoodiin perustuva ohjelmistopino, joka rakentuu Linux-ytimestä, sen päällä olevasta HAL-kerroksesta (Hardware Abstraction Layer), Android runtime -ajoympäristöstä (ART), natiiveista C/C++ -kirjastoista, sovelluskehyksestä sekä sovelluksista. Linux-ytimessä ovat laiteajurit, ja ydin vastaa muistin-, prosessien-, virran-, verkon sekä tietoturvallisuuden hallinnasta. HAL-kerroksesta löytyy kirjastomoduuleita, jotka kukin tarjoavat rajapinnan laitteiston komponenteille, kuten esimerkiksi sensoreille ja kameralle. (Google Android Developers 2021a.) Näiden kerrosten päällä sijaitsevat Android runtime eli Android-ohjelmien ajoympäristö ja siihen kuuluvat ydinkirjastot muodostavat pohjan sovelluskehykselle. Samalla tasolla ajoympäristön kanssa ovat natiivit kirjastot, joiden avulla voidaan käsitellä esimerkiksi media- ja grafiikkadataa sekä huolehtia datan tallennuksesta. Sovelluskehys (Java API Framework) tarjoaa luokat eli työkalut, joiden avulla Android-sovelluksia voidaan luoda. (Meier 2012.) Sovelluskehykseen kuuluvat näkymien hallinta (View System), resurssien hallinta (Resource Manager), ilmoitusten hallinta (Notification Manager), aktiviteettien hallinta (Activity Manager) ja sisällön tarjoajat (Content Providers) (Google Android Developers 2021a).



Kuva 1. Android-ohjelmistopino (mukaillen Google Android Developers 2021a)

Sekä natiivit että kolmannen osapuolen sovellukset ajetaan omina prosesseinaan oman virtuaali-ART:n sisällä (Google Android Developers 2021a). Jokainen sovellus on siis eristyksissä muista sovelluksista suorituksen aikana, ja Android-käyttöjärjestelmä hallinnoi sovelluksia käynnistämällä sekä sammuttamalla prosesseja tilanteen mukaan. Sovelluksen prosessi lakkautetaan esimerkiksi silloin, jos sovellusta ei enää tarvita, tai muut sovellukset vaativat lisää muistia käyttöönsä. Android-käyttöjärjestelmässä jokainen sovellus on eri Linux-käyttäjä, ja sovelluksilla on uniikki tunniste (id), jonka mukaan käyttöjärjestelmä asettaa sovelluksen tiedostojen käyttöoikeudet. Sovellukset voivat kuitenkin olla vuorovaikutuksessa keskenään jakamalla saman käyttäjätunnisteen. Järjestelmän resurssien säästämiseksi sovellukset voidaan ajaa samassa Linux-prosessissa saman virtuaalikoneen sisällä. Lisäksi sovellukset voivat pyytää käyttäjältä lupaa päästä käsiksi muihin resursseihin, kuten kameraan tai laitteen sijaintitietoihin. (Google Android Developers 2021b.)

3.2 Kotlin

JetBrains-yhtiö alkoi kehittää Kotlinia vuonna 2010, joten Kotlin on vielä suhteellisen tuore ohjelmointikieli, joka on kuitenkin jo saavuttanut aktiivisen kehittäjäyhteisön tuen. Kotlin on avoimeen lähdekoodiin perustuva staattisesti tyypitetty yleiskäyttöinen kieli, ja sillä voi ohjelmoida sekä funktionaalisen ohjelmoinnin että olio-ohjelmoinnin tapoja hyödyntäen. (Kotlin 2021.) Vuonna 2017 Google ilmoitti tukevansa Kotlinia Android-alustalla ja pari vuotta myöhemmin Google teki Kotlinista ensisijaisen kielen Android-ohjelmoinnille (Haase 2019; Moskala & Wojda 2017). Uudet Android-sovellukset tulisi kirjoittaa lähtökohtaisesti Kotlinilla (Haase 2019).

Kotlin on tyyppiturvallinen kieli, eli staattisen tyypityksen ansiosta ohjelman käännösaikana muuttujien tyypit tarkistetaan. Kotlinin tyyppijärjestelmän erikoisuus ja ohjelmointivirheitä vähentävä piirre on tyhjien arvojen käsittely. Kotlin-kääntäjälle tulee erikseen ilmoittaa, jos jokin muuttuja voi saada arvon ”null”, muussa tapauksessa käännös kaatuu virheeseen. Kotlin-koodista on tehty tiivistä eli tarvitaan vähemmän koodia yleisten toiminnallisuuksien toteuttamiseen. (Moskala & Wojda 2017.) Esimerkiksi hyödyntämällä Kotlinin data class -luokkaa saadaan sama toiminnallisuus yhdellä rivillä koodia verrattuna Javan useita kymmeniä riviä pitkään vastaavaan toteutukseen, sillä luokan konstuktori, attribuuttien get- ja set-metodit, toString()-, equals()-, hashCode()-, copy()- ja componentN()-metodit generoituvat automaattisesti (Baeldung 2021). Koodi on täten ylläpidettävämpää ja sitä on helpompaa lukea. Kotlinilla on läheinen suhde Java-ohjelmointikieleen, ja Kotlinia voidaan käyttää Javan rinnalla projekteissa. Javan kirjastot ja ohjelmointikehykset toimivat myös Kotlinilla. Mobiiliohjelmoinnin lisäksi Kotlinia käytetään palvelinohjelmoinnissa, työpöytäsovelluksissa ja verkkosovelluksissa. (Moskala & Wojda 2017.)

3.3 Android Studio ja Android-sovelluksen rakenne

Android Studio on Android-sovellusten ohjelmointia varten tarkoitettu ohjelmointiympäristö. Android Studio sisältää Android SDK -kehityspaketin, jonka työkalut helpottavat Android-kehitystä. Android Studion avulla voidaan kirjoittaa ohjelmia Kotlinilla, Javalla tai C++ -kielellä, hallinnoida sovellusten resursseja sekä testata sovelluksia erilaisilla emulaattoreilla. Android Studio tarjoaa työkaluja koodin analysoimiseen sekä julkaisemiseen. (Späth 2018.) Android SDK:n avulla lähdekoodi, resurssi- ja datatiedostot kääntyvät APK-pakettitiedostoksi (Android package), ja tätä .apk-päätteistä tiedostoa tarvitaan sovelluksen asentamiseksi (Google Android Developers 2021b).

Android-sovellus koostuu pääkomponenteista, niin sanotusta manifest-tiedostosta ja resurssitiedostoista. Sovellus ei käynnisty ilman manifest-tiedostoa (AndroidManifest.xml), joka määrittelee sovelluksen käyttämien komponenttien lisäksi sovelluksen käyttöluvat,

kuten pääsyn internetiin, sovelluskehyksen alimman tuetun version, sovelluksen tarvitsemat laitteisto- ja ohjelmisto-ominaisuudet ja muut mahdolliset erilliset API-kirjastot, kuten Google Maps -kirjaston. Resurssitiedostot koostuvat sovelluksen tarvitsemista kuvista, ääni- ja tyylitiedostoista yms., jotka ovat erillään sovelluksen lähdekoodista. Resurssit määritellään XML-tiedostoissa, ja niihin viitataan uniikeilla tunnisteilla koodissa. Erillisten resurssitiedostojen avulla voidaan helpommin määritellä vaihtoehtoisia resursseja erilaisille laitekonfiguraatioille. Sovelluksen pääkomponentteja on neljä, ja ne mahdollistavat käyttäjän tai järjestelmän pääsyn sovellukseen. Komponenttityypeillä on kullakin omat elinkaarensa, jotka määrittelevät komponenttien vaiheet komponentin luomisesta sen tuhoamiseen. Android-sovelluksilla ei ole yhtä yksittäistä sisääntuloa, ja muut sovellukset voivat käynnistää toisten sovellusten komponentteja käyttöjärjestelmän tuella. (Google Android Developers 2021b.)

Aktiviteetti-komponentti on yksittäinen ruutu näytöllä, jonka kanssa käyttäjä voi olla vuorovaikutuksessa. Sovellus voi koostua useasta eri aktiviteetista, ja käyttäjä tai muut sovellukset voivat kutsua sovelluksen aktiviteetteja missä järjestyksessä tahansa, sillä aktiviteetit ovat itsenäisiä kokonaisuuksia. Aktiviteettien elinkaaren tilan mukaan järjestelmä varmistaa, että tarvittavat prosessit ovat käynnissä tai että ne voidaan sammuttaa tarpeen vaatiessa. (Google Android Developers 2021b.)

Palvelu-komponentteihin ei liity käyttöliittymää, ja niitä hyödynnetään pitkäkestoisten operaatioiden suorittamiseen taustalla. Muut komponentit voivat käynnistää palveluita, kuten musiikin toistoa tai tiedon hakua verkosta, ja nämä palvelut pysyvät käynnissä niiden elinkaaren ajan, vaikka käyttäjä siirtyisi käyttämään toista sovellusta. Toinen komponentti voi myös sitoa palvelun itseensä ja olla vuorovaikutuksessa sen kanssa. Esimerkiksi saavutettavuustoiminnallisuudet, jotka tarjoavat toimintarajoitteisille ja vammaisille käyttäjille käyttöliittymään parannuksia, on tehty palveluina, joiden prosessit järjestelmä pitää käynnissä taustalla. (Google Android Developers 2021b.) Android-käyttöjärjestelmä hallinnoi saavutettavuuspalveluja, joiden elinkaari alkaa siitä, kun käyttäjä laittaa palvelun päälle laiteasetuksista (Google Android Developers 2021c). Android-käyttöjärjestelmällä on omat saavutettavuuspalvelunsa, kuten TalkBack, mutta Androidille voidaan myös luoda omia saavutettavuuspalveluja joko yhdistämällä ne toiseen sovellukseen tai tekemällä niistä erillisen sovelluksen. Mukautetut saavutettavuuspalvelut perivät AccessibilityService-luokan. (Google Android Developers 2021d.)

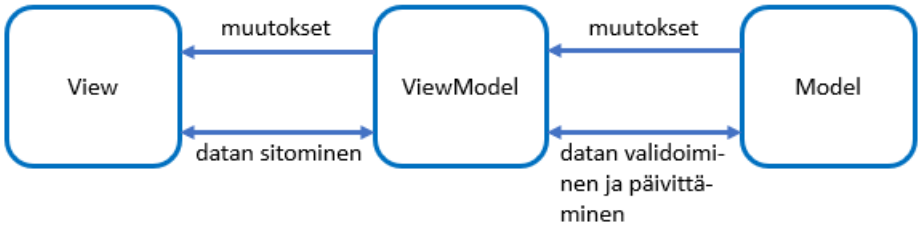
Broadcast Receiver -komponentin avulla voidaan vastaanottaa muiden sovellusten tai järjestelmän lähettämiä ilmoituksia. Tämä komponentti tarjoaa sisääntulon sovellukseen, jonka ei siis tarvitse olla käynnissä, jotta se voisi vastaanottaa ja reagoida ilmoitukseen. Broadcast Receiver -komponentti itsessään ei sisällä käyttöliittymää, mutta se voi antaa

käyttäjälle näkyvän ilmoituksen ja toimia välittäjinä muihin komponentteihin. Tämä komponentti on tärkeä esimerkiksi muistuttajasovelluksille. (Google Android Developers 2021b.)

Sisällön tarjoaja -komponentin (Content provider) avulla voidaan hallinnoida sovelluksen dataa, joka voi olla tallennettuna esimerkiksi tiedostojärjestelmään, tietokantaan tai johonkin muuhun pysyvään tallennustilaan, johon sovelluksella on pääsy. Sisällön tarjoaja huolehtii siitä, saavatko muut sovellukset muokata tai päästä käsiksi sen tarjoamaan dataan. Datalähteet ilmoitetaan URI-merkkijonotunnisteen avulla. Sisällön tarjoajat lisäävät tietoturvaa lupien hallinnan avulla eli rajoittamalla pääsyä sovelluksen dataan. (Google Android Developers 2021b.)

3.4 Model-View-ViewModel -arkkitehtuurimalli (MVVM)

Model-View-ViewModel (MVVM) -arkkitehtuurimallin pohjalla ovat Model-, View- ja ViewModel-luokat, joiden avulla voidaan selkiyttää rooleja ja jakaa vastuita esitys- ja sovelluslogiikkakerroksien välillä. Arkkitehtuurimallin Model-luokka enkapsuloi sovelluksen datan ja siihen liittyvän sovelluslogiikan. ViewModel-luokka vastaa esityslogiikasta ja datan käsittelystä View-luokkaa varten sekä huolehtii View- ja Model-luokkien välisestä vuorovaikutuksesta. View-luokka puolestaan hallinnoi sovelluksen käyttöliittymän rakennetta ja ulkoasua. (Ghoda & Ferracchiati 2012.) Arkkitehtuurimalli on esitetty kuvassa 2.



Kuva 2. MVVM-arkkitehtuurimalli (mukaillen Ghoda & Ferracchiati 2012)

MVVM-malli on yksi yleisimmistä Android-kehityksessä käytetyistä arkkitehtuurimalleista (Verdecchia, Malavolta & Lago 2019, 144). Mallin yleistymistä on edesauttanut Googlen vuonna 2017 julkaisema ViewModel-arkkitehtuurikomponentti (Google Android Developers Blog 2017; Verdecchia ym. 2019, 144). MVVM-arkkitehtuurin noudattaminen auttaa rakentamaan vakaampia sovelluksia, edistää sovelluksen eri vastuualueiden tunnistamista ja jakamista komponenttien välillä, tekee testaamisen helpommaksi ja vähentää tahattomia muistivuotoja (Verdecchia ym. 2019, 145–147).

3.5 Saavutettavuuden huomioiminen Android-sovelluskehityksessä

Android-käyttöjärjestelmän omiin saavutettavuusasetuksiin tai -sovelluksiin kuuluvat esimerkiksi ruudunlukijat, näytön ja fontin koon muokkaaminen, suurennustoiminto, kontrastin sekä värien säätö ja puheen sekä äänientunnistusapuvälineet. Google Play -kaupasta löytyy myös paljon ladattavia saavutettavuussovelluksia, joilla voi helpottaa älylaitteen käyttöä. (Google Android Accessibility Help 2021b.) Lisäksi puhelimessa voi ottaa käyttöön esteettömyysvalikon, Teksti puhuttuna -toiminnon, hyödyntää äänikomentoja ja mahdollistaa älypuhelimen käytön kytkimellä tai näppäimistöllä (Google Play kauppa 2021a).

Ruudunlukija, joka lukee ruudulla näkyvän sisällön ääneen, on tärkeä apuväline näkövammaisille henkilöille. Ruudunlukuohjelman avulla käyttäjä voi navigoida älypuhelimissa sormieleillä tai vaihtoehtoisesti puhelimeen liitetyn ulkoisen näppäimistön avulla. Android-käyttöjärjestelmän oman TalkBack-ruudunlukijan käynnistämisen jälkeen käyttäjällä on kaksi tapaa, miten navigoida laitteella: joko siirtämällä fokusta lineaarisesti pyyhkäisemällä näyttöä taakse- tai eteenpäin tai kuljettamalla sormea ruudulla etsien sisältöä. Elementit valitaan napauttamalla sormella kahdesti. (Google Material Design 2021a.)

Tiettyjen käyttöliittymäelementtien tarkoitus saattaa olla esitettynä käyttäjälle vain graafisesti. Jotta ruudunlukija voi ilmoittaa käyttäjälle, miten tällaiset elementit ovat vuorovaikutteisia, tulee näille käyttöliittymäelementille liittää kuvaus. Tämä saavutetaan Android-sovelluksessa lisäämällä elementeille android:contentDescription -attribuutti. Kuvauksen tulisi olla uniikki joka elementtiä kohden, ja kuvaukseen ei saa lisätä elementin tyyppiä, sillä ruudunlukija lukee tämän automaattisesti. Elementtien, jotka ovat vain koristeena, android:contentDescription-attribuutti tulisi asettaa tyhjäksi (@null) tai vaihtoehtoisesti alustaversioilla 16 ja ylöspäin käyttää attribuuttia android:importantForAccessibility. (Google Android Developers 2021e.) Pääsääntöisesti esimerkiksi ImageView-elementeillä, joissa ei ole tekstiä, tulisi olla contentDescription-attribuutti, mutta kuvan 3 esimerkissä on käytetty attribuuttia importantForAccessibility arvolla ”no”, sillä kyseinen elementti on vain koristeena eikä tuo käyttäjälle lisäinformaatiota.

```
<ImageView
    android:id="@+id/alarm_image"
    android:layout_width="36dp"
    android:layout_height="31dp"
    android:layout_marginStart="16dp"
    android:layout_marginTop="8dp"
    android:importantForAccessibility="no"
```

Kuva 3. Esimerkki attribuutin hyödyntämisestä ruudunlukijaa varten

EditText-objektit, joita hyödynnetään esimerkiksi lomakkeiden syöttökentissä, saadaan saavutettavammiksi lisäämällä niille android:hint-attribuutti. Lisäksi jos hyödynnetään View-elementtiä, kuten TextView-elementtiä kuvaamaan EditText-objektin sisältöä, kuvaavalla View-elementillä tulisi olla android:labelFor-attribuutti, jonka arvoksi laitetaan kuvattavan objektin tunniste. (Google Android Developers 2021f.)

Toinen tärkeä huomio on koota toisiinsa liittyvät elementit, kuten esimerkiksi taulukoiden rivit tai tuotteen tiedot, säiliöelementin sisään, jolloin ruudunlukijan kohdistus on tässä säiliössä. Tämä vähentää käyttäjän näytön pyyhkäisyjen tarvetta ja virtaviivaistaa ruudunlukijan puhetta. Tällainen ryhmittely ruudunlukijaa varten saadaan aikaan hyödyntämällä elementtien android:screenReaderFocusable ja focusable-attribuutteja. Jos sovelluksen View-elementin sisällä on otsikoita, jotka summaavat tekstialueiden sisältöä, apuvälineitä käyttävän käyttäjän navigaatiota helpottaa, jos elementille annetaan android:accessibility-Heading-attribuutti arvolla ’true’. Lisäksi sovelluksen eri näkymät voidaan nimetä android:accessibilityPaneTitle-attribuutin avulla, mikä helpottaa Androidin saavutettavuuspalveluiden kykyä antaa tarkempaa informaatiota käyttäjälle näkymän sisällön tai ulkonäön muuttuessa. (Google Android Developers 2021f.)

Käyttöliittymässä kannattaa hyödyntää tai periä mahdollisimman alhaalla luokkahierarkiassa olemassa olevia järjestelmän tarjoamia käyttöliittymäkomponentteja, sillä nämä tarjoavat myös saavutettavuustoiminnallisuuksia. Jokaisella uudelleenmääritellyllä callback-kutsulla tulisi kuitenkin olla sitä vastaava saavutettava toiminnallisuus (action). Muita tärkeitä tarkistettavia asioita ovat käyttöliittymäelementtien tunnistettavuus mahdollisimman monella eri tavalla, kuten väreillä, muodolla tai elementin tekstillä. (Google Android Developers 2021f.) Lisäksi mobiilisovelluksen käyttöliittymän painikkeiden kosketuksen kohdealueen tulisi olla vähintään 48dp x 48 dp, missä dp (device-independent pixel) on suhteellinen mittayksikkö, jonka avulla saadaan elementit skaalautumaan asianmukaisesti eri ruuduntiheyksille (Google Android Developers 2021e; Google Android Developers 2021g). Myös laitteen värinätoiminnallisuuden hallinta eli tuntoaistiin perustuvan palautteen hyödyntäminen esimerkiksi informaation vahvistamisessa tai muistutuksissa helpottaa kuulovammaisia henkilöitä. Värinää tulisi kuitenkin hyödyntää vain silloin, kun käyttäjä on sallinut niiden käytön järjestelmän asetuksissa. (Google Material Design 2021b.)

3.6 Saavutettavuuden testaaminen

Sovelluksen saavutettavuutta kannattaa testata ennen kaikkea ruudunlukijan ja esimerkiksi ulkoisen näppäimistön avulla, mutta Android-sovelluksille on tarjolla myös analyysityökaluja saavutettavuuden arviointia varten. Käyttäjätestauksessa voi hyödyntää ruudunlukijana Androidin omaa TalkBack-lukijaa (Google Android Developers 2021h.) Samsungin uudemmissa puhelimissa TalkBack-toiminnallisuutta kutsutaan nimellä Voice Assistant, ja vanhemmissa se on vielä TalkBack-nimellä (Samsung 2021). TalkBack-lukijan voi asettaa asetuksista näyttämään puheen ruudulla, mikä helpottaa sovelluskehittäjän testausta. Googlen Android Developers -sivusto (2021h) ohjeistaa kiinnittämään huomiota seuraaviin seikkoihin, kun sovellusta testataan ruudunlukijalla:

- onko käyttöliittymäelementeillä asianmukaiset kuvaukset ääneen puhuttuna
- ovatko ruudunlukijan ilmoitukset ytimekkäitä vai turhan yksityiskohtaisia
- onnistuuko sovelluksen päätoiminnallisuuksien käyttö vaivattomasti
- pystyykö sovelluksessa käymään läpi kaikki vuorovaikutteiset elementit pyyhkäisyeleillä
- lukeeko ruudunlukija ääneen ilmoitukset ja hälytykset

Android-puhelimia voidaan käyttää kosketusnäytön sijaan kytkintoiminnallisuuden (Switch Access) avulla, jolloin kytkintoiminnallisuus skannaa ruudulla olevia interaktiivisia elementtejä yksi kerrallaan, ja elementit voidaan valita joko ulkoisen näppäimistön tai kytkinlaitteen avulla (Google Android Accessibility Help 2021c). Lisäksi puhelimen omia painikkeita voidaan hyödyntää kytkiminä. Esimerkiksi äänen lisäyspainikkeen voi konfiguroida selauskytkimeksi, jolla käydään läpi käyttöliittymäelementtejä. Vastaavasti äänen vähennyspainikkeen voi asettaa valintakytkimeksi, jonka avulla valitaan elementti (Google Android Developers 2021h.). Googlen Android Developers -sivuston (2021h) mukaan kytkinominaisuutta testattaessa tulisi kiinnittää huomiota seuraaviin ongelmakohtiin:

- pystyykö sovelluksen päätoiminnallisuudet toteuttamaan kytkimen avulla
- pystyykö lomakkeisiin ja muihin syöttökenttiin täyttämään tekstiä vaivattomasti
- tuleeko elementteihin, joiden kanssa voi olla vuorovaikutuksessa, korostus aktivoitaessa ja vastaavasti ei-interaktiivisiin elementteihin ei
- ovatko elementit korostettuina vain kerran
- pystyykö kaikkia toiminnallisuuksia, jotka toimivat kosketuseleillä, toteuttamaan myös kytkimellä
- jos kytkintä käytetään ruudunlukijan kanssa, ilmoittaako ruudunlukija jokaisen elementin toiminnallisuuden asianmukaisesti

Google Play -kaupasta ladattavan Accessibility Scanner -sovelluksen avulla voi analysoida sovelluksia ja saada palautetta mahdollisista teknisistä toteutusvirheistä saavutettavuuteen liittyen. Accessibility Scanner -sovellus ilmoittaa WCAG-kriteereihin perustuvista saavutettavuusongelmista, kuten esimerkiksi huonosta kontrastista, liian pienestä elementin kosketuspinta-alasta tai puuttuvista kuvauksista elementeissä. Android-sovellusten käyttöliittymän automaattista testausta voidaan puolestaan tehdä monella erilaisella testauskehyksellä tai kirjastolla, kuten Espressolla. Espresso-kirjaston AccessibilityChecks-luokan avulla voidaan testata käyttöliittymäelementtien saavutettavuutta. Android-sovelluksia voidaan myös käyttää äänikomennoilla, mutta tämän toiminnallisuuden testaaminen on rajattu pois tästä opinnäytetyöstä. (Google Android Developers 2021h.)

4 Lääkitysmobiilisovellusten saavutettavuusanalyysit

Tässä osiossa analysoin olemassa olevia sovelluksia, joita voin hyödyntää oman lääkityslistasovellukseni kehitystyössä. Lääkityslista on luettelo käytössä olevista lääkkeistä sekä ravintolisistä, niiden annostuksista ja ottoajankohdista (Suomen Apteekkariliitto 2021). Esimerkki lääkityslistasta on esitetty kuvassa 4. Useimmiten lääkkeet on merkitty listaan kauppanimiltään, mutta lääkityslistassa voi olla myös tieto vaikuttavista aineista. Hain analyysia varten sovelluksia, jotka näyttävät käytössä olevat lääkkeet ja tämän lisäksi tarjoavat myös muistutustoiminnallisuuden lääkkeiden ottamiselle.

LÄÄKE	VAHVUUS	LÄÄKEMUOTO	ANNOSTUKSEN MÄÄRÄ JA AJANKOHTA				KESTO			KÄYTTÖTARKOITUS
			AAMU	PÄIVÄ	ILTA	YÖ	SÄÄNN.*	TARV.**	KUURI	
Amlodipin	5 mg	tabletti	1				x			verenpainelääke
Atorvastatin	20 mg	tabletti				1	x			kolesterolilääke
Panadol Forte	1 g	tabletti	1	1		1		x		särkylääke

*SÄÄN. = SÄÄNNÖLLINEN
**TARV. = TARVITTAESSA

Kuva 4. Esimerkki lääkityslistasta (mukaillen Suomen Apteekkariliitto 2021)

4.1 Sovellusten haku analyysia varten

Google Play -kauppa on merkittävin Android-sovelluksien jakelukanava, joten hain sovelluksia analyysia varten tästä sovelluskaupasta (Dogtiev 2021). Käytin englanninkielisiä hakutermejä, koska sain niiden avulla enemmän järkeviä hakutuloksia. Hakusanalla ”medicine list” ei löytynyt yhtä hyvin tuloksia verrattuna hakusanaan ”medicine reminder”. Tällä hakusanalla löytyy kymmeniä erilaisia lääkemuistuttajia, jotka tarjoavat muistutustoiminnallisuuden lisäksi myös jäljellä olevien lääkkeiden määrän seurantaa sekä muita mahdollisia terveysseurantatoimintoja, kuten verenpaineen tai painon seurantaa (Google Play kauppa 2021b). Heuristista analyysia varten valikoin näistä hakutuloksista 2 ilmaista sovellusta, joista löytyi suomenkieliset versiot ja joilla oli eniten latausmääriä sekä positiivisia arvioita. Arvioitavat sovellukset ovat:

- MyTherapy: Lääkemuistuttaja ja pilleriseuranta (Google Play kauppa 2021c)
- Medisafe: Lääkitys ja Pilleri Muistutus (Google Play kauppa 2021d)

Suomalaiselta UPharma Care -yritys julkaisi maaliskuussa 2021 Google Play -kauppaan oman sovelluksensa, jonka avulla voi hallita rokotus-, lääkitys- ja muita terveystietoja sekä asettaa lääkkeenottomuistutuksia (Google Play kauppa 2021e). Sovellus on maksullinen, mutta sitä pystyy testaamaan ilmaiseksi 30 päivän ajan (UPharma Care 2021a). Sovelluksessa on myös varsinainen lääkityslistanäkymä. Otin myös tämän sovelluksen mukaan saavutettavuusanalyysiin, jotta sain kotimaisen ja tämän opinnäytetyöprojektin aikana ilmestyneen uuden sovelluksen vertailukohteeksi.

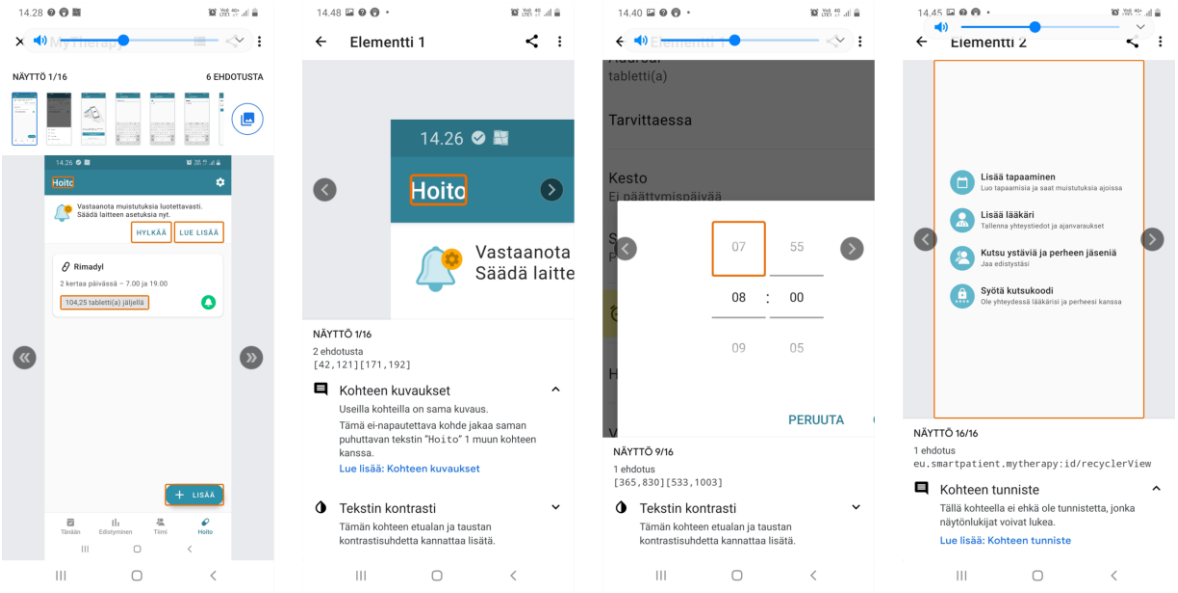
4.2 Lääkityssovellusten saavutettavuusanalyysin suoritus

Analysoin sovellukset hyödyntäen Accessibility Scanner -sovellusta, WCAG-kriteerejä, TalkBack -ruudunlukijaa sekä kytkintoiminnallisuutta. Analyysissa ei ollut tarkoitus käydä läpi koko WCAG-kriteerilistaa, vaan valikoidusti ja pistokokeenomaisesti erityisesti mobiilisovellusten kannalta olennaisia kriteerejä, joita olen tunnistanut tämän työn edellisessä osassa. Tärkeimmät mobiilisovelluksiin liittyvät WCAG-kriteerit, jotka valikoin analyysia varten, löytyvät liitteestä 1. Opettelin käyttämään TalkBack-ruudunlukijaa, ja asetin älypuhelimeni saavutettavuusasetuksissa äänenvoimakkuuden säätöpainikkeet Switch Access -kytkinpainikkeiksi. Asensin TalkBack-ruudunlukijaan suomenkielisen kielipaketin. Laadin kaikille sovelluksille erilliset analyysiraportit, jotka löytyvät liitteistä 2–4.

4.2.1 Lääkitysmuistuttaja ja pilleriseuranta (MyTherapy)

MyTherapyn lääkitysmuistuttajan päätoiminnallisuuksiin kuuluvat lääkkeiden ja niiden hälytysaikojen lisääminen, muistutusten antaminen käyttäjälle sekä lääkkeenoton seurantatietojen näyttäminen. Lääkitysmuistuttajan muistutustoiminnallisuus tukee useita erilaisia annostusvaihtoehtoja ja erikoisannosteluja. Sovelluksen avulla pystyy myös tallentamaan erilaisia terveydentilaan liittyviä mittausarvoja sekä jakamaan näitä tuloksia muille. Lisäksi lääkitysmuistuttaja seuraa jäljellä olevaa lääkkeiden määrää reseptin uusimistarvetta varten. (Google Play kauppa 2021c.) Lääkitysmuistuttajan Hoito-näkymä toimii myös eräänlaisena lääkityslistana.

Accessibility Scanner -työkalu antoi paljon huomautuksia huonoista kontrastisuhteista yläpalkin sekä painikkeiden etualan ja taustan välillä. Monessa painikkeessa oli myös liian pienikosketusalue. Lisäksi yhdestä näkymästä puuttui tunniste, mikä vaikeuttaa näkymän tarkoituksen hahmottamista. Esimerkkiruudunkaappauksia skannerin tuloksista esitellään kuvassa 5. Sovellus ei taipunut tekstin koon suurentamiseen järjestelmäasetuksista, sillä osa teksteistä oli katkaistu kolmella pisteellä. Lääkkeiden ajanottokohdan määrittäminen pyyhkäisyeleillä oli haastavaa pyyhkäisyalueen pienen koon takia. Sovellusta pystyi käyttämään sekä vaaka- että pystytasossa, eikä sisältöä tarvinnut vierittää kahdessa tasossa.



Kuva 5. Ruudunkaappauksia MyTherapyn sovelluksen Accessibility Scanner-analyysista

Sovelluksen käyttö TalkBackin kanssa oli pääosin sujuvaa, mutta sovelluksessa oli puutteita kuvien kohteiden ja yhden valikkokohteen kuvauksien suhteen. Näkymässä, jossa näytettiin päivän aikana suoritetut tehtävät, tehtävien lukumäärä oli ilmoitettu numeron sijaan kuvalla, joka kuvasi numeroa. Kuvalla ei kuitenkaan ollut tekstisisältöä, joten näkövammainen käyttäjä ei olisi saanut tietoa kuvan tarkoituksesta. Samasta näytöstä päädyttiin myös täysin epäolennaiseen kuvanäyttöön, jonka merkitys ei myöskään avautunut näkövammaiselle käyttäjälle. Ruudunlukija oli hetkittäin hieman jankkaava kuvaillessaan lääkkeitä listalla, mutta pääosin ilmoitukset olivat ytimekkäitä.

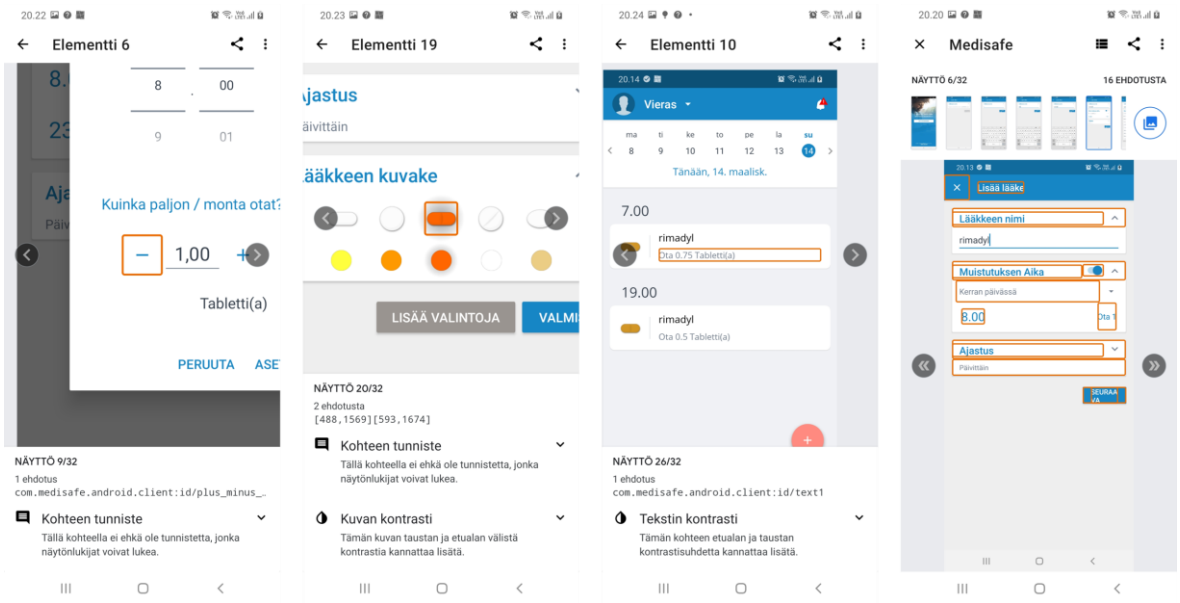
Sovellusta pystyi käyttämään kytkimen avulla asianmukaisesti; vain elementeille, joiden kanssa voi olla vuorovaikutuksessa, annetaan korostus. Suomalaisen käyttäjän kannalta on harmillista, että lomakkeiden automaattinen täyttö edellyttää lääketietokannan valitsemista, ja tällä hetkellä sovelluksessa ei pysty valitsemaan suomalaista lääketietokantaa.

4.2.2 Lääkitys ja Pilleri Muistutus (Medisafe)

Medisafen sovelluksen ominaisuuksiin kuuluvat lääkkeiden muistutukset, jotka tukevat myös erikoisannostuksia, muiden terveysmittausten tallentaminen, erilaiset seurantaraportit, mahdollisuus lähettää ystäville ja perheenjäsenille samoja muistutuksia kuin itselleen, sekä jäljellä olevien lääkkeiden määrän seuranta. Lisäksi sovellukseen on lisätty vinkkivideoita lääkityksen ottoon sekä tuki Android Wear -älykelloa varten. (Google Play kauppa 2021d.)

Accessibility Scanner -työkalun mukaan sovelluksella on todella paljon ongelmia kontrastisuhteiden kanssa. Lisäksi yllättävän monelta käyttöliittymäelementiltä puuttuu tunniste.

Kosketusalueet ovat myös pieniä. Kuvassa 6 on ruudunkaappauskuvia skannerityökalun analyysistä. Tekstin suurentaminen poisti sanoista kirjaimia painikkeissa. Sovellus toimii sekä pysty- että vaakasuunnassa, ja osoitineleet voi peruuttaa siirtämällä sormen kohdistuksen pois.



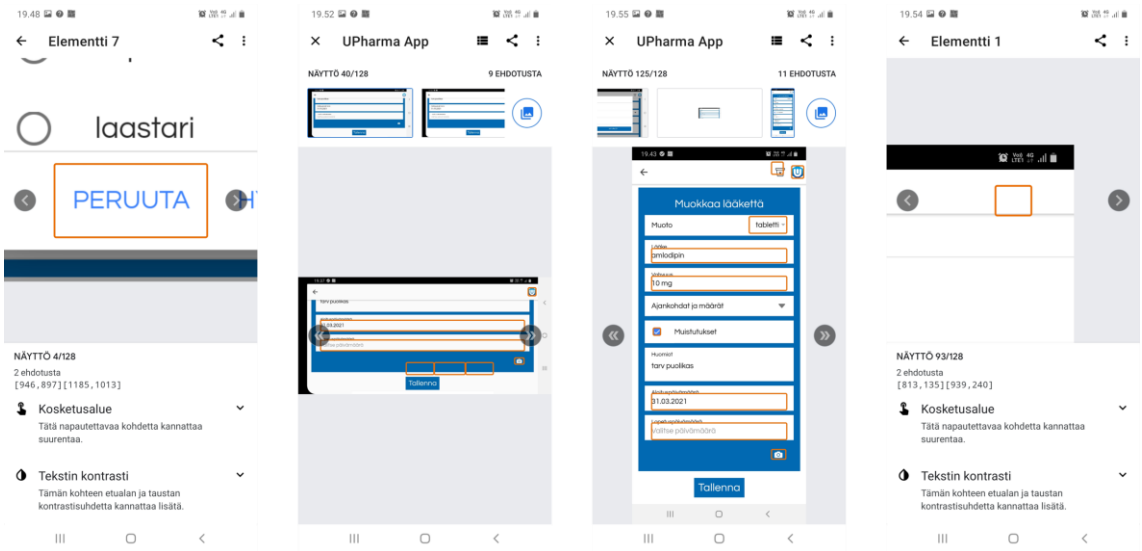
Kuva 6. Ruudunkaappauksia Medisafen sovelluksen Accessibility Scanner-analyysista

Tätä sovellusta oli paljon haastavampaa käyttää ruudunlukijan avulla. Todella monesta käyttöliittymäelementistä puuttuivat kuvaukset, mikä teki joistakin toiminnallisuuksista täysin mahdottomia näkövammaiselle käyttäjälle. Lisäksi lääkityksen muokkauselementin nimi oli kuvattu väärin, sillä ruudunlukija ilmoitti painikkeen kohdalla: ”tallenna”. Sovelluksen antamia ilmoituksia jäi lukematta. Sovelluksen tekstit myös vaihtuivat yllättäen paikoittain englanniksi, ja automaattinen kentän täyttö toimi vain amerikkalaisilla kauppanimillä. Kytkintä käytettäessä eräs näkymäelementti sai tarpeettoman kohdistuksen.

4.2.3 UPharma App

UPharma App -sovelluksen lääkityslistan ulkonäkö muistutti eniten Suomessa käytössä olevia lääkityslistoja. Sovelluksen kehittäjät ovat tunnistaneet tarpeen digitaaliselle ja helppokäyttöiselle lääkityslistalle, ja haluavat omalla ratkaisullaan lisätä lääkitysturvallisuutta ja helpottaa ajankohtaisten lääkitystietojen ylläpitämistä (UPharma Care 2021b). Ikävä kyllä sovellus ei ollut alkuunkaan saavutettava. Sitä ei pystynyt käyttämään kytkimen kanssa, ja sovelluksen kieliasetukset oli määritetty väärin, sillä ruudunlukija käytti englantia ja suomea sekaisin. Suomenkieliset tekstit luettiin englannin kielen mukaisella ääntämyksellä, ohjeet luettiin suomeksi ja osa elementtien nimistä (”camera”, ”people”, ”caret down”) ilmoitettiin sujuvalla englannin kielellä. Kieliasetusten vaihtaminen sovelluksessa ei auttanut, ainoastaan tekstit vaihtuivat.

Tässäkin sovelluksessa oli ongelmia kosketuspinta-alan koon ja kontrastien suhteen, mutta ne tuntuivat pieniltä verrattuna muihin teknisiin ongelmiin. Kuvassa 7 esitellään Accessibility Scanner -sovelluksen analyysin tuloksia. Osalla sovelluksen painikkeista ei ollut lainkaan nimiä, joten näkövammainen käyttäjä ei olisi pystynyt hyödyntämään sovelluksen päätoiminnallisuuksia. Fonttikoon suurentaminen sai aikaan joidenkin painikkeiden nimien peittymisen kuvien alle (kuva 8). Sovelluksessa oli myös muita teknisiä ongelmia, kuten ylimääräisiä näkymättömiä painikkeita. Esimerkiksi Tallenna-painike aktivoitui ruudun yläreunassa kuljettaessa sormea tyhjän kohdan yli ruudunlukijan ollessa päällä. Tietyissä näkymissä sovellus kääntyi ja lukittui yllättäen vaakatasoon, mikä haittaa käyttäjiä, joiden puhelin on kiinnitetty esimerkiksi pyörätuoliin kiinteästi. Sovellusta ei voi alkaa käyttää ilman, että luo aluksi tilin. Koska tietojen syöttäminen on täysin mahdotonta pelkän kytkintoiminnallisuuden varassa, osa käyttäjistä ei pääse tästä vaiheesta eteenpäin.



Kuva 7. Ruudunkaappauksia UPharma Caren sovelluksen Accessibility Scanner-analyysista

Kuva 8. Fonttikoon suurentamisen vaikutus UPharma Caren sovelluksessa

Kaikissa analysoiduissa sovelluksissa oli paljon teknisiä saavutettavuuspuutteita, kuten elementtien kontrastisuhteisiin liittyviä ongelmia, jotka pystyttäisiin korjaamaan suhteellisen vähällä vaivalla. MyTherapyn sovelluksen elementit oli järjestetty loogisesti, ja ainut selkeästi ongelmallinen elementti on lääkityksen annostuksen ajankohdan valitseminen. Medisafen sovelluksessa vastaava elementti oli toteutettu hieman paremmin, mutta sovelluksen käyttöliittymäelementtien satunnaisesti puuttuvat kuvaukset olivat suorastaan hämmentäviä ja yllättäviä. UPharma App hävisi kilpailijoilleen saavutettavuusasioissa selkeästi, sillä sitä ei pystynyt käyttämään lainkaan kytkimellä, ruudunlukijan kieliasetukset olivat väärin määritelty ja sovelluksen teknisessä toteutuksessa oli paljon muitakin virheitä, jotka haittasivat sovelluksen käyttämistä avustavilla teknologioilla.

5 Sovelluksen toteuttaminen

Kehitysympäristönä käytin Android Studio -ohjelmointiympäristöä, käännösautomaatiosovelluksena Gradlea, versionhallintaohjelmana hyödynsin Git:iä ja julkaisin lähdekoodin GitHub:iin. Gradle on vapaan lähdekoodin käännösautomaatiotyökalu, jota Android Studio tukee ja jolla voidaan kääntää myös Kotlin-ohjelmia (Gradle 2021). Git on hajautettu versionhallintajärjestelmä, joka soveltuu sekä pienille ja suurille projekteille ja jonka avulla voidaan hyödyntää paikallisia versiohaaroja ja erilaisia työnkulkumalleja (Git SCM 2021). Tähän projektiin valitsin Gitflow-haarautumismalliin perustuvan työnkulun, jossa hyödynnetään master-haaraa julkaisuversioille ja development-haaraa eri kehityshaarojen yhteensovittamista varten (Atlassian 2021). GitHub on verkkosivusto, jonne voi tallentaa omia Git-projekteja (GitHub 2021). Testasin sovellusta sekä oman Android-puhelimeni (Samsung Galaxy A50, Android versio 10) että Android Studion emulaattorin avulla. Valitsin sovelluksen minimi-SDK-versioksi version 19 ja käännetyksi versioksi version 30. Projektissa käytetty Gradle-versio on 4.0.1 ja Kotlin-versio 1.4.31.

5.1 Scrum-viitekehyksen hyödyntäminen

Sovelsin tässä projektissa Scrum-kehystä. Scrum on projektinhallinnan viitekehys, jonka avulla pyritään kehittämään asiakkaille eniten arvoa tuottavia ratkaisuja lyhyissä aikaväleissä sekä tarkastelemaan ja parantamaan tuotosta iteratiivisesti. Scrum on tarkoitettu hyödynnettäväksi pienessä tiimissä, ja kehys määrittelee eri roolit tiimin jäsenille. Scrum-tiimissä tuoteomistaja purkaa tehtävän työn tuotteen kehitysjonoksi, Scrum Master huolehtii Scrumin sääntöjen noudattamisesta sekä toimii tiimiä tukevassa roolissa, ja kehittäjät toteuttavat tuotteen kehitysjonon työt. Koska kehitin sovellusta yksin, en pystynyt noudattamaan Scrumia määritelmän mukaisesti ja jouduin soveltamaan rooleja. Hyödynsin kehyksen teorian mukaisia toistuvia iteraatiosyklejä eli sprinttejä, suunnittelu-, katselmus- sekä retrospektiivitapahtumia ja työn pilkkomista tuotteen kehitysjonoksi. Sprintti on tietynmittainen ajanjakso, jonka aikana toteutetaan tuotteen kehitysjonolta poimittuja töitä, ja sprintin tavoitteena on saada valmiita ominaisuuksia arvioitavaksi. Tuotteen kehitysjono on puolestaan priorisoitu lista tehtävästä työstä, jonka kärjessä ovat tärkeimmät ja eniten arvoa tuottavat toiminnallisuudet. (Schwaber & Sutherland 2020.)

Sprintin pituudeksi valitsin 3 viikkoa. Ennen sprinttiä suunnittelin ja pilkoin osiin sprintin aikana tehtävän työn, sekä arvioin ja priorisoin tarvittaessa uudelleen tuotteen kehitysjonoa. Esimerkiksi ensimmäiselle sprintille valitsin toteutettavaksi käyttäjätarinaksi lääkkeiden näyttämisen ja lisäämisen, koska muiden ominaisuuksien toteuttaminen riippui tämän käyttäjätarinan valmistumisesta. Pyrin toteuttamaan käyttäjätarinat tärkeysjärjestyksessä, mutta jouduin käytännössä joustamaan tästä ja tekemään ominaisuuksia niin sanotusti

osaamisjärjestyksessä. Esimerkiksi muistutusten lähettäminen ja niiden ajastaminen vaati minulta asian kypsyttelyä sekä opiskelua. Toteutin tämän toiminnallisuuden viimeisenä, vaikka olisin halunnut koestaa ominaisuuden paljon aikaisemmin. Teknisistä ja ajankäytöllisistä syistä jouduin myös jättämään pois ominaisuuksia, kuten erikoisannostusten valitsemisen, lopullisesta versiosta. Tuotteen kehitysjonon viimeinen versio on kuvattu liitteessä 5. Taulukossa 1 on esimerkki kahdesta käyttäjätarinasta tuotteen kehitysjonossa. Taulukko 2 puolestaan kuvaa sprintin kehitysjonoa, jossa tarinat on pilkottu osiin tehtäviksi toteutusta varten. Valitsin käyttäjätarinamuodon, koska koin, että sain tällä tavalla pilkottua sovelluksen toteutuksen järkeviin ja hyödyllisiin kokonaisuuksiin käyttäjän näkökulmasta.

Tässä työssä lääkkeellä tarkoitetaan käyttäjän käyttämää lääke-, ravintolisä- tai muuta vastaavaa valmistetta, jonka käyttäjä haluaa näkyvän lääkityslistallaan. Annostus on käyttäjän lääkkeen käyttöohje. Annoksella tarkoitetaan kerta-annosta, johon kuuluu otettavan lääkkeen määrä sekä ottoaika (esimerkiksi 2 annospussia klo 20.30). Valitsin tällaisen määrittelyn lääkkeen annoksille, jotta käyttäjä voi valita joustavasti eri määriä eri ottoaikoina.

Taulukko 1. Esimerkki käyttäjätarinoista tuotteen kehitysjonossa

Käyttäjätarina	Hyväksymiskriteeri
Käyttäjä voi muokata lääkkeen tietoja.	Lääkkeen muokkausnäkymän kenttiin voi syöttää uudet arvot. Näkymässä on painike, jota painamalla uudet tiedot tallentuvat tietokantaan ja päivittyvät näkymään.
Käyttäjä voi muokata annoksen tietoja.	Annostuksen muokkausnäkymässä voi vaihtaa määrän ja kellonajan. Edelliset arvot näkyvät. Tallennuspainike tallentaa tiedot tietokantaan ja siirrytään takaisin lääkkeen muokkausnäkymään.

Taulukko 2. Esimerkki sprintin kehitysjonosta

Käyttäjä voi muokata lääkkeen tietoja	Muokkausnäkymässä lääkkeen tiedot näkyvät EditText-kentissä, joissa niitä voi muokata
	Selkeämpi tyyli sille, että listalta pääsee muokkausnäkymään
	Painike, jota painamalla lääkkeen muokatut tiedot tallentuvat
	Tarkistukset muokkauksille (ettei tule tyhjiä arvoja)
	Turha roskakorin kuva pois
	Lisää NestedScrollView

Jokaisen sprintin jälkeen tein sprinttikatselmuksen, jossa tarkastelin aikaansaatuja toiminnallisuuksia ja suoritin saavutettavuusanalyysin sovellukselle. Aina kun pystyin, yritin kysyä myös ulkopuolisilta mielipidettä sovelluksesta, mutta pääosin arviointini oli omien havaintojen, testien ja tarkistuslistani varassa. Sprinttikatselmuksien avulla pystyin arvioimaan valmistuneiden ominaisuuksien lisäksi niiden saavutettavuuteen liittyviä mahdollisia

ongelmia ja parannuksia. Hyödynsin myös sprintin retrospektiivitapahtumaa, jonka avulla pystyin analysoimaan edistymistäni sekä pyrkimään parantamaan työskentelyäni. Kirjoitin itselleni ylös, mikä meni hyvin, mitä voisi parantaa ja mihin parannusehdotukseen sitoudun seuraavassa sprintissä (Scrum 2021).

Ensimmäisen sprintin aikana keskityin vielä uuden teknologian opiskeluun ja tutkin, miten vaadittavat ominaisuudet kannattaisi toteuttaa. Alussa haastetta aiheutti esimerkiksi oikeiden kirjastojen löytäminen ja niiden versioiden yhteensopivuus valitun minimi-SDK-version kanssa. Hain Google Android Developers -sivuston dokumentaatiosta tietoa viimeisimmistä versioista, sekä otin mallia Udacityn Developing Android Apps -verkkokurssin esimerkkiprojektien build.gradle -tiedostoista (Udacity 2021).

Olin iloinen, että sain aikaan toimivan tuotoksen jo ensimmäisellä sprintin jälkeen, vaikka en saanutkaan kaikkia käyttäjätarinoita toteutettua, joiden tekemiseen olin sitoutunut. Minulla meni alkuvaiheessa paljon aikaa itselleni vieraaseen teknologiaan tutustumiseen, mutta selvisin pienistä teknisistä haasteista sitkeällä tutkimus- ja tiedonhakutyöllä. Hain tietoa ensi sijassa Androidin virallisesta dokumentaatiosta, mutta jouduin hakemaan paljon tietoa myös hakukoneen avulla erilaisilta verkkofoorumeilta sekä hyödynsin paljon edellä mainittua Udacityn Developing Android Apps -verkkokurssia (Udacity 2021). Lisäksi osasin ajoissa rajata ja karsia tekemistäni, jotta sain valmiiksi ominaisuuksia, joita voi kokeilla ja analysoida. Kehitysympäristöni tuki työtäni ja helpotti saavutettavuuden tarkastelua esimerkiksi antamalla ilmoituksia puuttuvista sisältökuvauksista. Toisaalta huomioni oli enemmän teknologian opettelussa, ja saavutettavuuteen liittyvät seikat eivät korostuneet kehitystyön alussa.

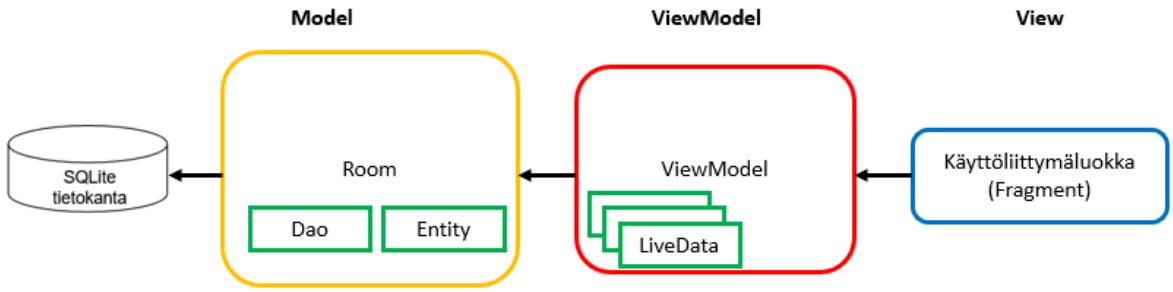
Seuraavan sprintin aikana en saanut toteutettua ainuttakaan käyttäjätarinaa valmiiksi teknisten vaikeuksien takia. Uusia asioita opetellessa on vaikeaa arvioida, kuinka paljon aikaa jonkin asian toteuttamiseen menee. Minulla meni yllättävän paljon aikaa esimerkiksi sen selvittämiseen, miten Room-kirjaston avulla tulisi käsitellä yhden suhde moneen -tietokantasuhdetta ja siihen liittyviä operaatioita. Lisäksi uudet käyttöliittymäelementit, kuten RecyclerView tuottivat ongelmia. Tämän sprintin aikana en löytänyt Androidin dokumentaatiosta kaikkea tarvitsemaa informaatiota ja jouduin etsimään apua muista lähteistä, kuten stackoverflow.com-sivustolta ja medium.com-sivuston blogeista. Jälkeen päin ajateltuna ongelmana oli, etten ollut kunnolla sisäistänyt tiiviistä dokumentaatiosta hätäisesti lukemiani asioita. Minun olisi pitänyt varata enemmän aikaa opiskeluun ja pilkkoa käyttäjätarinoita tarvittaessa pienemmiksi. Teknisten ongelmien takia en juurikaan ehtinyt pohtia saavutettavuuteen liittyviä asioita.

Sprinttien kuluessa ja ymmärryksen kasvaessa opin kuitenkin pilkkomaan käyttäjätarinoita ja toteutettavia toiminnallisuuksia järkevimmiksi kokonaisuuksiksi. Lisäksi nostin myöhempien sprinttien tehtävälistoille myös eri aihepiirejä, joihin tuli perehtyä, jotta pystyin huomioimaan paremmin uusien asioiden opiskeluun kuluvan ajan. Projektin loppupuolella koin vakavaksi puutteeksi, että en panostanut käyttöliittymätestaukseen riittävästi saavutettavuutta lukuun ottamatta. Myös yksikkötestejä olisi pitänyt olla enemmän. Minun olisi ehdottomasti pitänyt ottaa testit osaksi kriteerejä, joilla käyttäjätarina todetaan valmiiksi. Koin toisaalta, että sovelluksen ensimmäiset versiot olivat enemmän teknisiä harjoituksia ja idean toteuttamiskelpoisuutta koestavia kokonaisuuksia, jolloin testit olivat pienemmässä roolissa.

Saavutettavuustarkistuslistan avulla pystyin puuttumaan sprinttikatselmuksen puitteissa nopeasti ilmaantuneisiin ongelmiin. Opin myös paremmin testaamaan ja huomioimaan saavutettavuusasioita projektin aikana. Viimeisille sprinteille kasautui kuitenkin todella monta käytettävyyskorjausta. Tämä aiheutui siitä, että minulla meni enemmän aikaa muistutustoiminnallisuuden toteuttamiseen kuin olin arvioinut. Korona-aikana oli myös haastavaa saada testikäyttäjiä kokeilemaan sovellustani, mikä olisi auttanut minua huomaamaan käytettävyysongelmat aiemmin. Mahdollisuuksien mukaan pyysin kuitenkin muita henkilöitä arvioimaan sovelluksen toimintaa ja ulkonäköä. Sain arvokasta palautetta liian täyteen ahdetuista näkymistä, muistutuksen torkutusmahdollisuuden puuttumisesta sekä sovelluksen yläpalkin hyödyntämisestä navigaatiossa ja näkymien tunnistamisessa.

5.2 Sovelluksen rakenne

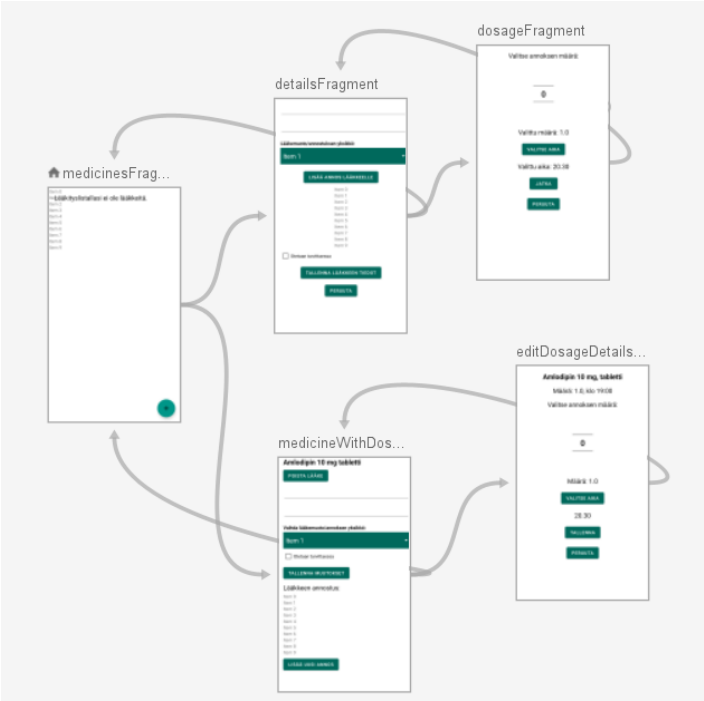
Sovelluksessa on pyritty noudattamaan MVVM-arkkitehtuuria. Tärkeimmät luokat on esitetty kuvassa 9. Sovelluksen Kotlin-paketit on järjestetty sovelluksen ominaisuuksien, kuten muistutuksen, lisäys-, ja muokkausnäkymän sekä lääkityslistanäkymän, mukaan.



Kuva 9. Sovelluksen arkkitehtuuri (mukaillen Google Android Developers 2021i)

5.2.1 Käyttöliittymä (View)

Sovellus koostuu yhdestä aktiviteetista ja siihen kuuluvista viidestä fragmentista. Fragmenttien (Fragment) avulla sovelluksen käyttöliittymä voidaan jakaa useisiin ja tarvittaessa uudelleenkäytettäviin osiin. Fragmentti on usein esimerkiksi yksi näkymä sovelluksessa. Fragmenteilla on omat XML-asetteluresurssitiedostonsa layout-resurssikansiossa ja elinkaarensa. Ne hallinnoivat omia syötetapahtumiaan, mutta eivät ole itsenäisiä kokonaisuuksia, vaan niiden pitää olla osana aktiviteetin tai toisen fragmentin näkymähierarkiaa. (Google Android Developers 2021j.) Siirtymät fragmentista toiseen määritellään navigaatiokaavioresurssitiedostossa. Lisäksi sovelluksen pääaktiviteetin activity_main.xml-resurssitiedostoon pitää lisätä navigation host -säilö, jonka sisällä sovelluksen navigaatiopäätepisteet esitetään. (Google Android Developers 2021k.) Kuvassa 10 on esitettynä sovelluksen fragmentit navigaatioeditorin design-näkymässä.



Kuva 10. Sovelluksen fragmentit navigaatioeditorin näkymässä

Esimerkkejä sovelluksen näkymistä ja toiminnoista löytyy liitteestä 6. Pääfragmentti näyttää lääkkeet listana. Lisäsin näkymään Google Material -kirjaston FAB-elementin (Floating Action Button), jonka edustaa näkymän päätoiminnallisuutta eli lääkkeen lisäämistä listaan (Google Material Design 2021c). FAB-painiketta napauttamalla päästään uuden lääkkeen tallennus -näkymään. Sitä vastaavassa fragmentissa on syöttökenttiä ja valintaruutuja lääkkeiden tietoja varten sekä lääkkeen tietojen tallennuspainike. Tallentamisen jälkeen käyttäjä ohjataan navigaatiolla lääkityslistanäkymään, jonne uusien lääkkeiden tiedot ilmestyvät. Päälistalta pääsee klikkaamalla lääkkeen korttia muokkausnäkymään, jossa voi poistaa ja muokata lääkkeen tietoja. Lääkkeen lisäys- ja muokkausnäkymistä pääsee

annosten lisäys- tai vastaavasti muokkausnäkymään. Näissä fragmenteissa hyödynnetään NumberPicker- ja TimePicker-elementtejä (Google Material Design 2021d; Google Android Developers 2021l).

Käytin RecyclerView-käyttöliittymäelementtiä lääkkeiden ja annosten näyttämisessä listalla. RecyclerView-elementtiä hyödynnetään erityisesti suuren datamäärän esittämisessä, ja se parantaa sovelluksen suorituskykyä luomalla tarvittaessa dynaamisesti näkymiä sekä kierrättämällä niitä uusien ruudulle ilmaantuvien elementtien näyttämiseen. (Google Android Developers 2021m.) Kyseinen elementti voi aiheuttaa saavutettavuusongelmia, jos sen yhteydessä ei määritellä RecyclerView:n esittämille toistuville elementeille yksilöllisiä tunnisteita. Itse unohdin antaa annoksien poistopainikkeille yksilöivät kuvaukset contentDescription-attribuutin avulla, mistä seurasi, että ruudunlukijakäyttäjä ei olisi tiennyt, mitä annosta on poistamassa.

Päätin hyödyntää Spinner-käyttöliittymäelementtia siihen, että käyttäjä voi valita pudotusvalikon ennalta määrätyistä vaihtoehdoista oikean lääkemuodon (Google Android Developers 2021n). Mitä vähemmän käyttäjä joutuu syöttämään tekstiä sovellukseen, sen helpompi sovellusta on käyttää. Mietin pitkään myös erilaisia valikkovaihtoehtoja lääkkeen vahvuuden valinnan helpottamiseen, mutta lopulta päädyin siihen, että joustavuuden kannalta sovelluksen tässä versiossa on selkeämpi, että käyttäjä määrittelee itse vahvuuden haluamallaan tavalla. Käyttöliittymässä tuli kiinnittää erityisesti huomiota siihen, miten elementtien tila vaihtuu esimerkiksi orientaation vaihtuessa. Äärimmäisin esimerkki tästä oli ensimmäinen versioni Spinner-valikosta, joka kaatoi sovelluksen ”Null Pointer Exception” -virheeseen kääntäessäni mobiililaitteen vaakatasoon. En ollut huomannut tehdä tarkistusta sille, oliko valikko saanut tiedon valitusta valikkovaihtoehdosta.

5.2.2 Datan käsittely käyttöliittymää varten (ViewModel)

Käyttöliittymässä fragmentti vastaa siihen liittyvän datan näyttämisestä, fragmenttia vastaava asetteluresurssitiedosto määrittelee fragmentin ulkoasun, mutta MVVM-arkkitehtuurimallin mukaisesti tarvitaan myös fragmenttia vastaava ViewModel-luokka, joka hallinnoi dataa fragmenttia varten. Jos dataa pidettäisiin vain Fragment-luokassa, se saattaa hävitä konfiguraatiomuutoksissa, kuten vaihdettaessa puhelimen orientaatiota. (Google Android Developers 2021o.) Datan sitomiskirjaston (Data binding) avulla helpotetaan UI-komponenttien ja sovelluksen datalähteiden kommunikaatiota kirjaston automaattisesti luomien luokkien avulla (Google Android Developers 2021p).

Käytin ViewModel-luokkieni kanssa Factory-luokkia, joiden avulla voin tehdä oman konstruktorin ViewModel-luokan ilmentymän luomista varten ja antaa tietokannan sekä sovelluksen globaalin tilan parametreina luonnin yhteydessä (Google Android Developers 2021q; Google Android Developers 2021r). Kuva 11 esittää MedicinesViewModel-luokan luonnissa käytetyn Factory-luokan.

```
class MedicinesViewModelFactory(
        private val dataSource: MedicineDao,
        private val application: Application) : ViewModelProvider.Factory {
    @Suppress( ...names: "unchecked_cast")
    override fun <T : ViewModel?> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(MedicinesViewModel::class.java)) {
            return MedicinesViewModel(dataSource, application) as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}
```

Kuva 11. Esimerkki ViewModelFactory-luokasta

Käytin lääkkeiden tietojen syötössä kahdensuuntaista datansitomista, jotta saan päivitettyä myös käyttöliittymässä tapahtuvat muutokset ViewModel-luokan muuttujiin. Tämän mahdollistaa kuvassa 12 näkyvä ”@={}”-merkintä, jonka avulla komponentti pystyy vastaanottamaan sekä datan että käyttäjän syötteestä tulevia muutoksia. Yhdensuuntaisessa datansidonnassa merkintä on muuten sama, mutta siitä puuttuu yhtäsuuruusmerkki. (Google Android Developers 2021s.)

```
<com.google.android.material.checkbox.MaterialCheckBox
    android:id="@+id/edit_check_if_taken_when_needed"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:checked="@={detailsViewModel.onlyWhenNeeded}"
    android:text="@string/taken_when_needed"
```

Kuva 12. Esimerkki kahdensuuntaisesta datansitomisesta ”@={}” -merkinnän avulla MaterialCheck-Box-käyttöliittymäelementissä

ViewModel-luokissa hyödynnettiin myös LiveData-luokkaa, jotta komponentit päivittyisivät vain, kun ne ovat aktiivisessa tilassa. LiveData-luokan avulla vältetään muistivuotoa, sovelluksen kaatumista sekä helpotetaan elinkaaren ja konfiguraatiomuutosten hallintaa. LiveData-muuttujia varten luodaan aktiviteetti- tai fragmenttiluokissa Observer-olio, jolle on annettu parametrina aktiviteetin- tai fragmentin LifecycleOwner-olio. Kun LiveData-objek-

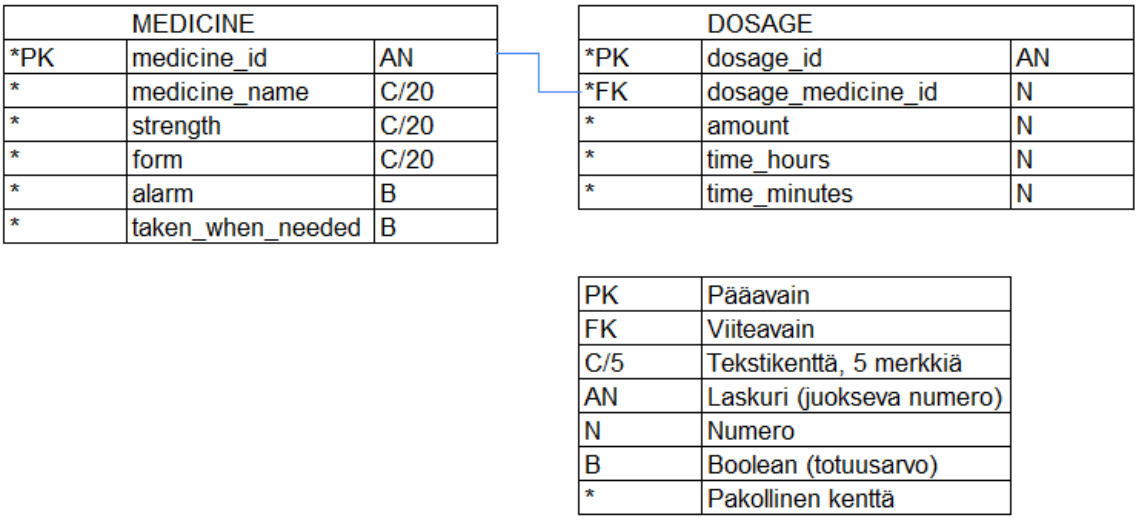
tiin säilytetyn muuttujan arvo muuttuu, siihen liitetyt Observer-oliot saavat tiedon muutoksesta, jos niiden LifecycleOwner on aktiivisessa tilassa. (Google Android Developers 2021t.) Kuvassa 13 on esimerkki LiveData-luokan käytöstä DetailsViewModel- ja DetailsFragment-luokkien välillä. Jos esimerkin _navigateToMedicines-muuttujan arvo muuttuu todeksi, Fragment-luokka saa tästä tiedon ja käynnistää navigaatiotapahtuman. Lopuksi Fragment-luokka palauttaa muuttujan arvon kutsumalla ViewModel-luokan finishedNavigating()-metodia.

```
//DetailsViewModel.kt
class DetailsViewModel(
        val database: MedicineDao, application: Application) : AndroidViewModel(application) {

    private val _navigateToMedicines = MutableLiveData<Boolean?>()
    val navigateToMedicines: LiveData<Boolean?>
        get() = _navigateToMedicines

//DetailsFragment.kt
detailsViewModel.navigateToMedicines.observe(viewLifecycleOwner, Observer { it: Boolean?
    if (it == true) { // Observed state = true
        this.findNavController().navigate(
                R.id.action_detailsFragment_to_medicinesFragment)
        detailsViewModel.finishedNavigating()
        detailsViewModel.clearDosageList()
    }
})
```

Kuva 13. Esimerkki LiveData-luokan käytöstä DetailsViewModel- ja DetailsFragment-luokissa

5.2.3 Tietokanta ja Room-kirjaston hyödyntäminen (Model)

Päätin aluksi mennä kahden taulun toteutuksella, sillä pyrin pitämään sovelluksen yksinkertaisempana alkuvaiheessa. Lisäksi monimutkaisempia suhteita tarvitaan vasta siinä vaiheessa, kun toteutetaan erityisannostuksia, joten Scrumiin kuuluvan Lean-ajattelun hengessä vähensin hukkaa keskittymällä olennaiseen (Schwaber & Sutherland 2020). Tietokannan malli on esitetty kuvassa 14.



Kuva 14. Sovelluksen tietokannan malli

Hyödynsin Room-kirjastoa datan tallentamiseen älypuhelimen paikalliseen SQLite-tietokantaan. Kirjastoa varten loin tietokantaluokan, entiteettiluokat, jotka edustavat tietokannan tauluun tallennettavia objekteja, sekä DAO-luokan, jonka avulla voidaan hakea ja muokata tietokannassa olevaa dataa. (Google Android Developers 2021u.) Tein entiteettiluokat Dosage- ja Medicine-objekteille, ja määrittelin niiden välisen yhden suhde moneen -relaation MedicineWithDosages-luokan avulla Room-kirjaston @Embedded ja @Relation annotaatioiden avulla (Google Android Developers 2021v). MedicineWithDosages-luokka esitetään kuvassa 15. Jouduin tekemään muokkauksia skeemaan, ja minun piti huomioida tämä päivittämällä tietokantaluokan versionumeroa sekä antamalla migraatiosuunnitelman (Muntenescu 2017). Tässä vaiheessa hyödynsin tietokannan rakennusluokan fallbackToDestructiveMigration()-metodia, sillä vaikka metodi hävittää sovelluksen datan päivityksen yhteydessä, sovellusta ei ole kuitenkaan vielä julkaistu, joten tässä kehitysvaiheessa ei tarvitse huolehtia sovelluksen käyttäjien vanhan datan säilyttämisestä.

```
@Parcelize
data class MedicineWithDosages (
        @Embedded val Medicine: Medicine,
        @Relation(
                parentColumn = "medicineId",
                entityColumn = "dosage_medicine_id"
        )
        val dosages: List<Dosage>
) : Parcelable
```

Kuva 15. Koodiesimerkki yhden suhde moneen -relaation määrittelystä

5.3 Lääkkeiden muistutustoiminnallisuus

Sovelluksen muistutustoiminnallisuuden toteuttamiseen käytettiin AlarmManager-luokkaa. Koska muistutuksesta haluttiin tarkka ja toistuva, hyödynsin setExactAndAllowWhileIdle()-metodia (kuva 16). Lääkemuistutuksen hälytyksen tyyppinä on RTC_WAKEUP, mikä tarkoittaa sitä, että aika on sidottu lokaaliin kellonaikaan, ja hälytys suoritetaan, vaikka näyttö olisi pois päältä. Kun hälytys laukeaa annettuna aikana, sille annettu pendingIntent-luokka aktivoituu. (Google Android Developers 2021w.)

```
alarmManager.setExactAndAllowWhileIdle(
        AlarmManager.RTC_WAKEUP,
        alarmTime,
        alarmPendingIntent
)
```

Kuva 16. AlarmManager-luokan setExactAndAllowWhileIdle-metodi

BroadCastReceiver-luokka (AlarmReceiver) vastaanottaa järjestelmän ilmoituksen (hälytyksen) ja saa tarvittavat tiedot pendingIntent-luokan avulla hälytyksen käsittelyä varten (Myrick 2019). Sovelluksen tapauksessa hälytys käynnistää notifikaation lähettämisen lääkkeen tiedoilla. Lähetyksen jälkeen seuraava hälytys ajastetaan tapahtumaan lääkkeenottoaikaan seuraavana päivänä. AlarmManager-luokalla on myös setRepeating()-metodi toistuvia hälytyksiä varten, jolle olisi voinut antaa toistovälin, mutta tämä ei ollut riittävän tarkka sovelluksen muistutustoiminnallisuuden kannalta. Jos puhelin käynnistetään uudelleen, AlarmManager hävittää kaikki olemassa olevat hälytykset. Tästä syystä lisäsin sovellukseen toisen BroadCastReceiver-luokan (BootReceiver), joka kuuntelee järjestelmän antamaa ACTION_BOOT_COMPLETED-lähetystä (kuva 17). Tämän lähetyksen saadessaan BootReceiver-luokka ajastaa hälytykset uudelleen onReceive-metodinsa avulla. Sekä AlarmReceiver että BroadcastReceiver tulee rekisteröidä AndroidManifest.xml-tiedostossa. (Google Android Developers 2021w.)

```
class BootReceiver : BroadcastReceiver() {

    override fun onReceive(context: Context, intent: Intent) {
        if (intent.action == "android.intent.action.BOOT_COMPLETED") {
            scheduleAllAlarmNotifications(context)
        }
    }
}
```

Kuva 17. BootReceiver-luokka

Versiosta 23 alkaen Android on ottanut käyttöön kaksi virransäästöön vaikuttavaa toiminnallisuutta: ”Doze” ja ”App Standby”. Nämä virransäästöominaisuudet vaikuttavat olennaisesti myös AlarmManager-luokan toimintaan. (Google Android Developers 2021x.) Omassa sovelluksessa huomasin, että jos puhelin oli pitkään käyttämättä, lääkelistasovelluksen muistutukset lakkasivat toimimasta. Androidin dokumentaatio suosittelee hyödyntämään FireBase Cloud Messaging -palvelua ja säätämään ilmoitusten prioriteetin korkeaksi, mutta päätin kuitenkin käyttää edellä mainittua setExactAndAllowWhileIdle()-metodia, jotta sovellusta voi käyttää puhelimen ollessa pois verkosta. Puhelimen virransäästöasetuksista voi myös ottaa sovellukselta pois virransäästöoptimoinnin, mutta tästä huolimatta ilmoitukset saattoivat tulla muutaman minuutin viiveellä.

5.4 Kooste sovelluksen saavutettavuusanalyyseista

Valmistuneiden uusien ominaisuuksien jälkeen testasin sovellusta tarkistuslistani sekä Accessibility Scanner -sovelluksen avulla. Sovelluksen tekemisen aikana havaitut saavutettavuusongelmat on listattu liitteessä 7. Sovelluksen alkuvaiheissa Accessibility Scanner -sovellus löysi huomautettavaa FAB-elementin kontrasteista sekä tekstin syöttökenttien kosketusalueiden koosta ja vihjetekstien huonosta kontrastisuhteesta. Ensimmäisissä versioissa myös elementtien sijoittelu aiheutti saavutettavuusongelmia. Tekstit ja painikkeet saattoivat mennä päällekkäin tai niitä joutui vierittämään esiin sekä pysty- että vaakasuunnassa asennon tai tekstikoon suurentuessa. Tämä johtui omasta kokemattomuudestani näkymän rakenteen asettelusta layout-resurssitiedostoissa.

Hyödynsin aluksi poistotoiminnallisuudessa pelkkää ImageView-käyttöliittymäelementtiä enkä esimerkiksi kuvallista painiketta. Tästä aiheutui se, että poistoa ei pystynyt peruuttamaan vierittämällä sormea pois kuvan päältä. Ruudunlukija ei myöskään lukenut tilaviestiä onnistuneesta poistosta. Tämän jälkeen päätin välttää kuvallisia pieniä painikkeita ja tehdä lisäksi painikkeista mahdollisimman selkeitä ja riittävän isokokoisia. Painikkeiden sijoittelusta aiheutui kuitenkin ongelmia. Lopulta tulin tulokseen, että käyttöliittymäelementtien sijoittelu lineaarisesti allekkain oli saavutettavuuden kannalta paras lähtökohta. Tällä sijoittelulla pystyin välttämään asioiden peittymisen tai tarpeen vierittää ruutua tekstikoon suurentuessa.

Lääkkeen annosten poistopainikkeilta puuttui yksilöivä kuvaus ruudunlukijakäyttäjälle. Korjasin sen koodissa säätämällä näiden painikkeiden contentDescription-attribuutin arvon adapterissa, joka mukauttaa annoksen tietoja näkymään (kuva 18). Korjauksen jälkeen TalkBack lukee käyttäjälle esimerkiksi seuraavan tekstin: ”Poista annos: määrä: 1; klo: 12:00”. Muussa tapauksessa jokaisella painikkeella olisi ollut ääneen luettuna teksti ”Poista”, ja käyttäjän olisi ollut hankalampi tietää, mitä annosta hän on poistamassa.

```
@BindingAdapter( ...value: "deleteButtonContentDescription")
fun Button.setContentDescriptionForDelete(item: Dosage?) {
    item?.let { it: Dosage
        val text = combineAmountAndTimes(item.amount, item.timeValueHours, item.timeValueMinutes)
        contentDescription = "Poista annos $text"
    }
}
```

Kuva 18. Painikkeen saavutettavuustekstin arvon antaminen adapterin avulla

Käyttöliittymäelementtien kuvauksissa, kuten painikkeissa ja syöttökentissä, oli siis helppo tehdä virheitä ruudunlukijakäyttäjän kannalta. Edellä mainittujen samanlaisten kuvausten lisäksi olin lisännyt tarpeettomia contentDescription-attribuutteja painikkeille, joille olin antanut kaiken lisäksi väärät arvot. Accessibility Scanner -sovellus ei olisi osannut valittaa tällaisesta virheestä, ja se tuli esille vain, kun testasin itse sovellusta TalkBack-ruudunlukijan avulla.

Spinner-elementissä jouduin korjaamaan elementin tyylimäärittelyjä, sillä kun suurensin tekstikokoa, puolet tekstistä peittyi eikä mahtunut valikon raameihin. Tämä olisi todennäköisesti jäänyt huomaamatta, jos tarkistuslistallani ei olisi ollut kohtaa tekstikoon suurentamisen testaamisesta. Spinner-elementtiin liittyen sovellukseen jäi ruudunlukijakäyttäjää mahdollisesti hämmentävä ominaisuus, jossa fragmentin vaihtuessa TalkBack lukee ääneen Spinner-elementissä olevan valitun kohteen arvon. Tämän huomasin vasta, kun poistin accessibilityPane-attribuutin, joka on tuettuna vasta versiosta 23 alkaen.

Kontrastiongelmia minulla oli erityisesti NumberPicker- ja TextEdit-käyttöliittymäelementtien kanssa. NumberPicker-elementin kanssa päätin hyväksyä sen, että valikon ulkopuolella olevien numeroiden kontrastisuhde ei ole riittävä. Itse elementin käyttö onnistui, sillä keskellä oleva numero näkyi selkeästi ja siihen pystyi napauttamaan ja syöttämään arvon ilman elementin pyöritystä. Accessibility Scanner -sovelluksen analyysi NumberPicker-elementistä on näytetty kuvassa 19. Alussa olin myös valinnut huonot värit sovellukselle. Hyödyntämällä Google Material Design -sivustolla olevaa Color Tool -työkalua sain helposti valittua värit, joiden kanssa ei tullut kontrastiongelmia (Google Material Design 2021e).

Kuva 19. Kontrastiongelma NumberPicker -käyttöliittymäelementissä

Sovelluksen alkuvaiheessa sovellus ilmoitti uuden lääkkeen tietojen syötevirheistä, mutta ei spesifioinut virheellistä kohtaa. Kokeilin TextView-luokan omia virheenkäsittelymetodeja, mutta tästä seurasi, että TalkBack luki virheilmoitukset sekakielellä englanniksi ja suomeksi. Hyödynsin lopulta TextInputLayout-luokkaa virheilmoituksen näyttämiseen, jonka jälkeen virheilmoitus luettiin moitteettomasti. Orientaation vaihtuessa tiedot virheestä kuitenkin hävisivät. Tämä johtui siitä, että olin rikkonut MVVM-periaatetta. Minun olisi kuulunut jättää virhelogiikan käsittely täysin ViewModel-luokan hallinnoitavaksi ja antaa fragmentin havainnoida virheen tilaa.

Kytkinkäytön kannalta mietin, olisiko minun pitänyt hallinnoida kohdistuksen siirtymisiä muutamissa kohdissa koodin avulla. Kaikkia toiminnallisuuksia pystyi käyttämään kytkimen avulla, mutta hetkittäin pohdin, olisiko käyttäjän kannalta ollut helpompi, jos kohdistus siirtyisi sujuvammin oikeaan kohtaan. Esimerkiksi poiston vahvistuksen yhteydessä kohdistus olisi voinut olla valmiiksi poistodialogielementissä. Koska läpikäytäviä elementtejä ei kuitenkaan ollut ruudulla mahdottoman paljon, annoin järjestelmän hoitaa kohdistuksen siirron.

6 Pohdinta

Saavutettavuusvaatimuksia eli WCAG-kriteereitä voidaan hyödyntää myös natiivissa mobiilisovelluskehityksessä, mutta WCAG-kriteereistä kannattaa huomioida erityisesti ne kriteerit, jotka koskevat mobiililaitteita (W3C 2015). Lisäksi WCAG 2.1-ohjeistukseen liittyvät oheisdokumentit, kuten kriteerien yhteydessä olevat ”How to meet” ja ”Understanding” -osiot hyödyttävät lähinnä verkkokehittäjiä. Esimerkiksi kriteeriin 1.4.10 (Responsiivisuus) ”How to meet” -osion esimerkkejä käsitellään vain verkkoteknologioiden kautta (W3C 2021b). Saavutettavuusvaatimukset sellaisenaan ovat teknologiariippumattomia, mutta tämä hankaloittaa mielestäni kriteerien sisäistämistä.

Onnistuin huomaamaan ja korjaamaan vähällä vaivannäöllä oman sovellukseni saavutettavuusongelmia tarkistuslistani ja saavutettavuustyökalujen avulla. Jos olisin jättänyt kaikki saavutettavuuskorjaukset työn loppuvaiheeseen, minulle olisi tullut kiire. Varhaisessa vaiheessa huomatut ongelmat, kuten huonosti toimivat käyttöliittymäelementit, oli myös helpompi korjata. Oman näkemykseni mukaan hyvä saavutettavuustarkistuslista Android-sovelluskehityksessä on tiivistetty ja mahdollisesti yksinkertaistettu versio WCAG 2.1 -saavutettavuusohjeistuksesta. Lista pohjautuu WCAG-kriteereihin ja esimerkiksi Googlen Android-sovellusten saavutettavuusohjeisiin yhdistellen molempia. Onnistumiskriteereitä täydentävä ohjeistus tulisi laatia mobiilisovelluskehityksen näkökulmasta, mikä helpottaisi kriteerien ymmärtämistä sekä niiden soveltamista käytännössä. Esimerkiksi kriteerin 4.1.2 (Nimi, rooli, arvo), jonka mukaan kaikkien käyttöliittymäkomponenttien nimi ja rooli täytyy olla selvitettävissä ohjelmallisesti, rinnalle voisi lisätä koodiesimerkin android:labelFor-attribuutin käytöstä EditText-elementeille. Toinen tärkeä huomio tähän kriteeriin liittyen on RecyclerView-elementin avulla muodostettujen toisteisten käyttöliittymäelementtien contentDescription-attribuutin asettaminen tarvittaessa adapterin avulla, jotta nämä toistuvat elementit ovat erotettavissa toisistaan myös avustavan teknologian avulla (Google Android Developers 2021e; W3C 2018).

Tässä opinnäytetyössä analysoitujen kaupallisten sovellusten otanta oli pieni, mutta oli kuitenkin yllättävää havainnoida niin monta saavutettavuusongelmaa jokaisessa näistä sovelluksista. Todennäköisiä syitä saavutettavuusongelmiin ovat mielestäni kiire, saavutettavuuskriteerien huono tuntemus sekä testauksen riittämättömyys. Saavutettavuus näyttäisi vieläkin olevan matalan prioriteetin lisäominaisuus, jonka toteuttamista ei nähdä tarpeellisena ilman lakisääteistä velvollisuutta. Tyypillisimpiä saavutettavuusongelmia olivat riittämättömät kontrastisuhteet, puutteelliset ja virheelliset käyttöliittymäelementtien kuvaukset sekä liian pienet kosketusalueet. Tekstikoon suurentamista ei myöskään ollut huomioitu, mikä aiheutti saavutettavuusongelmia, kuten käyttöliittymäelementtien peittymistä.

Kehittäjillä tai testaajilla saattaa olla hankaluuksia tulkita saavutettavuuskriteereitä, varsinkin jos he eivät tunne avustavaa teknologiaa tai eivät ole aiemmin käyttäneet sovelluksia sellaisten avulla. Esimerkiksi kriteerissä 1.1.1 (Ei-tekstuaalinen sisältö) vaaditaan sisällyttämään kaikkeen ei-tekstuaaliseen sisältöön tekstivastine, paitsi epäolennaiset koristekuvat ja muut vastaavat voidaan ohittaa avustavalla teknologialla (W3C 2018). Kehittäjä, joka on toteuttanut tyylikkäät kuvapainikkeet, ei välttämättä osaa ajatella, että henkilö, joka ei näe kuvan sisältöä, ei pysty ymmärtämään painikkeiden toiminnallisuutta. Tällaisessa tapauksessa painikkeilla tulee ehdottomasti olla tekstivastine ruudunlukijaa varten. Toisaalta hetkittäin voi olla hankalaa tulkita, mikä on epäolennaista kuvasisältöä. Toinen ääripää on, että kriteeriä tulkitaan yli, jolloin sovelluksissa saattaa olla tarpeettomia tekstivastineita, jotka pahimmassa tapauksessa hankaloittavat sisällön hahmottamista. Kriteerin 1.1.1 viesti on sinänsä selkeä, mutta se on monimutkaisesti ilmaistu. Kriteerien ylläpitäjä World Wide Web Consortium tiedostaa tämän ongelman, ja esimerkiksi WCAG 3.0 -vedoksessa edellä mainittu kriteeri on ilmaistu huomattavasti selkeämmin valaisevien esimerkkien kera (W3C 2021a).

Omassa toteutuksessani saavutettavuusongelmat johtuivat myös kokemattomuudesta uuden teknologian parissa. En tuntenut käyttämiäni luokkia riittävän hyvin, mistä aiheutui ongelmia saavutettavuuden sekä käytettävyyden kannalta. Vaikka olin etukäteen perehtynyt saavutettavuuskriteereihin ja Androidin omaan saavutettavuusdokumentaatioon, tein silti virheitä. Oli kuitenkin positiivista huomata, että hyödyntämällä tarkistuslistaa ja analysoimalla uudet valmistuneet ominaisuudet systemaattisesti pystyin huomaamaan ja korjaamaan saavutettavuusongelmat sujuvasti sovelluskehityksen ohessa. Scrum-viitekehyksen sprintit tarjoavat hyvän tarkistuspisteen saavutettavuusanalyyseille. Accessibility Scanner -työkalusta oli apua erityisesti kontrastiongelmien havaitsemisessa, ja saavutettavuustestit voi myös automatisoida, mutta mielestäni nämä eivät korvaa sovelluksen testaamista avustavan teknologian kanssa. Minulla oli muutama saavutettavuusongelma, kuten painikkeiden virheelliset kuvaukset sekä tekstin suurentamisesta aiheutuneet käyttöliittymäelementtien peittymiset, joita en olisi havainnut Accessibility Scanner -työkalun avulla. Kehittäjän ei kannatakaan tukeutua pelkästään analyysityökalun tai testien tuloksiin. Jos omat taidot eivät riitä saavutettavuuden arvioimiseen, sovelluksen testauksessa kannattaa hyödyntää saavutettavuusasiantuntijoita.

Opinnäytetyön pääasiallisena tarkoituksena oli perehtyä saavutettavuuteen ja sen soveltamiseen Android-sovelluskehityksessä, mutta työn sivujuonteena halusin myös ottaa haltuun omatoimisesti itselleni uuden teknologian sekä ohjelmointikielen. Työn kannalta tämä oli riski, joka toteutui, sillä minulla meni huomattavasti kauemmin aikaa työn toteutukseen

kuin mitä olin työn alussa arvioinut. En ollut varannut riittävästi aikaa uusien asioiden opiskeluun ja sisäistämiseen. Erityisesti lääkkeenoton muistutustoiminnallisuutta toteuttaessani jouduin oppimaan yrityksen ja erehdyksen kautta Android-käyttöjärjestelmän virransäästöominaisuuksien vaikutuksen järjestelmän hälytysten laukeamiseen. Siinä missä etsin virhettä koodissa, minun olisi pitänyt lukea dokumentaatiota tarkemmin sekä toteuttaa sovellukseen ominaisuus, joka pyytää käyttäjältä lupaa säätää virranhallinta-asetuksia.

Olisin halunnut panostaa enemmän sovelluksen käyttöliittymän suunnitteluun ja tutustua huolellisemmin Googlen Material Design -komponentteihin. Esimerkiksi Spinner-elementti kannattaisi korvata helppokäyttöisemmällä ja tyylikkäämmällä Material Design -valikkokomponentilla. Lääkkeen muokkausnäkymässä on mielestäni vielä liikaa asioita ruudulla. Jatkokehityksen kannalta minun tulisi hyödyntää enemmän valikoita, avautuvia dialogeja ja tarvittaessa esimerkiksi jakaa lääkkeen ja annostuksen muokkaus eri näkymiin. Lääkkeiden ja vahvuuksien syöttämistä tulisi myös selkeyttää ja helpottaa. Toivottavin ratkaisu tähän olisi, että sovellus olisi yhteydessä lääketietokantaan ja pystyisi täydentämään tekstiä automaattisesti tai antamaan ehdotuksia käyttäjän syötteen perusteella. Lisäksi joudun miettimään sovelluksen rakenteen ja tietokannan uusiksi, jotta sovellus tukisi erikoisannostelujen valintaa.

Opinnäytetyön ansiosta tietoni ja taitoni WCAG-kriteereistä sekä saavutettavuuden testaamisesta lisääntyivät. Pystyin myös hyödyntämään karttunutta osaamistani työssäni tehdessäni saavutettavuuskorjauksia verkkopalveluun. Opinnäytetyö harjoitti lisäksi tiedonhakutaitojani sekä epävarmuuden sietämistä. Opin myös uuden arkkitehtuurimallin ja pääsin syventämään osaamistani mobiilikehityksen parissa.

Android-sovellusmarkkinat ovat erittäin kilpaillut, ja Google Play -sovelluskaupassa oli 30.8.2021 melkein 3 miljoonaa Android-sovellusta (AppBrain 2021). Saavutettavaa sovellusta kannattaa ajatella kilpailuvalttina, joka erottuu edukseen vastaavista tuotoksista. Saavutettavan sovelluksen tekemiseen ei kuitenkaan ole yhtä yleispätevää ratkaisua, vaan suunnittelu ja ratkaisut riippuvat sovelluksesta ja sen käyttötarkoituksesta. Kehittäjän kannattaa tutustua avustavan teknologian, kuten esimerkiksi ruudunlukijan käyttöön, jotta hän ymmärtää, miten erilaisilla tavoilla ja apuvälineillä sovellusta voidaan hyödyntää. Tätä kautta hän myös pystyy paremmin huomioimaan teknisellä tasolla koodissa avustavien teknologioiden tarpeet. Loppujen lopuksi kaikki teknologia on avustavaa teknologiaa (Hendren 2014).

Vaikka digipalvelulaki velvoittaa muun muassa viranomaiset ja julkishallinnon organisaatiot noudattamaan saavutettavuusvaatimuksia, käyttäjien etuna on, että muutkin toimijat

pyrkivät huomioimaan tuottamiensa digitaalisten palveluidensa saavutettavuuden mahdollisuuksiensa mukaan (Laki digitaalisten palvelujen tarjoamisesta 306/2019). Saavutettavuusvaatimukset saattavat aluksi tuntua hankalalta sisäistää, mutta eivät loppujen lopuksi vaadi liikaa aikaa muulta kehitykseltä, jos niihin kiinnitetään huomiota koko kehitysprojektin aikana eikä vasta loppuvaiheessa. Lisäksi kehittäjät pystyvät edistämään yhteiskunnan yhdenvertaisuutta huolehtimalla digitaalisten palveluiden esteettömyydestä.

Lähteet

Aluehallintovirasto 2021a. Digipalvelulain vaatimukset. Luettavissa: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/. Luettu: 19.9.2021.

Aluehallintovirasto 2021b. WCAG 2.1: Lain vaatimukset. Luettavissa: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/wcag-2-1/. Luettu: 27.2.2021.

Aluehallintovirasto 2021c. Muita lakeja. Luettavissa: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/muita-lakeja/. Luettu: 27.2.2021.

AppBrain 2021. Number of Android apps on Google Play. Luettavissa: https://www.appbrain.com/stats/number-of-android-apps. Luettu: 30.8.2021.

Atlassian 2021. Gitflow Workflow. Luettavissa: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow. Luettu: 29.3.2021.

Baeldung 2021. Data Classes in Kotlin. Luettavissa: https://www.baeldung.com/kotlin/data-classes. Luettu: 10.4.2021.

Celia 2021. Saavutettavuus. Luettavissa: https://www.celia.fi/saavutettavuus/. Luettu: 10.4.2021.

Dogtiev, A. 2021. App Stores List (2020). Luettavissa: https://www.businessofapps.com/guide/app-stores-list/. Luettu: 11.4.2021.

Euroopan parlamentin ja neuvoston direktiivi julkisen sektorin elinten verkkosivustojen ja mobiilisovellusten saavutettavuudesta (EU) N:o 2102/2016. Luettavissa: https://eur-lex.europa.eu/legal-content/FI/TXT/HTML/?uri=CELEX:32016L2102&from=FI. Luettu: 27.2.2021.

Ghoda, A. & Ferracchiati, F. 2012. Windows 8 MVVM Patterns Revealed. Introduction. Apress.

Git SCM 2021. Git --distributed-even-if-your-workflow-isnt. Luettavissa: https://git-scm.com/. Luettu: 29.3.2021.

GitHub 2021. Where the world builds software. Luettavissa: https://github.com/. Luettu: 11.4.2021.

Google Android Accessibility Help 2021a. Get started on Android with TalkBack. Luettavissa: https://support.google.com/accessibility/android/answer/6283677?hl=en&ref_topic=3529932. Luettu: 10.4.2021.

Google Android Accessibility Help 2021b. Android accessibility overview. Luettavissa: https://support.google.com/accessibility/android/answer/6006564?hl=en&ref_topic=6007234. Luettu: 6.3.2021.

Google Android Accessibility Help 2021c. About Switch Access for Android. Luettavissa: https://support.google.com/accessibility/android/answer/6122836?hl=en. Luettu: 20.4.2021.

Google Android Accessibility Help 2021d. Content labels. Luettavissa: https://support.google.com/accessibility/android/answer/7158690?hl=en. Luettu: 18.9.2021.

Google Android Developers 2021a. Platform Architecture. Luettavissa: https://developer.android.com/guide/platform. Luettu: 8.3.2021.

Google Android Developers 2021b. Application Fundamentals. Luettavissa: https://developer.android.com/guide/components/fundamentals.html. Luettu: 11.3.2021.

Google Android Developers 2021c. AccessibilityService. Luettavissa: https://developer.android.com/reference/android/accessibilityservice/AccessibilityService. Luettu: 13.3.2021.

Google Android Developers 2021d. Create your own accessibility service. Luettavissa: https://developer.android.com/guide/topics/ui/accessibility/service. Luettu: 13.3.2021.

Google Android Developers 2021e. Make apps more accessible. Luettavissa: https://developer.android.com/guide/topics/ui/accessibility/apps. Luettu: 6.3.2021.

Google Android Developers 2021f. Principles for improving app accessibility. Luettavissa: https://developer.android.com/guide/topics/ui/accessibility/principles. Luettu: 6.3.2021.

Google Android Developers 2021g. More resource types. Luettavissa: https://developer.android.com/guide/topics/resources/more-resources. Luettu: 6.3.2021.

Google Android Developers 2021h. Test your app's accessibility. Luettavissa: https://developer.android.com/guide/topics/ui/accessibility/testing. Luettu: 13.3.2021.

Google Android Developers 2021i. Persist data with Room. Luettavissa: https://developer.android.com/codelabs/basic-android-kotlin-training-persisting-data-room#0. Luettu: 24.8.2021

Google Android Developers 2021j. Fragments. Luettavissa: https://developer.android.com/guide/fragments. Luettu: 5.4.2021.

Google Android Developers 2021k. Getting started with the Navigation component. Luettavissa: https://developer.android.com/guide/navigation/navigation-getting-started. Luettu: 5.4.2021.

Google Android Developers 2021l. NumberPicker. Luettavissa: https://developer.android.com/reference/kotlin/android/widget/NumberPicker. Luettu: 8.4.2021.

Google Android Developers 2021m. Create dynamic lists with RecyclerView. Luettavissa: https://developer.android.com/guide/topics/ui/layout/recyclerview. Luettu: 24.8.2021.

Google Android Developers 2021n. Spinners. Luettavissa: https://developer.android.com/guide/topics/ui/controls/spinner. Luettu: 27.8.2021.

Google Android Developers 2021o. Guide to app architecture. Luettavissa: https://developer.android.com/jetpack/guide. Luettu: 11.4.2021.

Google Android Developers 2021p. Data Binding Library. Luettavissa: https://developer.android.com/topic/libraries/data-binding/. Luettu: 5.4.2021.

Google Android Developers 2021q. Factory. Luettavissa: https://developer.android.com/reference/kotlin/androidx/lifecycle/ViewModelProvider.Factory.html. Luettu: 11.4.2021.

Google Android Developers 2021r. Task: Use a ViewModelFactory. Luettavissa: https://developer.android.com/codelabs/kotlin-android-training-view-model#7. Luettu: 11.4.2021.

Google Android Developers 2021s. Two-way data binding. Luettavissa: https://developer.android.com/topic/libraries/data-binding/two-way. Luettu: 11.4.2021.

Google Android Developers 2021t. LiveData Overview. Luettavissa: https://developer.android.com/topic/libraries/architecture/livedata. Luettu: 11.4.2021.

Google Android Developers 2021u. Save data in a local database using Room. Luettavissa: https://developer.android.com/training/data-storage/room. Luettu: 11.4.2021.

Google Android Developers 2021v. Define relationships between objects. Luettavissa: https://developer.android.com/training/data-storage/room/relationships. Luettu: 11.4.2021.

Google Android Developers 2021w. Schedule repeating alarms. Luettavissa: https://developer.android.com/training/scheduling/alarms. Luettu: 28.8.2021.

Google Android Developers 2021x. Optimize for Doze and App Standby. Luettavissa: https://developer.android.com/training/monitoring-device-state/doze-standby. Luettu: 28.8.2021.

Google Android Developers Blog 2017. Android and Architecture. Luettavissa: https://android-developers.googleblog.com/2017/05/android-and-architecture.html. Luettu: 24.8.2021.

Google Material Design 2021a. Accessibility. Luettavissa: https://material.io/design/usability/accessibility.html. Luettu: 6.3.2021.

Google Material Design 2021b. Android haptics. Luettavissa: https://material.io/design/platform-guidance/android-haptics.html#principles. Luettu: 7.3.2021.

Google Material Design 2021c. Buttons: floating action button. Luettavissa: https://material.io/components/buttons-floating-action-button/android. Luettu: 5.4.2021.

Google Material Design 2021d. Time pickers. Luettavissa: https://material.io/components/time-pickers/android#using-time-pickers. Luettu: 8.4.2021.

Google Material Design 2021e. Color Tool. Luettavissa: https://material.io/resources/color/#!/?view.left=0&view.right=0. Luettu: 29.8.2021.

Google Play kauppa 2021a. Androidin esteettömyystyökalut. Luettavissa: https://play.google.com/store/apps/details?id=com.google.android.marvin.talkback&e=-EnableAppDetailsPageRedesign. Luettu: 6.3.2021.

Google Play kauppa 2021b. Sovellukset (hakutulos). Luettavissa: https://play.google.com/store/search?q=medicine%20reminder&c=apps. Luettu: 14.3.2021.

Google Play kauppa 2021c. Lääkemuistuttaja ja pilleriseuranta. Luettavissa: https://play.google.com/store/apps/details?id=eu.smartpatient.mytherapy. Luettu: 14.3.2021.

Google Play kauppa 2021d. Lääkitys ja Pilleri Muistutus – Medisafe. Luettavissa: https://play.google.com/store/apps/details?id=com.medisafe.android.client. Luettu: 14.3.2021.

Google Play kauppa 2021e. UPharma App. Luettavissa: https://play.google.com/store/apps/details?id=app.upharma.mobile. Luettu: 31.3.2021.

Gradle 2021. What is Gradle? Luettavissa: https://docs.gradle.org/current/userguide/what_is_gradle.html. Luettu: 29.3.2021.

Haase, C. 2019. Google I/O 2019: Empowering developers to build the best experiences on Android + Play. Android Developers Blog. Luettavissa: https://android-developers.googleblog.com/2019/05/google-io-2019-empowering-developers-to-build-experiences-on-Android-Play.html. Luettu: 8.3.2021.

Hendren, S. 2014. The Mobility Project. Luettavissa: https://themobilityproject.net/2014/08/28/all-technology-is-assistive-technology-sara-hendren/. Luettu: 21.4.2021.

Hill, S. & Jansen M. 2021. Android vs. iOS: Which smartphone platform is the best? Luettavissa: https://www.digitaltrends.com/mobile/android-vs-ios/. Luettu: 21.4.2021.

Horton S. & Sloan D. 2014. Accessibility in Practice: A Process-Driven Approach to Accessibility. Teoksessa Langdon P., Lazar J., Heylighen A. & Dong H. (toim.). Inclusive Designing, s. 105–115. Springer, Cham. Luettavissa: https://doi.org/10.1007/978-3-319-05095-9_10. Luettu: 10.4.2021.

Kallionpää R. 2021. Ruudunlukija-käyttäjien ääni kuuluviin - yhteenveto saavutettavuuskyselyn tuloksista. Luettavissa: https://www.eficode.com/fi/blog/ruudunlukijakayttajien-aani-kuuluviin. Luettu: 17.8.2021.

Kela 2019. Kanta-palvelut Omakanta Käyttöohje. Luettavissa: https://www.kanta.fi/documents/20143/91498/Omakanta+palvelunkuvaus+K%C3%84YTT%C3%96OHJE.pdf/0977c89f-15e0-0430-a7dc-415ffc0ac7d6?t=1549625608692. Luettu: 10.4.2021.

Kela 2020. Resepti. Luettavissa: https://www.kanta.fi/ammattilaiset/resepti. Luettu: 10.4.2021.

Kela 2021. Kansallinen lääkityslista. Luettavissa: https://www.kanta.fi/ammattilaiset/kansallinen-laakityslista. Luettu: 15.2.2021.

Kotlin 2021. FAQ. Luettavissa: https://kotlinlang.org/docs/faq.html. Luettu: 9.3.2021.

Laki digitaalisten palvelujen tarjoamisesta 306/2019. Luettavissa: https://www.finlex.fi/fi/laki/alkup/2019/20190306#Pidp446830800. Luettu: 27.2.2021.

Meier, R. 2012. Professional Android 4 application development. Hello, Android. Wiley cop. Indianapolis.

Moskala, M. & Wojda, I. 2017. Android Development with Kotlin. Say hello to Kotlin. Packt Publishing.

Muntenescu, F. 2017. Understanding migrations with Room. Luettavissa: https://medium.com/androiddevelopers/understanding-migrations-with-room-f01e04b07929. Luettu: 11.4.2021.

Myrick, C. 2019. Android AlarmManager As Deep As Possible. Luettavissa: https://proandroiddev.com/android-alarmmanager-as-deep-as-possible-909bd5b64792. Luettu: 28.8.2021.

Papunet 2020. WCAG 2.1:n rakenne ja käyttö. Luettavissa: https://papunet.net/saavutettavuus/wcag-21n-rakenne-ja-kaytto. Luettu: 10.4.2021.

Samsung 2021. How do I enable and disable the Screen Reader on my Samsung Galaxy smartphone? Luettavissa: https://www.samsung.com/ie/support/mobile-devices/how-do-i-enable-and-disable-voice-assistant-talkback-on-my-samsung-galaxy-smartphone/. Luettu: 13.3.2021.

Schwaber, K. & Sutherland, J. 2020. The 2020 Scrum Guide. Luettavissa: https://scrum-guides.org/scrum-guide.html. Luettu: 29.3.2021.

Scrum 2021. What is a Sprint Retrospective? Luettavissa: https://www.scrum.org/resources/what-is-a-sprint-retrospective. Luettu: 5.4.2021.

Späth, P. 2018. Pro Android with Kotlin. System. Apress.

StatCounter 2021. Mobile Operating System Market Share Worldwide - February 2021 . Luettavissa: https://gs.statcounter.com/os-market-share/mobile/worldwide. Luettu: 13.3.2021.

Suomen Apteekkariliitto 2021. Lääkityslista. Luettavissa: https://www.apteekki.fi/apteekki-palvelut/laakityslista-2.html. Luettu: 7.3.2021.

Udacity 2021. Developing Android Apps with Kotlin. Luettavissa: https://www.udacity.com/course/developing-android-apps-with-kotlin--ud9012. Luettu: 4.4.2021.

UPharma Care 2021a. UPharma App. Luettavissa: https://www.upharma-care.com/upharma-app/. Luettu: 31.3.2021.

UPharma Care 2021b. Miksi juuri Sinä tarvitset lääkelistan? Luettavissa: https://www.upharmacare.com/2021/03/miksi-juuri-sina-tarvitset-laakelistan/. Luettu: 31.3.2021.

Valtiovarainministeriö 2021. Saavutettavuusdirektiivi. Luettavissa: https://vm.fi/saavutettavuusdirektiivi. Luettu: 14.2.2021.

Verdecchia, R., Malavolta, I. & Lago P. 2019. Guidelines for Architecting Android Apps: A Mixed-Method Empirical Study. 2019 IEEE International Conference on Software Architecture (ICSA), s. 141–150.

Virkkunen 2020. Valtakunnallisen lääkityslistan tilanne nyt. Luettavissa: https://www.fimea.fi/documents/160140/10248328/Virkkunen_L%C3%A4%C3%A4kityslista_Fimea_20200212.pdf/b9f5d6e6-bde6-9378-786c-59bde5001b63?t=1581936372468. Luettu: 20.4.2021.

W3C 2015. Mobile Accessibility: How WCAG 2.0 and Other W3C/WAI Guidelines Apply to Mobile. Luettavissa: https://www.w3.org/TR/mobile-accessibility-mapping/. Luettu: 28.2.2021.

W3C 2018. The World Wide Web Consortium. Web Content Accessibility Guidelines (WCAG) 2.1. Luettavissa: https://www.w3.org/TR/2018/REC-WCAG21-20180605/. Luettu: 14.2.2021.

W3C 2020. What's New in WCAG 2.1. Luettavissa: https://www.w3.org/WAI/standards-guidelines/wcag/new-in-21/#413-status-messages-aa. Luettu: 28.2.2021.

W3C 2021a. W3C Accessibility Guidelines (WCAG) 3.0. W3C Working Draft 08 June 2021. Luettavissa: https://www.w3.org/TR/2021/WD-wcag-3.0-20210608/. Luettu: 15.9.2021.

W3C 2021b. How to Meet WCAG (Quick Reference). Luettavissa: https://www.w3.org/WAI/WCAG21/quickref/#reflow. Luettu: 19.9.2021.

Liitteet

Liite 1. Mobiilisovellusten kannalta tärkeitä saavutettavuuskriteerejä

Alla olevassa taulukossa esitetyt WCAG-kriteerit ja niiden käännökset on otettu Aluehallintoviraston saavutettavuusvaatimukset-sivustolta (Aluehallintovirasto 2021b).

Kriteeri	Taso	Virallinen suomennos
1.3.4 Asento	AA	Sisältöä ei ole rajoitettu vain tiettyyn näyttölaitteen asentoon kuten pysty- tai vaakasuuntaan, lukuun ottamatta tapausta, jossa tietty asento on olennainen. Huomautus: Tapauksia, joissa tietty näyttölaitteen asento on olennainen ovat esimerkiksi šekki, pianosovellus, esitysdiat projektoria tai televisiota varten tai virtuaalitodellisuuden sisältö, joihin kahdensuuntainen näyttölaitteen asento ei sovellu.
1.4.3 Kontrasti (minimi)	AA	Tekstin ja tekstiä esittävien kuvien visuaalisen esitystavan kontrastisuhde on vähintään 4,5:1, paitsi seuraavissa tapauksissa: - Isokokoinen teksti: Isokokoisessa tekstissä uuteen ikkunaan ja isokokoista tekstiä esittävissä kuvissa kontrastisuhde on vähintään 3:1. - Oheissisältö: Tekstille tai tekstiä esittäville kuville ei ole kontrastivaatimusta, jos ne ovat osa inaktiivista käyttöliittymäkomponenttia, yksinomaan koristeita, eivät ole näkyvissä kenellekään tai ovat osa kuvaa, jossa on muuta merkittävää visuaalista sisältöä. - Logotyypit: Tekstille, joka on osa logoa tai brändin nimeä, ei ole kontrastivaatimusta.
1.4.4 Tekstin koon muuttaminen	AA	Lukuun ottamatta tekstitystä ja tekstiä esittäviä kuvia, tekstin kokoa voidaan muuttaa ilman avustavaa teknologiaa aina 200 prosenttiin asti ilman sisällön tai toiminnallisuuden menettämistä.
1.4.10 Responsiivisuus	AA	Sisältö voidaan esittää ilman sisällön tai toiminnallisuuden menettämistä ja ilman kahdensuuntaista vierittämistä, kun - pystysuuntaan vieritettävän sisällön leveys on 320 CSS-pikseliä. - vaakasuuntaan vieritettävän sisällön korkeus on 256 CSS-pikseliä. Lukuun ottamatta sisällön osia, jotka vaativat kahdensuuntaista esitystapaa käytön tai merkityksen vuoksi. Huomautus: 320 CSS-pikseliä vastaa 1280 CSS-pikselin levyistä selainikkunaa, joka on zoomattu 400 % kokoiseksi. Vaakasuuntaan vieritettävässä sisällössä (esim. pystysuuntainen kirjoitus), 256 CSS-pikseliä vastaa 1024 px korkeaa selainikkunaa, joka on zoomattu 400 % kokoiseksi. Huomautus: Esimerkkejä sisällöstä, joissa vaaditaan kahdensuuntaista esitystapaa, ovat kuvat, kartat, diagrammit, videopelit, esitykset, taulukkomuotoinen data ja käyttöliittymät, joissa on tarpeellista pitää työkalupalkki näkyvissä, kun käsitellään sisältöä.
2.5.1 Osoitineleet	A	Kaikkia toimintoja joissa hyödynnetään monipiste- tai reittiin perustuvia ohjauseleitä, voidaan käyttää myös yhdellä osoittimella ja ilman reittiin perustuvaa elettä, paitsi jos kyseinen ohjaustapa on olennainen. Huomautus: Tämä vaatimus koskee verkkosisältöä, joka vastaanottaa ja tulkitsee osoitinlaitteen toimintoja (ts. tämä ei koske toimintoja, joilla ohjataan käyttäjäagenttia tai avustavaa teknologiaa).

Kriteeri	Taso	Virallinen suomennos
2.5.2 Osoitineleellä tehdyn valinnan peruuttaminen	A	Toimintoihin, joita voidaan käyttää yhden osoittimen, pätee vähintään yksi seuraavista: - Ei alas-tapahtumaa: Mikään osa toiminnallisuudesta ei tapahdu alas-tapahtuman yhteydessä. - Keskeytä tai kumoa: Toiminnon päättäminen tapahtuu ylös-tapahtuman yhteydessä, ja on olemassa mekanismi, jolla toiminto voidaan perua ennen päättämistä tai kumota päättämisen jälkeen. - Vastakkaisuus: Ylös-tapahtuma kumoaa edeltävän alas-tapahtuman aiheuttaman toiminnon. - Olennainen: Toiminnon päättäminen alas-tapahtuman yhteydessä on olennaista. Huomautus: Toimintoja, jotka jäljittelevät näppäimistön tai numeronäppäimistön painalluksia pidetään olennaisina. Huomautus: Tämä vaatimus koskee verkkosisältöä, joka vastaanottaa ja tulkitsee osoitinlaitteen toimintoja (ts. tämä ei koske toimintoja, joilla ohjataan käyttäjäagenttia tai avustavaa teknologiaa).
2.5.4 Käyttö liikkeen avulla	A	Toiminnallisuus, jota voidaan käyttää liikuttamalla laitetta, voidaan käyttää myös käyttöliittymäkomponenttien avulla, ja liikeaktivointi voidaan ottaa pois päältä, jotta vältetään toiminnan aktivoiminen vahingossa. Tämä ei koske seuraavia tapauksia: - Tuettu rajapinta: Liikeaktivointi on toteutettu sellaisen rajapinnan kautta, joka on saavutettavuudeltaan tuettu. - Olennainen: Liike on toiminnon kannalta olennainen, ja sen poistaminen mitätöisi toiminnon.
4.1.3 Tilasta kertovat viestit	AA	Sisällössä, joka on toteutettu käyttäen merkkauskieliä, tilasta kertovat viestit voidaan selvittää ohjelmallisesti sellaisen roolin tai ominaisuuksien avulla, jotka mahdollistavat viestin esittämisen käyttäjälle avustavan teknologian avulla ilman kohdistuksen siirtämistä.

Liite 2. Saavutettavuusanalyysi 1: MyTherapy

Sovelluksen nimi: MyTherapy Lääkitysmuistuttaja ja pilleriseuranta

Accessibility Scanner -työkalun analyysi

Kahdella kohteella on sama kuvaus ”Hoito”. Yläpalkin etualan ja taustan sekä painikkeiden etualan ja taustan kontrastisuhde ei ole riittävä. Osassa painikkeita/napautettavia kohteita liian pieni kosketusalue. Kontrastisuhde on huono myös ilmoitusajan valinnassa (pyöritysvalikko). Tiimi-näkymässä kohteelta puuttuu tunniste.

WCAG-kriteerit

Kriteeri	Tulos
1.3.4 Asento	Pystyi käyttämään pysty- ja vaakasuunnassa. Tosin muokkaustila saattoi keskeytyä orientaation vaihtuessa yllättäen.
1.4.3 Kontrasti (minimi)	Kontrastitaso huono.
1.4.4 Tekstin koon muuttaminen	Osa teksteistä, jotka eivät mahtuneet ruutuun, katkaistiin kolmella pisteellä.
1.4.10 Responsiivisuus	Sisältöä ei tarvinnut vierittää.
2.5.1 Osoitineleet	Lääkkeiden ottoajankohdan määrittämistä ei pystynyt tekemään painamalla yhtä kohtaa ruudulla, vaan toiminnallisuuteen piti lisätä liike.
2.5.2 Osoitineleellä tehdyn valinnan peruuttaminen	Osoitineleen peruuttaminen onnistui viemällä sormen kohdistuksen pois.
2.5.4 Käyttö liikkeen avulla	(ei liittynyt sovelluksen toimintaan)
4.1.3 Tilasta kertovat viestit	Viestit ja hälytykset luettiin ääneen.

TalkBack

Tarkistuksen kohde	Tulos
onko käyttöliittymäelementeillä asianmukaiset kuvaukset ääneen puhuttuna	Yhdestä valikkokohteesta puuttui kuvaus (ruudunlukija luki: ”nimetön painike”). Ruudunlukija ei toisaalta ilmoittanut turhaan koristekuvista. Pahin puute oli suoritetut tehtävät näkymässä, jossa suoritettujen tehtävien lukumäärä oli ilmaistu numeron sijaan numeron kuvana, jota ei luettu ääneen.
ovatko ruudunlukijan ilmoitukset ytimekkäitä vai turhan yksityiskohtaisia	Ruudunlukija toisti turhaan lääkkeiden nimiä moneen kertaan.
onnistuuko sovelluksen päätoiminnallisuuksien käyttö vaivattomasti	Hälytyksen ajan asettaminen oli hankalaa, sillä pyöritysvalikko oli pieni, eikä pyyhkäisyn kohdistaminen ollut helppoa. Tästä olisi voinut olla ohjeet näkövammaiselle käyttäjälle. Jos sovelluksen asetuksista valitsi jonkin maan lääketietokannan, silloin sovellus tarjosi automaattista lomakkeen täyttöä kyseisen maan lääkkeiden tiedoilla.

Tarkistuksen kohde	Tulos
pystyykö sovelluksessa käymään läpi kaikki vuorovaikutteiset elementit pyyhkäisyeleillä	Pyyhkäisyeleet toimivat.
lukeeko ruudunlukija ääneen ilmoitukset ja hälytykset	Ruudunlukija luki ääneen sovelluksen ilmoitukset ja hälytykset.

Kytkintoiminnallisuus

Tarkistuksen kohde	Tulos
pystyykö sovelluksen päätoiminnallisuudet toteuttamaan kytkimen avulla	Pystyy.
pystyykö lomakkeisiin ja muihin syöttökenttiin täyttämään tekstiä vaivattomasti	Osittain. Harmi vain, että lääkkeiden nimien automaattinen täyttöominaisuus ei toiminut suomeksi.
tuleeko elementteihin, joiden kanssa voi olla vuorovaikutuksessa, korostus aktivoitaessa ja vastaavasti ei-interaktiivisiin elementteihin ei	Kyllä.
ovatko elementit korostettuina vain kerran	Kyllä.
pystyykö kaikkia toiminnallisuuksia, jotka toimivat kosketuseleillä, toteuttamaan myös kytkimellä	Kyllä, muun muassa lääkkeen ajankohdan valintaa pystyi ohjaamaan kytkimellä, vaikka ajan valinta oli karusellityyppinen.

Muita huomioita

Sovelluksessa ei ollut Suomen lääketietokantaa, joten lomakkeen automaattista täyttöä ei päässyt hyödyntämään suomeksi. Ruotsin lääketietokannalla pääsee kyllä lähelle, sillä samoja ja samantyylisiä kauppanimiä on paljon. Lääkkeenoton vahvistukselle on selkeä väripalkki, jossa on myös teksti- ja visuaalinen vihje pelkän värin sijaan. Kun asetin lääkkeelle varastoa ja vahingossa ruudun orientaatio vaihtui, sovellus palasi edelliseen ikkunaan.

Liite 3. Saavutettavuusanalyysi 2: Medisafe

Sovelluksen nimi: Medisafe Lääkitys ja Pilleri Muistutus

Accessibility Scanner -työkalun analyysi

Sovelluksella on paljon kontrastiongelmia tekstien, painikkeiden etualan ja taustan välillä. Liian pieniä kosketusalueita. Useammalta painikkeelta puuttuu tunnisteet.

WCAG-kriteerit

Kriteeri	Tulos
1.3.4 Asento	Toimi sekä pysty- että vaakasuunnassa.
1.4.3 Kontrasti (minimi)	Huonot kontrastisuhteet.
1.4.4 Tekstin koon muuttaminen	Painikkeiden teksteistä häviää kirjaimia.
1.4.10 Responsiivisuus	Sovellusta ei tarvitse vierittää kahdensuuntaisesti.
2.5.1 Osoitineleet	Annostuksen pystyy määrittämään painamalla yhtä kohtaa ruudulla.
2.5.2 Osoitineleellä tehdyn valinnan peruuttaminen	Eleen voi peruuttaa siirtämällä sormen kohdistuksen muualle.
2.5.4 Käyttö liikkeen avulla	(ei liittynyt sovelluksen toimintaan)
4.1.3 Tilasta kertovat viestit	Kohdistus ei siirtynyt, eikä ruudunlukija ilmoittanut viestin tilaa.

TalkBack

Tarkistuksen kohde	Tulos
onko käyttöliittymäelementeillä asianmukaiset kuvaukset ääneen puhuttuna	Ruudunlukija ilmoitti painikkeen nimeksi tallenna, mutta painikkeen takaa avautuu muokkaustoiminnallisuus. Sovelluksessa on todella paljon nimettömiä painikkeita.
ovatko ruudunlukijan ilmoitukset ytimekkäitä vai turhan yksityiskohtaisia	Pääsääntöisesti ytimekkäitä, mutta ilmoituksen yhteydessä ruudunlukija luki kummallisen numeroilmoituksen.
onnistuuko sovelluksen päätoiminnallisuuksien käyttö vaivattomasti	Nimettömät painikkeet hankaloittavat lääkeannoksen määrittelyä. Joissakin painikkeissa seliteteksti on alapuolella ja nimetön kuvake yläpuolella. Tallenna ja Valmis-painikkeiden sijainti vaihtelee ylhäällä palkissa tai alhaalla ruudussa. Lääkkeen täydennystoiminnallisuuden näyttö on outo.
pystyykö sovelluksessa käymään läpi kaikki vuorovaikutteiset elementit pyyhkäisyeleillä	Pystyy.
lukeeko ruudunlukija ääneen ilmoitukset ja hälytykset	Ruudunlukija ei lukenut lääkkeen ottamisen hyväksymisilmoitusta. Väliin jääneen lääkkeen ilmoitusta ei myöskään lueta.

Kytkintoiminnallisuus

Tarkistuksen kohde	Tulos
pystyykö sovelluksen päätoiminnallisuudet toteuttamaan kytkimen avulla	Pystyy.
pystyykö lomakkeisiin ja muihin syöttö-kenttiin täyttämään tekstiä vaivattomasti	Osittain. Automaattinen täyttö USA:n kauppanimillä.
tuleeko elementteihin, joiden kanssa voi olla vuorovaikutuksessa, korostus aktivoitaessa ja vastaavasti ei-interaktiivisiin elementteihin ei	Lääkkeet-näkymässä koko näkymä korostuu, vaikka sen valitsemisesta ei seuraa interaktiivista toimintaa.
ovatko elementit korostettuina vain kerran	Kyllä.
pystyykö kaikkia toiminnallisuuksia, jotka toimivat kosketuseleillä, toteuttamaan myös kytkimellä	Pystyy.

Muita huomioita

Sovelluksen tekstit saattavat yllättäen vaihtua englanniksi, mikä sotkee ruudunlukijan toimintaa. Automaattinen täyttö toimii vain englanninkielisillä kauppanimillä.

Liite 4. Saavutettavuusanalyysi 3: UPharma App

Sovelluksen nimi: UPharma App

Accessibility Scanner -työkalun analyysi

Painikkeiden kosketusalueet olivat pieniä ja niiden kontrastissa oli myös parantamisen varaa.

WCAG-kriteerit

Kriteeri	Tulos
1.3.4 Asento	Sisältö on rajoitettu tietyissä näkymissä vain vaakatasoon.
1.4.3 Kontrasti (minimi)	Kontrastit eivät olleet riittäviä.
1.4.4 Tekstin koon muuttaminen	Fonttikoon suurentaminen sai aikaan sen, että painikkeiden tekstejä jäi logojen alle.
1.4.10 Responsiivisuus	Ruutua ei tarvinnut vierittää tärkeiden toiminnallisuuksien löytämiseksi.
2.5.1 Osoitineleet	Pystyi käyttämään yhdellä sormella, mutta ruudun skrollaus ei onnistunut aina.
2.5.2 Osoitineleellä tehdyn valinnan peruuttaminen	Pystyi peruuttamaan.
2.5.4 Käyttö liikkeen avulla	(ei olennainen sovelluksen kannalta)
4.1.3 Tilasta kertovat viestit	Sovelluksessa ei ollut tilasta kertovia viestejä.

TalkBack

Tarkistuksen kohde	Tulos
onko käyttöliittymäelementeillä asianmukaiset kuvaukset ääneen puhuttuna	Ei, lisäksi painikkeiden nimet tulevat englanniksi.
ovatko ruudunlukijan ilmoitukset ytimekkäitä vai turhan yksityiskohtaisia	Taulukon elementit ei ollut ryhmitetty luettavaksi yhdessä.
onnistuuko sovelluksen päätoiminnallisuuksien käyttö vaivattomasti	Ei. Näkövammainen käyttäjä ei saa tarpeeksi informaatiota ja ruudunlukija lukee sekakielellä.
pystyykö sovelluksessa käymään läpi kaikki vuorovaikutteiset elementit pyyhkäisyeleillä	Ei.
lukeeko ruudunlukija ääneen ilmoitukset ja hälytykset	Sovelluksessa ei ollut tilasta kertovia viestejä.

Kytkintoiminnallisuus

Tarkistuksen kohde	Tulos
pystyykö sovelluksen päätoiminnallisuudet toteuttamaan kytkimen avulla	Ei. Sovellus ei anna navigoida kytkimen avulla.
pystyykö lomakkeisiin ja muihin syöttö-kenttiin täyttämään tekstiä vaivattomasti	Ei. Tähän pisteeseen ei päästä kytkimen avulla.
tuleeko elementteihin, joiden kanssa voi olla vuorovaikutuksessa, korostus aktivoitaessa ja vastaavasti ei-interaktiivisiin elementteihin ei	Ei tule korostusta, koska sovelluksen elementteihin ei edes pääse käsiksi kytkimen avulla.
ovatko elementit korostettuina vain kerran	Ei voi arvioida edellä mainituista syistä.
pystyykö kaikkia toiminnallisuuksia, jotka toimivat kosketuseleillä, toteuttamaan myös kytkimellä	Ei.

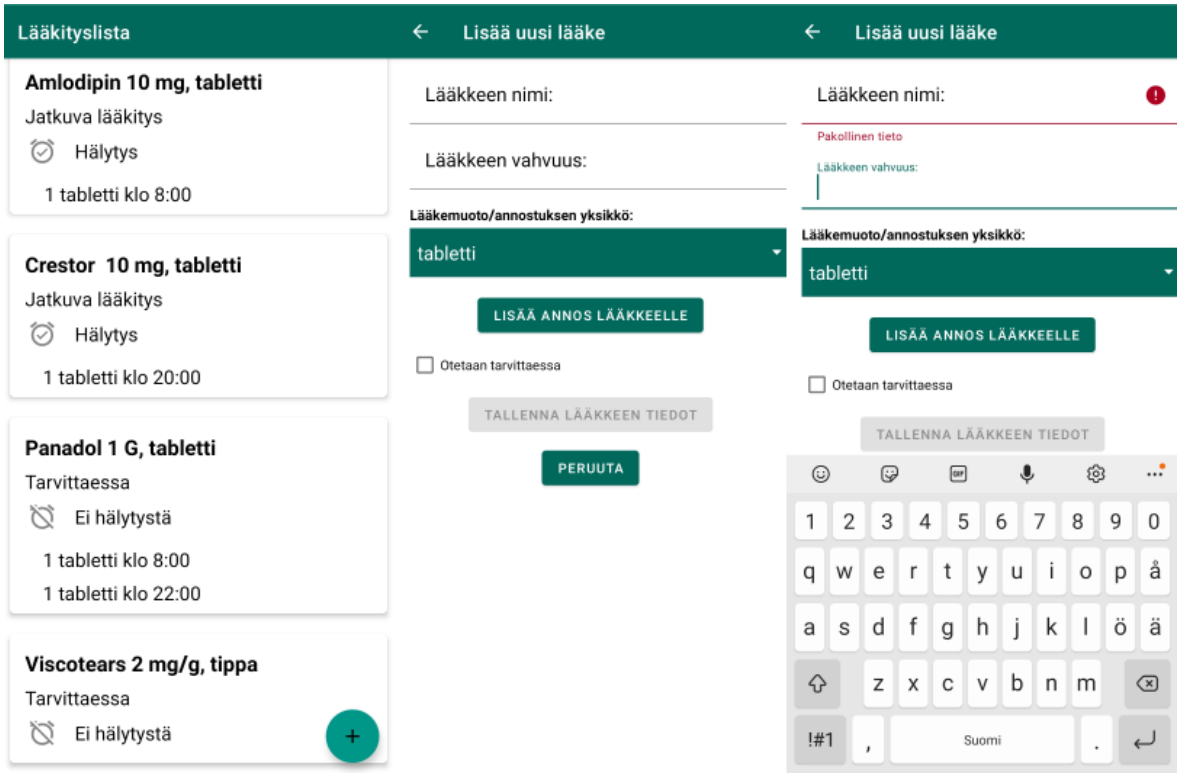
Muita huomioita

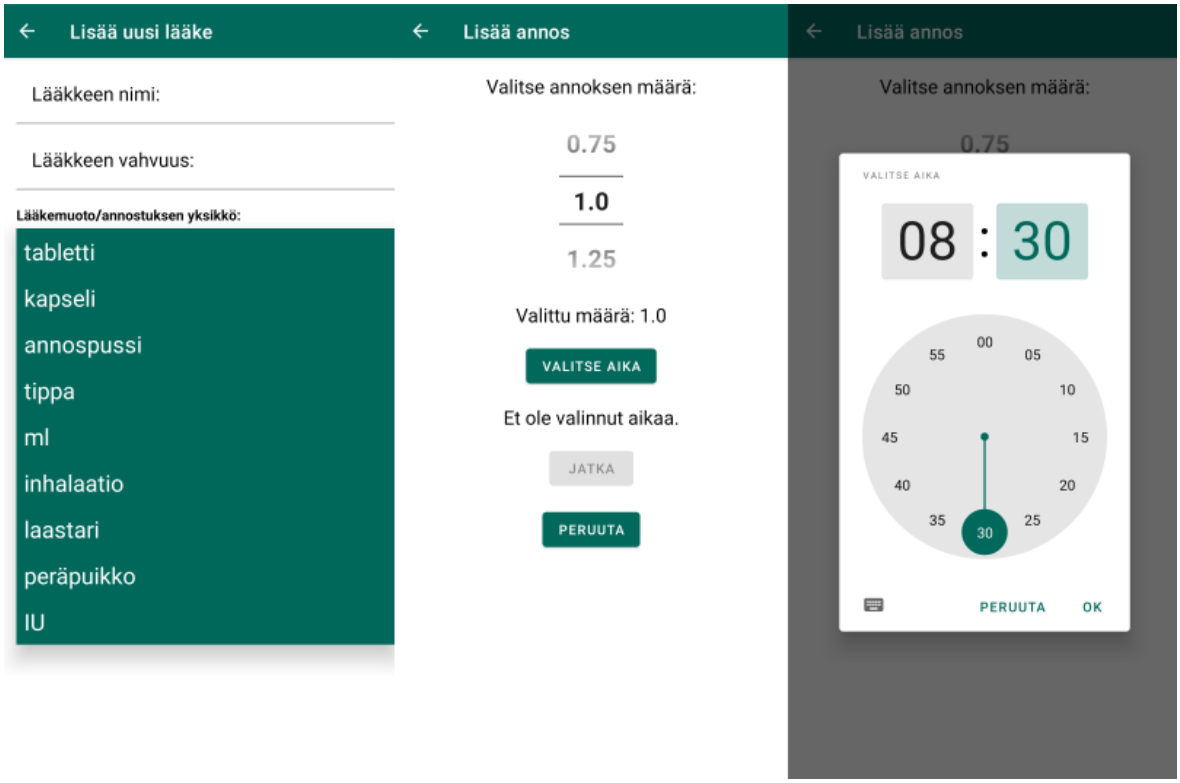
Sovelluksen latauksen jälkeen sovellus pakotti luomaan tilin. Säännöllinen lääkitys, tarvittaessa otettavat lääkkeet sekä rokotekorttinäkymä on lukittu tiettyyn orientaatioon. Jos lääkettä halusi muokata, tuli painaa kapeaa taulukon riviä. Skannerin avulla paljastui ylimääräisiä napautettavia alueita, joissa ei ollut tekstiä eikä toiminnallisuutta. Lääkkeenoton kellonaikoja ei pystynyt vaihtamaan (määrät klo 8, 14, 16, 20) lääkenäkymässä vaan asetuksissa, ja muistutusten määrä per lääke on rajattu neljään. Sovellus käyttää englantia ruudunlukijan kanssa, vaikka näytön kielenä on suomi. Vaihdoin vielä varmuuden vuoksi asetuksista kielen suomeksi. Painikkeiden nimeä ei lueta. Ruudunlukijaa käytettäessä selauksen kohdistus osui haamupainikkeisiin, ja joidenkin painikkeiden kosketuspinta-ala on olematon eikä vastannut ruudulla näkyvää painiketta.

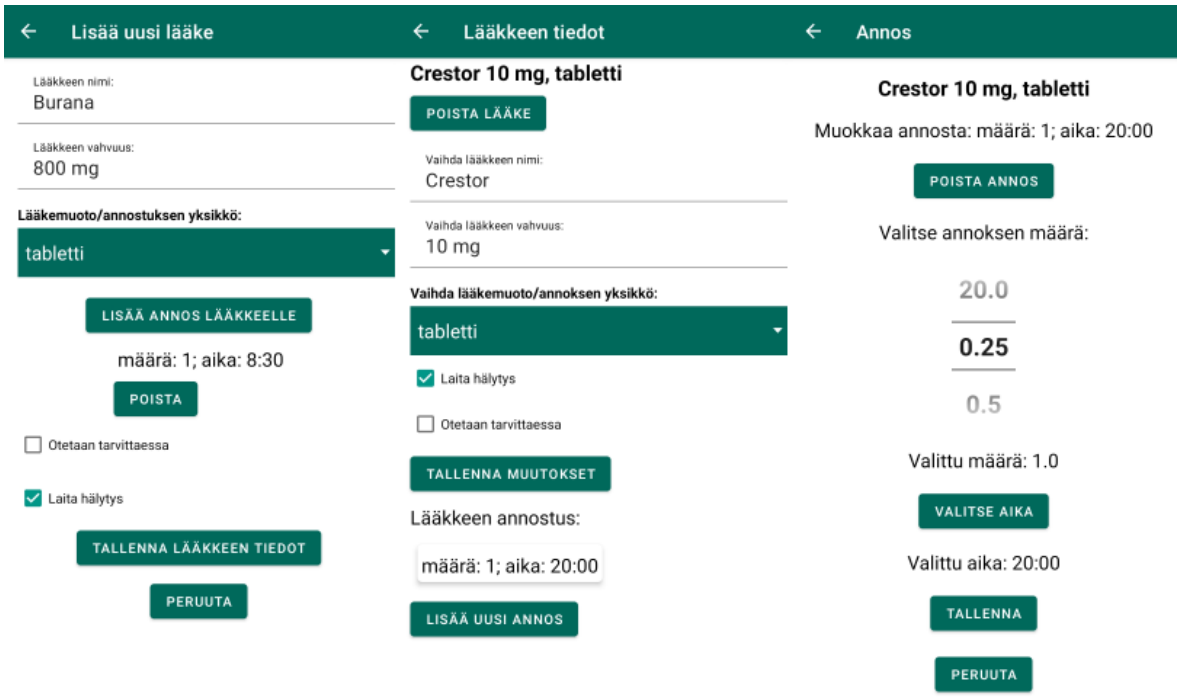
Liite 5. Tuotteen kehitysjono käyttäjätarinamuodossa

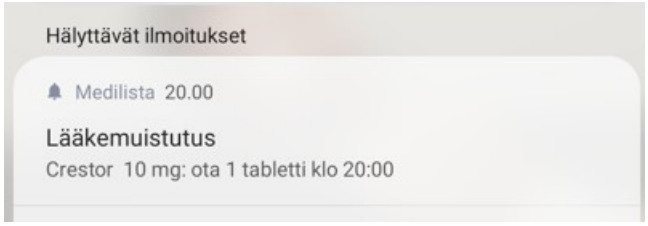
Käyttäjätarina	Hyväksymiskriteeri
Käyttäjä voi lisätä lääkkeen listaan.	Päänäkymässä on painike, jota painamalla pääsee lääkkeen lisäysnäkymään. Lomakkeen täyttämisen ja tallennuspainikkeen painamisen jälkeen lääke tallentuu tietokantaan.
Käyttäjä voi tarkastella lääkkeitä listalla.	Tallennettujen lääkkeiden tiedot näkyvät päänäkymän listassa.
Käyttäjä voi lisätä lääkkeelle perusannostuksen.	Lääkkeen lisäysnäkymästä pääsee toiseen näkymään, jossa voi tallentaa määrän ja ottoajan lääkkeelle. Lisätty annostus näkyy lääkkeen lisäysnäkymässä, ja tallennuksen jälkeen myös päänäkymässä lääkkeen muiden tietojen yhteydessä.
Käyttäjä voi poistaa lääkkeen.	Päänäkymässä lääkkeen korttia painamalla päästään muokkausnäkymään, jossa on painike lääkkeen poistolle. Painikkeen painamisen jälkeen lääke ja sen annostukset poistuvat tietokannasta ja päänäkymästä.
Käyttäjä voi poistaa annostuksen.	Lääkkeen muokkausnäkymässä annostuksen korttia painamalla pääsee näkymään, jossa on painike annostuksen poistolle. Painikkeen painamisen jälkeen kyseinen annostus poistuu tietokannasta ja näkymistä.
Käyttäjä voi muokata lääkkeen tietoja.	Lääkkeen muokkausnäkymän kenttiin voi syöttää uudet arvot. Näkymässä on painike, jota painamalla uudet tiedot tallentuvat tietokantaan ja päivittyvät näkymään.
Käyttäjä voi muokata annostuksen tietoja.	Annostuksen muokkausnäkymässä voi vaihtaa määrän ja kellonajan. Edelliset arvot näkyvät. Tallennuspainike tallentaa tiedot tietokantaan ja siirrytään takaisin lääkkeen muokkausnäkymään.
Käyttäjä voi lisätä lääkkeelle muistutuksen.	Jos lääkkeelle on valittu hälytys, käyttäjä saa muistutuksen (notifikaation äänellä) lääkkeen annostuksista ottoaikoina päivittäin. Jos hälytys otetaan pois, muistutukset lakkaavat.
Käyttäjä voi lisätä lääkkeelle erikoisannostuksen.	Lääkkeen lisäysnäkymässä kentät eri annostiheyksille (x kertaa päivässä, tunnissa, viikossa, tietyt viikonpäivät, toistuva kierto). Muistutukset tulevat erikoisannostusten ottoaikojen ja -päivien mukaan.

Liite 6. Esimerkkejä sovelluksen näkymistä






Lisää uusi lääke
Lääkkeen nimi:
Burana
Lääkkeen vahvuus:
800 mg
Lääkemuoto/annostuksen yksikkö:
tabletti
LISÄÄ ANNOS LÄÄKKEELLE
määrä: 1; aika: 8:30
POISTA
Otetaan tarvittaessa
Laita hälytys
TALLENNA LÄÄKKEEN TIEDOT
PERUUTA
Lääkkeen tiedot
Crestor 10 mg, tabletti
POISTA LÄÄKE
Vaihda lääkkeen nimi:
Crestor
Vaihda lääkkeen vahvuus:
10 mg
Vaihda lääkemuoto/annoksen yksikkö:
tabletti
Laita hälytys
Otetaan tarvittaessa
TALLENNA MUUTOKSET
Lääkkeen annostus:
määrä: 1; aika: 20:00
LISÄÄ UUSI ANNOS
Annos
Crestor 10 mg, tabletti
Muokkaa annosta: määrä: 1; aika: 20:00
POISTA ANNOS
Valitse annoksen määrä:
20.0
0.25
0.5
Valittu määrä: 1.0
VALITSE AIKA
Valittu aika: 20:00
TALLENNA
PERUUTA


Hälyttävät ilmoitukset
Medilista 20.00
Lääkemuistutus
Crestor 10 mg: ota 1 tabletti klo 20:00

Liite 7. Sovelluksen kehityksen aikana havaitut saavutettavuusongelmat

Kriteeri	Kuvaus
1.3.4 Asento	- vaakatasoon käännettäessä elementit limittyivät päällekkäin
1.4.3 Kontrasti (minimi)	- uuden lääkkeen lisäyspainikkeessa riittämätön kontrastisuhde ikonin ja taustan välillä - riittämätön kontrastisuhde syöttökenttien tekstin taustan ja tekstin välillä - riittämätön kontrastisuhde NumberPicker- ja TimePicker -käyttöliittymäelementissä
1.4.4 Tekstin koon muuttaminen	- suurentamisen jälkeen valintaruutuja ja painikkeita ei pystynyt käyttämään pystyasennossa - Spinner-käyttöliittymäelementissä tekstistä jäi puolet näkymättä
2.5.2 Osoitineleellä tehdyn valinnan peruuttaminen	- poiston napautusta ei pystynyt peruuttamaan siirtämällä kohdistusta pois sormella (pelkkä kuva eikä painike)
3.3.1 Virheen tunnistaminen	- syötevirheen kohtaa ei ilmaistu - ruudunlukija luki syötevirheilmoituksen sekakielellä suomeksi ja englanniksi - syötevirhe häviää orientaation vaihtuessa
4.1.2 Nimi, rooli, arvo	- toistuvilla luettelorivikohteilla oli sama kuvaus - virheellisiä contentDescription-attribuutteja (nimi ei vastannut painikkeen toimintaa)
4.1.3 Tilasta kertovat viestit	- kun annoksen poisto aktivoitiin napauttamalla ruksin kuvaa, ruudunlukija ei lukenut tilaviestiä