Jenni Harakka, Mikko Tanhola, Olli Varila, Joonas Viljanen

Trainer-sovellus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintäteknologian tutkinto-ohjelma

Toteutusdokumentti

3.3.2023

Sisällysluettelo

1 Johdanto

Trainer on sovellus, joka on suunnattu kuntosalia harrastavilla tai harrastusta aloittaville. Sovelluksessa käyttäjän on mahdollista aloittaa uusi harjoitus valmiista pohjasta tai aloittaa kokonaan uusi halutesssa. Sovellukseen voi myös luoda uusia liikeitä tai harjoituksia halutessa.

Dokumentaatio selittää Trainer- sovelluksen toimintaa ja kehitystä. Dokumentaatiota voi hyödyntää kehitystiimi kehitysprosessien parantamiseen tai jatkokehitykseen.

Luvussa 3 kerrotaan mitä ohjelma tekee, miksi se on kehitetty ja kuvataan ohjelmistolle asetetut vaatimukset.

Luvussa 4 on ohjeet sovelluksen käyttöönottoa varten.

Luvuissa 6–8 esitellään sovelluksen tietomallia, rakennetta ja toimintaa. Dokumentaation lopussa on yhteenveto, jossa kerrotaan tulevaisuuden suunnitelmista.

2 Käsitteet

Activity – Android-sovelluksen komponentti, joka tarjoaa näkymän, jonka kanssa käyttäjä on vuorovaikutuksessa

Adapter – Luokka, joka määrittelee Android Studiossa RecyclerView:n toiminnan

Exercise – Yksittäinen liike harjoituksessa, sisältää tiedon suoritetuista toistoista (seteistä) ja painosta, jolla toistot on tehty.

ExerciseType – Liikkeen nimi, esim. kyykky, penkkipunnerrus, jne.

Fragment – Activityn osa, jonka voi saada muuttumaan, kun muu näkymä pysyy samana

Set – Set eli setti kertoo yksittäisen liikkeen yksittäisen suorituksen. Sisältää toistojen määrän ja painon.

Workout – Kokonainen harjoitus, joka sisältää kaikki suoritetut liikkeet (Exercise).

3 Tuotteen vaatimukset

3.1 Toiminnalliset vaatimukset

Vaatimukset valmiille sovellukselle on luotu määrittelyjen pohjalta. Käyttäjän on pystyttävä aloittamaan harjoitusohjelma valmiiksi tehdystä pohjasta tai halutessaan aloittamaan tyhjä ohjelma, jonka voi mukauttaa itselleen sopivaksi. Trainer on aloittelijoiden lisäksi tähdätty kokeneemmille harjoittelijoille heille annettava mahdollisuus luoda uusia liikkeitä tai harjoitusohjelmia.

Sovelluksen päätarkoitus on auttaa käyttäjää kehittymään, joten oman treenihistorian tarkastelu on myös olennainen osa sovelluksen toimintaa.

3.2 Laadulliset vaatimukset

Laadun varmistamisen takaamiseksi on backend koodilla oltava riittävä testikattavuus, jotta oikeanlaisesta toiminnallisuudesta pystytään olla varmoja.

Frontend koodin laadun vaatimuksena on käyttöliittymän selkeys ja helppokäyttöisyys. Käyttöliittymän on myös noudatettava WCAG-standardeja, visuaalisten elementtien osalta.

Kehityksessä myös seurataan WCAG-standardeja visuaalisten elementtien selkeydestä ja näkyvyydestä. Myös subjektiivisia standardeja seurataan, sovelluksen käyttöliittymä ei saa aiheuttaa käyttäjässä turhautumista tai hämmennystä.

4 Käyttöönotto

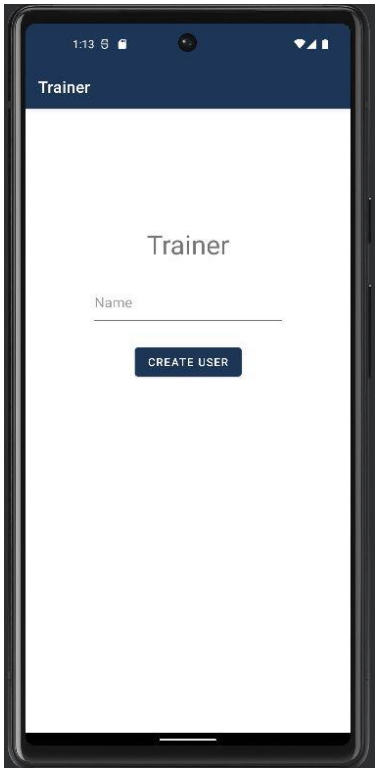
Sovelluksen voi ottaa käyttöön lataamalla stabiilin version .apk tiedoston GitHub[1] sivulta ja asentamalle sen omalle Android laitteelle. Uusimman version saa käyttöön kloonaamalla projektin tiedostot omalle koneelle ja asentamalla sen esimerkiksi Android Studion avulla omalle puhelimelle.

Sovelluksen käyttöönotto ei vaadi muuta erillistä konfiguraatiota. Sovellus kysyy käynnistyksen yhteydessä tarvittavat lisätiedot.

Sovelluksen jatkokehitystä varten suositellaan Android Studiota.

5 Sovelluksen käyttö

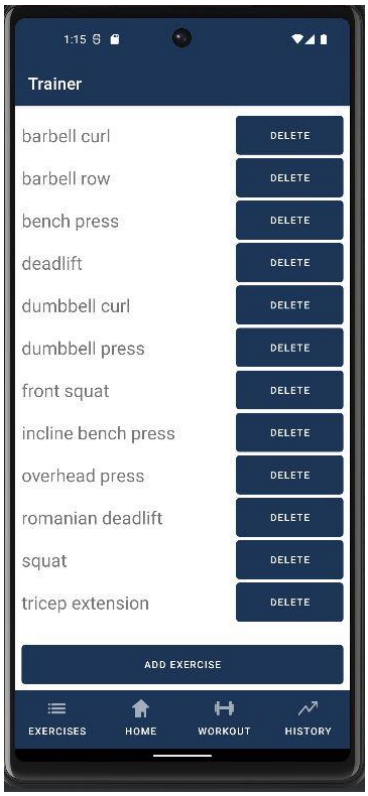
Trainer sovelluksella on mahdollisuus seurata omaa kuntosaliharjoittelua. Sovellukseen pystyy luomaan omia harjoituksia, sekä tallentamaan tehtyjä harjoituksia. Sovelluksessa on mahdollista luoda uusia harjoituksia, luoda niihin liikkeitä sekä antaa liikkeille sarjoja joihin voi määrittää suoritetut toistot ja käytetty paino. Harjoitukset tallennetaan käyttäjän lopettaessa harjoituksen, ellei tämä sitä hylkää. Tehdyistä harjoituksista on mahdollista luoda harjoituspohjia uudelleenkäyttöä varten.



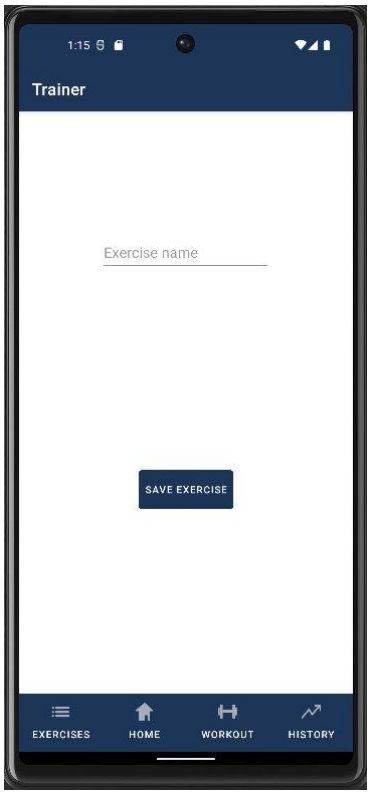
Kuva 1. Sovellus avataan ensimmäistä kertaa

Kuva 2. Sovelluksen päänäkymä

Ensikertaa avattaessa käyttäjä antaa käyttäjänimen sovelluksen, jonka jälkeen sovelluksen päänäkymä avautuu. Päänäkymästä voi navigoida ’Exercises’, ’Workout’ tai ’History’ näkymään.



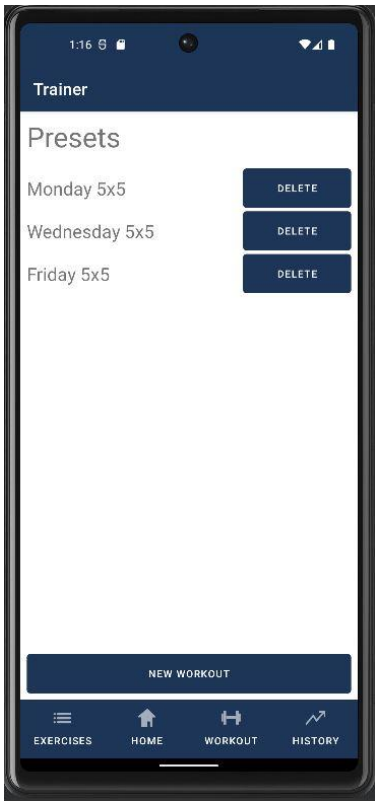
Kuva 3. Liikelista



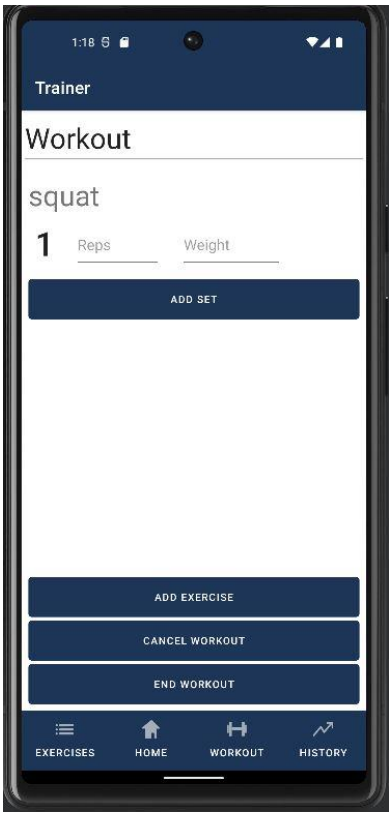
Kuva 4. Liikkeen lisäys

Exercises-näkymässä käyttäjälle avautuu (kuva 3) mukainen näkymä. Tästä luotuja liikkeitä voi hallita poistamalla tai lisäämällä niitä.

Lisätessä liikkeitä avautuu (kuva 4) näkymä, jossa annetaan halutun liikeen nimi.



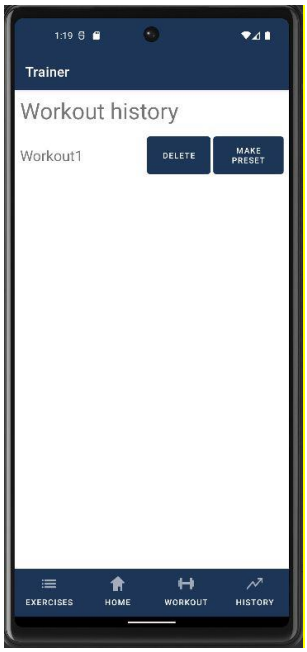
Kuva 5. valmiit harjoituspohjat



Kuva 6. aktiivinen harjoitus

Workout-välilehdellä (kuva 5) voi valita valmiin harjoituspohjan tai luoda kokonaan uuden harjoituksen.

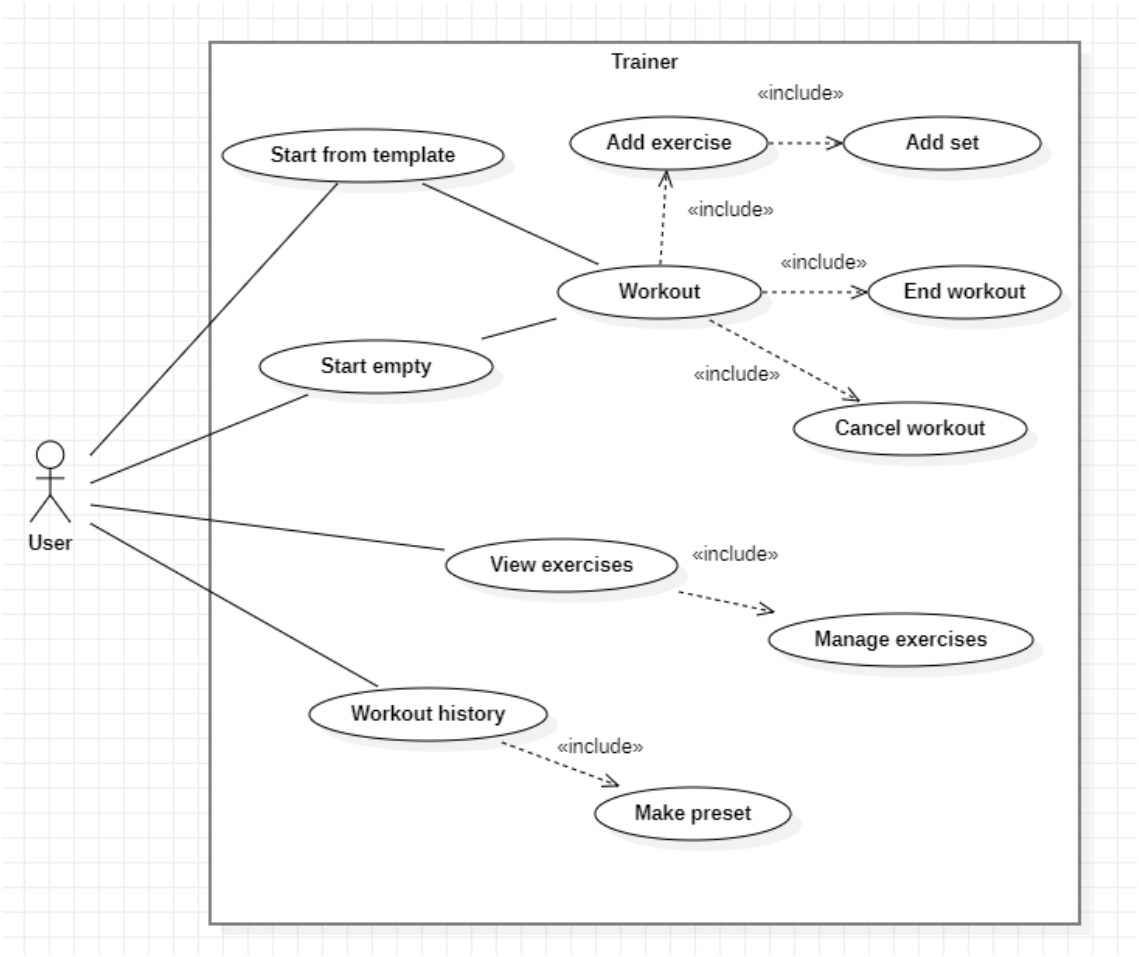
Aktiivisen harjoituksen näkymässä (kuva 6) pystyy lisäämään harjoitukseen tietoja tehdyistä liikkeistä ja harjoituksen loputtua harjoitus voidaan tallentaa ”End workout”- napilla tai hylätä ”Cancel workout”- napilla.



Kuva 7. Harjoitushistoria

History-välilehdellä (kuva 7) voi tarkastella suoritettuja harjoituksia.

6 Käyttäjäroolit ja käyttötapaukset



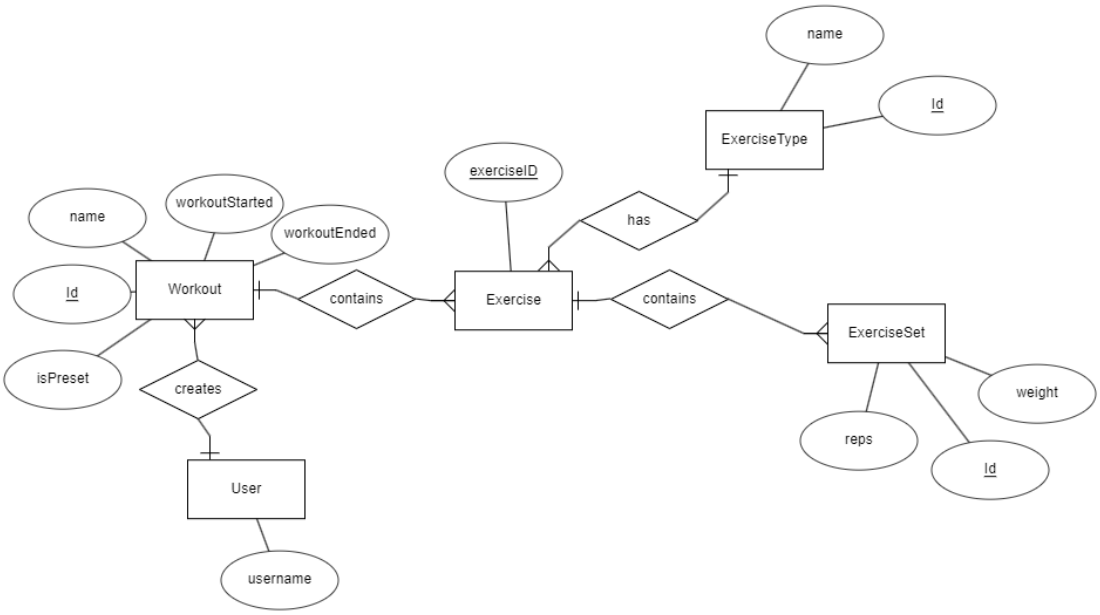
Kuva 8. Käyttötapauskaavio

Kuvattuna käyttäjän vuorovaikutusmahdollisuudet. Käyttäjä voi aloittaa uuden harjoituksen, jonka jälkeen sen voi lopettaa tai hylätä. Harjoitusta voi myös muokata tarvittaessa liikkeitä tai sarjoja lisäämällä.

Käyttäjällä on mahdollisuus selata harjoitushistoriaa ja luoda uusia liikkeitä.

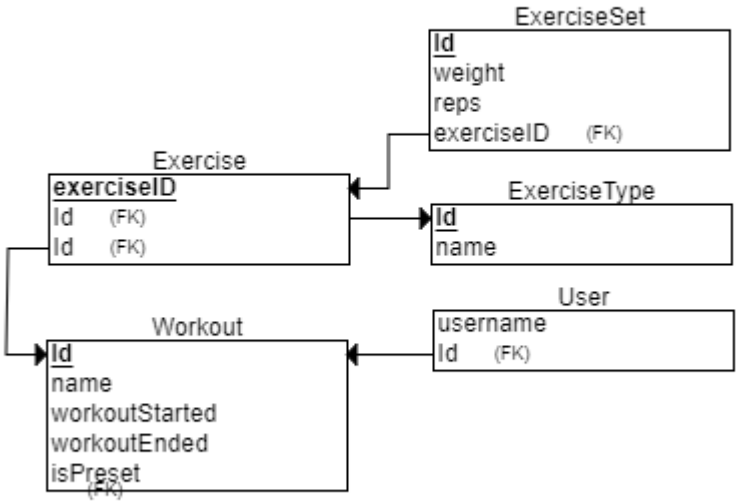
7 Ohjelmiston tietomalli

Sovelluksessa luodaan Workout-olioita ja tallennetaan niitä tietokantaan. Workout oliot sisältävät Exercise-olioita, jotka sisältävät ExerciseSet-olioita. Tämän lisäksi Exercise-olioilla on myös tyyppi, joka sisältää tietyn liikkeen nimen. Alla olevassa ER-diagrammissa on kuvattu tarkasti olioiden suhteet ja mitä kukin sisältää.



Kuva 9. ER-kaavio

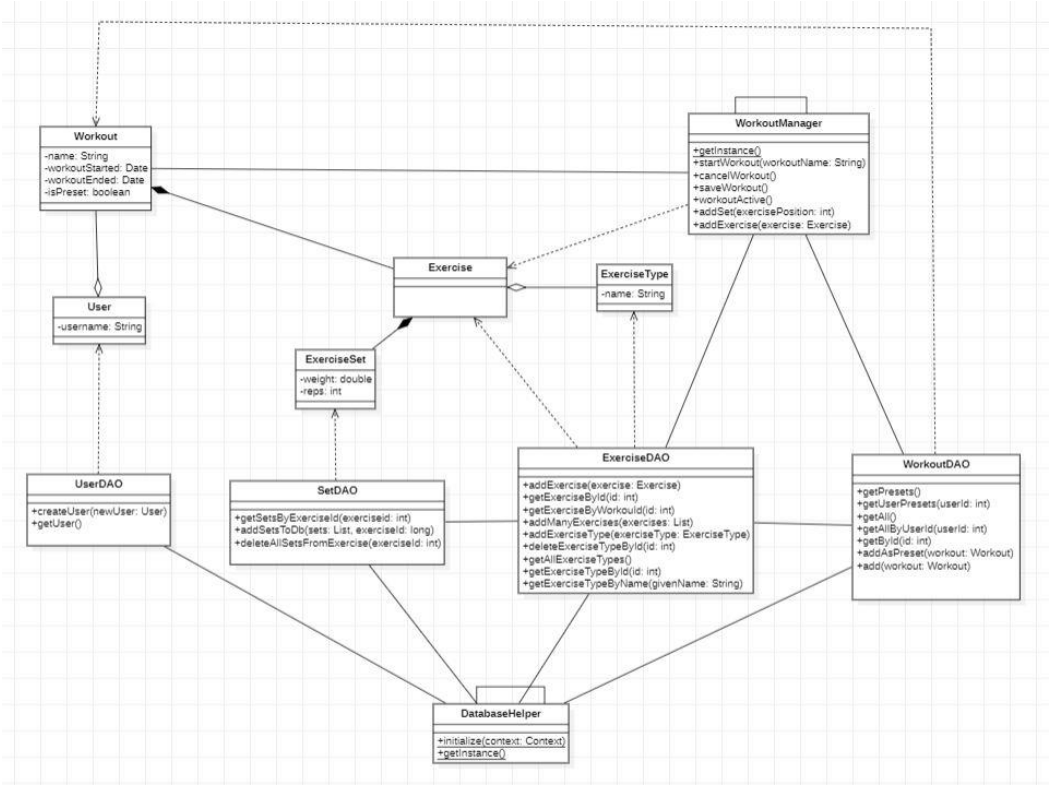
ER-diagrammin pohjalta on myös luotu relaatiotietokantakaavio, jonka avulla sovellukseen luodaan ensimmäisen käynnistyksen yhteydessä taulut. Tietokanta on normalisoitu kolmanteen normaaliin muotoon (third normal form).



Kuva 10, Relaatiotietokanta

8 Ohjelmiston rakenne

Alla olevassa luokkakaaviossa (Kuva 11) on kuvattu sovelluksemme luokkarakenne, poissulkien sovelluksemme käyttöliittymäluokat. Kaaviossa on kuvattu logiikkaa käsittelevät luokat, kuten Workout ja User, DAO-luokat, WorkoutManager-luokka, joka hoitaa Workout-olion säilyttämisen ja muokkaamisen, sekä DatabaseHelper-luokka, joka luo tietokannan ja yhteyden sinne. WorkoutManager ja DatabaseHelper ovat Singleton-luokkia. Kuvauksen ulkopuolelle jää käyttöliittymäluokat, eli fragment-, adapter- ja activity-luokat.



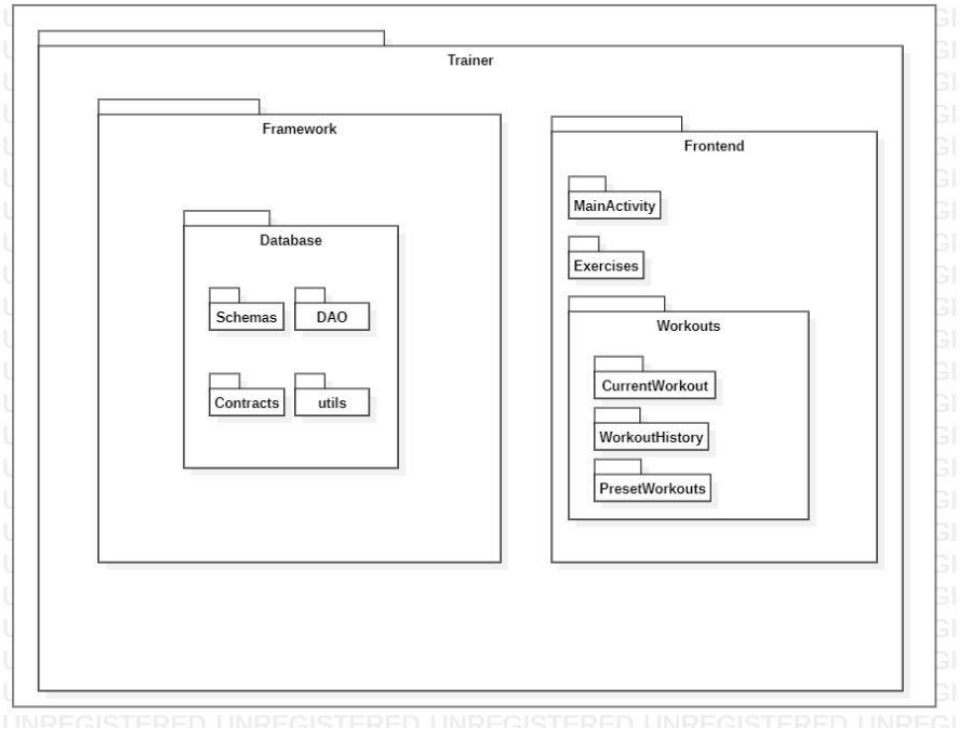
Kuva 11. Trainer-sovelluksen luokkakaavio

Kaaviossa (Kuva 11) kuvattavissa luokissa ei ole kuvattu välttämättä kaikkia kyseisen luokan muuttujia tai metodeja, ainoastaan oleellisimmat. Esimerkiksi kaikilla perusolioilla on Id:t, mutta niitä ei ole toiston takia merkitty kaavioon.

Kuva 12 kuvaa sovelluksemme pakkausrakennetta. Uloimmaisin pakkaus on itse sovellus, jonka sisällä on kaksi erottelevaa pakkausta, Framework ja Frontend.

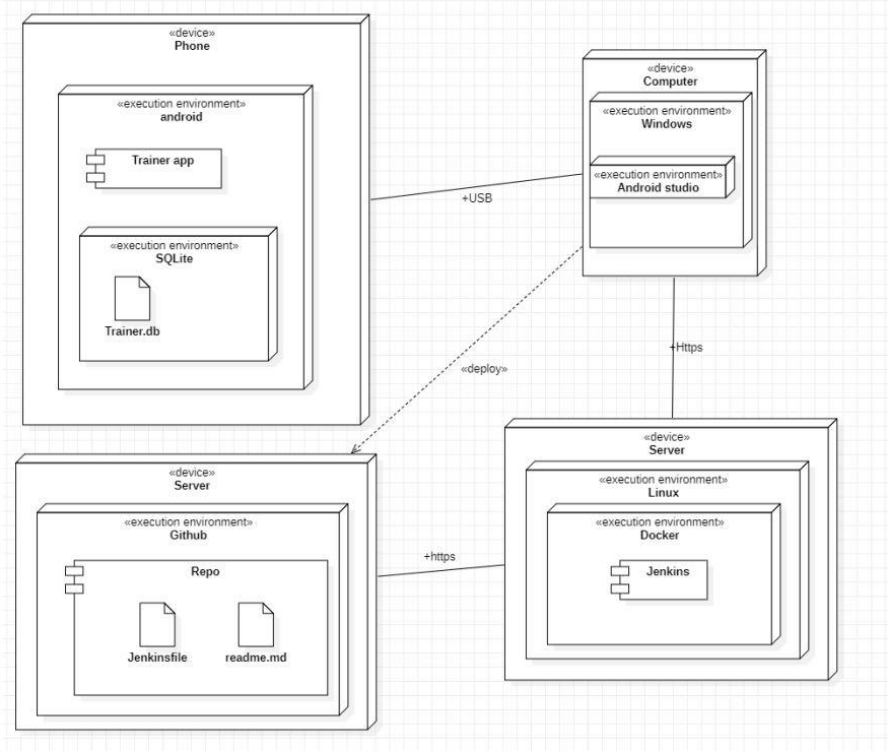
Framework-pakkauksen sisälle on sijoitettu Database-pakkaus, jonne on sijoitettu kaikki tietokantaan liittyvät pakkauksemme, kuten pakkaukset DAO- ja skeemaluokille.

Frontend-pakkauksessa on pakkaukset Exercise-toiminnoille ja MainActivitylle, joka on sovelluksemme pääsivu, jonka sisällä fragmentit vaihtuvat. Frontend-pakkauksen sisällä on vielä Workouts-pakkaus, jonka sisällä on omat pakkauksensa nykyiselle workoutille, workout historialle sekä preset workouteille.



Kuva 12. Trainer-sovelluksen pakkauskaavio

Alla oleva sijoittelukaavio (Kuva 13) kuvaa sovelluksemme fyysistä rakennetta. Olemme käyttäneet Android Studiota sovelluksen pääasiallisena kehitysalustana. Jos olemme halunneet nähdä, miltä sovellus näyttää käytössä, olemme yhdistäneet koneeseen puhelimen USB-liitännällä. Kehittäessämme tuotetta olemme lähettäneet muutokset Githubiin. Erillisellä palvelimella olevalla Jenkinssillä olemme suorittaneet testausta.



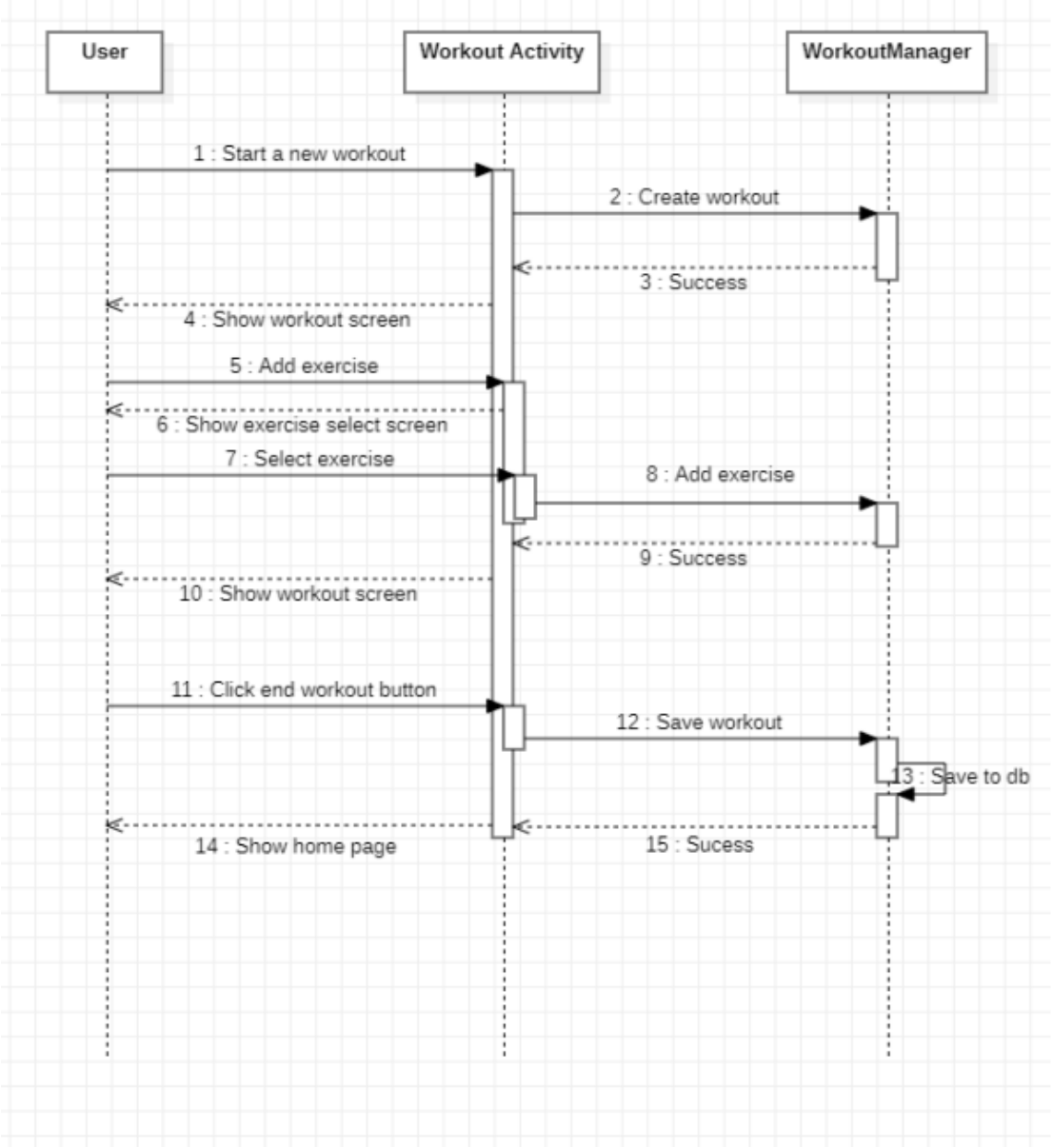
Kuva 13. Trainer-sovelluksen sijoittelukaavio

9 Ohjelmiston toiminta

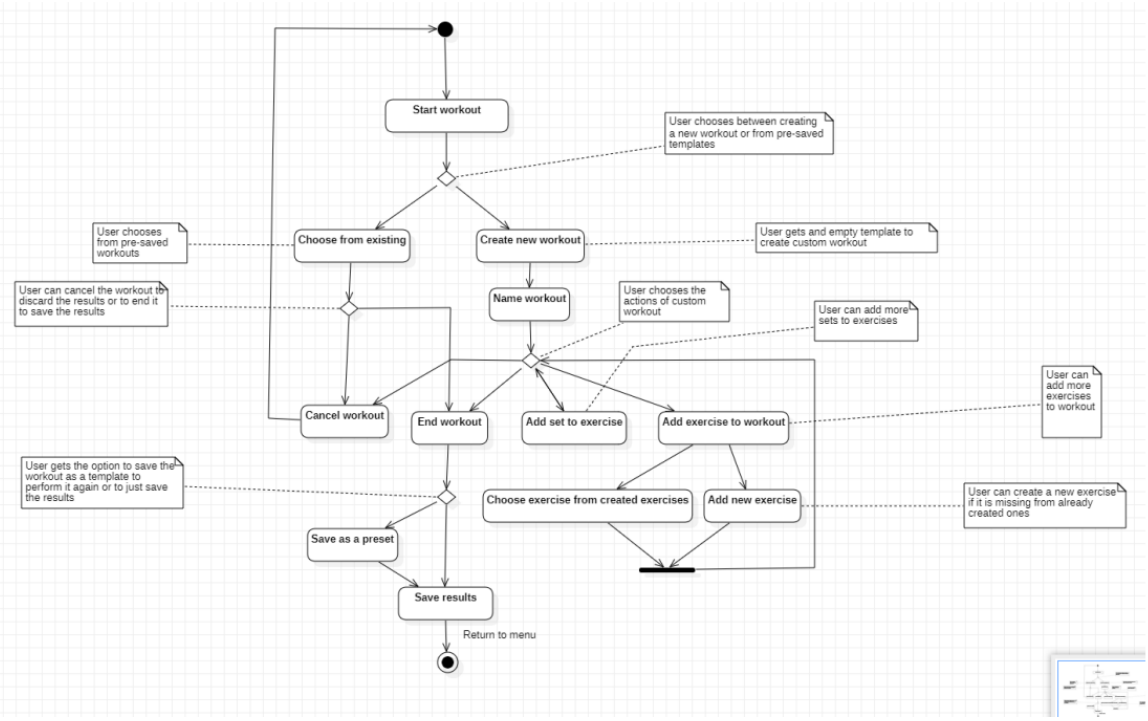
Sovelluksen päätoiminto on omien harjoitusten luominen, niiden muokkaaminen sekä tallentaminen tulosten seuraamiseksi.

Kuvassa 14 on kuvattu käyttötapaus uuden harjoituksen luomiseksi sekvenssikaavion avulla ja kuvassa 15 sama käyttötapaus aktiviteettikaaviolla selitettynä.

Käyttäjä aloittaa luomalla uuden harjoituksen tai valitsemalla jonkun jo valmiin harjoituksen. Harjoituksen sisällä on mahdollisuus luoda uusia liikkeitä, joihin pystyy lisätä tiedot toistojen määrästä sekä lisätystä painosta. Harjoituksen tiedot tallennetaan väliaikaisesti WorkoutManager-luokkaan. Kun käyttäjä lopettaa harjoituksen, tallennetaan se puhelimen lokaaliin tietokantaan.



Kuva 14. 1 Sekvenssikaavio harjoituksen luomisesta Trainer sovelluksella



15. 2 aktiviteettikaavio harjoituksen luomisesta Trainer sovelluksella

10 Kehitysprosessi ja kehitysvaiheen tekniikat

Kehitysprosessi alkoi tuotteen halutun lopputuloksen määrittelystä. Määrittelyn aikana tutkittiin mitä ominaisuuksia kohderyhmä tarvitsisi sovellukselta. Määrittelyjen jälkeen luotiin backlog ominaisuuksista.

Kehityksen aikana seurattiin agile-metodeja scrum kehyksellä. Määrittelyjen perusteella luotiin tuotteelle backlogi, jota lähdettiin toteuttamaan neljän 2 viikon pituisen sprintin aikana.

10.1 Kehitysympäristö

Sovellus on kehitetty Android sovelluksille ja sovelluksen kehitys tapahtui käyttämällä Android Studiota. Kehityksessä hyödynnettiin myös GitHubia, jonka avulla kehitystiimi teki yhteistyötä.

10.2 Testausympäristö

Testaus suoritetaan Jenkinsillä, joka toimii Metropolian servereillä. Jenkins varmistaa sovelluksen jatkuvan toiminnan suorittamalla sovelluksen yksikkötestit tasaisin väliajoin.

11 Yhteenveto

Sovellus saavutti asetetut vaatimukset mutta jatkokehitystä varten on suunnitteilla. Applikaation ”backend” eli tietokantatoteutus siirretään pilveen ja tietokannalle tehdään REST-api Java Spring kehyksen avulla. Käyttäjille pitää tehdä autentikaatio ja kirjautumistoiminnallisuus.

Sovelluksen nykyistä toteutusta voisi myös refaktoroida ja muokata modulaarisemmaksi. Treeni historian visualisointi on myös tärkeä ominaisuus joka pitäisi lisätä. Myös harjouitusten jakaminen ja vertailu muiden käyttäjien kesken ovat mahdollisia jatkokehitysaiheita.

Agile metodien mukainen toteutus onnistui hyvin scrumia hyödyntämällä. Sprintit olivat kaikki onnistuineita, toista lukuunottamatta, jolloin sprintin backlogiin otettiin liikaa käyttäjätarinoita toteutettavaksi. Anroid Studioon tottumattomuuden vuoksi kehityksessä kesti myös odotettua pidempään.

Scrumin toteutus onnistui hyvin. Kehitystiimi sai pidetyksi lähes päivittäisiä palavereja ja kehitys oli sujuvaa. Viimeisimmän sprintin lopuksi saatiin myös valmiiksi sovellus joka toteutti asetetut vaatimukset.

12 Liitteet

1. https://github.com/JoonasMV/Trainer