Jenni Harakka, Mikko Tanhola, Olli Varila, Joonas Viljanen

Trainer-sovellus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintäteknologian tutkinto-ohjelma

Toteutusdokumentti

5.5.2023

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 halutessaan. Sovellukseen voi myös luoda uusia liikeitä tai treenejä halutessa. Sovelluksen uusin ominaisuus on treenien jakaminen muille käyttäjille ja toisen käyttäjän treenin tallentaminen itselle.

Tämä 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. Luvussa 9 kerrotaan sovelluksen lokalisaatiosta ja luvuissa 10 ja 11 on ohjeet jatkokehittäjälle sekä tietoa kehitysprosessista- ja ympäristöstä. Dokumentaation lopussa on yhteenveto ja jatkokehityssuunnitelmat.

2 Käsitteet

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

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

Exercise – Treeniin (workout) lisätty liike, sisältää tiedon liikkeen nimestä, sekä seteistä, jotka treenin aikana on suoritettu eri määrillä toistoja ja painoja

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

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

Quote – Motivoiva lainaus/sanonta sovelluksen etusivulla. Lokalisoitu.

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

Workout – Kokonainen treeni tai 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 suunnattu kokeneemmille harjoittelijoille. Kyseiselle käyttäjäryhmälle on annettava mahdollisuus luoda uusia liikkeitä tai harjoitusohjelmia. Käyttäjän on myös pystyttävä jakamaan uusia luomiaan harjoituksia muille käyttäjille, jotka pystyvät näitä hakemaan käyttäjänimen avulla.

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.

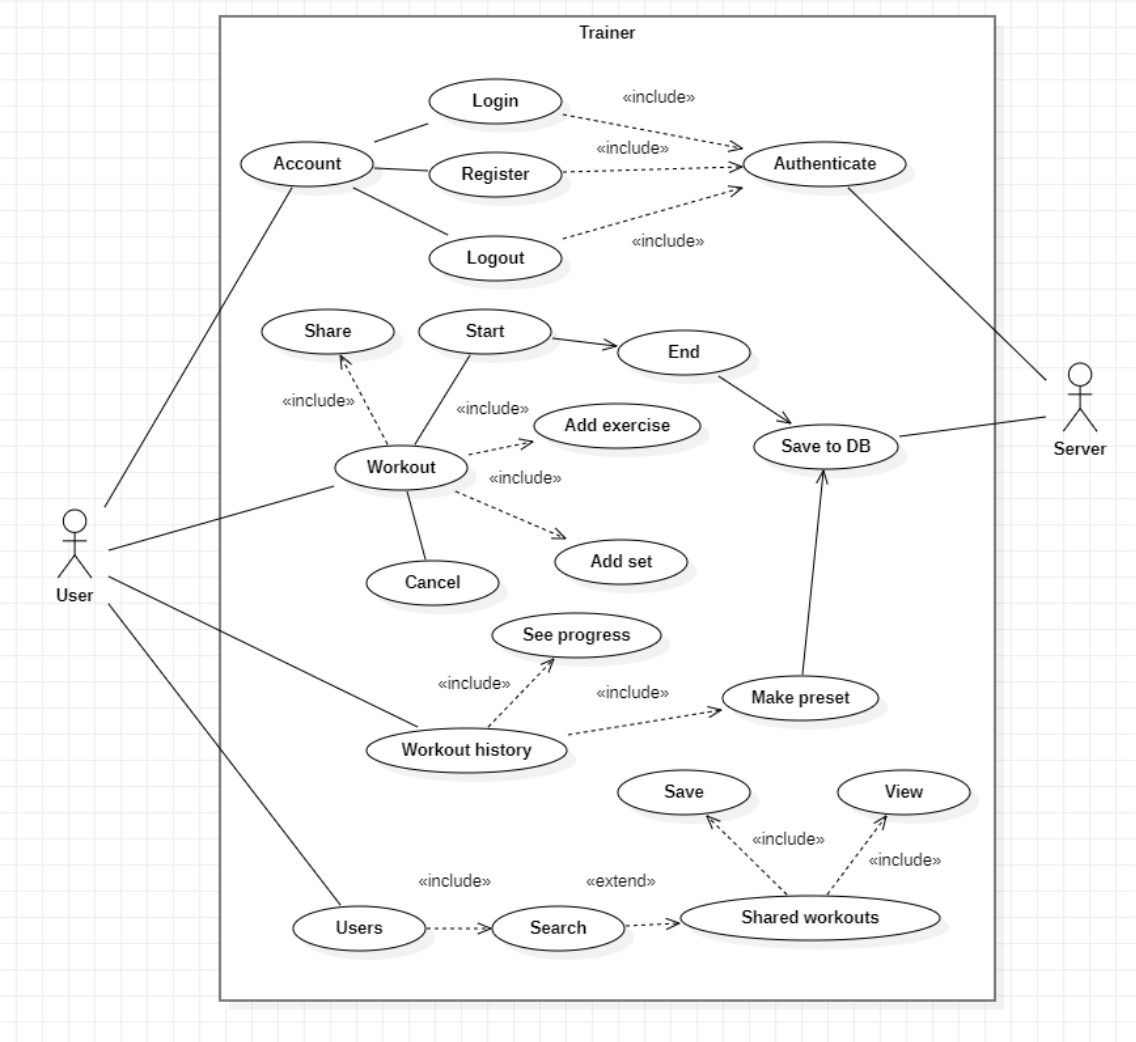
4 Käyttöönotto

Sovelluksen voi ottaa käyttöön lataamalla stabiilin version .apk tiedoston GitHub[1] sivulta ja asentamalle sen omalle Android laitteelle. Tämän jälkeen on luotava uusi käyttäjä, jonka jälkeen sovellus on käyttövalmis.

Uusimman version saa käyttöön kloonaamalla projektin tiedostot omalle koneelle ja asentamalla sen esimerkiksi Android Studion avulla omalle puhelimelle.

Sovelluksella on MIT-lisenssi, joten kuka vain on vapaa tekemään jatkokehitystään halutessaan. Mahdollisen jatkokehitys tiimin on kuitenkin luotava oma postgreSQL- tietokantansa ja liitettävät se Trainer-API:iin, johon löytyy linkki Trainerin GitHub-sivulta.

5 Käyttäjäroolit ja käyttötapaukset



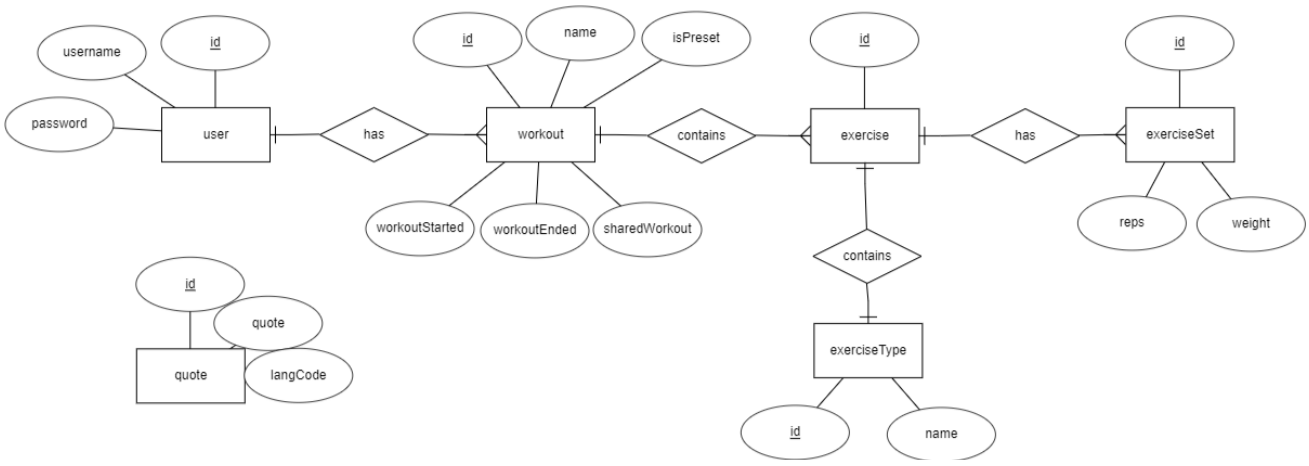
Kuva 1. Käyttötapauskaavio

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, tarkastella edistymistään ja luoda uusia liikkeitä. Uusin ominaisuus on harjoitusten jakaminen muille käyttäjille, sekä muiden jakamien harjoitusten tallentaminen itselle.

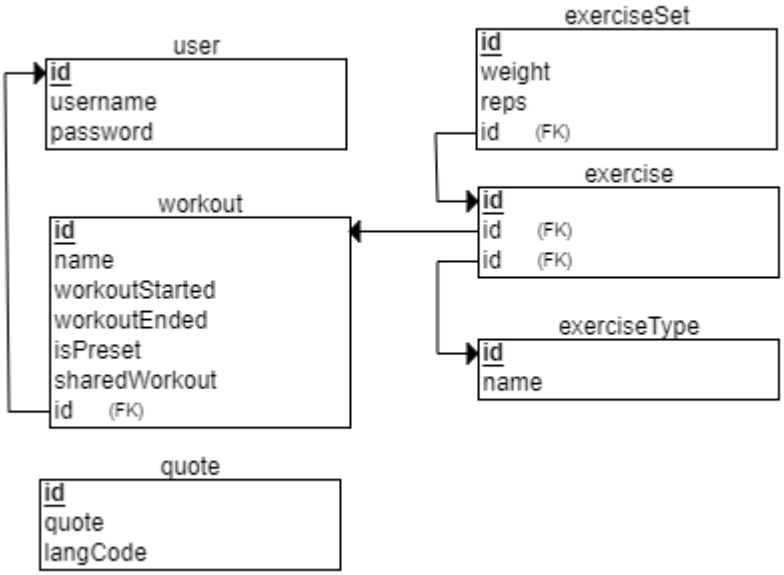
6 Ohjelmiston tietomalli

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. Lisäksi tietokannassa on Quote pöytä, joka sisältää motivaatiota lisääviä lainauksia. Näitä näytetään käyttäjälle sovelluksen etusivulla. Quotet on lokalisoitu tietokannassa. Yksittäisen rivin kolumni langCode kertoo millä kielellä se on.



Kuva 2. ER-kaavio

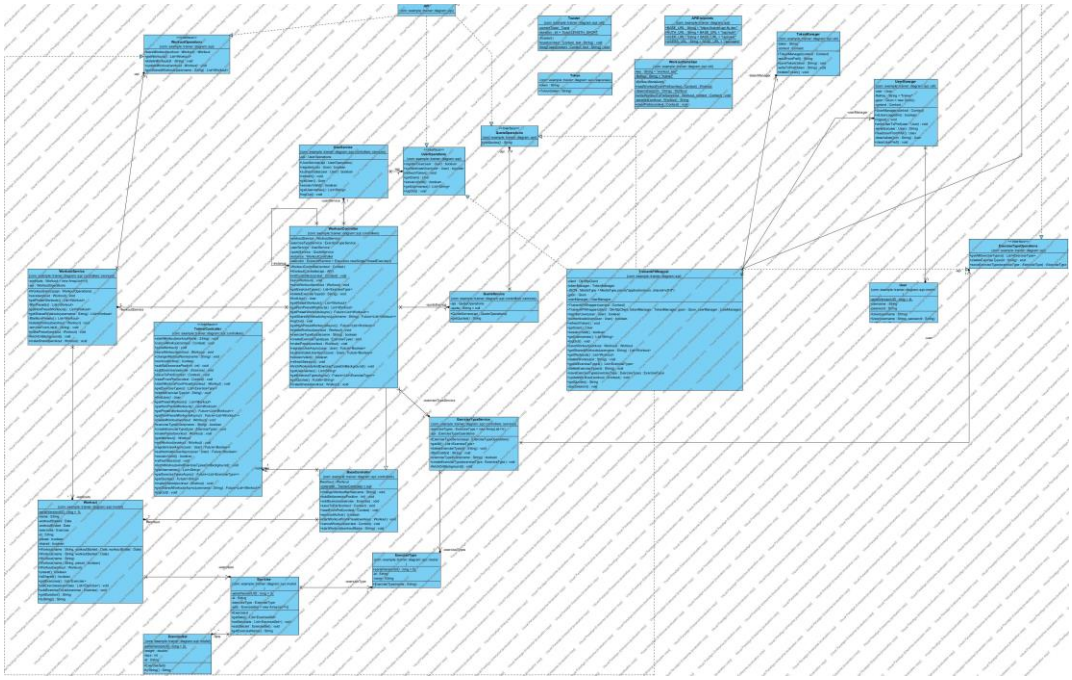
Alla olevassa kuvassa näkyy ER-diagrammin pohjalta luotu relaatiotietokanta-kaavio. Tietokannan luomisesta ja päivittämisestä on vastuussa Hibernate sekä JPA. Tietokanta on normalisoitu kolmanteen normaaliin muotoon (third normal form).



Kuva 3. Relaatiotietokanta

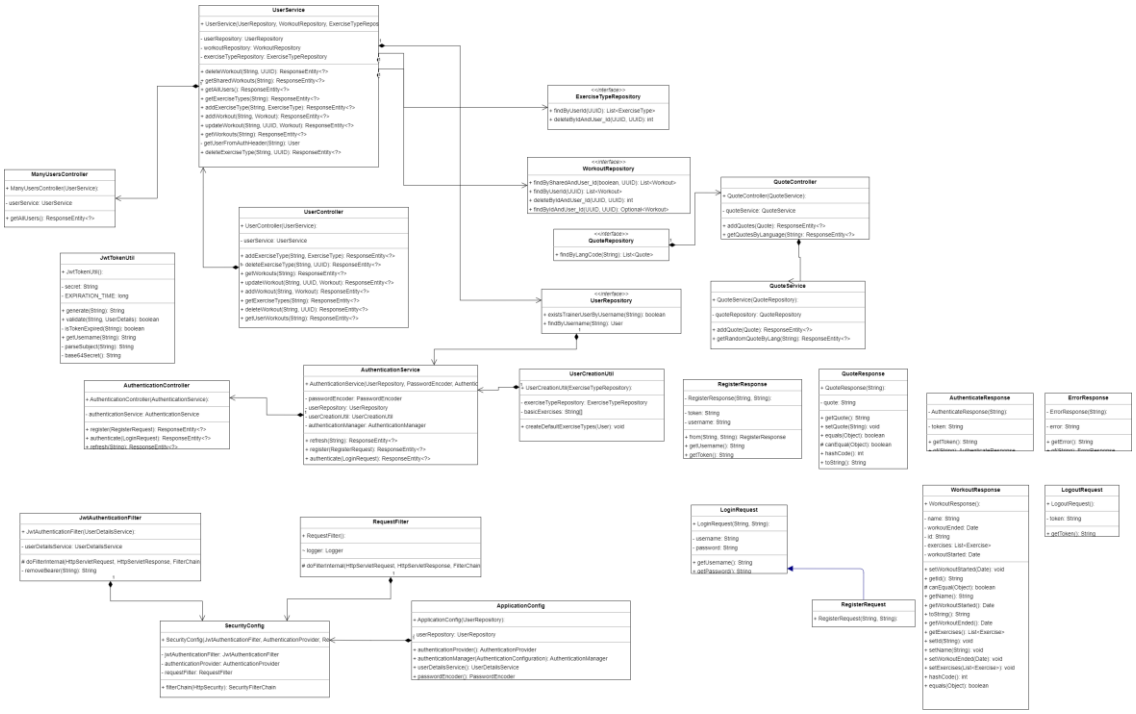
7 Ohjelmiston rakenne

Alla olevassa luokkakaaviossa (Kuva 11) on kuvattu sovelluksemme luokkarakenne, poissulkien sovelluksemme käyttöliittymäluokat. Kaaviossa on kuvattu logiikkaa käsittelevät luokat, kuten Workout ja User, Controller-, Manager- ja Service-luokat sekä Operations-rajapinnat.



Kuva 4. Trainer-sovelluksen luokkakaavio

Kaaviossa (Kuva 4) kuvattavissa luokissa ei ole kuvattu välttämättä kaikkia kyseisen luokan muuttujia tai metodeja, ainoastaan oleellisimmat.



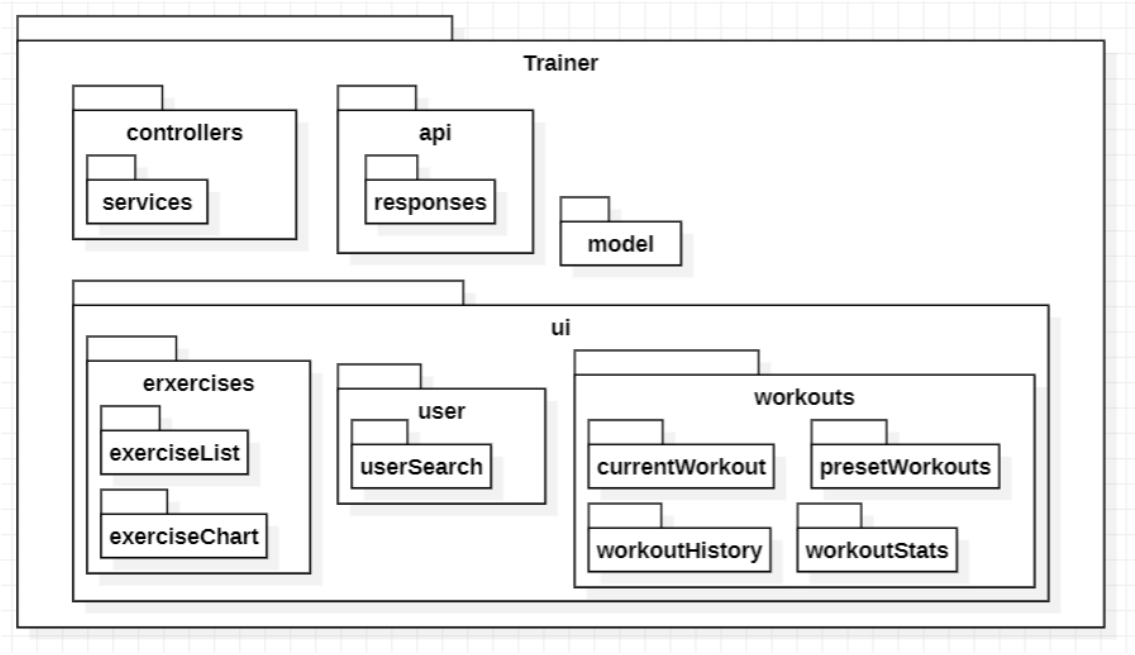
Kuva 5. Trainer-API luokkakaavio

Yllä olevassa kaaviossa (Kuva 5) on kuvattu backendin luokat ja niiden suhteet. Kaaviosta on jätetty pois model-pakkauksen luokat koska niitä käsiteltiin aikaisemmin kappaleessa 6.

Kuva 6 kuvaa sovelluksemme pakkausrakennetta. Uloimmaisin pakkaus on itse sovellus, jonka sisällä on kaksi erottelevaa pakkausta, Framework ja Frontend.

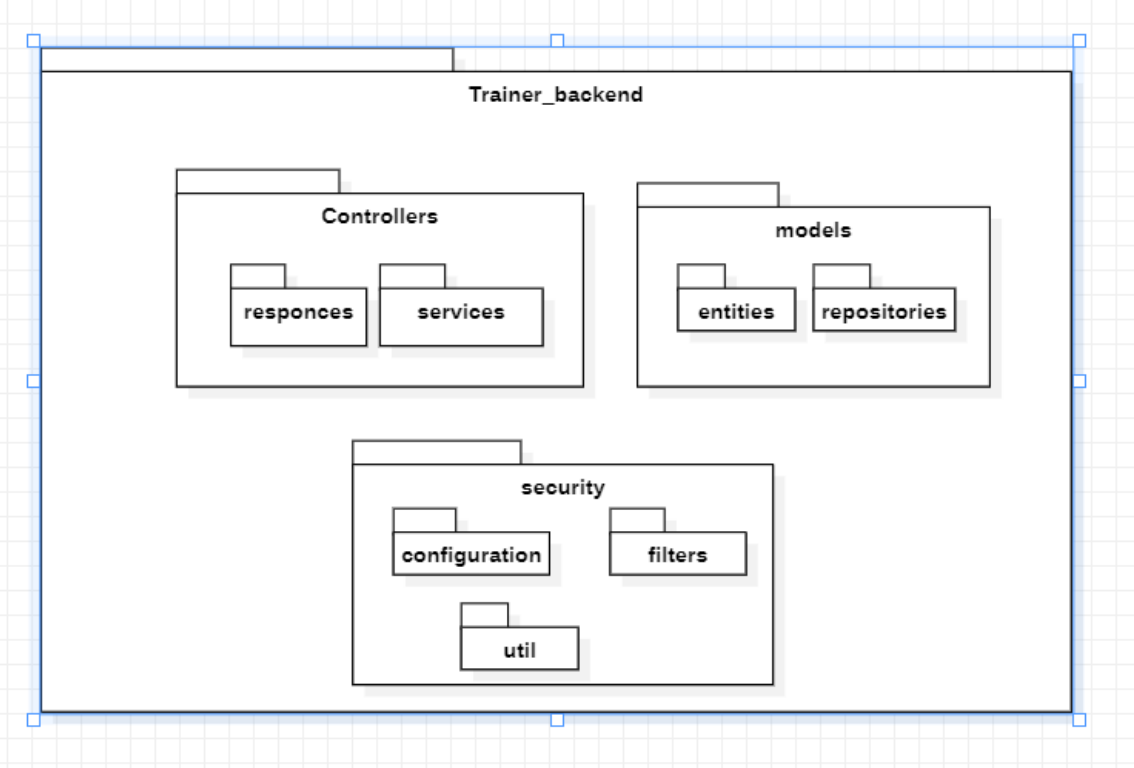
Controllers-paketti sisältää käyttöliittymäkomponenttien kanssa interaktion ylläpitävän ohjaimen. Services paketti pitää sisällään metodit, joita tämä ohjain käyttää.

API-paketti sisältää kaikki API:n kanssa keskustelun hoitavat luokat. Sen sisällä oleva responses-kansio sisältää käyttäjän tokeniin liittyvät toiminnot. Model-kansio sisältää sovelluksessa olevat oliot. Controllers-luokka sisältää API:n kanssa keskustelun hoitavat luokat. Ui-pakkauksessa ovat kaikki luokat, jotka määrittelevät sovelluksen eri näkymien toiminnan.



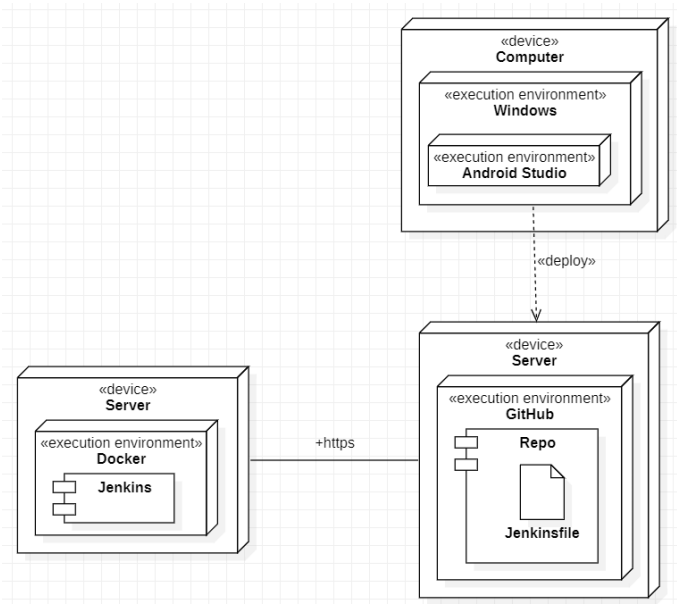
Kuva 6. Trainer-sovelluksen pakkauskaavio

Kuvassa 7. kuvataan backendin rakennetta pakettikaaviolla. Controllerit ovat vastuussa serverille lähetettävien pyyntöjen suorittamisesta ja vastausten lähettämisestä. Modelit sisältävät käytettävät tietorakenteet ja security paketista löytyy tietoturvaan liittyviä toteutuksia.



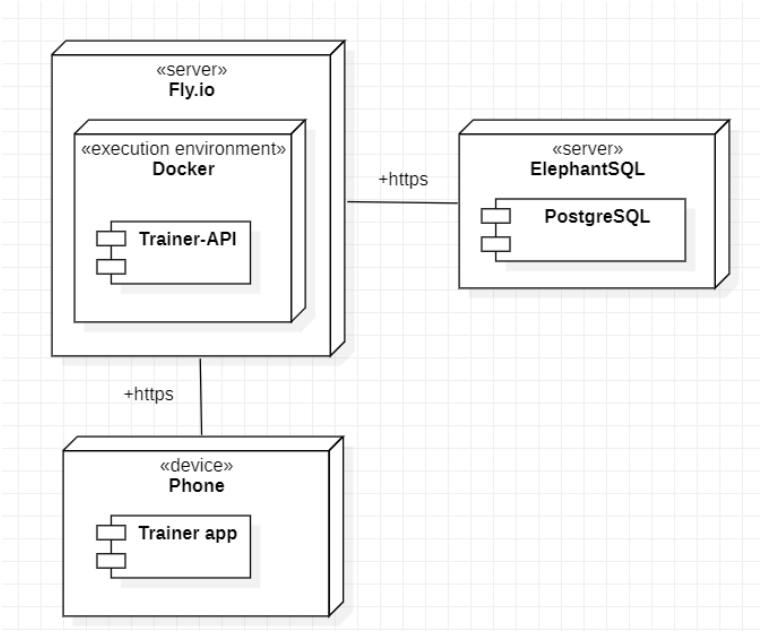
Kuva 7. Backendin pakkauskaavio

Alla oleva sijoittelukaavio (Kuva 8) kuvaa sovelluksemme kehitysympäristöä. Sovellusta kehitettiin Android Studiolla, josta muutokset päivitetään GitHubiin. Repositoriossa on myös Jenkinsfile, jossa on määritelty Jenkins-palvelimella ajettava testiautomaatio pipeline.



Kuva 8. Kehitysympäristön sijoittelukaavio

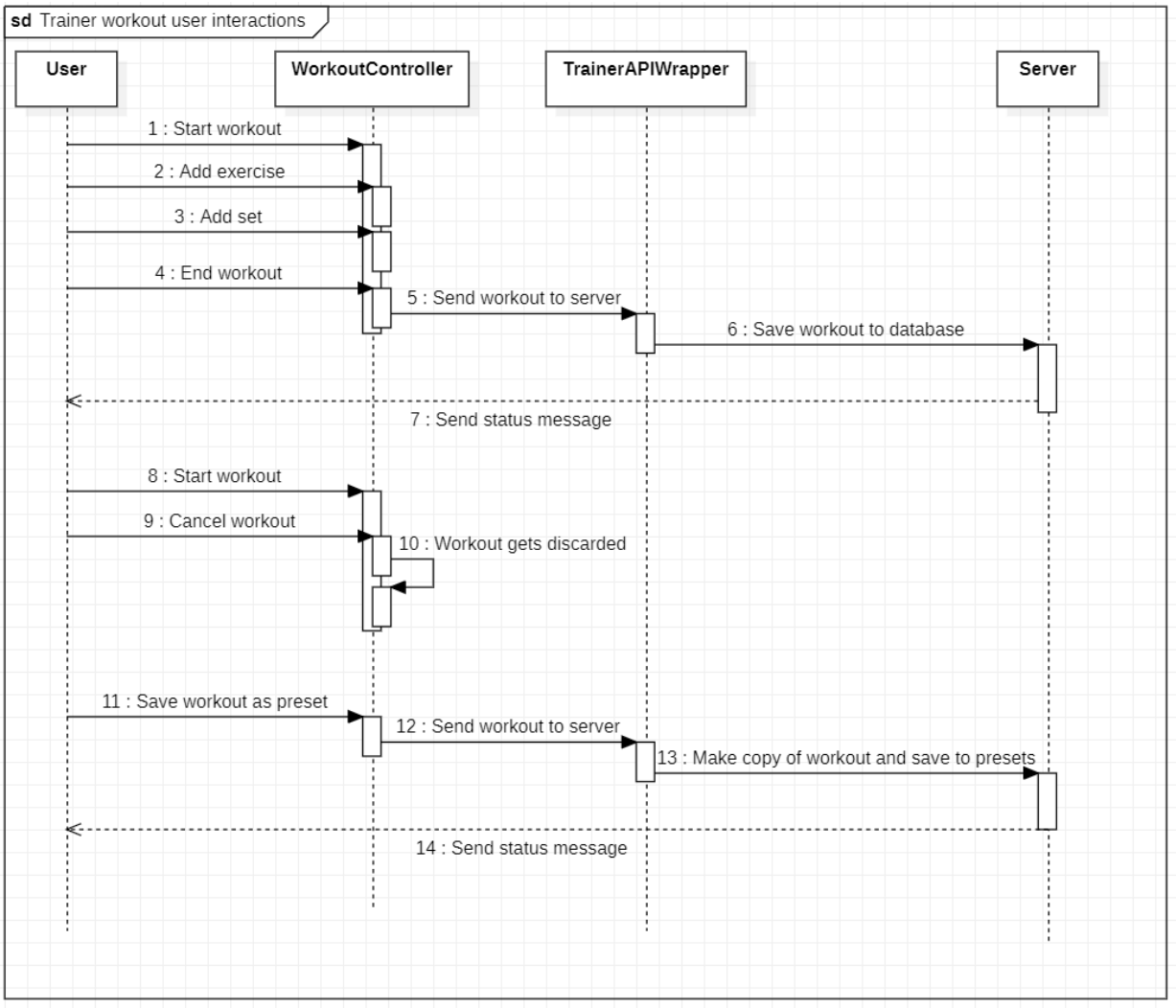
Kuvassa 9 kuvattuna, miten sovelluksen eri osa keskustelevat toistensa kanssa. Kuvasta käyttäjän laitteella on käynnissä pelkästään sovellus, joka keskustelee API:n kanssa, jonka kautta käyttäjää tallentaa ja lukee omia tietojaan.



Kuva 9. Sovelluksen sijoittelukaavio

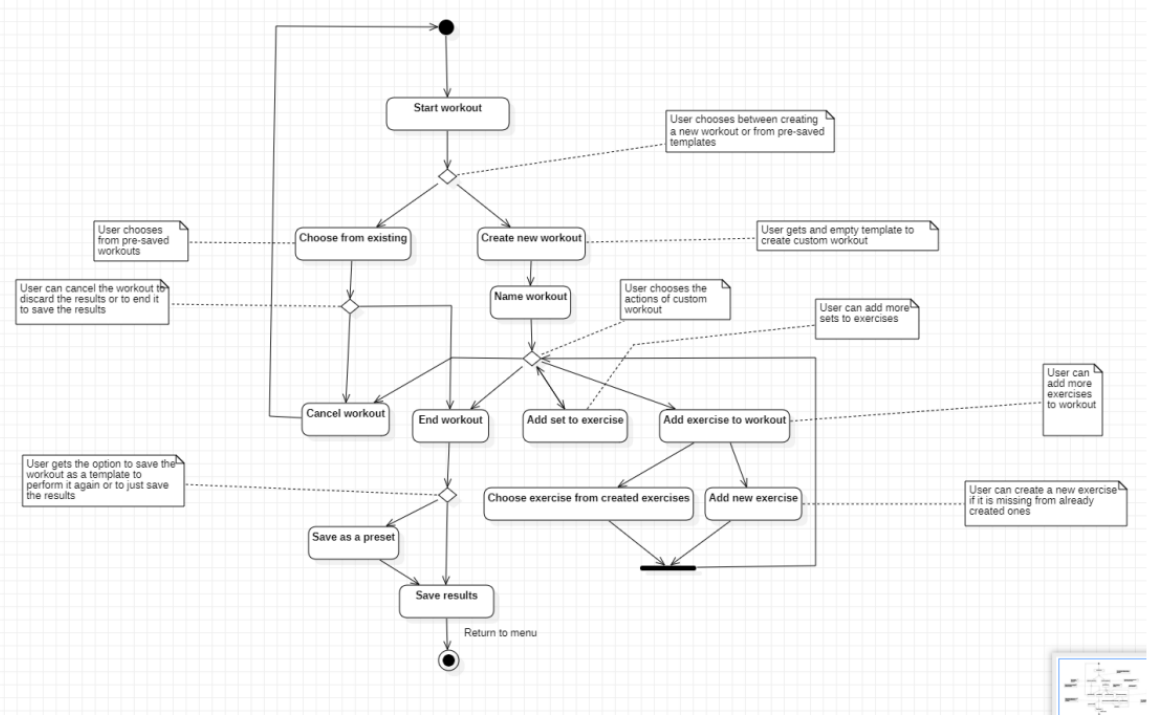
8 Ohjelmiston toiminta

Sovelluksen päätoimintoihin kuuluu liikkeiden ja harjoitusten luominen, sekä harjoitusten jakaminen. Kuvassa 10 näytetään sekvenssikaavion avulla uusien harjoitusten luominen, sekä niiden tallentaminen treenipohjiksi.



Kuva 10. Sekvenssikaavio harjoituksen luomisesta Trainer-sovelluksella

Kuvassa 11 on kuvattu aktiviteettikaaviolla, miten käyttäjä pystyy käyttämään sovellusta. Kuvattuna on treenin aloittaminen ja siihen liittyvät toiminnot, kuten setin tai liikkeen lisääminen ja treenin lopettaminen.



Kuva 11. Aktiviteettikaavio harjoituksen luomisesta Trainer-sovelluksella

9 Lokalisaatio

Sovelluksen oletuskielenä on englanti, mutta sovellus on myös lokalisoitu suomeksi, armeniaksi, sekä espanjaksi. Sovelluksen kieli riippuu käytettävän puhelimen kielestä.

Sovelluksen lokalisointi tehtiin käyttämällä Android Studion sisäänrakennettuja lokalisaatiotyökaluja. Tämä tarkoittaa sitä, että jokaiselle kielelle luotiin oma string.xml-tiedosto. Sovellus osaa itse vaihtaa kielensä oikeaan, ja hakee oikeat string-resurssit kyseisistä tiedostoista.

Sillä tietokantamme ovat pääasiassa käyttäjien itse tekemiä, emme nähneet järkeväksi lokalisoida näitä tietoja. Tämän takia loimme uuden Quote-taulukon, jonka lokalisoimme riveittäin. Jokaisella Quotella on langCode-kenttään merkittynä kyseisen lainauksen kieli, jonka perusteella se haetaan tietokannasta.

10 Jatkokehittäjälle

Seuraava kappale on tarkoitettu mahdollisille jatkokehittäjille. Kappaleessa käydään läpi, miten sovellukseen saadaan lisätyksi uusia näkymiä ja API- pyyntöjä. Sovellus on kehitetty Android studiolla ja jatkokehitystä suositellaan samalla sovelluksella.

10.1 Näkymien lisäys

Sovelluksen rakenteen vuoksi, lähes kaikki uudet näkymät ovat fragmentteja muutamaa erityistapausta laskematta. Uudet näkymät tulee näyttää FragmentContainerView:ssa nimeltä ”mainContainer”. Android studiossa olevan bugin vuoksi jokaisen fragmentin onCreateView() metodiin on lisättävä seuraava koodin pätkä.

```
if (container != null) {
    container.removeAllViews();
}
```

Koodi varmistaa, että näkymä vaihtaa oikein tietyissä tapauksissa, joissa joudutaan backStackissa hyppimään jonon eri kohtiin.

10.2 API- pyyntöjen lisäys

API- pyynnöt lisätään TrainerApiWrapper.java luokkaan, tätä ennen on kuitenkin haluttu metodi lisättävä johonkin operations- rajanpintaan esim. QuoteOperations tai uusi rajapinta on luotava, jonka jälkeen tämä on implementoitava API.java, sekä TrainerApiWrapper.java luokkiin.

Pyynnöt backendiin suoritetaan TrainerApiWrapper.java luokassa, käyttämällä okHttp- kirjastoa (3). Uusien ominaisuuksien toiminnallisuus luodaan TrainerAPI:ssa.

10.3 Trainer-API

Trainer sovellus toimii yhdessä Trainer-API:n (2) kanssa. Mikäli jatkokehittäjä haluaa jatkaa kehitystä omaan suuntaan, on tämän luotava oma postgreSQL-tietokanta. Tämän voi tehdä kätevästi esimerkiksi Docker Composen avulla (Liite 4).

11 Kehitysprosessi ja kehitysvaiheen tekniikat

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

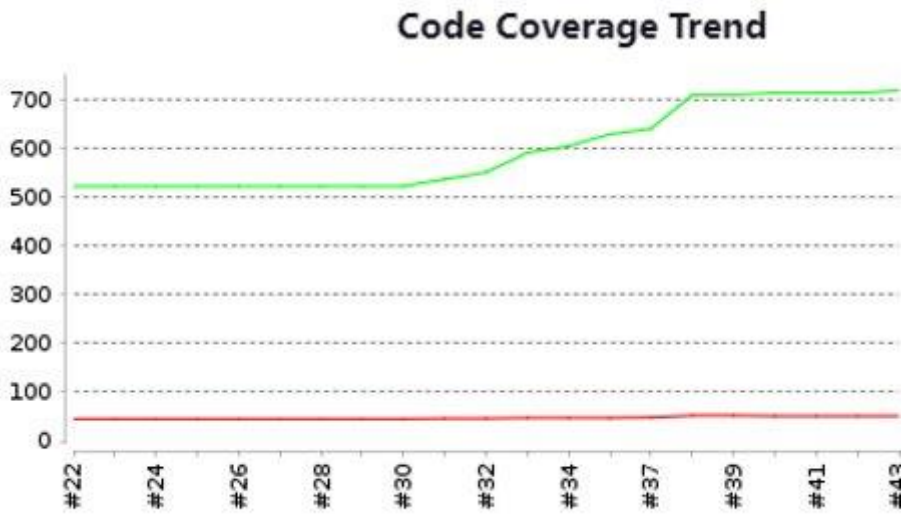
Kehityksen aikana seurattiin Agile-metodeja Scrum-kehyksellä. Määrittelyjen perusteella luotiin tuotteelle backlogi, jota lähdettiin toteuttamaan neljän 2 viikon pituisen sprintin aikana, jonka aikana perusominaisuudet lisättiin. Tämän jälkeen kehitystä jatkettiin kolmella 3 viikon pituisella sprintillä, jonka aikana sovellukseen lisättiin sosiaalisia toimintoja ja tietojen tallentaminen siirrettiin pilveen.

11.1 Kehitysympäristö

Sovellus on kehitetty Android-alustalle ja sovelluksen kehitys tapahtui käyttämällä Android Studiota. Sovelluksen backend kehitettiin Spring boot -kehyksen avulla ja tähän käytettiin IntelliJ Idea (Ultimate tai community edition) editoria. Versionhallinnassa hyödynnettiin Githubia. Backendin kehityksessä oli myös käytössä Github Actions, jonka avulla buildataan sovellus ja julkaistaan uusin versio palvelimelle automaattisesti.

11.2 Testausympäristö

Testaus suoritetaan Jenkinsillä, joka toimii Metropolian educloud ympäristössä. Jenkins varmistaa sovelluksen jatkuvan toiminnan suorittamalla sovelluksen yksikkötestit tasaisin väliajoin. Jenkinssissä on määritetty pipeline Trainer applikaatiolle sekä Trainer API:lle. Apin testit voidaan ajaa missä tahansa ympäristössä ilman suurempaa konfiguraatiota, koska testeissä käytetään tietokantana H2-tietokantaa, joka on siis in-memory tietokanta.



Kuva 10 Backendin test coverage

Kuva 11 Frontendin test coverage

12 Yhteenveto

Sovellus saavutti alkuperäiset vaatimukset, sekä suurimman osan jatkokehitysideoista. Ainoaksi puutteeksi jäi kaverilista ominaisuus, joka oli suunnitteilla.

Alkuperäisen kehityksen aikana sovellukseen saatiin lisätyksi kaikki perusominaisuudet. Sovellukseen luotiin mahdollisuus luoda liikkeitä ja treenejä, sekä mahdollisuus tarkastella treenihistoriaa yksinkertaisella tavalla. Tämän jälkeen luotiin jatkokehitysideoita, jotka pääasiallisesti liittyivät sovelluksen pilveen siirtämiseen.

Jatkokehityksen aikana sovelluksen koko tietokanta siirrettiin kokonaan pilveen. Sovellukselle luotiin REST- API Spring boot -kehyksen avulla. Käyttäjiä varten luotiin autentikaatio ja kirjautumistoiminnallisuus. Oman kehityksen seuraaminen tehtiin visuaalisemmaksi.

Alkuperäisen kehitysvaiheen jälkeen koodia myös refaktoroitiin suuresti pilveen siirtämisen yhteydessä, koodiin lisättiin kommentteja ja frontti koodia uusittiin.

Agile metodien mukainen toteutus onnistui hyvin Scrum-metodologiaa hyödyntämällä. Sprintit olivat kaikki onnistuneita alkuperäisessä kehityksessä, toista lukuunottamatta, jolloin sprintin backlogiin otettiin liikaa käyttäjätarinoita toteutettavaksi. Android Studioon tottumattomuuden vuoksi kehityksessä kesti myös odotettua pidempään.

Jatkokehityksessä Scrum-metodit olivat paljon paremmin opittuja ja aikasemman kehityksen vaikeudet olivat kokonaan hävinneet. Kaikki suoritettavaksi otetut käyttäjätarinat saatiin suoritetuksi ja Android Studio oli jo hallinnassa, joten kehitys oli paljon nopeampaa.

Scrumin toteutus onnistui koko kehityskaaren ajan. Tiimi sai pidetyksi lähes päivittäisiä palavereja ja kehitys oli sujuvaa.

13 Jatkokehitys

Sovelluksesta puuttuu vielä ystävien lisäys ominaisuus. Jaetuille harjoituksille olisi myös mahdollista lisätä äänestysjärjestelmä, jolloin yhteisön hyväksi toteamat harjoitukset saisivat enemmän näkyvyyttä.

Sovelluksessa on myös ongelma, jos nettiyhteys katkeaa tallennuksen yhteydessä, harjoitus katoaa. Myös jonkinlainen lokaali tietokanta voisi olla hyvä lisätä, jolloin API-pyyntöjen vuoksi syntyvä viive ei näkyisi käyttäjälle. Sovellusta voisi myös lokalisoida useammalle alueelle sekä tehdä se kokonaisvaltaisemmin.

14 Liitteet

1. https://github.com/JoonasMV/Trainer
2. https://github.com/ollivarila/Trainer-API
3. https://square.github.io/okhttp/
4. https://geshan.com.np/blog/2021/12/docker-postgres/

Luokkakaaviot, löytyvät paremmalla laadulla ”kaaviot” kansiosta