LUT CT60A2411 Olio-ohjelmointi

Kevät 2021

HARJOITUSTYÖSUUNNITELMA

Ver. 1.0

Hyvinvointisovellus - **LiveGreen**

Team DPS:

Konsta Keski-Mattinen

Arttu Käyhkö

Juha Kajanen

LiveGreen

"Hyvinvointisovellus parempaan elämään"

Sisällysluettelo

1. Johdanto

Tässä dokumentissa kuvataan Team DPS:n harjoitustyön toteutussuunnitelma LUT-yliopiston CT60A2411 Olio-ohjelmointi -kurssia varten. Kurssin suoritusaika on kevät 2021. Harjoitustyöryhmän “Team DPS”:n muodostavat seuraavat henkilöt:

Konsta Keski-Mattinen
Arttu Käyhkö
Juha Kajanen

2. Tehtävänanto

Harjoitustyön tehtävänä on suunnitella ja ohjelmoida hyvinvointisovellus Java-ohjelmointikieltä käyttäen Android-alustalle. Sovelluksen tarkoituksena on auttaa käyttäjää havainnoimaan ja sitä kautta mahdollisesti muuttamaan elintottumuksiaan parempaan suuntaan. Sovelluksen antamien tietojen avulla käyttäjä voi myös vaikuttaa laajemmin ympäristöönsä esim. muuttamalla elintottumuksiaan ympäristön kannalta kestävämmälle pohjalle.

Harjoitustyöhön liittyviä pakollisia suunnitteluratkaisuja ovat olio-ohjelmoinnin mukainen toteutus sisältäen vähintään viisi (5) eri luokkaa/oliota ja vähintään yhden ulkoisen ohjelmointirajapinnan käyttö, jolla haetaan käyttäjälle vertailutietoja valitusta tietolähteestä. Sovelluksen on myös pystyttävä pitämään lokia käyttäjän syöttämistä tiedoista erillisessä tiedostossa ja tietoja on pystyttävä tarkastelemaan sovelluksen avulla. Harjoitustyön pakolliset toiminnallisuudet ja sovellukseen kehitettäviä muita mahdollisesti sovellukseen vapaasti toteutettavia ominaisuuksia esitetään liitteessä 1. Liitteessä esitetään myös toiminnallisuuksiin liittyvä pisteytys.

Harjoitustyön palautus sisältää:

- Sovelluksen lähdekoodin versionhallintajärjestelmässä
- Sovelluksen esittelyvideon
- Dokumentaation sovelluksesta (vaatimukset liitteessä 2.)

3. Toiminnallisuus

Harjoitustyössä on tavoitteena toteuttaa sovellukseen seuraavia toimintoja:

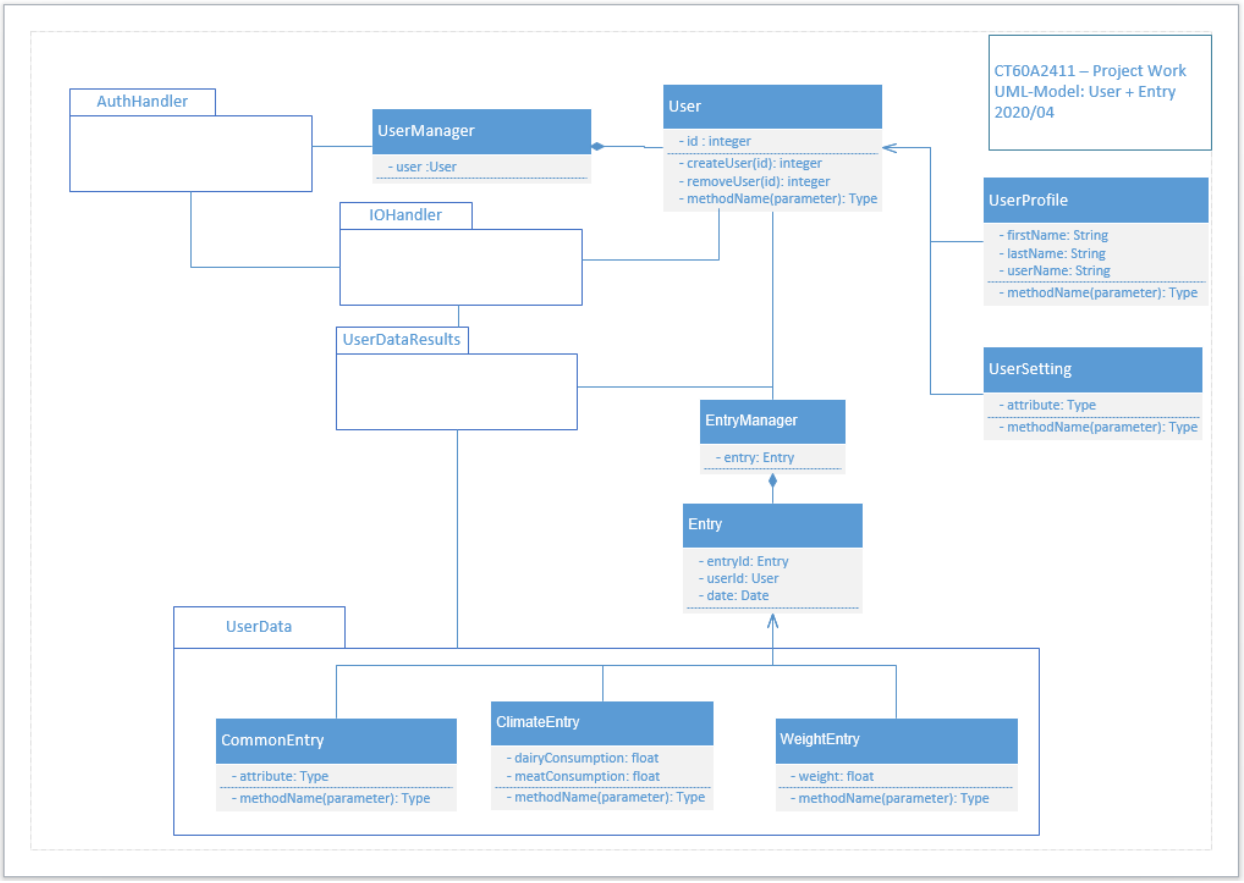
- Sovelluksen käynnistykseen käynnistysikoni ja splash-screen
- Sovellukseen sisäänkirjautuminen ja uloskirjautuminen
    - uuden käyttäjän sisäänkirjautuminen ja uuden tunnuksen lisääminen
        - käyttäjätunnus
        - salasana
    - olemassa olevan käyttäjän sisäänkirjautuminen
    - tilan muistaminen ulos-/sisäänkirjautumisessa
    - kirjautuminen toteutetaan ulkoisella kirjastolla jos mahdollista
- Autentikointi/salasanojen hallinta
    - tunnuksen ja salasanan hallinta toteutetaan ulkoisella kirjastolla jos mahdollista, muuten yksinkertainen käyttäjäkohtainen tiedostotallennus soveltuvalla salauksella
- Käyttäjäprofiilin hallinta
    - perustietojen täyttö
        - nimitiedot
        - ikä
        - asetukset
- Käyttäjätietojen lisäys
    - vertailutietojen täyttö
        - pituus
        - paino
        - asuinpaikka
        - elintavat, esim. maito/liha/kasvis/yms -käyttö
- Ulkoisen tietolähteen käyttö
    - datan haku (asynkronisesti) ja käyttö sovelluksen tarpeisiin
    - datan lataus sovelluksen muistiin (tarvittaessa)
    - tietojen luku toteutetaan tietolähteen tarjoaman formaatin mukaisesti
- Tietojen käyttö

    - käyttäjän itsensä tallentamien tietojen käyttö hyvinvointiin liittyvien arvojen laskennassa (esim. BMI)
    - ulkoisten tietolähteiden tietojen käyttö ja analysointi käyttäjätietojen rinnalla
- Tietojen esittäminen
    - numero- ja taulukkomuodossa
    - numeerisen tiedon graafinen esitys ulkoisen kirjaston avulla
- Lokin hallinta
    - sovelluksen käyttöloki
        - kirjautumiset
    - loki käyttäjän syöttämistä tiedoista
    - tietojen uloskirjoitus .csv-formaatissa (Excel + Google Sheets käyttö)

Käyttöliittymätoiminnot kuvataan erikseen kohdassa 5. Käyttöliittymä.

4. Data-arkkitehtuuri

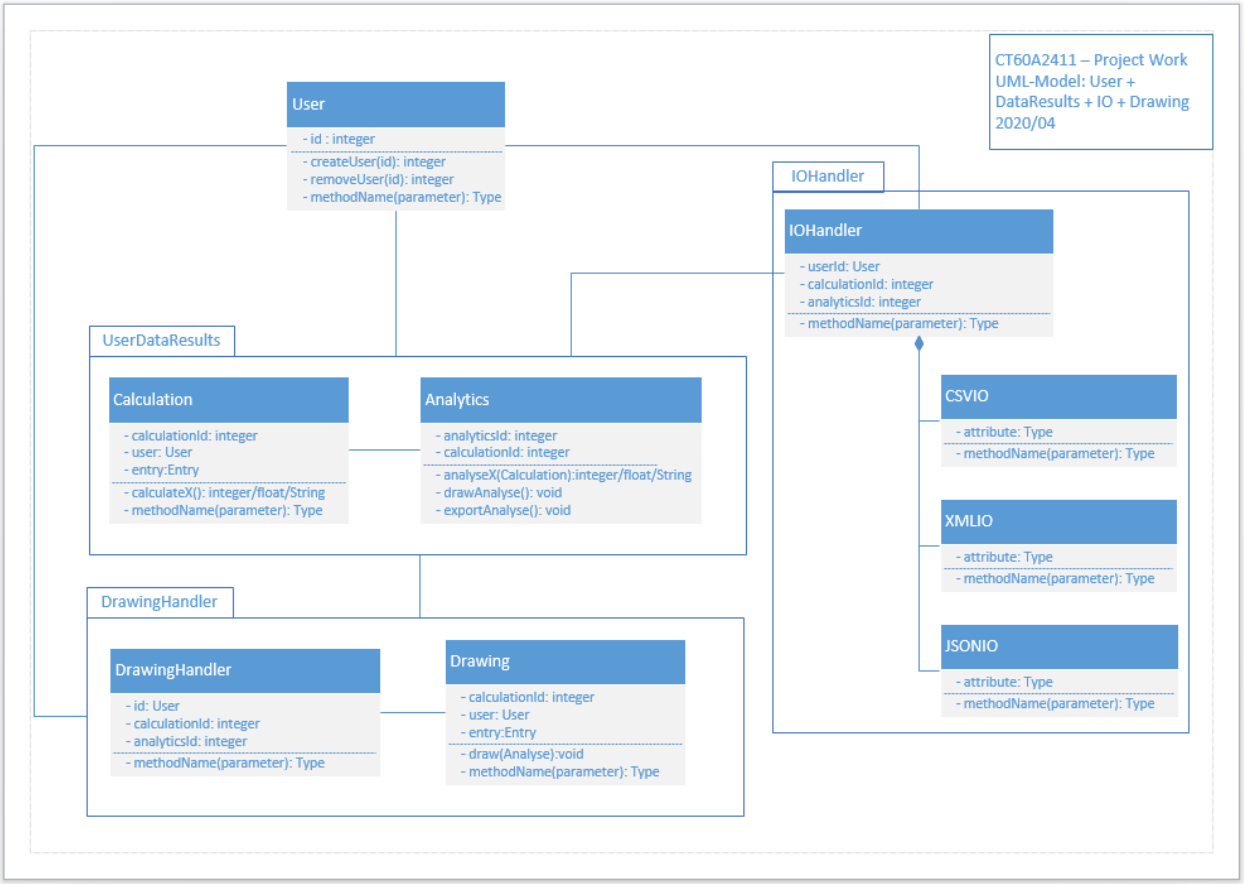
Alustava suunnitelma sovelluksen tietorakenteista ja sen käyttämistä luokista esitetään kuvissa 1 - 3.



Kuva 1. Käyttäjä-entiteetit

Käyttäjän hallinta tapahtuu UserManager-luokan kautta. Käyttäjäluokka (User) jakaantuu kolmeen osaan, itse käyttäjäolioon ja siihen liittyviin aliluokkiin profiiliin (UserProfile) ja asetuksiin (UserSetting). Profiiliin tallennetaan käyttäjän perustiedot (esim. nimi, asuinpaikka, käyttäjätunnus). Asetuksiin tallennetaan sovelluksen toimintaan liittyviä käyttäjäkohtaisia valintoja.

Käyttäjään liittyvät mittaustiedot hallitaan EntryManager-luokan kautta. Se toimii rajapintana mittaustietojen kirjaukseen, jotka tallentuvat Entry-luokkaan ja sen aliluokkiin (UserData-luokat).



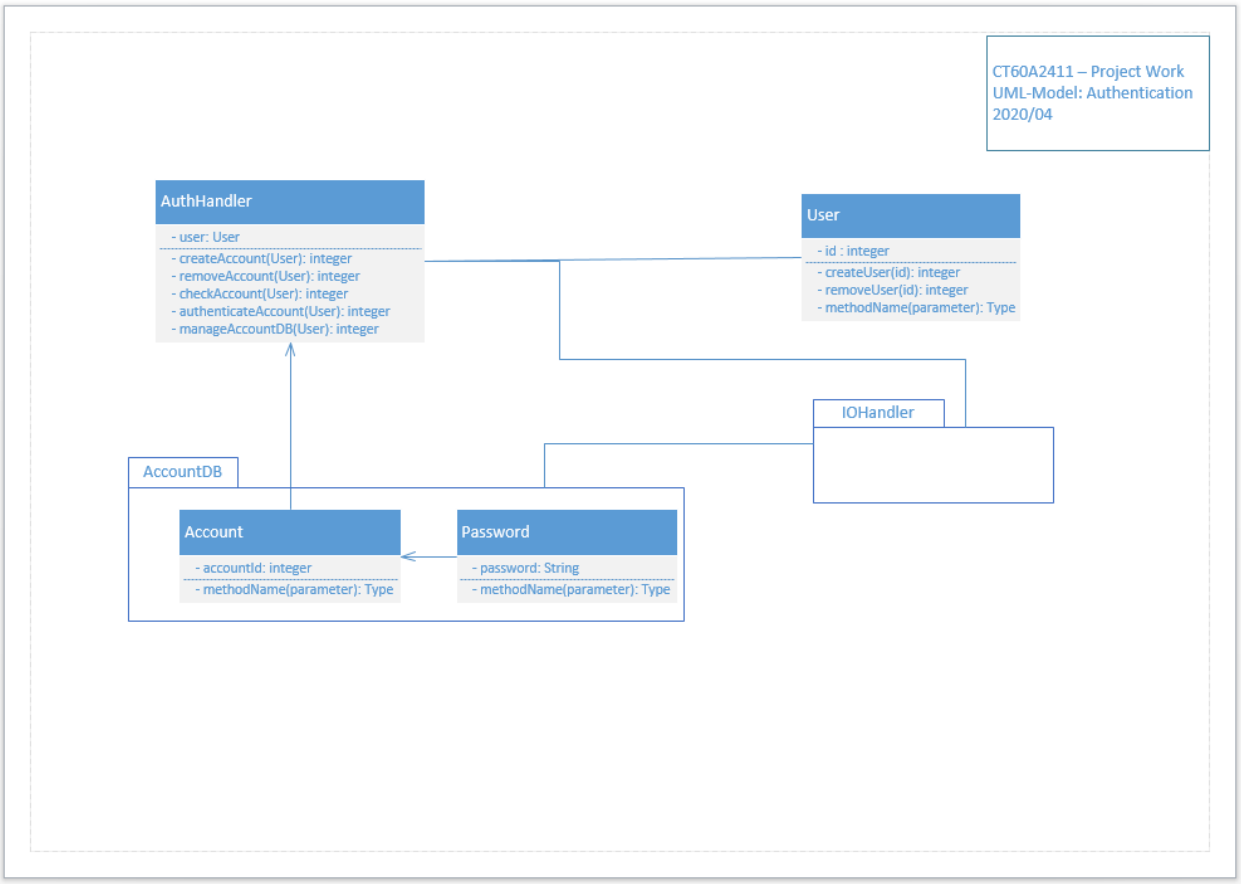
Kuva 2. Käyttäjän tallentamien tietojen käsittely ja niiden uloskirjoitus

Tietojen tallennus (lokit) tehdään IOHandler-luokkien avulla. Tallennus pysyvään tietovarastoon toteutetaan ulkoisen kirjaston avulla, mikäli mahdollista. Muutoin toteutus tehdään yksinkertaisena käyttäjäkohtaisena tiedostotallennuksena laitteen muistiin. IO-luokkia käytetään myös liitynnöissä ulkopuolisiin tietovarastoihin.

Käyttäjän syöttämien tietojen käyttö toteutetaan UserDataResults-luokissa (laskennat ja analysointi). Nämä toiminnallisuudet päätetään toteutusvaiheessa. Tietojen

esittämiseen numeerisessa ja/tai graafisessa muodossa käytetään ulkoista kirjastoa, mikäli mahdollista.

Käyttäjä liittyy autentikointitoimintoihin AuthHandler-luokan kautta ja se tarjoaa käyttäjän tunnistuksen salasanavarmistuksella. Autentikointi toteutetaan kolmannen osapuolen kirjastoja käyttäen, jos soveltuva kirjasto löytyy. Mikäli autentikointi toteutetaan omalla ratkaisulla, käyttäjätunnus- ja salasanatietojen tallennus ja salaus tehdään IO-luokkien avulla.



Kuva 3. Autentikoinnin hallinta ja sen tallennus

5. Käyttöliittymä

Käyttöliittymään suunnitellut näkymät ja elementit.

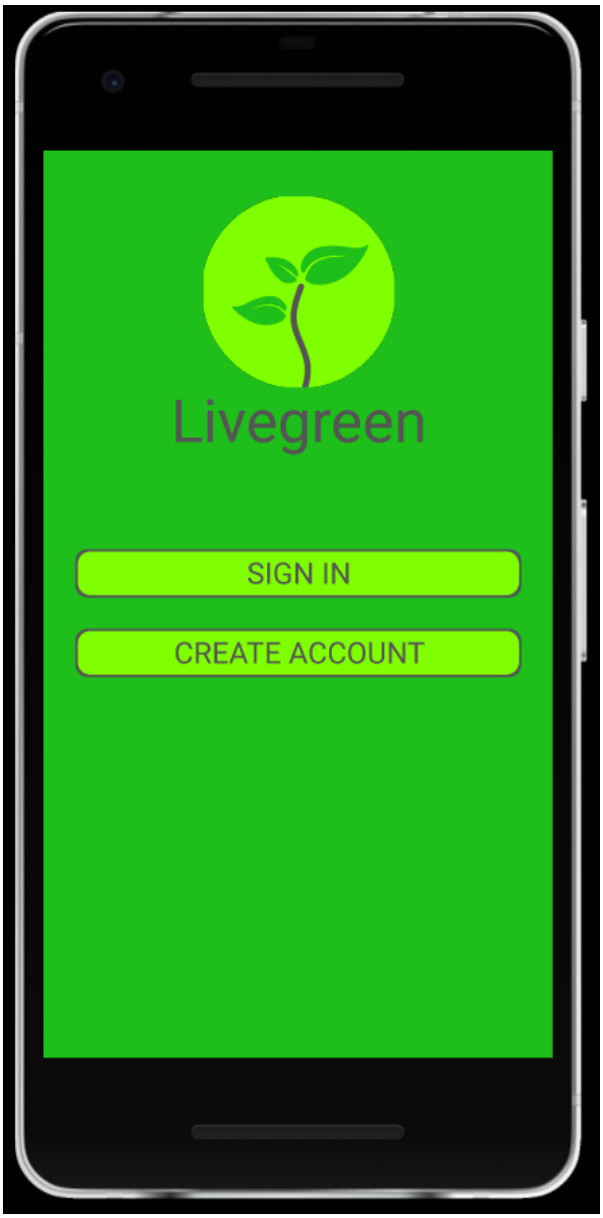
Yleiskatsaus tämänhetkisestä digitaalisesta käyttöliittymä prototyypistä. Antaa yleiskuvaa sovelluksen toiminnasta ja erillisistä näkymistä.

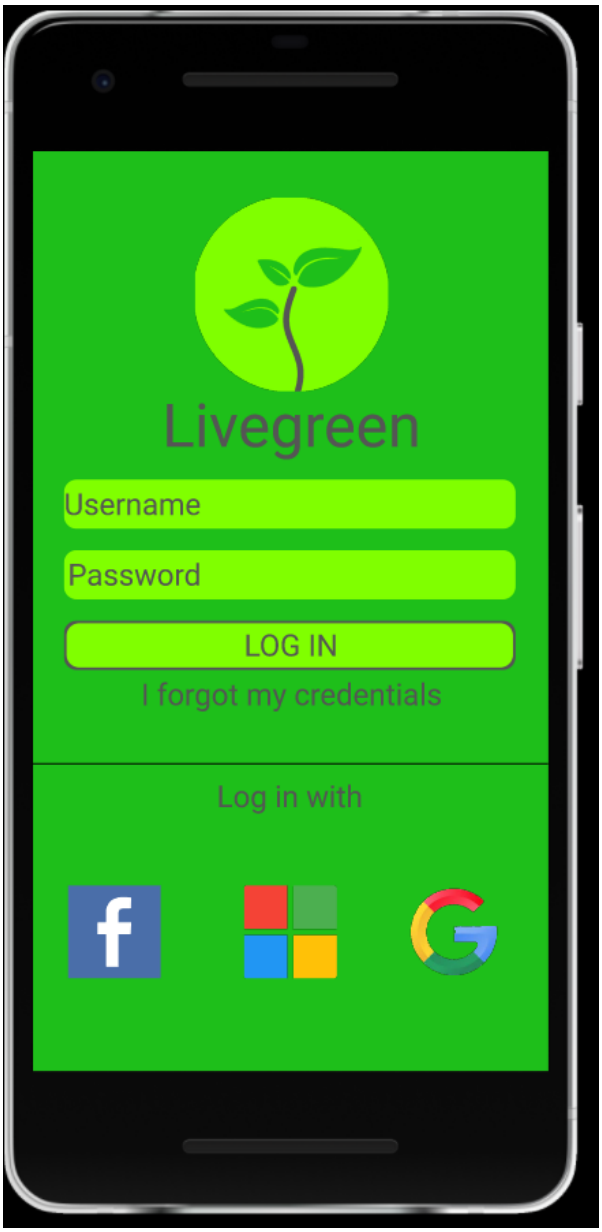
Tarkemmat näyttö komponentit ovat hyvin riippuvaisia myöhemmässä vaiheessa päätettävistä rajapinnoista, joten ne tullaan sopimaan myöhemmin.

Sovelluksen Entry-ikkuna on käyttäjän datan syöttöä varten. Tämä valinta on tehty, koska datan syöttö on eniten käytetyin ja käyttäjän kannalta vähiten kiinnostava osa sovelluksesta. Samalla data-analyysin siirtäminen avattavaan näyttöön ja pelillistäminen tuo käyttäjälle pienen odotuksen ja yllätyksen tunteen joka kerta kun hän avaa sovelluksen.

Splashscreen eli näkymä mikä latautuu sovelluksessa aivan ensin. Tätä näyttöä näytetään käyttäjille, joiden puhelimet ovat hitaita.

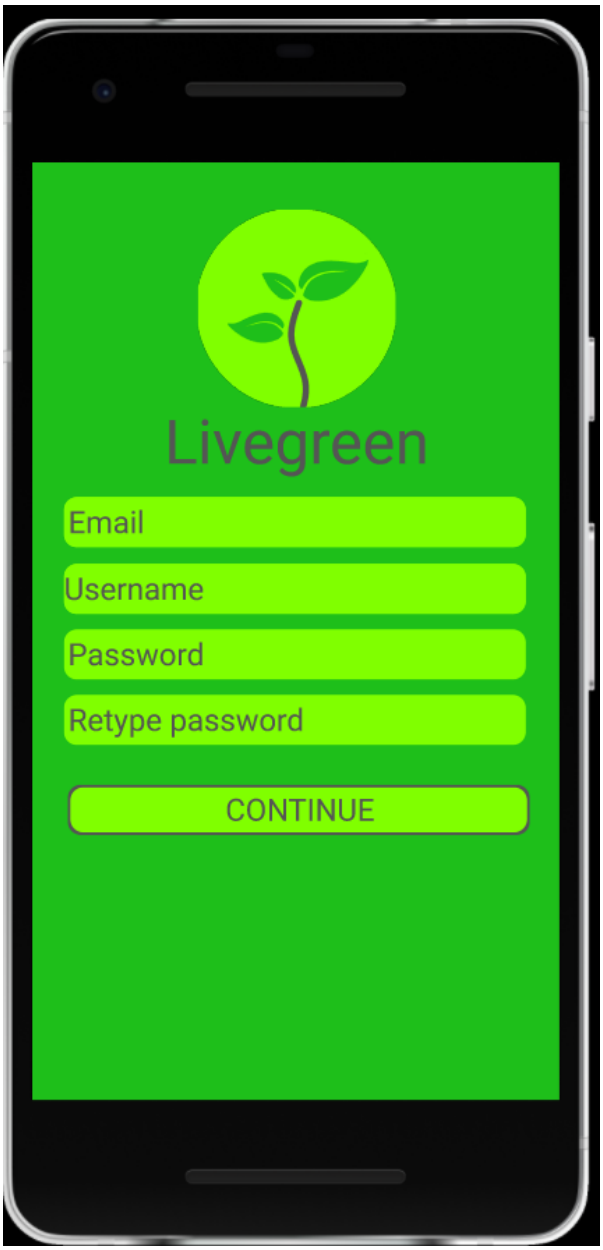
Splashscreenissä näkyy ohjelman logo ja nimi.

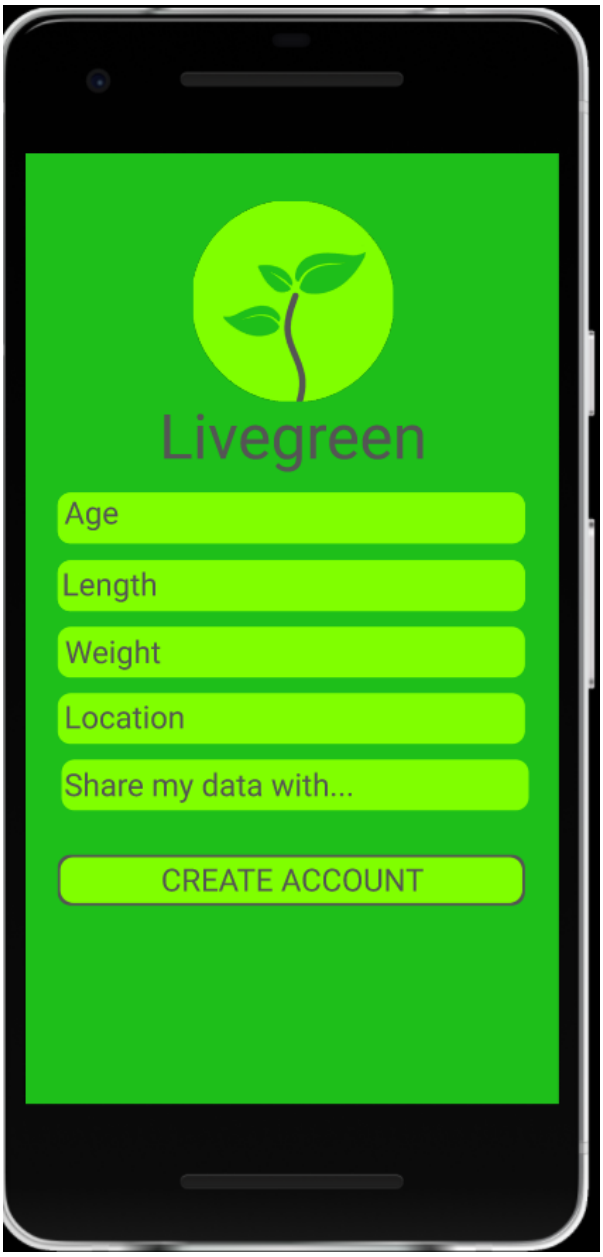
	Ensimmäinen oikea näkymä minkä ensimmäisellä käynnistyksellä näkee.

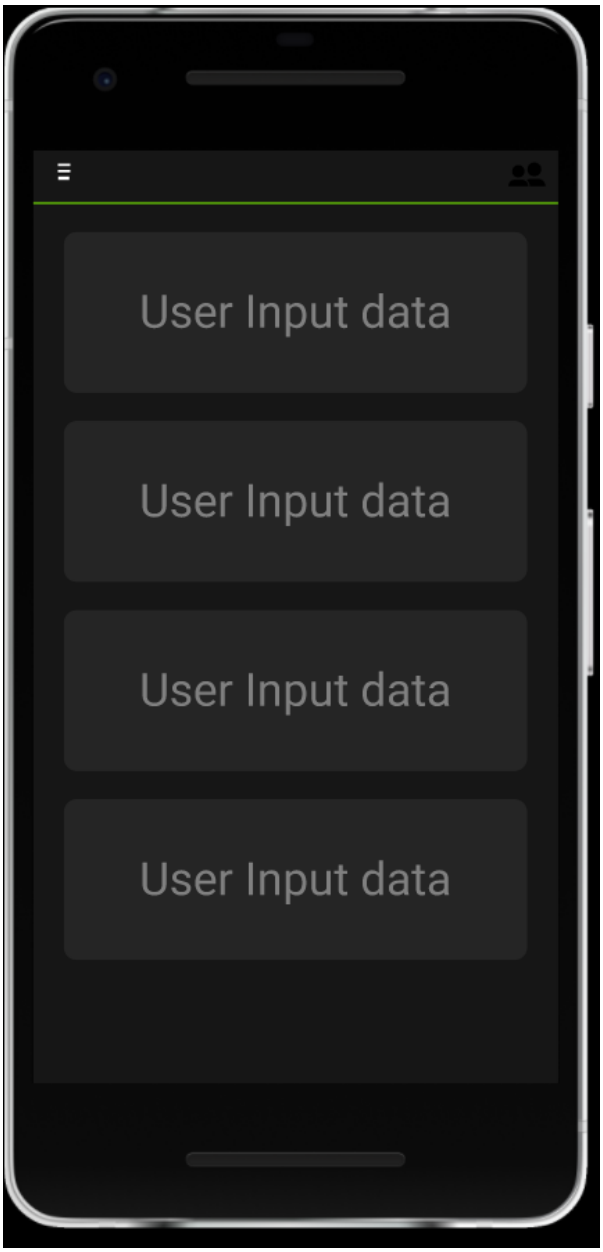


Kirjautumisnäkymä, joko ulkopuolisella autentikaatiolla tai omalla systeemillä.

Huomio login with X ominaisuus on vain visuaalinen eikä sen toiminnallisuus ole tämän harjoitustyön asia.

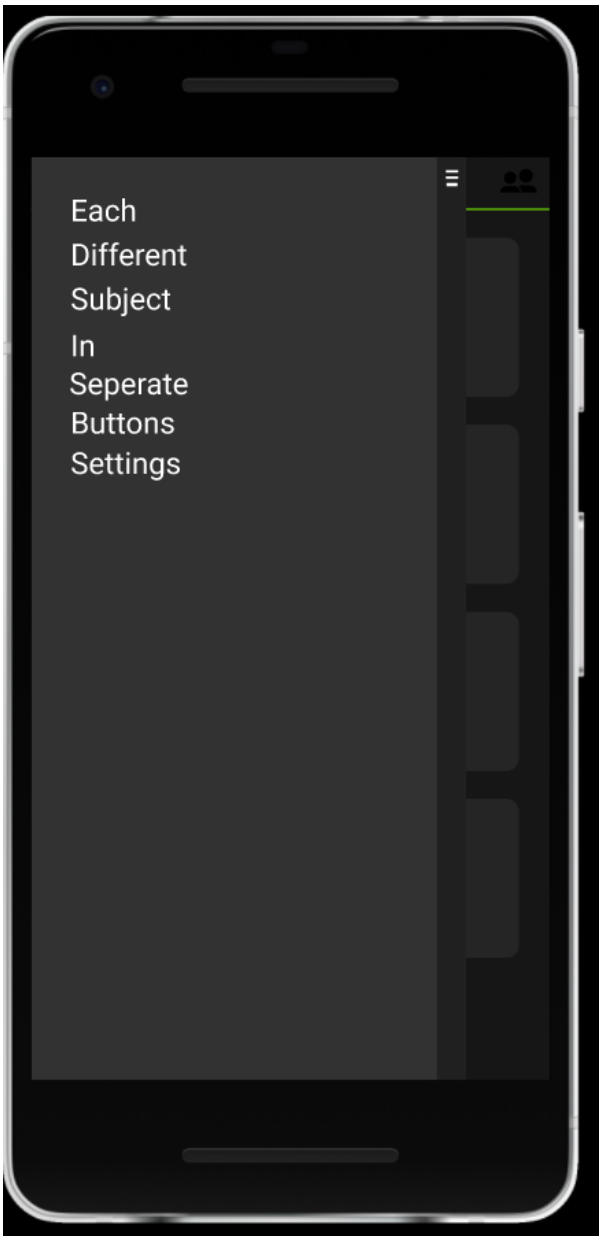
	Uuden profiilin tekonäkymä osa 1, jossa luodaan tilin hallintaa vaadittavat tiedot.

	Uuden profiilin luonti -näkymä osa 2, jossa saadaan käyttäjän tiedot.



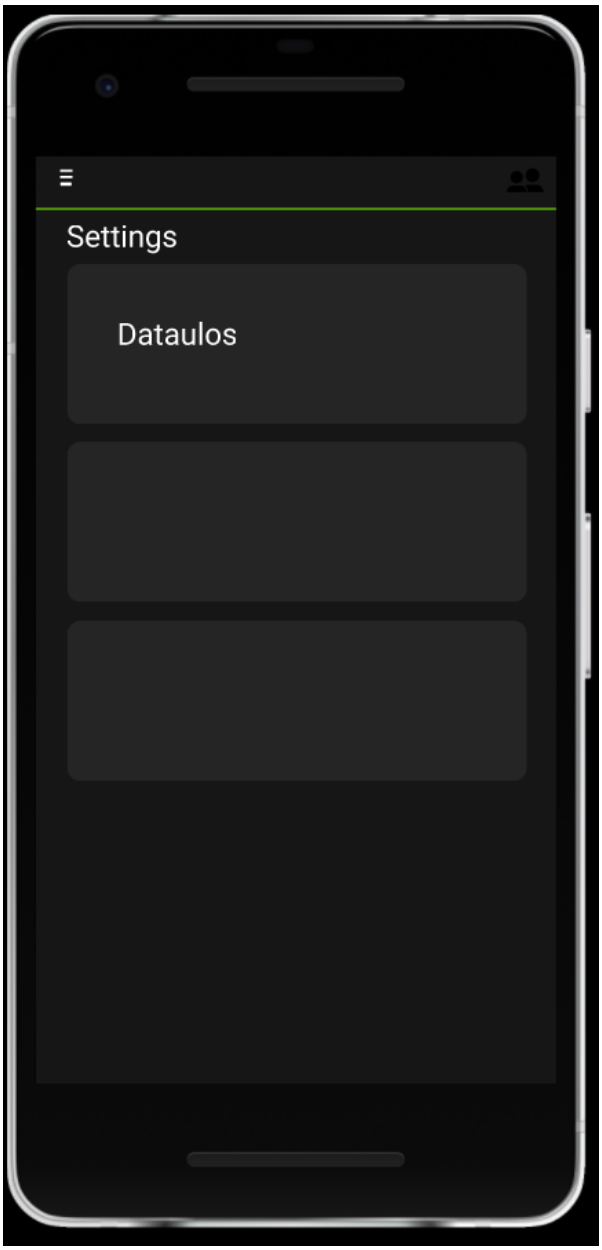
Ensimmäinen ikkuna minkä käyttäjä näkee sovellusta avatessaan, jotta halutun datan syöttö olisi mahdollisimman helppoa.

Vetoluukku löytyy vasemmalta ja data-analyysi -ikkuna löytyy oikealta. Näihin voi käyttäjä ohjata itsensä painamalla yläosan ohjausalueesta tai vetoeleillä sopiviin suuntiin.



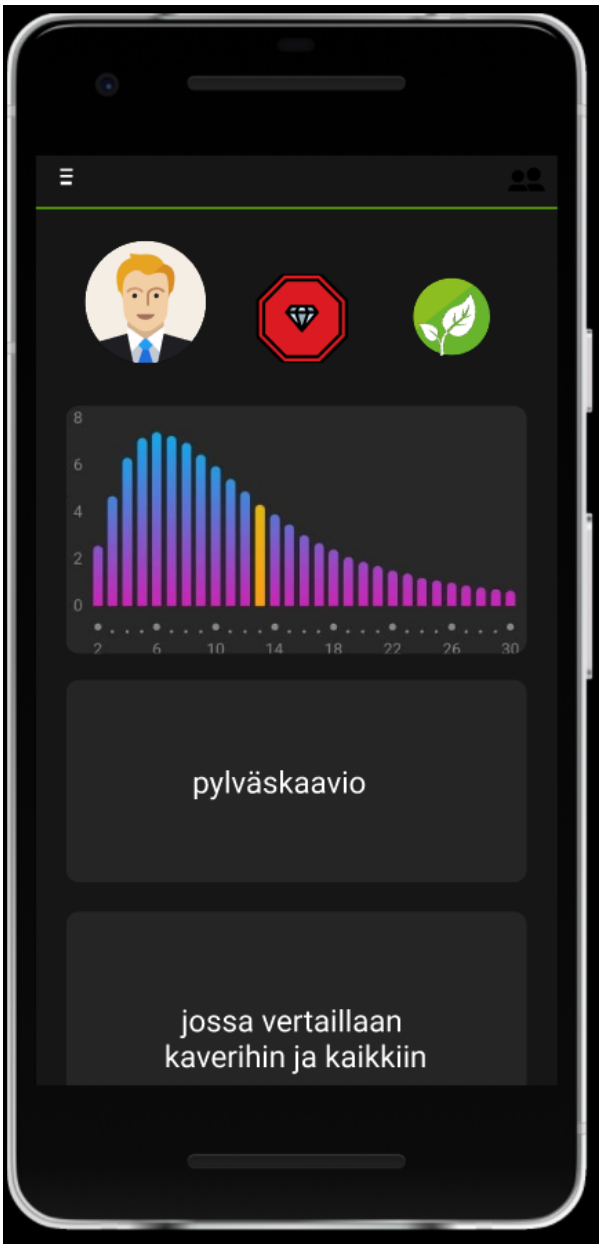
Vetolaatikosta löytyy ohjausnappulat ohjelman jokaiselle erilliselle sivulle.

Lisäksi vetolaatikon ui puolella on tummempi reuna tuoden käyttäjälle vetolaatikon mielimallia. Lisäksi hampurilaismenulogo pysyy näkyvissä kertoen käyttäjälle ohjelman tilasta.



Tämä ikkuna on kuvastamassa nappuloiden toimintaa ja sitä että formatoidun datan hakeminen sovelluksesta on säilytetty “ammattilaisille”. Samasta näkymästä voidaan antaa käyttäjälle tieto mistä ulkoinen data on.

Asetuksista löytyy sovelluksen asetukset.



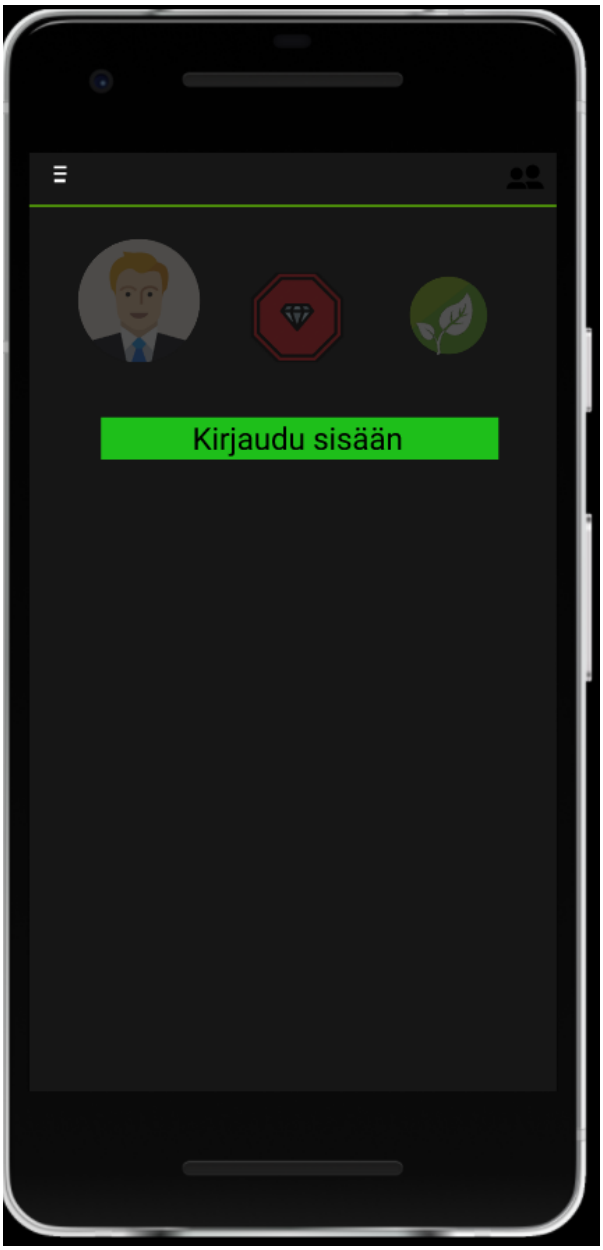
Data-analyysi -sivu, jossa näytetään käyttäjän profiilikuva ja käyttäjän lohko arvo kussakin mitattavassa osa-alueista.

Pylväskaavioissa näkyy kunkin mitattavan aiheen desiilit ja miten käyttäjä ja käyttäjän kaverit sijoittuvat niissä.

Profiili-ikkuna johon pääsee painamalla käyttäjän kuvaketta tai vetolaatikosta.

Profiili näkymästä näkyy käyttäjän tiedot ja sieltä löytyy kavereiden hallintaa.

Asetusnäkymä voi mahdollisesti yhdistää profiili näkymään, sillä sisältö on hyvin samanlainen.

	Ikkuna missä näky, jos käyttäjä yrittää avata profiili näkymän tai data-analyysi näkymän ilman kirjautumatta. Mieluusti arvo lohkot ja tyhjä käyttäjäkuva olisi sopivan epämukavat suunnaten käyttäjät käyttämään sovellusta ja personalisoimaan omaa tiliä.

Taulukko 1: Käyttöliittymä ja valintaperusteet

6. Ympäristöt

Harjoitustyön tekemisessä käytetään taulukossa 1. esiteltäviä suunnittelu-, ohjelmointi- ja tiimityöympäristöjä ja -välineitä.

Tehtävä	Järjestelmä/ympäristö
Suunnittelu/Dokumentointi	Google Docs/Sheets/Slides
Suunnittelu/Aikataulutus	Quire
Digitaalinen prototyyppi	Figma
Data-arkkitehtuurisuunnittelu	Microsoft Visio
Ohjelmointi	Android Studio 4.1.2
Virtuaaliemulaattori	Pixel_3a_API_30_x86 / Android 11.0
Versionhallinta	Git/GitHub
Dokumentinhallinta	Git/GitHub
Tiimityö	Slack

Taulukko 2: Käytettävät kehitys- ja työympäristöt

Suunnittelu ja töiden dokumentointi tehdään Googlen työvälineillä ja Google Driveä käyttäen. Näin dokumenttien yhteiskäyttö on sujuvinta. Aikataulusuunnitteluun ja työnseurantaan on käytössä helppokäyttöinen ilmaisohjelmisto Quire (https://quire.io/). Data-arkkitehtuuri kuvataan Microsoftin Visiota käyttäen. Tarjolla olevat ilmaisohjelmistot kuten Lucidchart tai Draw.io/Diagrams.net eivät ole käytettävyydeltään tai ominaisuuksiltaan samalla tasoilla, joten opiskelijakäytössä oleva Visio on tehokkain valinta.

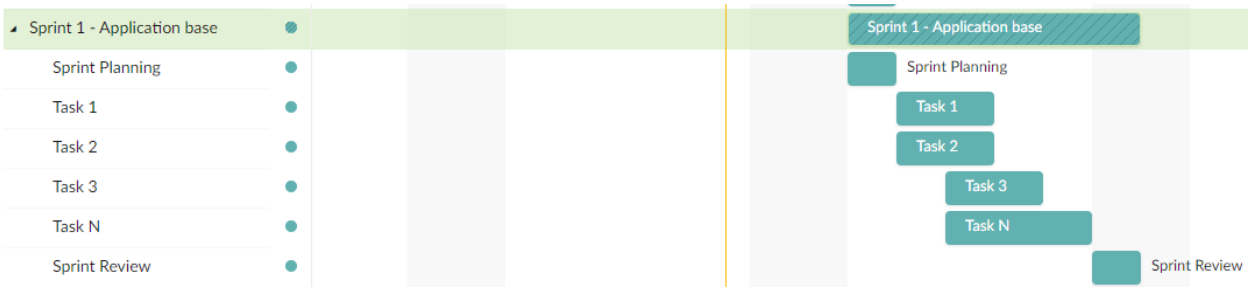
Ohjelmointiympäristönä on kurssilla käytössä oleva Android Studio ja sen tarjoamat virtuaaliemulaattorit. Versionhallinnassa käytetään Git-versionhallintaa ja GitHubia ohjelmavaraston (repository) hallintaan ja arkistointiin.

Tiimityövälineenä on Slack, jolla hoidetaan tiimin sisäinen viestintä ja palaverointi. Quirea käytetään Slackin ohella töiden seurantaan ja ohjaukseen.

7. Työtavat

Harjoitustyö toteutetaan Scrum-tyylisesti sprintteinä (Kuva 4.). Sprintit koostuvat

- Alkupalaverista (Sprint planning) jossa sovitaan sprintin sisältö ja kuka toteuttaa minkäkin osuuden
- Tehtävien itsenäisestä toteutuksesta
- “Daily” Scrumeista, joissa kerrotaan mitä on tehty, mitä tehdään ja onko työtä haittaavia esteitä (2-3 kertaa / viikko)
- Sprintin tarkastuksesta (Sprint review) jossa käydään läpi valmistuneet sprintin inkrementit. Tämä yhdistetään tässä tapauksessa seuraavan sprintin suunnittelupalaveriin.



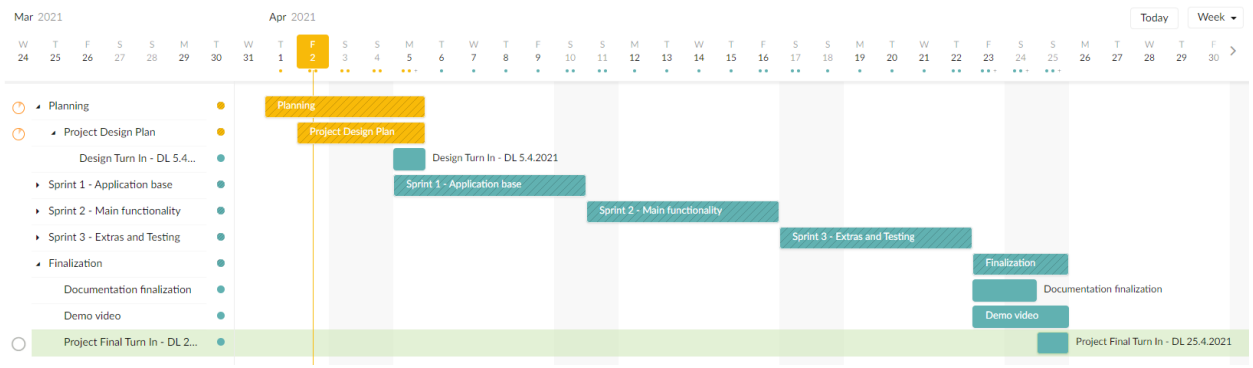
Kuva 4. Esimerkki toteutussprintistä

8. Aikataulu

Harjoitustyön palautuksiin liittyvät tärkeät päivämäärät:

- Harjoitustyösuunnitelman palautus DL 5.4.2021
- Harjoitustyön palautus 25.4.2021

Harjoitustyö toteutetaan suunnitteluvaiheen jälkeen kolmessa sprintissä. Ensimmäinen sprint keskittyy perussovelluksen toteutukseen (käynnistyminen, pääikkunat/aktiviteetit/fragmentit, käyttäjän perustoiminnallisuus). Toisessa sprintissä toteutetaan loput pakolliset alustatoiminnallisuudet (liitynnät ulkoisiin järjestelmiin) ja käyttäjän syöttämien tietojen käsittelyt ja analyysit. Viimeinen sprint keskittyy sovelluksen viimeistelyyn, mahdollisiin lisätoiminnallisuuksiin ja testaukseen. Dokumentaatio toteutetaan sprinteissä kehitystyön rinnalla ja viimeistellään viimeisen sprintin jälkeen. Viimeisen sprintin jälkeen koostetaan palautettava materiaali (lähdekoodit, dokumentaatio, esittelyvideo) palautuksen vaatimaan muotoon.



Kuva 5. Harjoitustyön aikataulutus

9. Liitteet

Liite 1. Harjoitustyön pakolliset- ja lisätoiminnallisuudet

Ominaisuus	Pisteet
Olio-ohjelmoitu	Pakollinen
Vähintään viisi erilaista luokkaa & oliota (käyttöliittymäluokkia ei lasketa)	Pakollinen
Vähintään yhden APIn käyttö, esim. Ilmastodieetti: https://ilmastodieetti.ymparisto.fi/ilmastodieetti/swagger/ui/index	Pakollinen
Sovellus tallentaa käyttäjän toiminnan (käyttäjän syöttämät arvot / tulokset) logiin (JSON, XML jne.)	Pakollinen
Logia on mahdollista tarkastella (puhtaana tekstinä, graafisilla käppyröillä jne.), eli voidaan tutkia arvojen (esim. oma massa) kehitystä kirjausten edetessä	Pakollinen

Ominaisuus	Pisteet
Ohjelma on rakennettu hyvin suunnitelluista UI-komponenteista	1 – 5 pistettä
Kirjautuminen applikaatioon	3 pistettä
Sovelluksella voi olla useampi käyttäjä (ja niiden luominen), tietojen tallennus järkevästi jonnekin	3 pistettä
Kirjautumisen salasana noudattaa hyvän salasanan sääntöjä (sisältää vähintään yhden numeron, erikoismerkin, ison ja pienen kirjaimen, on vähintään 12 merkkiä pitkä)	2 pistettä
Salasanan tallennus käyttää jonkinlaista hash-menetelmää ja suolausta (esim SHA-512 + salt)	2 pistettä
Jokin toinen datalähde fiksusti implementoituna (näitä löytää esim. https://www.avoindata.fi/, https://www.europeandataportal.eu/fi tai https://thl.fi/fi/tilastot-ja-data/aineistot-ja-palvelut/avoin-data/avoimet-rajapinnat) eli sovellus käyttää siis useampaa datalähdettä kerralla	2 – 5 pistettä
Ohjelmaan on mahdollista syöttää perustiedot (esim. pituus, paino, ikä(/syntymävuosi), kuva, asuinkunta) käyttäjästä ja näitä arvoja käytetään jossakin	2 pistettä
Ohjelma kerää käyttäjän massan kehityksestä dataa ja näyttää muutokset graafisesti havainnollistaen ruudulla	3 pistettä
Ohjelma näyttää graafisesti ilmastodieetin tarjoamien arvojen muutokset käppyröillä (esim. kuinka lihan kulutus ja hiilijalanjälki on muuttunut aikojen saatossa)	3 pistettä
Ohjelma kertoo asuinkunnan, ikäluokan yms. riskitekijät (esim. kunnassa X tupakoitsijoita on Y%) pohjautuen THL:n dataan omien syötteiden lisäksi	2 pistettä

Ominaisuus	Pisteet
Ohjelma peilaa eri datalähteidä ristiin ja muodostaa uutta tietoa (THL:n dataa höystettynä tilastokeskuksen datalla ja saldona saadaan ulos kaavio Z)	3 pistettä
Ohjelma muistaa käynnistämisen / kirjautumisen jälkeen missä näkymässä käyttäjä oli ennen ohjelman sulkemista	2 pistettä
Asynkronisten HTTP-kutsujen käyttö dataa haettaessa	2 pistettä
Fragmenttien hyödyntäminen aktiviteettien sijasta käyttöliittymiä rakennettaessa	2 pistettä
Scoped storagen käyttäminen tiedon tallennuksessa (ei vaadi käyttäjän myöntämiä oikeuksia laitteen massamuistiin, vaan toimii omassa "hiekkalaatikossaan")	2 pistettä
Responsiivinen käyttöliittymä (toimii siis erikokoisilla ruuduilla sulavasti)	2 pistettä
Puutteellinen dokumentaatio	-1 – -5 pistettä
Ohjelmassa on ongelmia, jotka haittaavat sen käyttöä	-1 – -5 pistettä
Ohjelma ei noudata kunnollista oliomallia	-1 – -10 pistettä
Ohjelmakoodia on kirjoitettu/kommentoitu suomeks (huomaa, että käyttöliittymätekstit voivat tietysti olla suomeksi)	-3 pistettä
Ohjelmassa ja/tai dokumentaatiosta on rasismia, vihapuhetta tai muuta epäsoveliasta	-20 pistettä
Jokin oma hieno ominaisuus tai toiminto (tai useampi)	Max 5 pistettä per ominaisuus
Yhteensä (mikäli kaikki toteutetut ominaisuudet tuottavat enemmän kuin 40 voidaan lisäominaisuuksilla korvata huonompia, mutta yli 40 pistettä ei voi harjoitustyöstä saada)	Max 40 pistettä.

Liite 2. Dokumentaation sisältö:

- Lyhyt kuvaus ohjelmasta, jonka toteutitte
- Kuka teki, mitä teki (ryhmätyössä ei pelkästään nimiä etusivulla, vaan kirjoittakaa, kuka teki minkäkin osuuden)
- Miten ohjelma suunniteltiin ja toteutettiin
- Luokkakaavio (vähintäänkin tärkeimmät metodit ja attribuutit, mutta sellainen mistä saa selvää. Mahdollista olla erillinen kuva dokumentaation ohella, jos dokumentaatiossa kuva on epäselvä)
- Lista toteutetuista ominaisuuksista ja toiminnoista ja lopullinen pistemäärä näiden perusteella
- Jonkinlainen arvio käytetyistä tunneista per ryhmän jäsen (esim. 20pvä ja 5h per pvä = 100h)
- Jokaisen ryhmän jäsenen vastaus kysymykseen: "Mitä opin harjoitustyöstä?" (vastaus vähintään 3 virkettä) ja muita yleisiä kommentteja (jos sellaisia on)
- Bonus (ei pakollinen olla dokumentissa, mutta suotavaa): Mitkä ominaisuudet / toiminnot olivat helppoja / vaikeita toteuttaa. Tämä auttaa tulevaisuudessa kehittämään harjoitustyön ohjeistusta paremmaksi ja pisteytystä sopivammaksi
- Dokumentaation muoto on **PDF-tiedosto**