Diabetes-sovelluksen toteutus Android-käyttöjärjestelmälle

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus
Kevät, 2023

Juho Glumov

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutus Tiivistelmä

Tekijä Juho Glumov Vuosi 2023

Työn nimi Diabetes-sovelluksen toteutus Android-käyttöjärjestelmälle

Ohjaaja Tommi Lahti

TIIVISTELMÄ

Opinnäytetyössä tutustuttiin mobiilisovelluksen ohjelmointiin käyttäen Java-ohjelmointikieltä, sekä Android Studio ohjelmistoa. Sovelluskehityksen tarkoituksena oli luoda toimiva Android-sovellus, jolla diabeetikot voivat helposti kirjata ylös sokeritasoja. Sovellukselta vaadittiin toimivuus Android-käyttöjärjestelmässä, sekä yksinkertainen ja toimiva käyttöliittymä. Opinnäytetyö toteutettiin itselle kehittääkseni ohjelmointitaitojani.

Opinnäytetyössä perehdyttiin Android Studion käyttöön mobiiliohjelmoinnissa, sekä erilaisiin Java-kirjastoihin, joita hyödynnettiin sovelluksen luomisessa. Lisäksi suunniteltiin yksinkertainen ja helppokäyttöinen käyttöliittymä sovellukselle. Aineistoina käytettiin erilaisia internet-lähteitä, joita hyödynnettiin ohjelmoinnissa. Opinnäytetyö on toiminnallinen.

Opinnäytetyön valmista sovellusta testattiin kolmen eri henkilön avulla. Testauksessa löytyviä ongelmia pohditaan ja ratkotaan. Valmista sovellusta pystyisi kuitenkin vielä kehittämään eteenpäin entistä käyttäjäystävällisemmäksi, sekä tukemaan useampaa mobiililaitetta.

Avainsanat Java, Android, Android Studio, Diabetes, SQLite

Sivut 29 sivua ja liitteitä 3 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology

Author Juho Glumov

Subject Diabetes application for Android operating system

Supervisor Tommi Lahti

Abstract

Year 2023

ABSTRACT

The purpose of this thesis was to create a working application for mobile devices that runs on the Android operating system. The application was created using Java programming language and Android Studio software. The application would serve as a tool for diabetics for an easier way of documenting sugar levels, carbohydrates and used insulin units. The thesis was made for myself to improve my programming skill set.

In this thesis, we familiarize ourselves with Android Studio software and different types of Java libraries that were used in the application. We also learn how to create a simple and user-friendly user interface for our application. Different types of internet materials were used for programming the software.

The application made in this thesis was tested by three different persons. The problems found in this testing were studied and some were fixed. The application could still be improved with better and more user-friendly user interface, and with wider device support, to make the application run on a larger variety of devices.

Keywords Java, Android, Android-Studio, Diabetes, SQLite

Pages 29 pages and appendices 3 pages

Sanasto

Java	Työssä käytettävä ohjelmointikieli
Hypoglykemia	Verensokerin liian alhainen taso
Hyperglykemia	Verensokerin liian korkea taso
Variable	Ohjelmoinnissa käytettävä termi muuttujalle
Android Studio	Ohjelmointiympäristö
IntelliJ	Ohjelmointiympäristö
Gradle	Rakennusjärjestelmä
SDK	Ohjelmistokehityspaketti (Software development kit)

Sisällys

Kuvat ja ohjelmakoodit

Liitteet

Liite 1 Aineistonhallinta

Liite 2 PDF-Tiedostoon piirtäminen

1 Johdanto

Diabetes on yleistynyt suomessa, jonka seurauksena sitä kutsutaan kansantaudiksi. Suomessa on noin 50,000 tyypin 1 diabeetikkoa, jotka joutuvat elämään sairautensa kanssa lopun elämänsä. Opinnäytetyössäni suunnittelen ja kehitän sovelluksen Android-puhelimille, jonka avulla tyypin 1 diabeetikot voivat helposti kirjata mitatut sokerit, syödyt hiilihydraatit sekä pistetyt insuliinit.

Opinnäytetyössä tehdään Android-sovellus alusta loppuun, eli ohjelmistojen asennuksesta sovelluksen käyttöönottoon puhelimessa. Keskityn opinnäytetyössä kuitenkin pääsääntöisesti itse sovellukseen ja sovelluksen ominaisuuksiin, sekä ulkonäköön ja käytettävyyteen, enkä niinkään sovelluksien tekoon aloittelijan näkökulmasta, joten kerron näistä vain lyhyesti. Teen opinnäytetyön itselleni harjoittaakseni Java-ohjelmointitaitojani. Tämä on ensimmäinen mobiilisovellus jonka luon käyttäen Java-ohjelmointikieltä.

Sovelluksessa käytettävyys on tärkeässä roolissa. Sovelluksen käyttöliittymän tulee olla selkeä, helposti käytettävä sekä innovatiivinen. Sovelluksen testaamiseen ollaan pyydetty kahta diabeetikkoa testaamaan käytettävyyttä käyttäjän perspektiivistä.

Tutkimuskysymykset:

- Mitä dataa sovelluksessa pitäisi pystyä hallitsemaan?
- Miten ohjelmoidaan tarvittavat toiminnot datan tallennukseen, muokkaamiseen, poistamiseen ja tulostamiseen?
- Miten valmis sovellus rakennetaan .apk asennustiedostoksi?

2 Sovelluksen idea

Sovelluksen idea on olla oiva työkalu diabeetikoille, joka kulkee kätevästi taskussa. Sovellus antaa mahdollisuuden helppoon tuloksien kirjaamiseen, sekä lukemiseen. Sovelluksella pystyy myös tallentamaan valitut mittaustulokset PDF tiedostoksi, joka mahdollistaa helpon jaettavuuden muun muassa oman diabeteshoitajan kanssa. Nykypäivänä lähes jokaisella on kannossa älypuhelin ympäri vuorokauden, jota pystymme hyödyntämään sovelluksella mittaamiseen ja kirjanpitoon.

2.1 Diabeteksen haasteet

Diabetesta on kahta erilaista, tyypin 1-, ja tyypin 2 diabetes. Tyypin 1 diabetes on sisäsyntyinen tulehdussairaus johon kuka tahansa voi elämäntavoista riippumatta sairastua. Tyypin 2 diabetes puolestaan johtuu elämäntavoista ja useimmiten alkaa vasta aikuisiän saavuttamisen jälkeen. Opinnäytetyössä keskitymme vain tyypin 1 diabetekseen. (Ilanne-Parikka, 2021)

Diabetes hankaloittaa normaalia elämistä, ja sen hoidossa hyvä tasapaino on kaikki kaikessa. Diabeetikon täytyy pystyä ylläpitämään hyvä sokeritasapaino ympäri vuorokauden. Tasapainon ylläpitäminen voi olla erittäin hankalaa aktiivisessa elämässä. Sokereiden mittaaminen ja seuraaminen on suuri osa jokapäiväistä elämää. Diabeetikot joutuvat käyttämään insuliinia pitämään heidän sokeritason kurissa. Tämä tapahtuu yleensä ruokailun yhteydessä injektioneulalla, joka on kynän mallisessa pakkauksessa.

Kirjanpito auttaa diabeetikkoa tunnistamaan hypo- ja hyperglykemian mahdollisia aiheuttajia yhdistämällä sokeritason kellonaikaan, syötyihin hiilihydraatteihin sekä pistettyyn insuliiniin. Sovellus on suunniteltu tällaista kirjanpitoa varten.

2.2 Sovelluksien kartoitus

Kyseiseen tarkoitukseen on jo olemassa useampia sovelluksia, kuten MySugr ja Diabetes:M. Kyseiset sovellukset ovat kuitenkin tehty testaajien mukaan turhan monimutkaiseksi jonka seurauksena he ovat päätyneet käyttämään vanhanaikaisesti kynää, sekä paperia tarvittaessa. Kyseiset sovellukset vaativat kirjautumista, joka nostaa kynnystä sovelluksen käyttöönottoon.

Sovelluksien käyttöliittymässä on testaajien mukaan liikaa informaatioo, jota he eivät hyödynnä päivittäisessä käytössä mitenkään, vaikeuttaen hyödyllisen datan luettavuutta, sekä löytämistä.

Kuva 1 nähdään MySugr ohjelmiston käyttöliittymä. Käyttöliittymästä huomataan viimeisen 14-päivän keskiarvot, poikkeamat, hiilihydraatit ja aktiivisuus.

Kuva 1: MySugr käyttöliittymä (MySugr app, n.d.).

Opinnäytetyössä tehdyn sovelluksen pääpiirre on yksinkertaisuus käyttäjälle. Käyttöliittymässä näytetään vain tarvittavat tulokset, kuten mitatut Sokerit, syödyt hiilihydraatit sekä pistetyt insuliinit. Sovelluksen käyttöönoton tulee olla mutkatonta, eikä kirjautumista vaadita. Mittaustuloksien lisääminen on oltava erittäin helppoa jopa aloittelijalle. Sovellus suunnitellaan vanhemmat käyttäjät pitäen mielessä.

3 Android-ohjelmointi

Androidille voidaan ohjelmoida sovelluksia useilla eri ohjelmointikielillä, kuten Java, Kotlin, C++, C# sekä JavaScript, jota käytetään nyt suuressa huudossa olevan React-Nativen kanssa. Opinnäytetyössä keskitymme vain Javan käyttöön. Työhön valittiin Java parantaakseen jo opittuja alkeita, sekä opettelemaan Android Studio ohjelmiston käyttämistä. Android Studio tukee Javaa ilman lisä-asennuksia.

Android-ohjelmointi Java-kielellä on yleisin tapa kehittää Android-sovelluksia. Java on alustariippumaton kieli, joka on suunniteltu helposti opittavaksi ja käytettäväksi. Android SDK tarjoaa erilaisia Java-luokkia ja -rajapintoja, jotka auttavat kehittäjää luomaan Android-sovelluksia. (Meet Android Studio, n.d.)

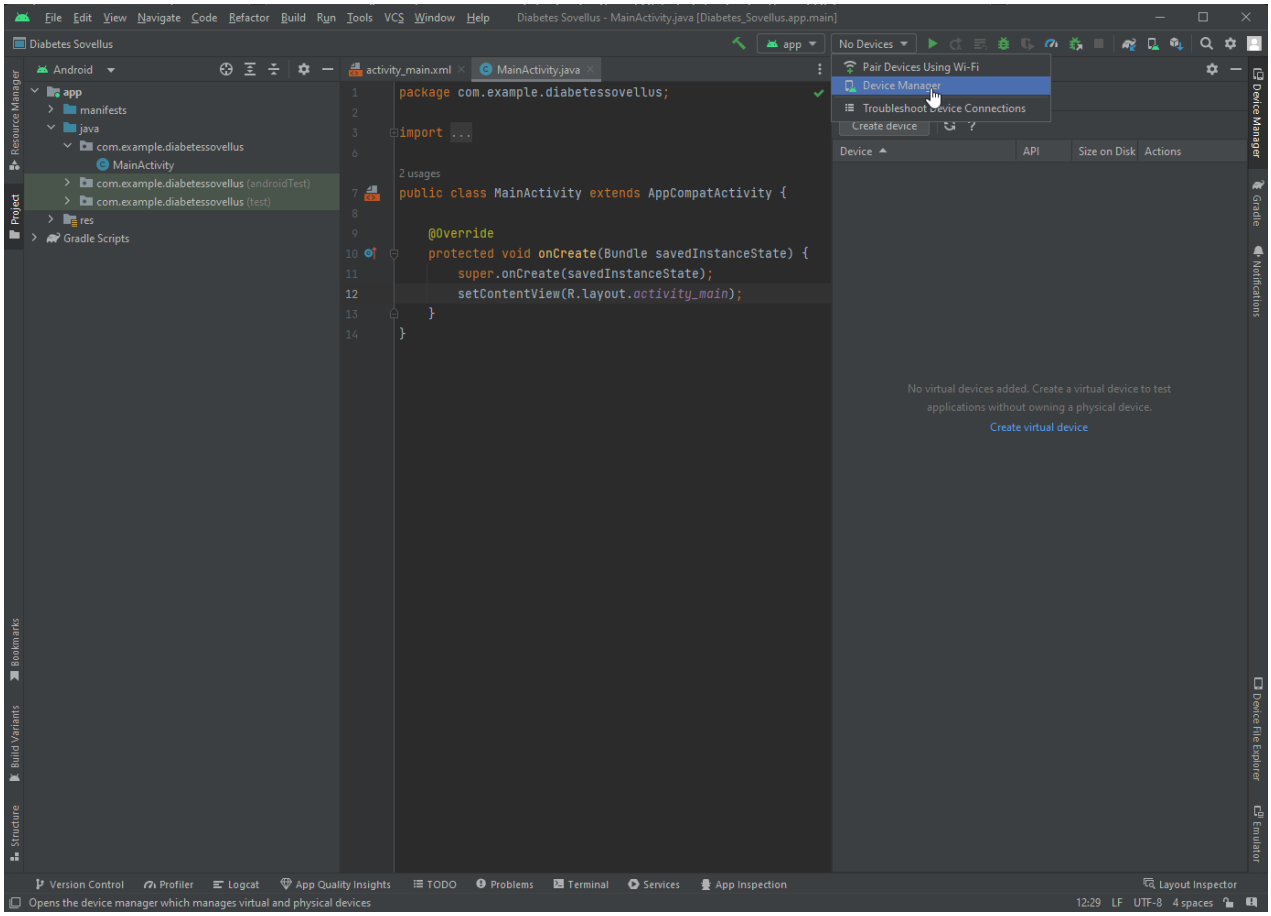
Android-ohjelmointi Java-kielellä on suosittu valinta kehittäjille, sillä se tarjoaa monia etuja, kuten monipuolisuutta, skaalautuvuutta ja helppokäyttöisyyttä. Lisäksi Java-kirjastot ja työkalut ovat laajalti saatavilla ja niitä pystytään helposti soveltamaan Android-sovellusten kehittämisessä.

3.1 Android Studio

Android Studio on Android-sovellusten kehitystyökalu, joka on kehitetty ja julkaistu Googlella. Android Studio on IntelliJ IDEA -pohjainen kehitysympäristö, ja tarjoaa monipuoliset työkalut Android-sovellusten kehittämiseen. Android Studio on myös ilmainen, sekä avoimen lähdekoodin ohjelmisto, tukien Windows, macOS ja Linux -käyttöjärjestelmiä. (Meet Android Studio, n.d.) Android Studio tarjoaa paljon toimintoja, kuten koodieditorin ja emulaattorit. Emulaattorit helpottavat Android ohjelmointia erittäin paljon, sillä voit emuloida Android puhelinta, jolla ohjelmoitava ohjelma on käynnissä. Emulaattoriin pääset käsiksi Andoid Studion Device managerista. Emulaattorin käyttäminen vaatii tietokoneelta virtualisoinnin päälle kytkemistä, joka tapahtuu tietokoneesi bios-asetuksista.

Kuva 2 nähdään Android Studion käyttöliittymä. Device manager löytyy ylävalikosta kuvan näyttämästä kohdasta.

Kuva 2: Android Studio ja Device manager



Kuva 3 nähdään emuloitu Android-mobiilipuhelin. Emuloitu puhelin käyttäytyy kuin fyysinen mobiililaite. Muutoksia voi kuitenkin olla asetuksissa, kuinka mobiililaite kerää dataa sekä käyttää sensoreita.

Kuva 3: Emuloitu Android-mobiilipuhelin

3.2 Sovelluksen julkaiseminen

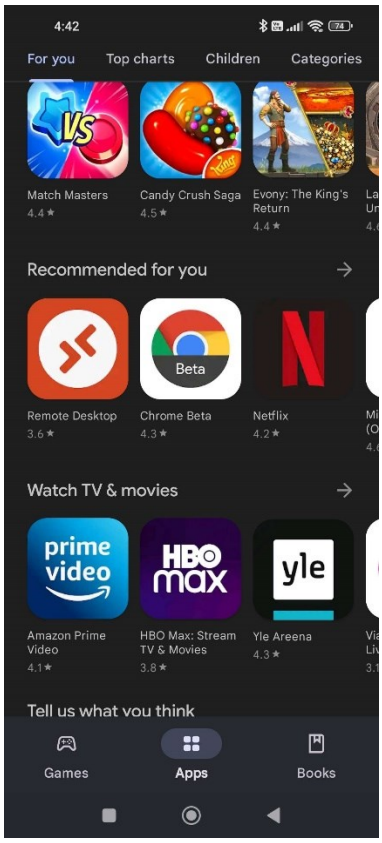
Google Play kauppaan julkaiseminen vaatii ohjelmistokehittäjän tilin. Tilin rekisteröiminen kustantaa 25-dollarin kertamaksun (How to use Play Console, n.d.). Tilin luomisen jälkeen sovellukseen täytyy lisätä allekirjoitettu APK tiedosto. Tämän prosessin pystyy suorittamaan käyttäen Android Studiota, kohdasta Build: Generate signed Bundle/APK.

APK-tiedoston luomisen jälkeen kirjaudutaan ohjelmistokehittäjän konsoliin, josta julkaisu tehdään täyttäen vaaditut tiedot, kuten sovelluksen nimi, kuvaus, kuvakaappaukset ja videot. Allekirjoitettu APK-tiedosto lähetetään julkaisun mukana. Sovelluksen hyväksymisen jälkeen julkaisusi on Play kaupassa, josta kuka tahansa pystyy sen lataamaan. (Create and set up your app, n.d.)

Opinnäytetyössä luotua sovellusta ei toistaiseksi aiota julkaista Play kaupassa, joten ohjelmistovälitys tulee toimimaan lähettämällä asennettava ohjelmisto testaajille. Mahdollisuus julkaista sovellus myöhemmin Play kaupassa pidetään avoimena.

Kuva 4 sisältää Google Play-sovelluksen etusivun. Sovellus on Androidin virallinen kauppapaikka johon kehittäjät voivat julkaista ohjelmistojaan. Play-kaupassa on miljoonia sovelluksia ladattavissa, niin ilmaisena kuin maksullisenakin.

Kuva 4: Google Play kauppa



4 Sovelluksen suunnittelu

Opinnäytetyössä sovelletaan vesiputousmallia projektinhallintamenetelmänä. Kyseisen menetelmä on valittu, sillä sovellus suunnitellaan, sekä toteutetaan yksi. Projekti suunnitellaan alusta loppuun ennen sovelluksen ohjelmointia.

Sovelluksen pääominaisuus on tuloksien kirjanpito. Kirjanpitoon voidaan lisätä mitatut sokerit, syödyt hiilihydraatit, pistetty insuliinin määrä sekä vapaa-sanainen lisätieto. Käyttöliittymä näyttää listana päivän lisätyt tulokse. Nappia painamalla päästään lisäämään uusi tulos.

Sovellukseen luodaan kalenteri, joka näyttää kyseisen päivän. Kalenteria käyttämällä päästään katsomaan kuluneiden päivien tuloksia ja muokkaamaan niitä.

Sovelluksessa on mahdollisuus tallentaa tuloksia PDF-tiedostoksi. Voidaan valita haluttu aikavälin, esimerkiksi viimeinen kulunut viikko ja tallentaa koko viikon tulokset yhteen PDF-tiedostoon. Tästä ominaisuudesta on hyötyä jos tulokset halutaan diabeetikon toimesta lähettää hoitajalle tarkistettavaksi.

Sovellus toteutetaan käyttäen Android Studio ohjelmistoa, sekä Java-ohjelmointikieltä. Kyseinen kieli on valittu aikaisemman kokemuksen perusteella. Sovellus toimii hyvänä harjoituksena Java-mobiiliohjelmointiin.

4.1 Kirjanpito

Kirjanpito pääominaisuus toteutuu käyttäen SQLite tietokantaa. SQLite on maailman käytetyin tietokanta moottori (What is SQLite?, n.d.). Kyseisen moottorin avulla pystytään luomaan tietokanta lokaalisti puhelimeen, jonka tehtävänä on pitää huoli luvuista ja tiedoista, jotka kirjataan ylös sovellukseen. Tietokanta tallentaa lisätyt numerot, josta niitä voidaan muokata tai poistaa. SQLite-tietokanta luodaan ja hallinnoidaan käyttämällä SQLiteOpenHelper-luokkaa. Tämä luokka tarjoaa metodeja tietokannan luomiseen, päivittämiseen ja avaamiseen. Tietokannan sisältöä voidaan hallinnoida käyttämällä SQL-kyselyitä ja tietokantatauluja.

4.2 Kalenteri

Kalenterin idea on näyttää käyttäjälle päivämäärä, sekä luoda mahdollisuus kulkea kalenteripäivissä taaksepäin ja tutkia edeltävien päivien mittaustuloksia. Ominaisuus tehdään käyttäen java.util.Calendar luokkaa. Kyseinen luokka on Java-kirjaston tarjoama luokka, joka tarjoaa täyden kalenterijärjestelmän tukea. Luokka tarjoaa metodeja päivämäärän- ja ajan asettamiseen, hakemiseen, muokkaamiseen sekä kalenterin käyttäytymisen määrittämiseen. Luokka toimii yleisesti kalenterina ja tarjoaa tukea monille eri kalentereille ja aikavyöhykkeille.

4.3 PDF-tallennus

PDF-tallennuksen ideana on mahdollisuus pystyä tallentamaan useamman päivän mittaustuloksia PDF-tiedostoksi, jonka avulla mittaustulokset ovat helppo jakaa oman hoitajan kanssa. Ominaisuus luodaan käyttäen Canvas luokkaa. Canvas on Android-kirjaston tarjoama luokka, joka tarjoaa mahdollisuuden piirtämiseen. Luokka tarjoaa metodeja, jotka mahdollistavat piirto-operaatiot, kuten viivojen, ympyröiden, tekstin, kuvien ja monimutkaisten muotojen piirtämisen. Ominaisuudessa piirretään halutut tiedot käyttäen kyseistä luokkaa ja tallennetaan piirros PDF-tiedostoksi. Tallentamisessa pitää ottaa huomioon tallentamiseen liittyvät luvat Android-käyttöjärjestelmässä.

4.4 Käyttöliittymä

Käyttöliittymä on osa Android-sovellusta, jonka avulla käyttäjä operoi sovellusta. Käyttöliittymä koostuu erilaisista elementeistä, kuten nappeista, tekstikentistä, valikkoista ja kuvista. Käyttöliittymän tarkoituksena on antaa käyttäjälle mahdollisuus käyttää sovellusta ja tarjota tietoa sovelluksen toiminnasta.

Selkeä käyttöliittymä on käyttäjäystävällinen ja helppokäyttöinen. Käyttöliittymä suunnitellaan niin, että käyttäjä voi helposti löytää ja käyttää sovelluksen toimintoja. Selkeä käyttöliittymä noudattaa Android-käyttöliittymän suosituksia, kuten Googlen luomaa Material Design järjestelmää. Selkeä käyttöliittymä on tärkeä osa hyvän Android-sovelluksen kehitystä, sillä se parantaa käyttäjäkokemusta ja edistää sovelluksen käyttöä.

Kuva 5 sisältää ensimmäisen mallinnuksen suunnitellusta käyttöliittymästä. Käyttöliittymän etusivulla on päivän tulokset listattuna, painikkeesta päästään lisäämään uusi tulos. Tulosta painamalla päästään katsastamaan lisätietoja tuloksesta. Halutessa käyttäjä voi myös poistaa tuloksen.

Kuva 5: Alkuperäinen suunnitelma käyttöliittymälle

5 Valmiin sovelluksen esittely

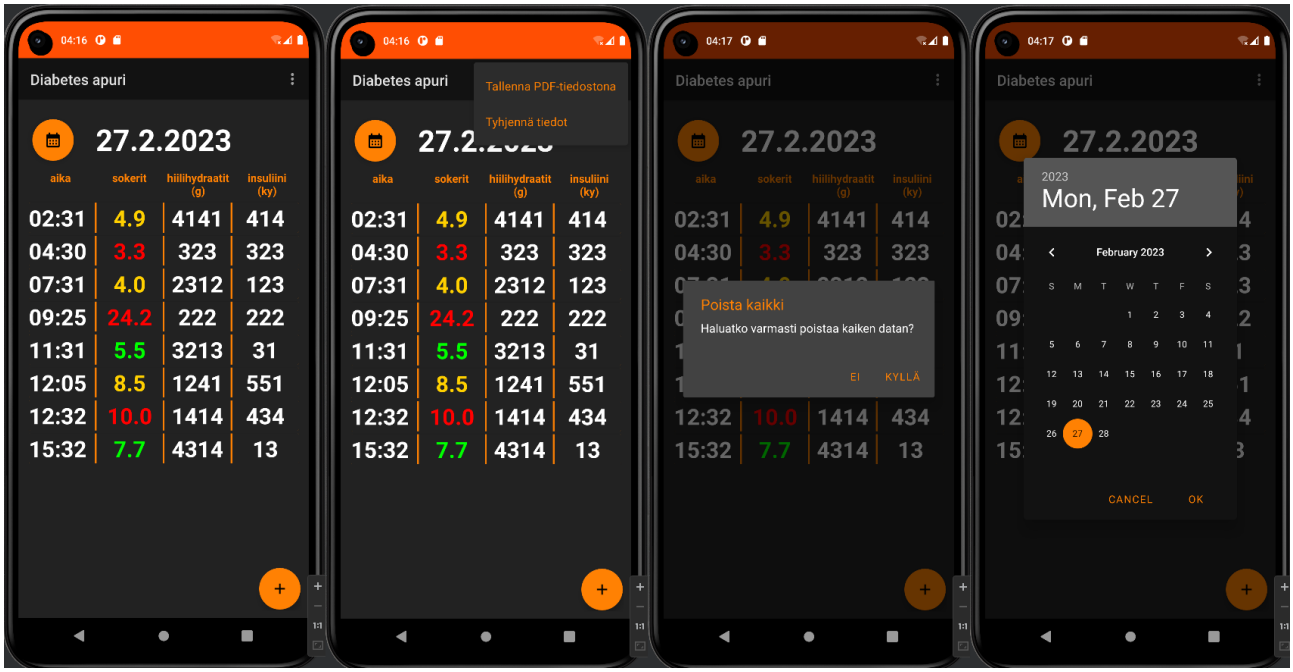
Tästä kappaleesta eteenpäin kirjoitetaan näkökulmasta, jossa sovelluksen ensimmäinen version on valmis, sekä jaettu testaukseen.

Sovelluksen ulkonäön pääpiirteet pysyivät hyvin samankaltaisina suunnitelman kanssa, vaikkakin ulkonäöstä on tehty kauniimpi ja selkeämpi käyttäjälle. Kuva 6 nähdään sovelluksen päänäkymä. Päänäkymä hakee automaattisesti kuluvan päivän kalenteriin ja näyttää lisätyt mittaustulokset kyseiselle päivälle. Oikean-alakulman plus-napista päästään lisäämään uusi tulos. Mittaustulosta painamalla siirrytään tuloksen lisätietoihin, josta tulosta voidaan muokata tai poistaa. Mittaustuloksien sokereiden fontin värit vaihtelevat liikennevalomallin mukaisesti. Vihreä tarkoittaa hyvää tulosta, keltainen kohtaloista ja punainen huonoa.

Työkalupalkki sisältää valikon, josta löytyvät painikkeet PDF-tallennukselle ja kaikkien mittaustuloksien poistamiselle. Valikon ”tyhjennä tiedot” painikkeen valitseminen varmistaa ettei painiketta olla painettu vahingossa kysyen ”haluatko varmasti poistaa kaiken datan?”. Valitsemalla kyllä, poistetaan jokainen talletettu mittaustulos.

Päänäkymä sisältää kalenterin. Kalenterilla voidaan kulkea menneissä päivissä jos halutaan tutkia, lisätä tai muokata kuluneiden päivien tuloksia. Kalenteri hakee automaattisesti nykyisen päivän sovelluksen avautuessa.

Kuva 6: Sovelluksen päänäkymä

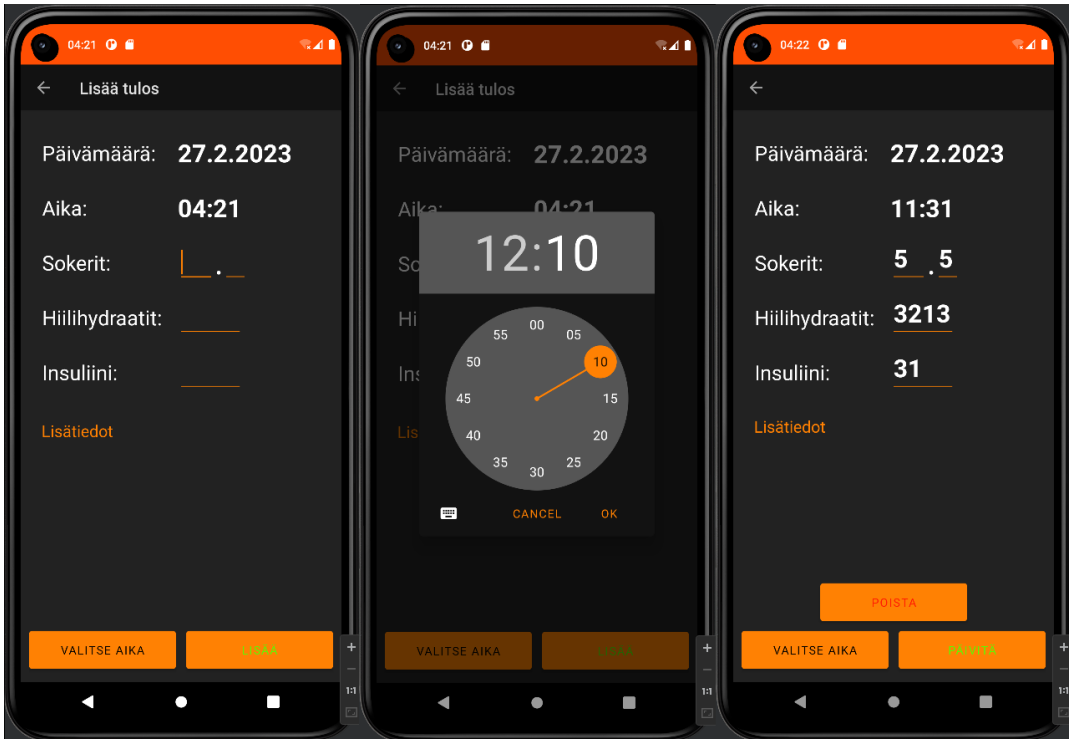


Kuva 7: Sovelluksen Lisäys-, ja muokkausnäkymä. Päänäkymästä siirtyy päivämäärä tälle näkymälle. Aika haetaan automaattisesti käyttöjärjestelmän kellosta. Aikaa pystytään kuitenkin halutessa muuttamaan manuaalisesti ”Valitse aika” painikkeesta. Mittaustulokseen voidaan päivän ja kellonajan lisäksi kirjata mitatut sokerit, syödyt hiilihydraatit, pistetty insuliini sekä Vapaavalintaisesti lisätietoja. Jokaisen kentän voi kuitenkin jättää tyhjäksi halutessa. Kentän täydennettyä seuraavaan kenttään siirtyminen onnistuu kätevästi enter-painikkeesta. Lisää-painike tallentaa kirjatut tulokset tietokantaan, josta niitä voidaan katsastella päänäkymästä.

Näkymässä huomataan myös kello. Tämän avulla muutetaan tulokselle haluttu kellonaika. Kellonaika hakeutuu automaattisesti tämänhetkiseen kellonaikaan. Kello-näkymän saa auki painamalla ”valitse aika” painiketta.

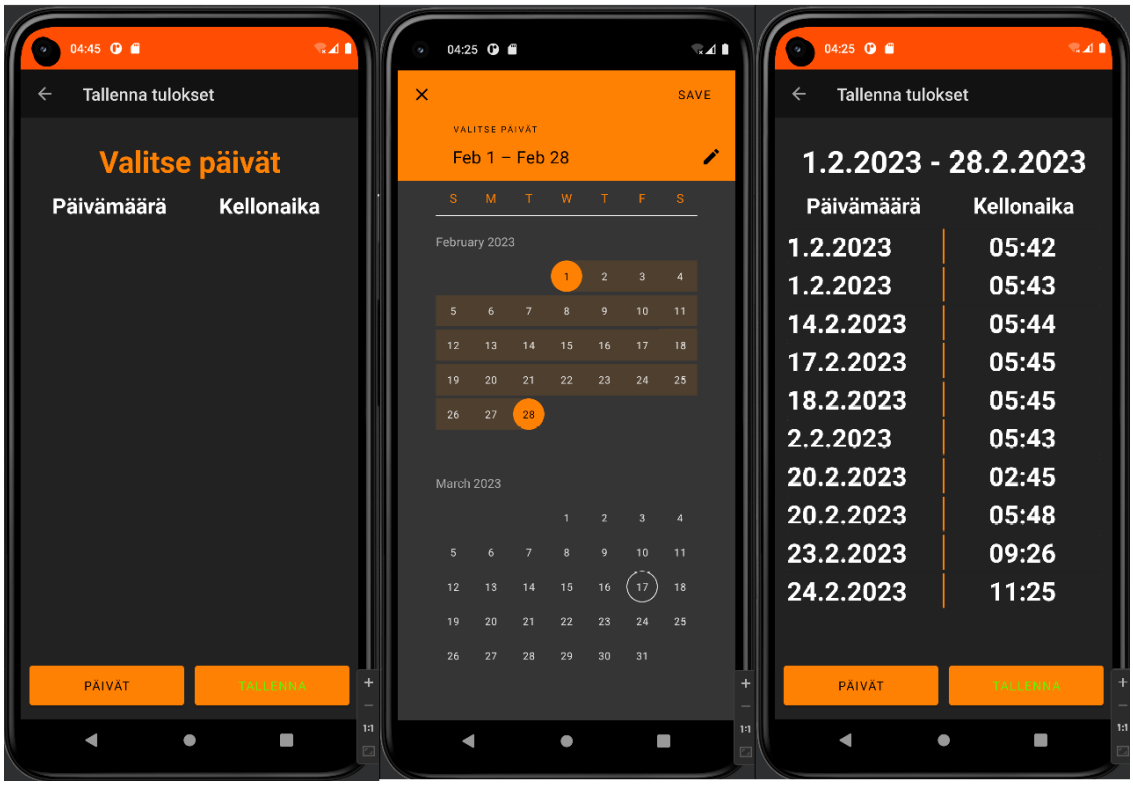
Tuloksia päästään muokkaamaan päänäkymästä painaen mittaustulosta jota halutaan muokata. Tässä näkymässä voidaan muokata tuloksen kaikkia tietoja paitsi päivämäärää. Tuloks pystytään myös poistamaan ”poista” painikkeesta. Uusien tietojen jälkeen ”päivitä” nappi päivittää tuloksen tietokantaan ja palauttaa käyttäjän päänäkymään.

Kuva 7: Sovelluksen Lisäys-, ja muokkausnäkymä



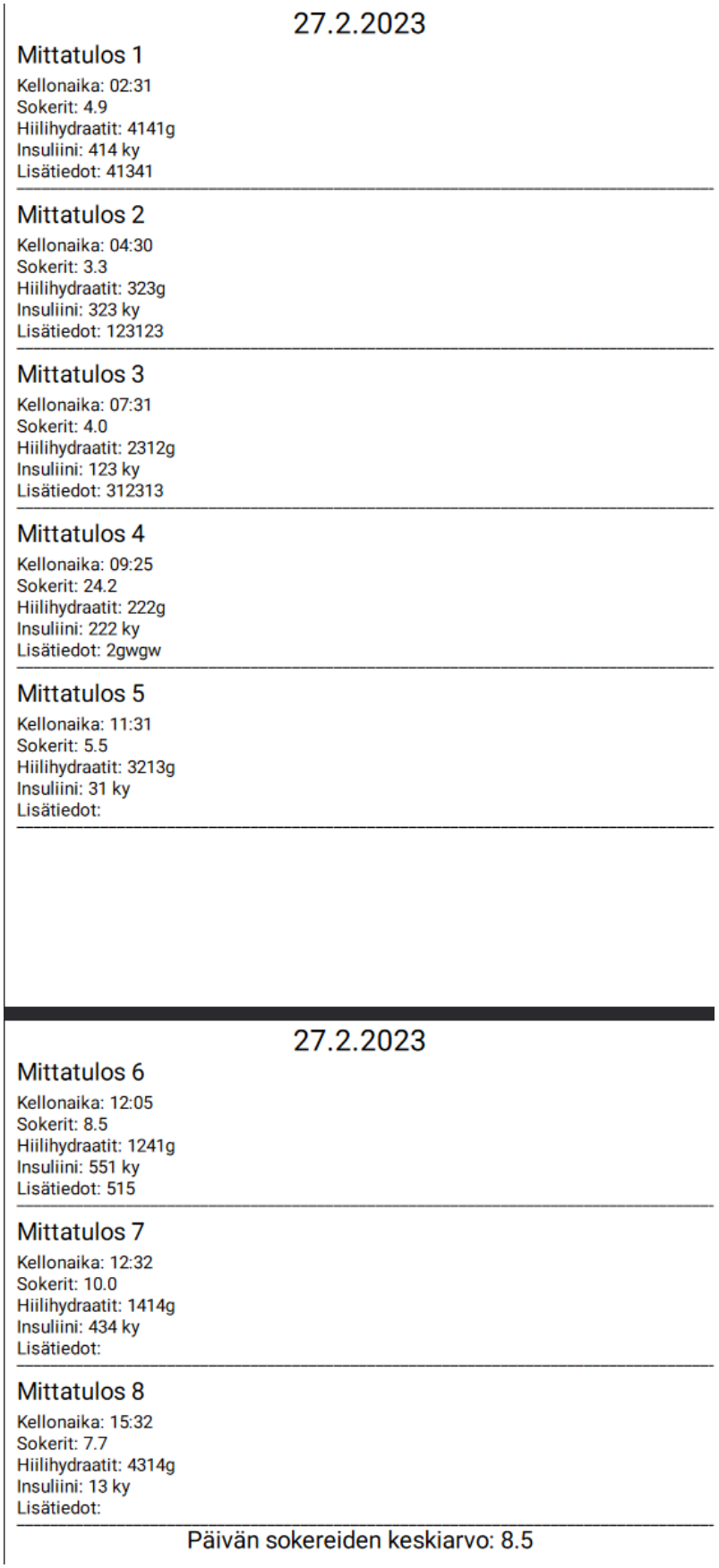
Kuva 8: Sovelluksen PDF-tallennuksen näkymän. Kyseiseen näkymään päästään päänäkymän työkalupalkista. Käyttäjän tarvitsee valita haluamat päivät joilta tulokset halutaan tallentaa PDF-tiedostoksi. Päivät voidaan valita ”Päivät” painikkeesta. Painike avaa Kalenteri-näkymä, josta valitaan päivät. Kuvassa on valittu päivät 1-28.2.2023. Valittujen päivien jälkeen näkymä näyttää valituilta päiviltä jokaisen tuloksen lyhyesti. Tuloksista näytetään vain päivämäärä ja kellonaika. ”Tallenna” painikkeesta sovellus tallentaa PDF-tiedostona valittujen päivien tulokset. Tiedosto tallentuu Android-käyttöjärjestelmässä sijaitsevaan Download kansioon.

Kuva 8: Sovelluksen PDF-tallennuksen näkymä



Kuva 9: Tallennettu PDF-tiedostosta kaksi sivua. Mittaustulokset numeroidaan päivittäin ja päivämäärä löytyy sivun alusta. Tiedostossa näkyy myös päivän sokereiden keskiarvo sekä 7 päivän välein viikon keskiarvo. Tiedosto tallentuu A4 paperin koossa.

Kuva 9: Tallennettu PDF-tiedosto



27.2.2023

Mittatulos 1

Kellonaika: 02:31
Sokerit: 4.9
Hiilihydraatit: 4141g
Insuliini: 414 ky
Lisätiedot: 41341

Mittatulos 2

Kellonaika: 04:30
Sokerit: 3.3
Hiilihydraatit: 323g
Insuliini: 323 ky
Lisätiedot: 123123

Mittatulos 3

Kellonaika: 07:31
Sokerit: 4.0
Hiilihydraatit: 2312g
Insuliini: 123 ky
Lisätiedot: 312313

Mittatulos 4

Kellonaika: 09:25
Sokerit: 24.2
Hiilihydraatit: 222g
Insuliini: 222 ky
Lisätiedot: 2gwgw

Mittatulos 5

Kellonaika: 11:31
Sokerit: 5.5
Hiilihydraatit: 3213g
Insuliini: 31 ky
Lisätiedot:

27.2.2023

Mittatulos 6

Kellonaika: 12:05
Sokerit: 8.5
Hiilihydraatit: 1241g
Insuliini: 551 ky
Lisätiedot: 515

Mittatulos 7

Kellonaika: 12:32
Sokerit: 10.0
Hiilihydraatit: 1414g
Insuliini: 434 ky
Lisätiedot:

Mittatulos 8

Kellonaika: 15:32
Sokerit: 7.7
Hiilihydraatit: 4314g
Insuliini: 13 ky
Lisätiedot:

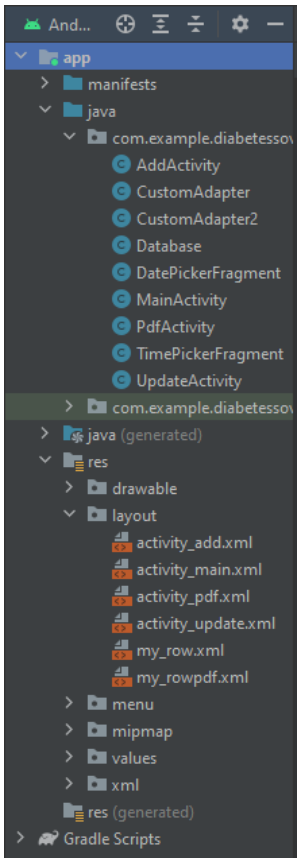
Päivän sokereiden keskiarvo: 8.5

6 Sovelluksen ohjelmointi

Android Studiolla sovelluksen ohjelmointi perustuu aktiviteetteihin. Aktiviteetit ovat esimerkiksi sovelluksen sivuja ja muita näkymiä. Aktiviteetit koostuvat kahdesta osasta, Java-tiedostosta, joka sisältää aktiviteetin ohjelmoidut toiminnot, sekä .xml tiedostosta, joka sisältää näkymän ulkonäön erilaisilla elementeillä.

Kuva 10 sisältää sovelluksen aktiviteetit. Päänäkymä ”MainActivity.java + activity_main.xml”, Lisäysnäkymä ”AddActivity.java + activity_add.xml”, Päivitysnäkymä “UpdateActivity.java + activity_update.xml” sekä PDFnäkymä “PdfActivity.java + activity_pdf.xml”. Kuvassa näkyvät muut tiedostot eivät ole aktiviteetteja, vaan lisäyksiä muihin aktiviteetteihin, kuten tuloksien rivit, kalenteri, kellonaika sekä tietokanta luokka.

Kuva 10: Sovelluksen aktiviteetit



Sovellukset sisältävät myös ”AndroidManifest.xml” tiedoston, joka löytyy kansion ”manifests” alta. Tiedosto sisältää asetukset itse sovellukselle ja jokaiselle aktiviteetille erikseen.

Ohjelmakoodi 1 sisältää AndroidManifest tiedostossa olevia konfiguraatioita. Label tarkoittaa otsikkoa, joka näkyy työkalupalkissa. ScreenOrientation liittyy sovelluksen kääntymiseen puhelimen kääntyessä, tässä tapauksessa pidämme sovelluksen muotokuva tilassa. parentActivityName kertoo aktiviteetin vanhemman, jonka avulla pääsemme työkalupalkin takaisin nuolesta edelliseen aktiviteettiin.

Ohjelmakoodi 1: AndroidManifest konfiguraatioita

```
android:label="Diabetes apuri"
android:screenOrientation="portrait"
android:parentActivityName=".MainActivity"
```

Ohjelmakoodi 2 ja Ohjelmakoodi 3 sisältävät Android-ohjelmoinnissa käytettäviä oikeuksia. Joudumme pyytämään käyttäjältä oikeuksia käyttääksemme käyttöjärjestelmän ominaisuuksia, kuten gps paikanninta, kameraa, mikrofonia tai tallennustilaa. Tämä tapahtuu AndroidManifest-konfiguraatiotiedostossa, sekä Java-tiedostossa jossa kyseistä oikeutta tarvitaan. Sovelluksessa tarvittiin käyttäjältä oikeudet tallennustilaan PDF-tallennuksen vuoksi.

Ohjelmakoodi 2: AndroidManifest konfiguraatiossa sijaitsevat asetukset lupiin.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.MANAGE_EXTERNAL_STORAGE"/>
```

Ohjelmakoodi 3: PDF aktiviteetissa pyydettävät luvat.

```
int PERMISSION_ALL = 1;
String[] PERMISSIONS =
{Manifest.permission.READ_EXTERNAL_STORAGE,Manifest.permission.WRITE_EXTERN
AL_STORAGE};

        if (!hasPermissions(this, PERMISSIONS)) {
            ActivityCompat.requestPermissions(this, PERMISSIONS,
PERMISSION_ALL);
        }

        ActivityCompat.requestPermissions(this,
                new  String[]{WRITE_EXTERNAL_STORAGE,
READ_EXTERNAL_STORAGE,MANAGE_EXTERNAL_STORAGE},
                PackageManager.PERMISSION_GRANTED);
```

Sovelluksessa ollaan pyydetty käyttäjältä kolmea eri lupaa tallennustilaan: Lupaa kirjoittaa, lukea sekä muokata. PDF-tallennukseen tarvitaan ainoastaan lupa kirjoittaa, jonka vuoksi kaksi muuta

lupaa pyydetään suotta. Tämä on huono tapa, sillä lupia pitäisi pyytää vain sen perusteella mitä ohjelman käyttäminen vaatii, ei ylimääräisiä.

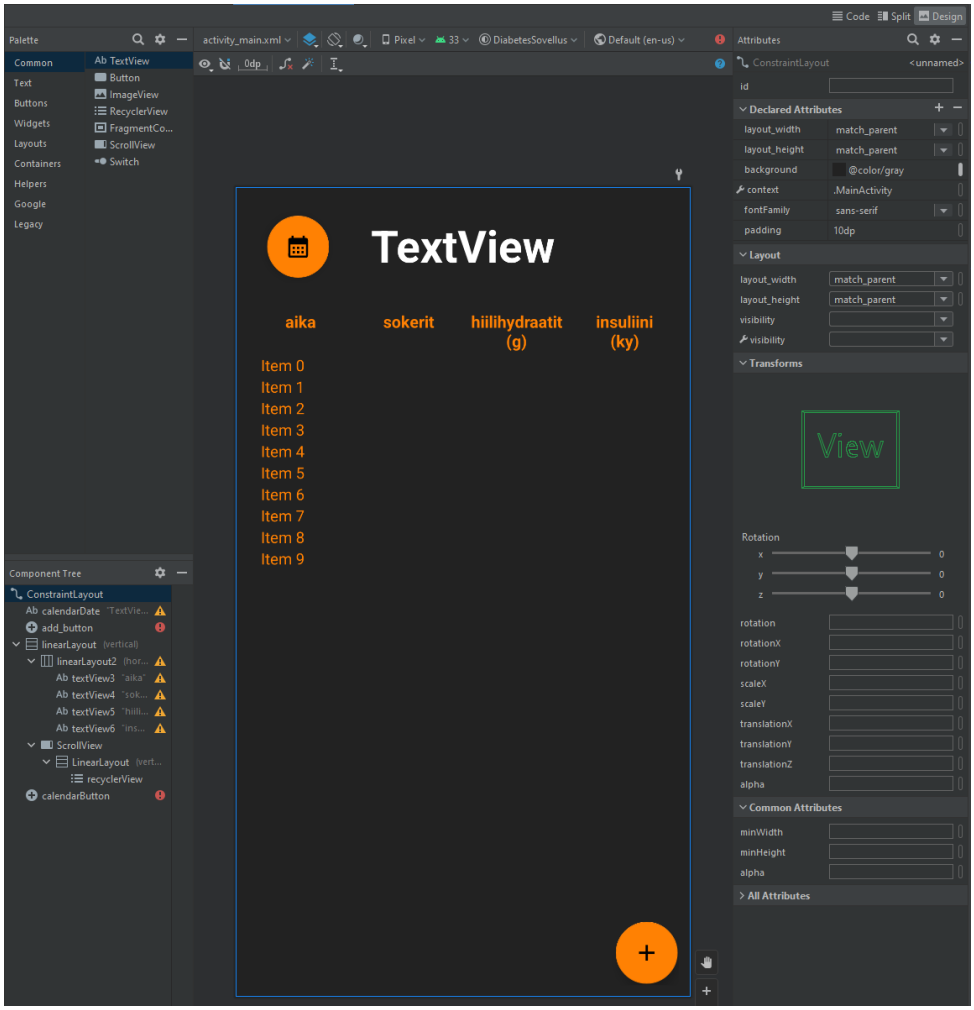
6.1 Käyttöliittymä

Käyttöliittymän luominen tapahtuu .xml tiedostojen kautta. Tiedostossa on käyttöliittymän jokainen elementti ja niille asetetut attribuutit koodina. Android Studio sisältää järjestelmän, jolla pystytään lisäämään erilaisia elementtejä helposti valikosta. Elementin lisääminen valikosta lisää tarvittavan koodipätkän kyseiselle elementille, eikä sitä näin tarvitse kirjoittaa käsin. Elementeillä on erilaisia attribuutteja joilla ne saadaan pysymään halutuissa paikoissa, sekä näyttämään halutunlaiselta. Elementtejä on useita erilaisia, sovelluksessa käytetään painikkeita, tekstikenttiä, säiliöitä sekä pohjapiirustuksia.

Elementit ovat aina pohjapiirustuksen, eli layoutin sisällä. Layoutteja voi olla näkymässä useita ja nämä helpottavat elementtejen sijoittelua haluttuun tapaan. Elementti-puusta näemme helposti eri layoutit, containerit ja komponentit näiden sisällä.

Kuva 11 nähdään päänäkymän aktiviteetin muotoilunäkymä. Vasemmalla ylhäällä on valikko, josta komponentteja voidaan helposti lisätä. Vasemmalla alhaalla on komponentti-puu, jossa näkyy aktiviteetin jokainen komponentti. Oikealla näkyy valitun komponentin attribuutteja joita pystytään muokkaamaan tämän kautta.

Kuva 11: Muotoilunäkymä

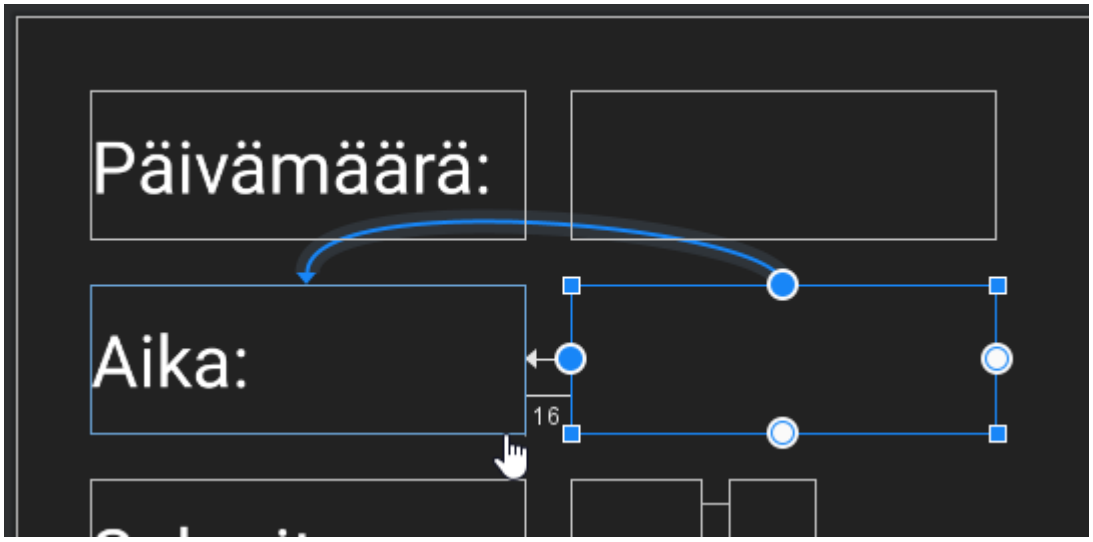


Muotoilunäkymän lisäksi on koodinäkymä. Aktiviteettiin lisätyt elementit ja niille asetetut attribuutit näkyvät täällä. Pystymme helpommin lisäämään ja muokkaamaan attribuutteja koodinäkymästä kuin muotoilunäkymästä.

Elementtien tärkein attribuutti on ”constraint”. Tämä attribuutti kertoo missä kohtaa komponentin pitäisi istua näkymässä. Jokaisella komponentilla tulisi olla vähintään kaksi eri constraint attribuuttia, yksi X-akselille ja toinen Y-akselille. Muotoiluikkunassa pystymme helposti lisäämään kyseisen attribuutin vetämällä komponentin reunoista joko toiseen komponenttiin tai layoutin reunoihin.

Kuva 12 sisältää constraint-attribuutin. Näemmä kuinka valittu elementti on yhdistettynä toiseen attribuutin avulla.

Kuva 12: Constraint-attribuutti



6.2 SQLite

Sovelluksessa käytetään SQLiteä tietokantaratkaisuna. Tietokantaa varten sovelluksessa on database.java luokka tietokannalle. Luokassa on erilaisia metodeita tietokannan käyttämiseen, kuten lukeminen, lisäys, päivitys sekä poistaminen.

Kuva 13 sisältää käytettävän tietokannan rakenteen. Ymmärtääksemme paremmin koodia, tarvitsee meidän tietää tietokannan rakenne. Tietokannassa on seitsemän saraketta: id, date, time, sokerit, hiilarit, insuliini, sekä info. Sarake id on primary key ja luodaan automaattisesti. Muut sarakkeet ovat yksinkertaisesti stringejä. Päivämäärien tallentaminen stringeinä on huono tapa, sovelluksessa olisi kannattanut alusta asti käyttää käyttötarkoitukseen tehtyjä formaatteja, joka olisi helpottanut ominaisuuksien ohjelmoinnissa.

Kuva 13: Tietokannan rakenne

	id	date	time	sokerit	hiilarit	insuliini	info
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	43	27.2.2023	09:25	24.2	222	222	2gwgw
2	44	26.2.2023	09:25	43.3	2	234	3yywrhg
3	45	25.2.2023	09:25	23.4	2262	235	estwt
4	46	24.2.2023	11:25	32.4	423	33	gsgsgwe
5	47	23.2.2023	09:26	23.2	22	22	wgwgwgw
6	48	27.2.2023	04:30	3.3	323	323	123123
7	49	27.2.2023	07:31	4.0	2312	123	312313
8	50	27.2.2023	02:31	4.9	4141	414	41341
9	51	27.2.2023	11:31	5.5	3213	31	
10	52	27.2.2023	15:32	7.7	4314	13	
11	53	27.2.2023	12:05	8.5	1241	551	515
12	54	27.2.2023	12:32	10.0	1414	434	
13	55	1.2.2023	05:42	12.2	22	41	joo kylla nain on tietty
14	56	1.2.2023	05:43	12.3	1231	23	aafiaffa
15	57	2.2.2023	05:43	7.5	35	23	kyll
16	58	9.2.2023	05:43	34.4	66	54	faghhb
17	59	14.2.2023	05:44	0.2	32	234	123131
18	60	17.2.2023	05:45	1.2	4222	342	heppa
19	61	20.2.2023	02:45	2.2	14	314	juppe
20	62	18.2.2023	05:45	14.5	241	145	asasdqqwwrq
21	63	20.2.2023	05:48	11.2	213	114	fqfqq
22	64	8.2.2023	05:48	22.2	1231	13	fafafa
23	65	3.3.2023	00:10	22.2	222	22	moi
24	66	3.3.2023	04:12	0.0			

Tietokanta luokka Database.java sisältää sovelluksen database-luokan metodit tietokannan luomiseen. Luettavuuden ja muokattavuuden helpottamiseksi, on luotu variablet tietokannan nimelle, versiolle, sekä sarakkeiden nimille. Ensimmäistä metodia readData() käytetään sovelluksen pääsivulla näyttääkseen tulokset päivän mukaan. Toinen metodi readDataArray(), on käytössä PDF-tallennuksen sivulla ja hakee tulokset useammalta päivältä.

Luokka sisältää myös metodit tietokannasta poistamiseen sekä muokkaamiseen. Muokkaamista käytetään update-näkymästä, jossa tulosta voidaan muokata halutuksi. Update-näkymässä voidaan myös halutessa poistaa koko tulos käyttäen deleteOneRow() metodia. Toinen metodi poistamiselle on deleteAllData(). Metodi poistaa jokaisen tallennetun tuloksen tietokannasta. Tätä metodia käytetään päänäkymän työkalupalkista.

6.3 Kello ja kalenteri

Sovelluksessa aikaan ja päivämäärään liittyvät ominaisuudet ovat tehty Javan Calendar luokkaa käyttäen. Luokka mahdollistaa helpon päivämäärän sekä ajan hakemisen. Ohjelmakoodi 4 haetaan päänäkymään sovelluksen käynnistyessä tämänhetkinen päivämäärä, sekä asetetaan päivämäärä näkymään otsikkoon. Calendar.getInstance() metodi hakee päivämäärän, jonka jälkeen haetaan päivämäärästä vuosi, kuukausi sekä päivä, jotka tallennetaan variableihin.

Ohjelmakoodi 4: Päänäkymän päivämäärä

```
Calendar c = Calendar.getInstance();
int year = c.get(Calendar.YEAR);
int month = c.get(Calendar.MONTH);
int day = c.get(Calendar.DAY_OF_MONTH);
int realMonth = month + 1;
String currentDateString = day + "." + realMonth + "." + year;
calendarDate = findViewById(R.id.calendarDate);
calendarDate.setText(currentDateString);
```

Ohjelmakoodi 5 ja Ohjelmakoodi 6 nähdään kalentereiden ja kellonajan valitsemisessa käytettävä DialogFragment luokka, joka mahdollistaa uuden näkymän lisäämistä näkymän päälle. Tämän avulla voidaan tuottaa uusi näkymä ilman siirtymistä vanhasta näkymästä pois.

Ohjelmakoodi 5: TimePickerFragment luokka kellolle

```
public class TimePickerFragment extends DialogFragment {
    @NonNull
    @Override
    public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
        Calendar c = Calendar.getInstance();
        int hour = c.get(Calendar.HOUR_OF_DAY);
        int minute = c.get(Calendar.MINUTE);
        return new TimePickerDialog(getActivity(),
(TimePickerDialog.OnTimeSetListener) getActivity(), hour, minute,
DateFormat.is24HourFormat(getActivity()));
    }
}
```

Ohjelmakoodi 6: DatePickerFragment luokka kalenterille

```
public class DatePickerFragment extends DialogFragment {
    @NonNull
    @Override
    public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
        Calendar c = Calendar.getInstance();
        int year = c.get(Calendar.YEAR);
        int month = c.get(Calendar.MONTH);
        int day = c.get(Calendar.DAY_OF_MONTH);


        return new DatePickerDialog(getActivity(),
(DatePickerDialog.OnDateSetListener) getActivity(), year, month, day);
    }
}
```

Ohjelmakoodi 7 huomataan kuinka Fragment luokkaa käytetään päänäkymässä. Kyseistä metodia kutsutaan kun kalenterista on valittu haluama päivämäärä. Metodi hakee kyseisen päivämäärän, sekä sijoittaa päivämäärän näkymään pääsivun otsikkoon. Metodissa kutsutaan myös toista metodia, joka hakee tietokannasta tallennetut tulokset kyseisestä päivästä.

Ohjelmakoodi 7: Fragment-luokan käyttäminen

```
@Override
    public void onDateSet(DatePicker view, int year, int month, int
dayOfMonth) {
        Calendar c = Calendar.getInstance();
        c.set(Calendar.YEAR, year);
        c.set(Calendar.MONTH, month);
        c.set(Calendar.DAY_OF_MONTH, dayOfMonth);
        int realMonth = month + 1;

        String currentDateString = dayOfMonth + "." + realMonth + "." +
        year;

        calendarDate = findViewById(R.id.calendarDate);
        calendarDate.setText(currentDateString);

        storeDataInArrays();
        customAdapter = new CustomAdapter(MainActivity.this, this, time,
      date,sokerit, hiilarit, insuliini, info, id);
        recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setAdapter(customAdapter);
        recyclerView.setLayoutManager(new
LinearLayoutManager(MainActivity.this));
    }
```

Ohjelmakoodi 8 huomataan kalenterin käyttäminen PDF-tallennuksessa. Tämä metodi on lievästi monimutkaisempi kuin aikaisempi, sillä käyttäjä valitsee kaksi päivämäärää joiden välistä valitaan jokainen päivä. Haetut päivät tallennetaan millisekunteina.

Tämän ohjelmoinnissa törmättiin ongelmaan jossa käyttäjän valitsemat päivät olivat aina yhdellä pielessä. Tämä ongelma saatiin korjattua miinustamalla ensimmäisestä tallennetusta päivästä päivän verran millisekunteja, eli 86400000. Metodin lopputuloksena saamme tallennetut tulokset jokaiselta valitulta päivältä.

Ohjelmakoodi 8: Kalenteri PDF-tallennuksessa

```
private void showDatePickerDialog() {

      dates = new ArrayList<>(); // the list to store all the dates
      MaterialDatePicker.Builder<Pair<Long, Long>> builder =
      MaterialDatePicker.Builder.dateRangePicker();
      builder.setTitleText("Valitse päivät");
      MaterialDatePicker<Pair<Long, Long>> materialDatePicker =
      builder.build();
      materialDatePicker.show(getSupportFragmentManager(), "Test");
      materialDatePicker.addOnPositiveButtonClickListener(new
      MaterialPickerOnPositiveButtonClickListener<Pair<Long, Long>>() {
            @Override
            public void onPositiveButtonClick(Pair<Long, Long> selection) {
                  if (selection.first != null && selection.second != null) {
                     // start date
                     Calendar start = Calendar.getInstance();
                     start.setTimeInMillis(selection.first);
                     Long time = start.getTimeInMillis() - 86400000;
                     start.setTimeInMillis(time);
                     // end date
                     Calendar end = Calendar.getInstance();
                     end.setTimeInMillis(selection.second);

                     while (start.before(end)) {
                        start.add(Calendar.DAY_OF_MONTH,1); // add one day
                        String day =
                        String.valueOf(start.get(Calendar.DAY_OF_MONTH));
                        String month =
                        String.valueOf(start.get(Calendar.MONTH) + 1);
                        String year =
                        String.valueOf(start.get(Calendar.YEAR));
                        String add = day + "." + month + "." + year;
                        dates.add(add);
                     }
                  dateText.setText(dates.get(0) + " - " +
                  dates.get(dates.size() - 1));
                 }
                 storeDataInArrays();
                  customAdapter2 = new CustomAdapter2(PdfActivity.this,
                  PdfActivity.this, time, date, sokerit, hiilarit,
                  insuliini, info, id);
                  recyclerView = findViewById(R.id.recyclerView);
                  recyclerView.setAdapter(customAdapter2);
                  recyclerView.setLayoutManager(new
                  LinearLayoutManager(PdfActivity.this));
            }
         });
   }
```

6.4 PDF-tiedosto

PDF-tiedoston tallentaminen oli yksinkertaista PdfDocument luokkaa käyttäen. Haasteena oli sivujen asettelu ja tuloksien piirtäminen sivuille. Tekstin piirtäminen sivuihin tapahtuu käyttämällä Paint luokkaa.

Ohjelmakoodi 9 nähdään PDF-tiedoston luonti. Jos tiedosto on jo olemassa, lisätään nimen perään numero. Sivun kooksi asetetaan 595x842px joka vastaa A4 kokoista paperia.

Ohjelmakoodi 9: PDF-tiedoston luominen

```
String stringFilePath =
String.valueOf(Environment.getExternalStoragePublicDirectory(Environment.DI
RECTORY_DOWNLOADS) + "/Diabetes_tulokset.pdf");
File file = new File(stringFilePath);
int fileNumber = 1;
while (file.exists()) {
            stringFilePath =
            String.valueOf(Environment.getExternalStoragePublicDirectory(En
            vironment.DIRECTORY_DOWNLOADS) + "/Diabetes_tulokset_" +
            fileNumber + ".pdf");
            file = new File(stringFilePath);
            fileNumber++;
        }
PdfDocument pdfDocument = new PdfDocument();
PdfDocument.PageInfo pageInfo = new PdfDocument.PageInfo.Builder(595, 842,
1).create();
```

Tulokset piirretään paperiin useiden looppejen avulla (Liite2). For looppi kiertää niin kauan kuin tuloksia on jäljellä. Ensimmäisessä If lausekkeessa päätetään uuden sivun tekeminen. Jokaisen päivän tulokset tulevat omalle paperilla, mutta yhdelle sivulle piirretään vain 5 tulosta. Jos tuloksia on enemmän kuin 5 päivää kohden, luodaan uusi sivu. Tekstin piirtäminen sivulle tapahtuu käyttäen koordinaatteja. Koordinaatteihin lisätään summa jokaista mittatulosta kohden, muuten tulokset piirtyisivät päällekkäin sivulle. Jokaisen päivän tuloksien perään lisätään päivän keskiarvo sokereista. Seitsemän tuloksen välein lisätään myös viikoittainen keskiarvo.

7 Sovelluksen testaus

Jotta sovellusta pystytään jakamaan, täytyy se saada .apk tiedostoksi jonka Android-käyttöjärjestelmä osaa asentaa. Tämä prosessi on hyvinkin yksinkertainen Android Studiossa. Studion valikoista valitaan ”Build-APK’s” jonka jälkeen .APK tiedosto on luotuna. Seuraavaksi tiedosto pitää allekirjoittaa muutamalla lisätiedoilla. Allekirjoittamiseen luodaan avain, jonka avulla kukaan muu ei voi allekirjoittaa sovellusta muokattuaan sitä. Avain tallentuu lokaalisti tietokoneelle ja on salasanan takana.

Sovellus jaettiin kolmelle henkilölle testattavaksi lähettämällä heille sovelluksen .apk asennusohjelma. Henkilöt 1 ja 2 ovat diabeetikoita ja voivat hyödyntää sovellusta oikeaan käyttötarkoitukseen, eli mittaustuloksien kirjanpitoon. Henkilö 3 testasi sovellusta vain löytääkseen ja ilmoittaakseen ongelmista sovelluksessa.

Sovelluksesta löytyi hyvinkin nopeaan 4 ongelmaa.

1. Fontti sovellukseen tulee automaattisesti käyttöjärjestelmästä. Eri puhelinvalmistajat vaihtelevat fonttia, joka saattoi saada tekstin ulos elementeistä.
2. Jos puhelinta käyttää vaakatasossa, elementit menevät sekaisin.
3. Jos puhelimen Android asetuksissa on Yötila/Darkmode päällä, sovelluksen värit menevät sekaisin ja hankaloittaa luettavuutta.
4. Puhelimien näyttöjen eri resoluutiot saattoivat saada elementit sekaisin.

Ohjelmakoodi 10 huomataan, kuinka helposti ongelma fontin kanssa oli hoidettavissa pakottamalla sovellus käyttämään tiettyä fonttia. Sovellukselle asetettiin Sans-Serif fontti xml-tiedostoissa.

Ohjelmakoodi 10: Ratkaisu ongelmaan fontin kanssa

```
android:fontFamily="sans-serif"
```

Ohjelmakoodi 11 huomataan vaakatason korjaaminen pakottamalla sovellus olemaan aina pystyasennossa. Jos puhelimen kääntää vaakaan, sovellus ei käänny puhelimen mukana.

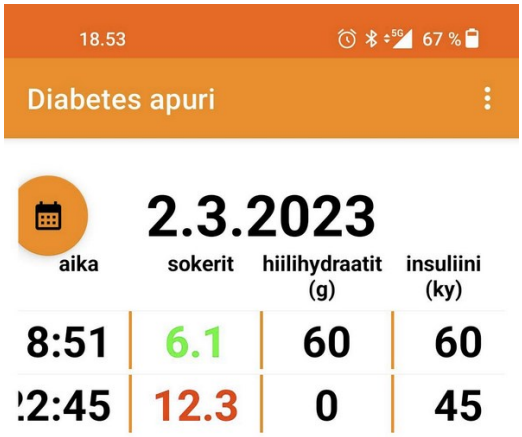
Ohjelmakoodi 11: Ratkaisu ongelmaan puhelimen kääntyvyyden kanssa

```
android:screenOrientation="portrait"
```

Teemoista löytyi ratkaisu korjaamaan yötila ongelma. Teemoissa on valmiina värit kahdelle eri teemalle: Light- ja Darkmodelle. Tämä mahdollistaisi eri värien käyttämistä mukautuen käyttöjärjestelmän asetuksista. Sovelluksessa kuitenkin käytettiin samoja värejä molempiin teemoihin, korjatakseen ongelma jossa sovelluksesta ei enään saanut selvää Darkmode valittuna.

Kuva 14 nähdään neljäs ongelma, jossa elementit eivät pysy halutuilla paikoilla. Tämä johtuu eri laitteiden resoluutioista. Tämä ongelma korjaantuisi suunnittelemalla käyttöliittymää uudelleen antamalla elementeille eri attribuutteja kuin nyt on käytössä. Kuvassa elementeille on annettu attribuutteina koot pikseleinä, jonka vuoksi kyseiset elementit eivät skaalaannu resoluution muuttuessa.

Kuva 14: Ongelma skaalauksen kanssa



8 Yhteenveto

Opinnäytetyössä valmistettu sovellus onnistui mielestäni hyvin. Sovellus sisälsi pääsääntöisesti Android-kehitystä Java-ohjelmointikielellä. Java-ohjelmointikielestä omasin alkeet joita pystyin hyödyntämään ohjelmoinnissa, mutta en ollut ennen hyödyntänyt näitä mobiiliohjelmoinnissa, joten uutta opittavaa oli paljon. Android Studio oli myöskin täysin uusi työkalu itselleni ja tämän opetteleminen tulee varmasti hyödyttämään tulevaisuudessa.

Lopputuloksena saatiin toimiva sovellus Android-käyttöjärjestelmälle. Sovellus on suunnattu diabeetikoille helppoa kirjanpitoa varten. Sovelluksen ominaisuuksiin kuuluu mittaustuloksien kirjanpito, sisältäen mitattu sokeriarvo, syödyt hiilihydraatit, pistetyt insuliinit sekä vapaa-sanainen kenttä lisätiedoille. Ominaisuuksiin kuuluu myöskin näiden tuloksien tarkisteleminen, sekä tallennus PDF-tiedostoksi halutulta aikaväliltä.

Jokaiseen tutkimuskysymykseen löydettiin vastaukset. Kertaan viellä lyhyesti kysymykset ja vastaukset.

Mitä dataa sovelluksessa pitäisi pystyä hallitsemaan? Sovelluksen suunnitteluvaiheessa todettiin tarvitsevan päivämäärää, aikaa, mitattuja sokeritasoja, syötyjä hiilihydraatteja, pistettyä insuliinia sekä vapaa-sanaista kohtaa lisätiedoille.

Miten ohjelmoidaan tarvittavat toiminnot datan tallennukseen, muokkaamiseen, poistamiseen, sekä tulostamiseen? Sovelluksen ohjelmoinnissa käydään läpi SQLiten käyttäminen tallennukseen, muokkaamiseen ja poistamiseen. Ohjelmoinnissa käydään myös läpi miten tulokset saadaan tallennettua luomalla uusi PDF-tiedosto johon tulokset piirretään.

Miten valmis sovellus rakennetaan .apk asennustiedostoksi? Sovelluksen testaamisessa käydään läpi kuinka valmis sovellus saadaan Android Studion avulla luotua allekirjoitettu .apk tiedosto, jonka pystyy helposti asentamaan Android laitteeseen.

Opinnäytetyössä tehtyä sovellusta pystyttäisiin halutessa jatkokehittämään paremmalla käyttöliittymällä, sekä parantamaan laitetukea, jotta sovellusta pystyttäisiin käyttämään mahdollisimman monella laitteella.

Lähteet

Ilanne-parikka, P. (2021). Tyypin 1 diabeteksen hoito. Terveyskirjasto Duodecim. https://www.terveyskirjasto.fi/dlk00011

MySugr App. (n.d.). Roche. https://www.rochediabetescaremea.com/en/mysugr-app-en

Meet Android Studio. (n.d.). Google. https://developer.android.com/studio/intro

How to use Play Console. (n.d.). Google. https://support.google.com/googleplay/android-developer/answer/6112435?sjid=10157913421640428942-EU#zippy=%2Cstep-pay-registration-fee

Create and set up your app. (n.d.). Google. https://support.google.com/googleplay/android-developer/answer/9859152?hl=en

What is SQLite?. (n.d.). SQLite Consortium. https://www.sqlite.org/index.html

Liite 1: Aineistonhallintasuunnitelma

Opinnäytetyössä on käytetty erilaisia tietoja sivustoilta, jotka ovat luotu harjoituskäyttöön. Työtä tehdessä kirjoitettiin erilaisia muistiinpanoja .txt tiedostoina. Tiedostot tallennettiin lokaalisti ja myös varmuuskopioitiin pilvipalveluun. Opinnäytetyössä ei käytetä dataa, jonka jakamisessa ja levittämisessä pitäisi kiinnittää huomiota tietosuojaan.

Liite 2: PDF-tiedostoon piirtäminen

```
for (int i = 0; i <date.size(); i++) {

            Paint paint = new Paint();

            if (startPage) {
                page = pdfDocument.startPage(pageInfo);
                paint.setTextSize(25);
                paint.setTextAlign(Paint.Align.CENTER);
                page.getCanvas().drawText(date.get(i),298,25, paint);

            }

            paint.setTextAlign(Paint.Align.LEFT);
            paint.setTextSize(20);
            page.getCanvas().drawText("Mittatulos " + mittaTulos,10,50 + y,
paint);
            paint.setTextSize(15);
            page.getCanvas().drawText("Kellonaika: " + time.get(i),10,74 +
y, paint);
            page.getCanvas().drawText("Sokerit: " + sokerit.get(i),10,92 +
y, paint);
            keskiArvo.add(Double.valueOf(sokerit.get(i)));
            keskiArvoV.add(Double.valueOf(sokerit.get(i)));
            page.getCanvas().drawText("Hiilihydraatit: " + hiilarit.get(i)
+ "g",10,110 + y, paint);
            page.getCanvas().drawText("Insuliini: " +insuliini.get(i) + "
ky",10,128 + y, paint);
            page.getCanvas().drawText("Lisätiedot: " + info.get(i),10, 146
+ y, paint);

page.getCanvas().drawText("_____
_____
_____"
                    ,10,155 + y, paint);
            startPage = false;
            y = y + 134;
            perPage++;
            mittaTulos++;



            if (date.size() == i + 1) {
                paint.setTextAlign(Paint.Align.CENTER);
                paint.setTextSize(20);
                double KA = keskiArvo.stream().mapToDouble(d ->
d).average().orElse(0.0);
                DecimalFormat df = new DecimalFormat("#.#");
                page.getCanvas().drawText("Päivän sokereiden keskiarvo: " +
df.format(KA),298, 175 + y-134, paint);
                if (dayCount == 7) {
                    dayCount = 0;
                    double KAV = keskiArvoV.stream().mapToDouble(d ->
d).average().orElse(0.0);
                    DecimalFormat df2 = new DecimalFormat("#.#");
                    page.getCanvas().drawText("7-päivän sokereiden
keskiarvo: " + df2.format(KAV),298, 198 + y-134, paint);
                    keskiArvoV = new ArrayList<>();
                }
```

```
                keskiArvo = new ArrayList<>();
                pdfDocument.finishPage(page);
                startPage = true;
                y = 0;
                perPage = 0;
                mittaTulos = 1;
                dayCount++;

            }
            else if (!date.get(i).equals(date.get(i+1))) {
                paint.setTextAlign(Paint.Align.CENTER);
                paint.setTextSize(20);
                double KA = keskiArvo.stream().mapToDouble(d ->
d).average().orElse(0.0);
                DecimalFormat df = new DecimalFormat("#.#");
                page.getCanvas().drawText("Päivän sokereiden keskiarvo: " +
df.format(KA),298, 175 + y-134, paint);
                if (dayCount == 7) {
                    dayCount = 0;
                    double KAV = keskiArvoV.stream().mapToDouble(d ->
d).average().orElse(0.0);
                    DecimalFormat df2 = new DecimalFormat("#.#");
                    page.getCanvas().drawText("7-päivän sokereiden
keskiarvo: " + df2.format(KAV),298, 198 + y-134, paint);
                    keskiArvoV = new ArrayList<>();
                }
                keskiArvo = new ArrayList<>();
                pdfDocument.finishPage(page);
                startPage = true;
                y = 0;
                perPage = 0;
                mittaTulos = 1;
                dayCount++;
            }
            else if (perPage == 5) {
                pdfDocument.finishPage(page);
                startPage = true;
                y = 0;
                perPage = 0;
            }
        }
```