

Vera Finogenova, Valtteri Kuitula, Jhon Rastrojo, Korpi Tolonen
Raportti Word Wonders -projektityöstä
Ryhmä 7

Metropolia Ammattikorkeakoulu

Tieto- ja viestintätekniikka

Ohjelmistotuotanto

11.12.2023

Sisällysluettelo

1	Johdanto	4
2	Asiakkaan vaatimukset	4
2.1	Toiminnalliset vaatimukset	4
2.2	Laadulliset vaatimukset.....	5
3	Kehitysmenetelmät	5
4	Ohjelmiston suunnittelu ja arkkitehtuuri	6
4.1	Käyttäjäroolit ja käyttötapaukset	7
4.2	Ohjelmiston rakenne	9
4.3	Ohjelmiston toiminta	10
5	Ohjelmiston ohjelmointitekniinen toteutus.....	14
5.1	Käytetyt ohjelmointikielet ja kirjastot (ulkoiset API:t)	14
5.2	Tietokannan kuvaus.....	14
5.3	Lokalisointi	15
5.3.1	Käyttöliittymän lokalisointi.....	15
5.3.2	Lokalisointiresurssit	16
6	Ohjelmiston testaus	17
6.1	Testitapaukset.....	18
6.1.1	Session luonti ja käyttöönotto	18

6.1.2	Profiilikuvan vaihtaminen	19
6.1.3	Kielen vaihto	20
6.1.4	Kokemuspisteiden ansaitseminen	21
6.2	Testien ajaminen.....	22
7	Yhteenveto.....	23
8	Liitteet	24

1 Johdanto

Tässä dokumentissa määritellään Metropolia-ammattikorkeakoulun järjestämien Ohjelmistotuotantoprojekti 1 ja 2 -kurssien aikana toteutettu projektityö osana ohjelmistotuotannon opintosuunnitelmaa. Dokumentissa käydään läpi projektityön toiminnalliset ja laadulliset vaatimukset, kehitysmenetelmät, suunnittelu ja arkkitehtuuri, ohjelmointitekniinen toteutus sekä testaus. Projektityön aiheena on englannin kielen oppimiseen tarkoitettu mobiilisovellus, joka on suunnattu 6–8-vuotiaille lapsille. Projektityön toteutuksessa olivat mukana opiskelijat Vera Finoganova, Valtteri Kuitula, Jhon Rastrojo ja Korpi Tolonen.

2 Sovelluksen vaatimukset

2.1 Toiminnalliset vaatimukset

Mobiilisovellus koostuu useista eri ominaisuuksista, jotka muodostavat selkeän kokonaisuuden. Käyttäjä voi luoda sovellukseen oman tallennuksen ja ainoat tietovaatimukset ovat profiilikuva sovelluksen tarjoamista kuvavaihtoehtoista, nimimerkki sekä käyttäjän ikä. Käyttäjä valitsee myös kielen, jolla haluaa käyttää sovellusta. Kieli voidaan vaihtaa sovelluksen asetuksista. Käyttäjän on mahdollista poistaa luomansa tallennus jälkikäteen sovelluksen asetuksista. Lisäksi käyttäjä voi profiilinäkymässä vaihtaa aikaisemmin asettamansa profiilikuvan uuteen.

Sovelluksen perimmäisenä tarkoituksena on toimia opettavaisena kielisovelluksena käyttäjälle pelillistämisen kautta. Sovellus sisältää neljä peliä, joita käyttäjä voi pelata ja samalla oppia englanninkielisiä sanoja. Käyttäjän on mahdollista poistua pelistä kesken pelaamisen. Jokainen peli sisältää etenemispalkin, joka viestii käyttäjän etenemistä kyseisessä pelissä.

Käyttäjä ansaitsee sovelluksen peleistä kokemuspisteitä, jotka määräävät käyttäjän henkilökohtaisen tason. Kokemuspisteet lisätään käyttäjälle pelin jälkeen ja ne visualisoidaan käyttäjälle edistyspalkin muodossa sovelluksen koti- ja asetusnäkymissä. Käyttäjän taso vaikuttaa suoraan sovelluksen peleihin; mitä suurempi taso, sitä enemmän sana- ja kuvavaihtoehtoja peleissä.

2.2 Laadulliset vaatimukset

Yksi projektiryhmän pääprioriteeteista on sovelluksen helppokäyttöisyys. Tämä pitää sisällään muun muassa selkeän käyttöliittymän, helposti opittavan pelilogiikan sekä yksinkertaisen käyttäjän luonnin. Lisäksi sovellus tukee kolmea eri kieltä: suomea, espanjaa ja arabiaa. Kokonaisuudessaan sovellus on hyvin selkeä ja yksinkertainen, joka mahdollistaa sujuvan jatkokehityksen.

Sovelluksen peleissä oikeat ja väärät vastaukset viestitään käyttäjälle selkeiden ponnahdusikkunoiden muodossa. Tämä parantaa käyttäjäkokemusta ja lisää kiinnostusta pelin suhteen. Lisäksi sovelluksen taustamusiikit ja erilaiset ääniefektit luovat rauhallista tunnelmaa ja vaikuttavat positiivisesti käyttäjän keskittymiskykyyn.

3 Kehitysmenetelmät

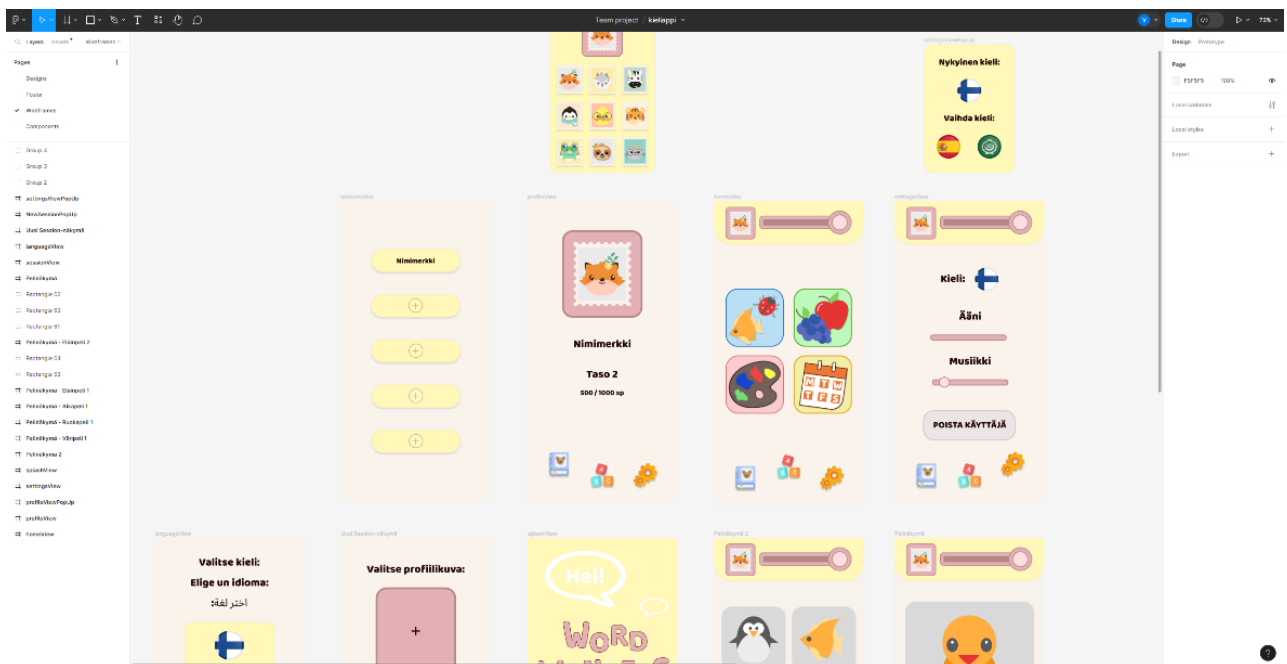
Projektityön kehityksessä hyödynnettiin Scrum-kehitysmenetelmää, jossa työ jaettiin lyhyisiin aikajaksoihin eli sprintteihin. Sprinttejä oli yhteensä seitsemän ja jokainen sprint kesti pääosin noin kaksi viikkoa. Ryhmän jäsenet toimivat jokainen ryhmänvetäjänä, eli Scrum Masterina, vähintään yhden sprintin ajan.

Jokainen sprint koostui erilaisista tehtävistä liittyen mobiilisovelluksen kehitykseen ja suunnitteluun. Ryhmä piti yhteyttä lähes päivittäin Discord-sovelluksen välityksellä.

Projektinhallintaan käytettiin Trello-nimistä verkkopohjaista työkalua, johon merkittiin kaikki toteutettavat tehtävät. Trellossa oli mahdollista jakaa tehtäviä ryhmän jäsenten kesken ja jakaa tehtäviä pienemmiksi suoritettaviksi osiksi. Projektityön versionhallintana toimi GitHub ja ryhmän työtunnit merkittiin Exceliin.

4 Ohjelmiston suunnittelu ja arkkitehtuuri

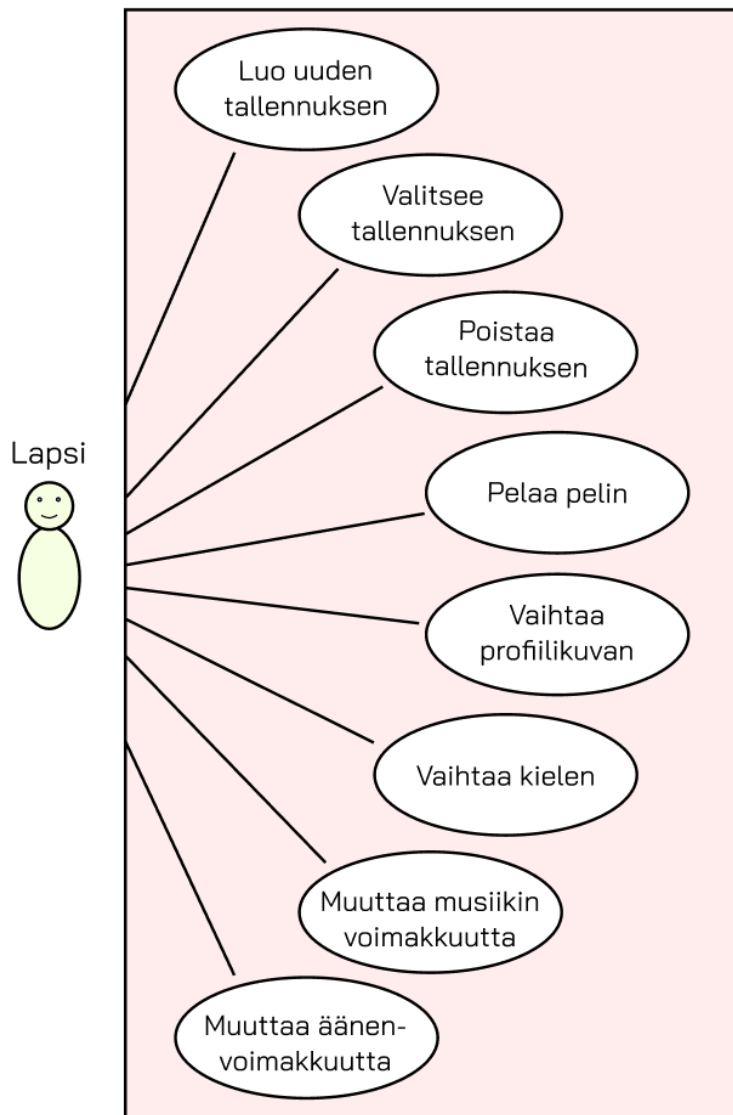
Ohjelmiston suunnittelussa hyödynnettiin laajasti erilaisia UML-kaavioita ja koko sovelluksen käyttöliittymä laadittiin suunnittelutyökalu Figmassa ennen lopullista toteutusta Android Studiossa (Kuva 1). Tässä osiossa kuvataan yleisesti erilaisia kaavioita, jotka toteutettiin projektityön suunnitteluvaiheessa. Sovelluksen tietomalli käy ilmi raportin ”Tietokannan kuvaus” -osiossa.



Kuva 1. Kuvakaappaus Figma-suunnittelutyökalusta, jossa sovelluksen näkymät suunniteltiin.

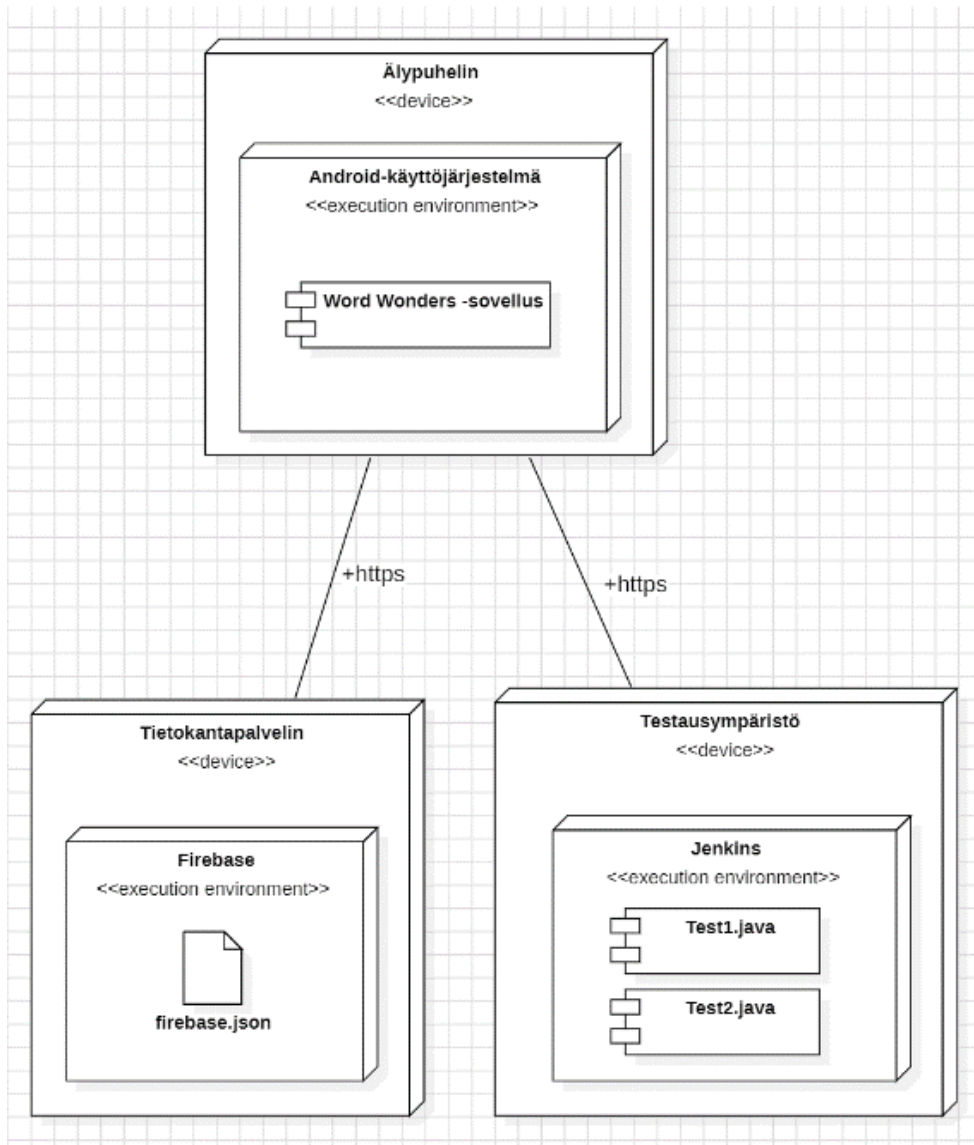
4.1 Käyttäjäroolit ja käyttötapaukset

Sovelluksen käyttäjäkunta koostuu pääosin lapsista. Sovelluksen toiminnallisuus on toteutettu hyvin yksinkertaistetusti, jonka myötä se ei sisällä käyttäjän lisäksi muita käyttäjärooleja. Kuva 2 havainnollistaa sovelluksen eri ominaisuuksia, joita käyttäjä (tässä tapauksessa lapsi) voi käyttää sovelluksessa.



Kuva 2. Käyttötapauskaavio lapsen näkökulmasta.

Sovelluksen kokonaisuuteen kuuluvat käyttäjän käyttämä laite (älypuhelin), tietokanta sekä testausympäristö. Sovelluksen tietokantayhteys ja automatisoitu testaus ovat yhteydessä laitteeseen Internetin välityksellä. (Kuva 3)

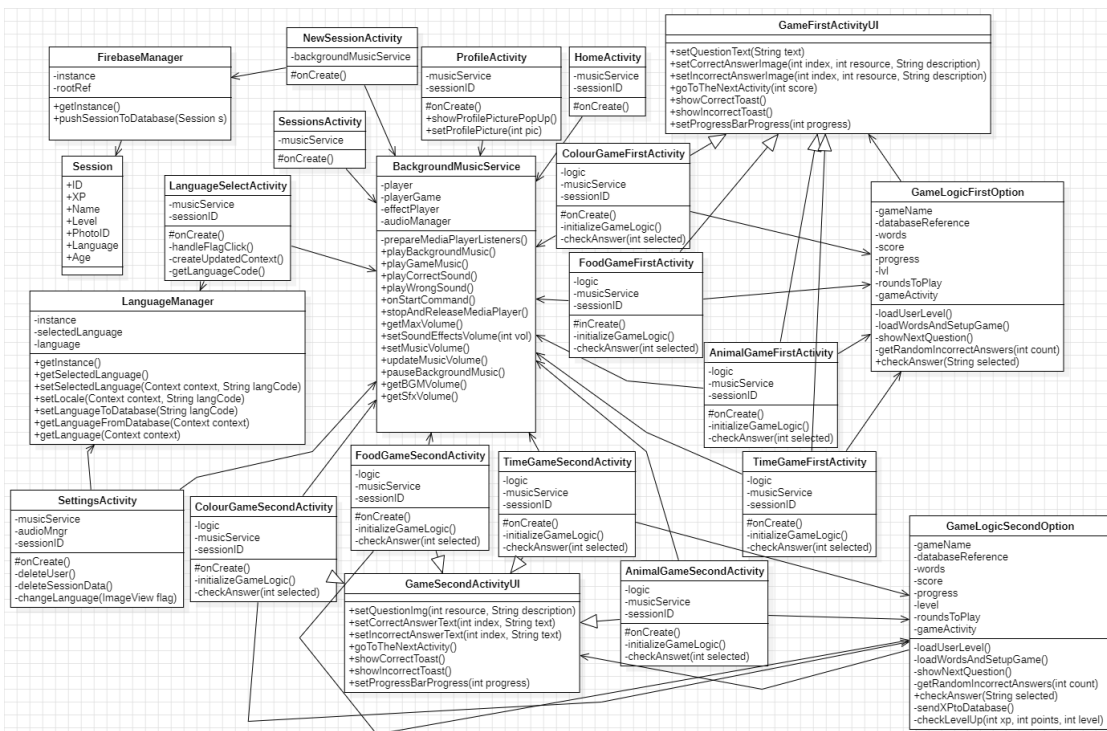


Kuva 3. Sijoittelukaavio järjestelmän eri osista.

Ohjelmiston rakenne

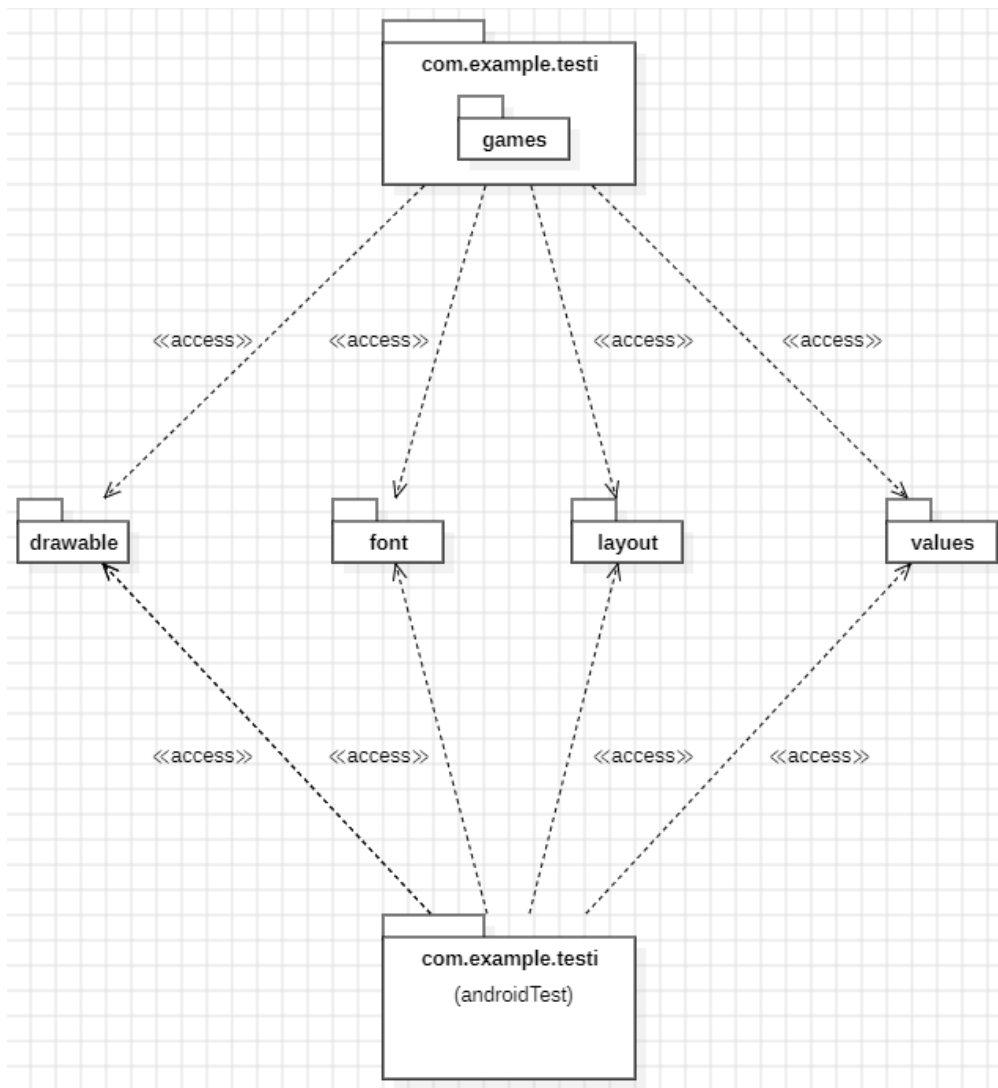
Ohjelma koostuu 22 luokasta, joista 15 on niin sanottuja aktiviteettiluokkia. Aktiviteettiluokat vastaavat sovelluksen eri näkymien, eli aktiviteettien, toiminnasta. Tässä tapauksessa aktiviteettiluokista löytyy käyttöliittymän käytökseen liittyviä metodeja sekä yleistä sovelluksen logiikkaa kuten pelien logiikkaa tai tietojen hakemista ja tallentamista tietokantaan. Aktiviteettiluokat hyödyntävät myös muita ohjelman luokkia, esimerkiksi LanguageManager-luokka vastaa sovelluksen kielen muuttamisesta.

Ohjelman luokkarakenne on esitetty kuvassa 4. Kaikki ohjelman aktiviteettiiluokat sisältävät sessionID-nimisen muuttujan, jolla paikannetaan oikea käyttäjä. Tämä tehdään session tietojen haun helpottamiseksi, sillä niitä käytetään kaikissa aktiviteeteissa.



Kuva 4. Sovelluksen luokkakaavio.

Ohjelman kooditiedostot, testitiedostot sekä tyylitiedostot ja -resurssit löytyvät omista pakkauksista. Sekä koodi- että testipakkauksen sisällä olevilla tiedostoilla on pääsy tyylipakkauksissa oleviin tiedostoihin. Ohjelman pakkausrakennetta kuvaava kaavio löytyy kuvasta 5.



Kuva 5. Sovelluksen pakkauskavio.

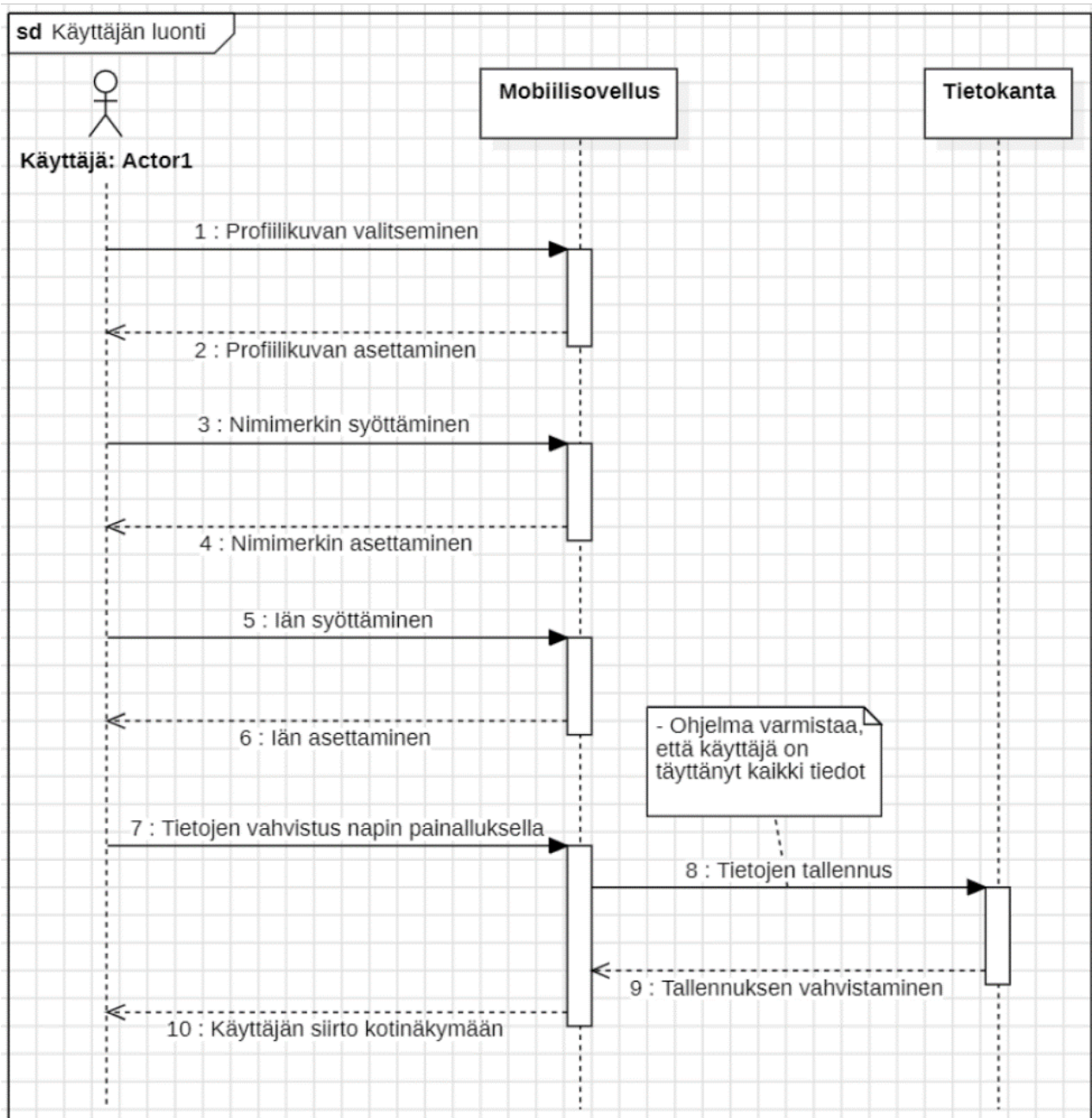
4.3 Ohjelmiston toiminta

Ohjelmisto koostuu viidestätoista eri näkymästä, joiden välillä käyttäjä liikkuu. Näkymiä ovat muun muassa käyttäjän tallennuksen luominen, asetusnäkymä, profiilinäkymä sekä kaksi erilaista pelinäkymää per peli (yhteensä kahdeksan pelinäkymää). Jokainen näistä sisältää toiminnallisuuksia. Kolme sovelluksen näkymistä sisältävät myös ponnahdusikkunan. Kuvassa 6 näkyy sovelluksen aloitusnäkymä, kotinäkymä sekä yksi ruokapelin näkymistä.



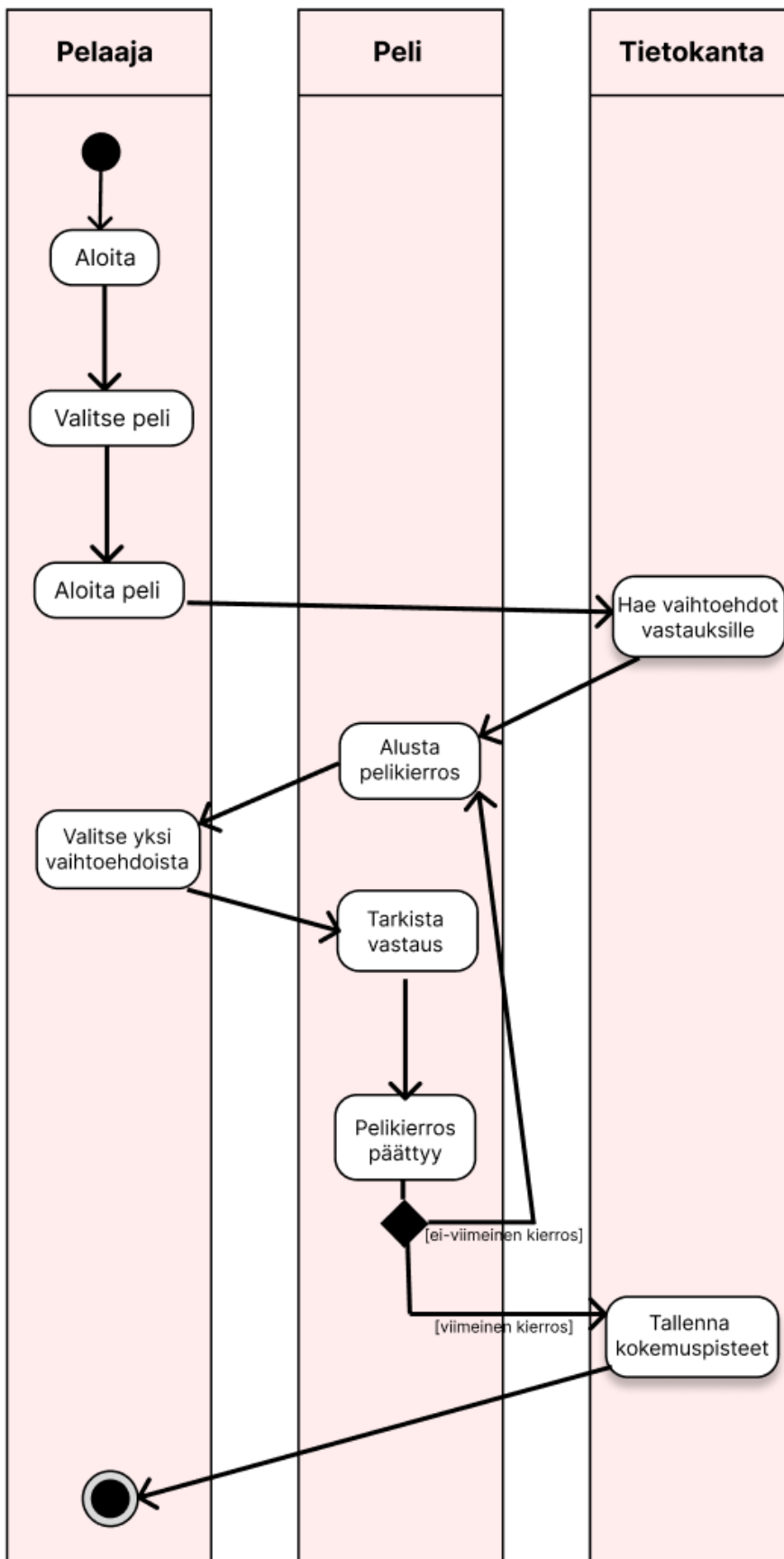
Kuva 6. Sovelluksen eri näkymiä.

Kuva 7 havainnollistaa miten käyttäjä luo itselleen uuden tallennuksen ollessaan siihen tarkoitetussa näkymässä. Kaikki täytettävät kentät eli profilikuva, nimimerkki ja ikä ovat pakollisia täyttää. Kun käyttäjä klikkaa "Valmis"-nappia, sovellus tarkistaa tietojen oikeanlaisen syöttötavan ja ilmoittaa, mikäli kentissä on sopimattomia merkkejä tai jokin kenttä on jätetty tyhjäksi.



Kuva 7. Käyttäjän luonti sekvenssikaaviona.

Kun käyttäjä on oman tallennuksensa aloitusnäkyssä, hän voi valita neljästä pelistä itselleen sopivan. Vaihtoehtoja ovat väripeli, eläinpeli, aikapeli ja ruokapeli, joista jokainen sisältää teemansa mukaista sanastoa ja kuvia. Kuvassa 8 on kuvattuna esimerkkitapaus sovelluksen pelin kulusta käyttäjän, pelin ja tietokannan välillä.



Kuva 8. Aktiviteettikaavio pelin kulusta.

5 Ohjelmiston ohjelmointitekhninen toteutus

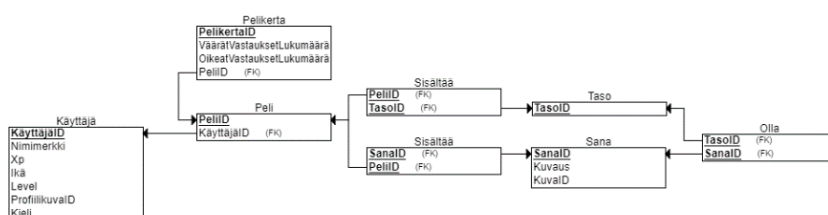
5.1 Käytetyt ohjelmointikielet ja kirjastot (ulkoiset API:t)

Projektityön pääohjelmointikielenä toimi Java, jolla kirjoitettiin kaikki luokat, testit ja sovelluksen aktiviteetit. Sovelluksen käyttöliittymän toteutuksessa hyödynnettiin XML-nimistä merkintäkieltä (eXtensible Markup Language). Kehitysympäristönä toimi Android Studio. Käytettyihin ulkoisiin kirjastoihin sisältyvät AndroidX, Espresso, Firebase SDK sekä JUnit.

5.2 Tietokannan kuvaus

Sovelluksen tietokantana toimii Firebase Realtime Database, joka on osa Googlen Firebase-palvelua. Tietokantaa hyödynnetään sovelluksen jokaisessa aktiviteetissa, sieltä haetaan ja sinne tallennetaan käyttäjän tietoja. Lisäksi tietokanta on tärkeässä osassa sovelluksen testeissä.

Huomioitavaa on se, että Firebase Realtime Database on NoSQL-tietokanta poiketen perinteisistä relaatiotietokannoista. Tämän vuoksi sovelluksen tietokantaa ei ole mahdollista mallintaa relaatiotietokantana. Kuva 9 mallintaa sovelluksen tietokannan rakennetta tapauksessa, jossa se toteutettaisiin SQL-tietokantana. Tietokantaan tallentuvat käyttäjän ja pelien tiedot. Lisäksi pelien eri sanat ja kuvat haetaan tietokannasta, kun käyttäjä aloittaa uuden pelin.



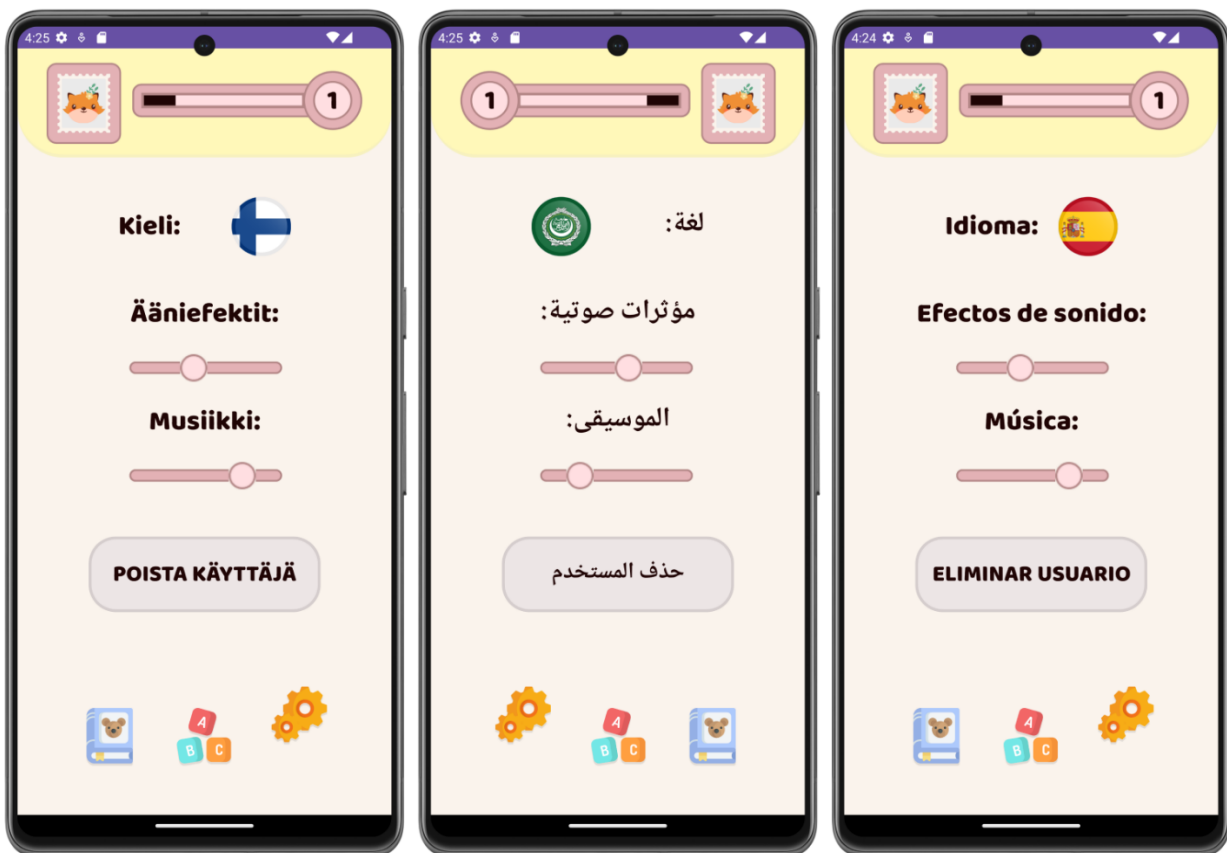
Kuva 9. Sovelluksen relaatiotietokanta.

5.3 Lokalisointi

Sovelluksen käyttöliittymä lokalisoitiin kolmelle eri kielelle vaatimusten mukaisesti: suomen, espanjan ja arabian kielelle. Ohjelman tietokantaa ei lokalisoitu siitä syystä, että Firebase Realtime Database ei itsessään tarjoa sisäänrakennettua lokalisaatiotukea. Alla kuvataan tarkemmin lokalisointiprosessia sekä mainitaan käytetyt lokalisointiresurssit.

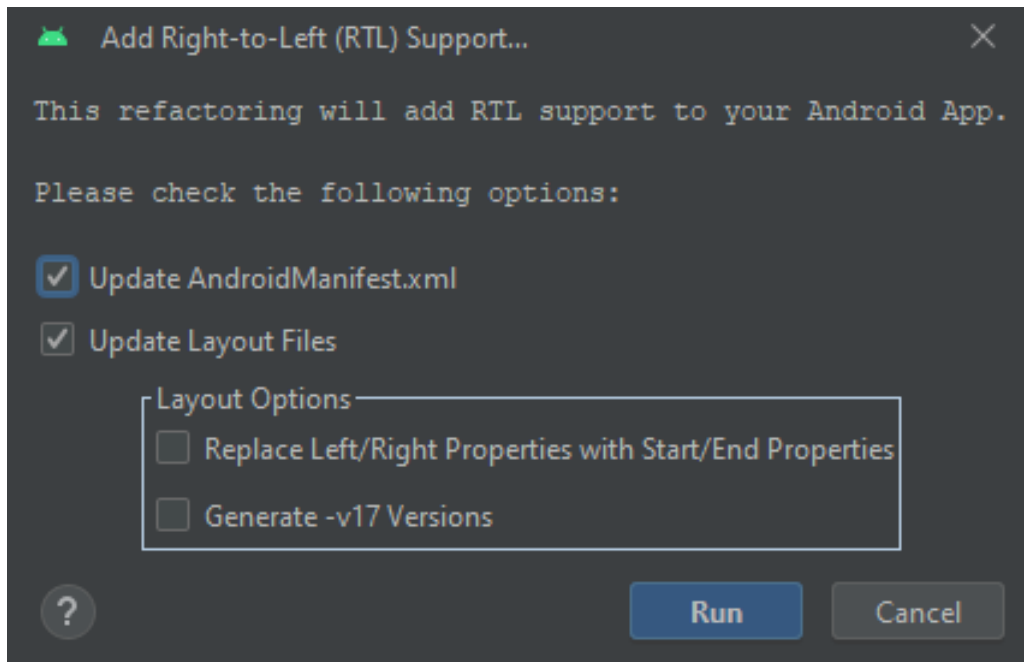
5.3.1 Käyttöliittymän lokalisointi

Käyttöliittymän lokalisointiin lukeutuivat kaikki sovelluksen tekstielementit, kuten tekstinäkymät, alertit ja Android Studio tarjoamat käyttöliittymäelementit. Kuvassa 10 näkyy sovelluksen asetukset lokalisoituna jokaiselle tuetulle kielelle. Huomioitavaa on se, että arabian kielen tapauksessa käyttöliittymä muuttuu luettavaksi oikealta vasemmalle.



Kuva 10. Käyttäjän asetukset lokalisoinnin jälkeen.

Arabian kielen kirjoitussuunta on oikealta vasemmalle, jonka vuoksi lokalisoinnissa täytyi huomioida käyttöliittymäelementtien uudelleensijoittaminen. Android Studio tarjoaa sisäänrakennetun asetuksen käyttöliittymäelementtien uudelleensijoittamiselle valitun kielen perusteella. Tämä asetus tuli laittaa päälle Android Studiossa. (Kuva 11)



Kuva 11. Android Studion asetus, joka lisää RTL-tuen Android-sovellukseen.

5.3.2 Lokalisointiresurssit

Sovelluksen eri kielet jaettiin lokalisaatioprosessissa values-ar-, values-es- ja values-fi-nimisiin kansioihin. Kansioihin lisättiin kaikki kielten sanat ja lauseet. Eri kielten käännöksissä hyödynnettiin Google Kääntäjää sekä ilmaisia sanakirjoja Internetistä. Android Studion sisäinen Translations Editor -työkalu toimi erinomaisena välineenä käännöstyön hallinnassa (Kuva 12).

Arabic (ar)	Finnish (fi)	Spanish (es)
اختر صورة الملف الشخصي:	Valitse profiilikuva:	Selecciona un avatar:
أدخل اسم المستخدم:	Anna käyttäjänimi:	Nombre de usuario:
العمر	Ikä	Edad
أدخل صورك:	Anna ikäsi:	Edad:
اسم المستخدم	Käyttäjänimi	Nombre de usuario
الاسم	Nimi	Nombre
المستوى	Taso	Nivel:
الموسيقى:	Musiikki:	Música:
مؤثرات صوتية:	Ääniefektit:	Efectos de sonido:
صحيح!	Oikein!	Correcto!
خطأ	Väärin	Incorrecto
المقبل	SEURAAVA	SIGUIENTE
مستعد	VALMIS	LISTO
حذف المستخدم	POISTA KÄYTTÄJÄ	ELIMINAR USUARIO
لغة:	Kieli:	Idioma:
املأ جميع المعلومات	Täytä kaikki tiedot	Completa toda la información
املأ كل الحقول	Täytä kaikki kentät	Complete todos los campos
املأ كل الحقول	Täytä kaikki kentät	Complete todos los campos
من فضلك، قم بملء حقول الاسم والعمر للتدقيق	Ole hyvä ja täytä nimi- ja ikäkentät j	Por favor, completa los campos de
املأ كل الحقول	Täytä kaikki kentät	Complete todos los campos

Kuva 12. Android Studio Translations Editor -työkalu.

6 Ohjelmiston testaus

Projektin aikana toteutettiin useita testejä, joissa testattiin sekä käyttöliittymän toimivuutta että sovelluksen logiikkaa ja tietojen eheyttä. Kaikki luodut testit olivat niin sanottuja instrumentoituja testejä. Luodut testit kattavat sovelluksen keskeisimpiä toimintoja. Tietokantahakuihin hyödynnettiin Google Play -palveluiden Tasks-rajapintaa sekä Firebase Databasen ValueEventListener-rajapintaa.

Jokaisen testin alussa luotiin testisessio hyödyntämällä TestSessionManager-luokkaa sekä @Before ja @After-annotaatioita. Ensimmäinen annotaatio loi session ja tallensi sen

tietokantaan, ja toinen puolestaan poisti session sekä sovelluksesta että tietokannasta testin jälkeen. Tämä esti tietokannan täyttymästä turhilla käyttäjätiedoilla.

Testausvaiheessa suurin osa ongelmista liittyi tietokantaan. Firebase-tietokannan toiminta oli varsin hidas, ja tietojen hakemisessa saattoi kestää liian pitkään, mikä puolestaan vaikutti testin läpäisyyn. Tällaisissa tapauksissa testit eivät menneet läpi, sillä ne eivät ehtineet saada oikeita tietoja tietokannasta, vaikka muuten kaikki oli oikein. Useissa testeissä jouduttiin lisäämään manuaalisesti testausaikaa, jotta tietokannalla oli tarpeeksi aikaa hakea tiedot. Firebase-tietokannan hitaus voi johtua esimerkiksi sen asynkronisesta luonteesta, epävakaasta Internet-yhteydestä tai palvelimen ylikuormituksesta.

6.1 Testitapaukset

Alla kuvataan neljän eri testitapauksen prosessi yksitellen. Jokainen testitapaus toteutettiin aikaisemmin mainitulla tavalla.

6.1.1 Session luonti ja käyttöönotto

Tässä testitapauksessa testattiin sovelluksen kykyä luoda uusi sessio sekä sen tietojen tallentamista tietokantaan. Testin teon aikana ei ollut merkittävää määrää haasteita tai bugeja Firebase-tietokannan hitautta lukuun ottamatta. Aluksi yritettiin hyödyntää Mockito-kirjastoa, mutta lopulta päädyttiin käyttämään Espresso-testauskehystä tässä ja kaikissa muissa testeissä.

Kuvassa 13 näkyy testin kulku. Testi alkaa käyttäjän luonnilla ja käyttäjälle asetetaan nimimerkki, ikä sekä profiilikuva. Kun kaikki edellä mainittu on tehty, painetaan "Valmis"-nappia ja käyttäjä siirtyy sovelluksen kotinäkömään. Tietokantaan tallentuu käyttäjän asettamat tiedot, joka varmistetaan testin lopuksi.

Testitapaus #: 1	Testitapauksen nimi: Session luonti ja käyttöönnotto	Sivu: 1 / 1
Järjestelmä: Android 13.0	Alajärjestelmä: Käyttäjäprofiili	
Suunnitellut: Jhon Rastrojo	Suunnittelupäivä: 20.11.2023	
Suorittanut: Jhon Rastrojo	Suorituspäivä: 20.11.2023	
Lyhyt kuvaus: Testataan session luomista		

Esiehdot:

1. Sovellus on asennettu ja käynnistetty.
2. Voimassa oleva internet-yhteys on käytettävissä.
3. Firebase-tunnistus ja tietokanta on konfiguroitu asianmukaisesti.
4. Sovelluksen nykyinen näkymä on Käyttäjän luonti -näkymä

Vaihe	Toimenpide	Odotettu tulos	Läpi	Huomioitavaa
1	Syötä käyttäjän nimeksi "TestUser"	Käyttäjän nimeksi asetetaan TestUser	Kyllä	
2	Syötä käyttäjän iäksi 3	Käyttäjän iäksi asetetaan 3	Kyllä	
3	Klikkaa profiilikuvan kuvaketta	Näytölle avautuu ponnahdusikkuna, jossa näkyvät vaihtoehdot profiilikuvalle	Kyllä	
4	Valitse ensimmäinen kuva vaihtoehdoista	Ponnahdusikkuna sulkeutuu ja käyttäjän profiilikuvaksi asetetaan ensimmäinen kuva	Kyllä	
5	Klikkaa "Valmis"-nappia	Käyttäjä siirretään sovelluksen kotinäkömään	Kyllä	
6	Tarkista jälkiehto 1			

Jälkiehdot:

1. Käyttäjän syöttämät tiedot tallentuvat tietokantaan

Kuva 13. Ensimmäinen testitapaus raportoituna.

6.1.2 Profiilikuvan vaihtaminen

Tämän testitapauksen tavoitteena oli varmistaa, että profiilikuvan vaihtaminen toimi halutulla tavalla eikä sen yhteydessä tullut ongelmia. Ensin testissä testataan käyttöliittymän toimivuutta Espresso-testauskehysellä. Varmistetaan, että jokaisessa vaiheessa käyttöliittymä vastaa käyttäjän toimintoihin odotetulla tavalla, ja että uusi profiilikuva tulee vanhan tilalle profiilinäkymässä ja sovelluksen yläpalkissa. Lopuksi JUnit-testillä tarkistetaan, että uusi kuva on tallentunut tietokantaan. (Kuva 14)

Tässä testitapauksessa haastavinta oli testata kuvien täsmäämistä. Ominaisuutta oli kuitenkin pakko testata, ja lopulta päädyttiin ratkaisuun, jossa tehtiin erillinen luokka, joka tarkisti kuvien täsmäämistä.

Testitapaus #: 2	Testitapauksen nimi: Profiilikuvan vaihtaminen	Sivu: 1 / 1
Järjestelmä: Android 13.0	Alajärjestelmä: Käyttäjäprofiili	
Suunnitellut: Vera Finogenova	Suunnittelupäivä: 21.11.2023	
Suorittanut: Vera Finogenova	Suorituspäivä: 24.11.2023	
Lyhyt kuvaus: Varmistetaan, että käyttäjän profiilikuvan vaihtaminen onnistuu		

Esiehdot:
1. Word Wonders -sovellus on asennettu Android-pohjaiseen älypuhelimelle
2. Puhelimen nettiyhteys on päällä
3. Sovellukseen on luotu sessio
4. Sovelluksen nykyinen näkymä on profiilinäkymä

Vaihe	Toimenpide	Odotettu tulos	Läpi	Huomioitavaa
1	Paina profiilinäkymän kuvakkeesta	Profiilinäkymä aukeaa	Kyllä	
2	Paina profiilikuvasta	Aukeaa ponnahdusikkuna, jossa ovat listattuna profiilikuvavaihtoehdot	Kyllä	
3	Paina listassa viimeisenä olevasta vaihtoehdosta	Ponnahdusikkuna menee kiinni ja äsken valittu kuva tulee vanhan profiilikuvan tilalle sekä profiilinäkymässä että muissa näkymissä	Kyllä	
4	Tarkista jälkiehto 1		Kyllä	

Jälkiehdot:
1. Uusi profiilikuva on tullut vanhan tilalle sekä sovelluksessa että tietokannassa

Kuva 14. Toinen testitapaus raportoituna.

6.1.3 Kielen vaihto

Tässä testitapauksessa testattiin sovelluksen kielen vaihtoa Asetukset-näkymässä (Kuva 15). Kielinä tapauksessa toimivat espanjan kieli ja arabian kieli, mutta samalla tavalla voisi testata myös muiden sovelluksen kielten vaihtamista. Testaus osoittautui yllättävän haastavaksi, sillä sen aikana löytyi useita bugeja liittyen Asetukset-näkymään ja kielen vaihtamiseen, ja niiden selvittämiseen kului paljon aikaa. Lisäksi on huomioitavaa se, että arabian kieleen vaihtamisen jälkeen sovellus lokalisoituu oikein muutamaa käyttöliittymäkomponentteja lukuun ottamatta. Tämä ongelma ei ratkennut useista yrityksistä huolimatta.

Testi alkaa sovelluksen Asetukset-näkymästä ja siellä klikataan Espanjan lipun kuvaa. Klikkaus avaa ponnahdusikkunan, josta klikataan arabian kieltä kuvastavaa lippua. Sovelluksen näkymä päivittyy ja varmistetaan, että tekstit ovat lokalisoitu ja vaihdettu kieli tallentuu tietokantaan.

Testitapaus #: 3	Testitapauksen nimi: Kielen vaihto	Sivu: 1 / 1
Järjestelmä: Android 13.0	Alajärjestelmä: Kieli	
Suunnitellut: Valtteri Kuitula	Suunnittelupäivä: 25.11.2023	
Suorittanut: Valtteri Kuitula	Suorituspäivä: 26.11.2023	
Lyhyt kuvaus: Testataan sovelluksen kielen vaihtoa espanjan kielestä arabian kieleksi		

Esiehdot:
1. Sovellukseen on luotu sessio
2. Luodun session kieli on espanjan kieli
3. Sovelluksen nykyinen näkymä on Asetukset-näkymä

Vaihe	Toimenpide	Odotettu tulos	Läpi	Huomioitavaa
1	Klikkaa Espanjan lipun kuvaelementtiä	Näytölle avautuu ponnahdusikkuna, jossa näkyvät nykyinen kieli sekä muut kiellivaihtoehdot	Kyllä	Espanjan lippu kuvaa sovelluksen nykyistä kieltä
2	Klikkaa Arabian lipun kuvaelementtiä	Ponnahdusikkuna sulkeutuu ja näkymä päivittyy	Kyllä	
3	Tarkista jälkiehto 1		Kyllä	
4	Odota 10 sekuntia	Testi odottaa	Kyllä	Testi odottaa, jotta sovellus ehtii päivittämään uuden kielen Firebase-tietokantaan
5	Tarkista jälkiehto 2		Kyllä	

Jälkiehdot:
1. Sovelluksen tekstit vaihtuvat arabiankielisiksi
2. Vaihdettu kieli tallentuu tietokantaan

Kuva 15. Kolmas testitapaus raportoituna.

6.1.4 Kokemuspisteiden ansaitseminen

Tässä testitapauksessa testattiin käyttäjän kokemuspisteiden kertymistä sovelluksen pelin pelaamisen jälkeen. Sovelluksen peleistä testattavaksi valikoitui väripeli. Huomioitavaa testissä on se, että on hyvin pieni mahdollisuus, että testi ei mene läpi tilanteessa, jossa käyttäjä vastaa väärin pelin jokaiseen vaihtoehtoon. Tällaista tilannetta ei testauksen aikana tapahtunut, mutta se on teoriassa mahdollista. Tällaisen tilanteen välttäminen on mahdollista siten, että käyttäjä ansaitsee pienen määrän kokemuspisteitä myös vääristä vastauksista.

Kuva 16 kuvaa testin kulkua. Testi alkaa sovelluksen kotinäköymästä ja siellä painetaan väripelin kuvaketta. Pelin pelaaminen toimii siten, että painetaan viisi kertaa vastausvaihtoehtoa, odotetaan hetki ja toistetaan sama uudestaan. Testiä varten luodun session taso on 1, jonka vuoksi peli kestää yhteensä kymmenen kierrosta. Pelin päätyttyä varmistetaan, että käyttäjä ansaitsi kokemuspisteitä tarkistamalla se tietokannasta.

Testitapaus #: 6	Testitapauksen nimi: Kokemuspisteiden ansaitseminen
Järjestelmä: Android 13.0	Sivu: 1 / 1
Suunnittelut: Valtteri Kuitula	Alajärjestelmä: Kokemuspisteet
Suorittanut: Valtteri Kuitula	Suunnittelupäivä: 25.11.2023
Lyhyt kuvaus: Testataan käyttäjän kokemuspisteiden kertymistä väripelin jälkeen	Suorituspäivä: 26.11.2023

Esiehdot:
1. Sovellukseen on luotu sessio
2. Luodulla sessiolla ei ole kokemuspisteitä
3. Sovelluksen nykyinen näkymä on kotinäkymä

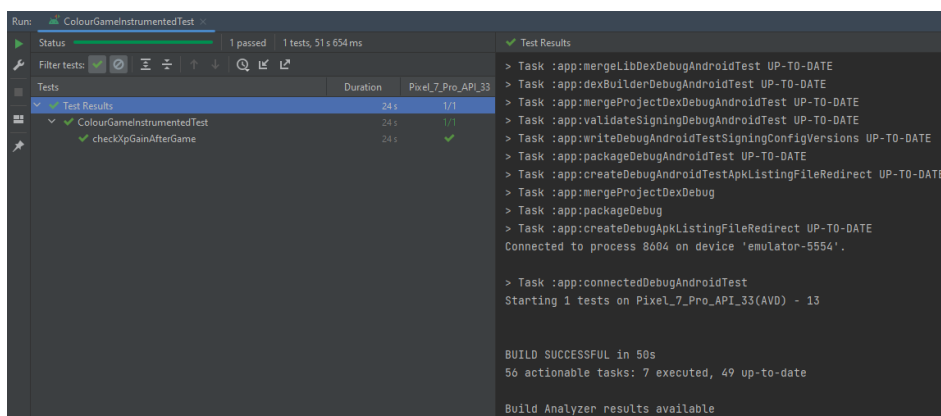
Vaihe	Toimenpide	Odotettu tulos	Läpi	Huomioitavaa
1	Klikkaa väripelin kuvaketta	Käyttäjä siirtyy väripelin ensimmäiseen aktiviteettiin	Kyllä	
2	Klikkaa vastausvaihtoehtoa	Pelin vastausvaihtoehdot muuttuvat	Kyllä	Käyttäjä ansaitsee kokemuspisteitä, jos vastaus meni oikein
3	Toista vaihe 2 neljä kertaa	Pelin vastausvaihtoehdot muuttuvat joka vastaukserralla	Kyllä	Pelikierroksia on yhteensä viisi per aktiviteetti
4	Odota 5 sekuntia	Käyttäjä siirtyy väripelin seuraavaan aktiviteettiin	Kyllä	
5	Toista vaihe 2 viisi kertaa	Pelin vastausvaihtoehdot muuttuvat joka vastaukserralla	Kyllä	
6	Odota 10 sekuntia	Peli päättyy ja käyttäjä siirtyy takaisin sovelluksen kotinäkymään	Kyllä	
7	Tarkista jälkiehto 1		Kyllä	

Jälkiehdot:
1. Käyttäjän ansaitsemat kokemuspisteet tallentuvat tietokantaan

Kuva 16. Neljäs testitapaus raportoituna.

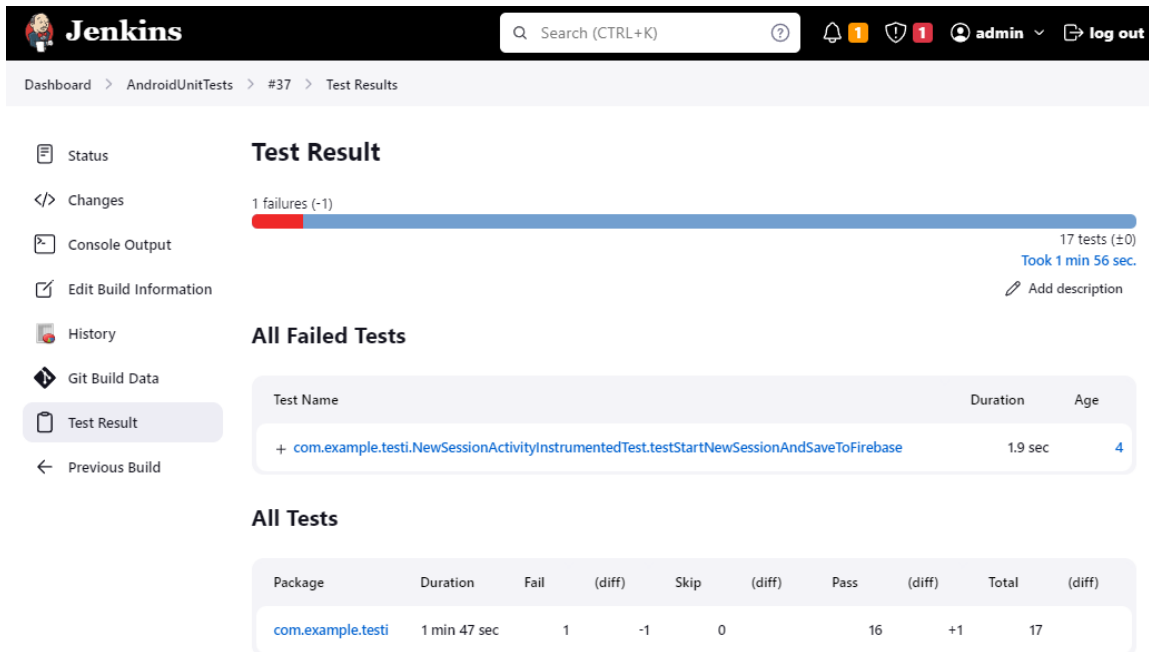
6.2 Testien ajaminen

Kaikki luodut testit ajettiin aluksi suoraan emulaattorissa Android Studiassa. Kuvassa 17 näkyy neljännen testitapauksen ajon tulos Android Studiassa.



Kuva 17. Neljäs testitapaus ajettuna Android Studiassa.

Projektityön edetessä hyödynnettiin myös Jenkins-työkalua testien ajamiseen. Kaikki testit menivät läpi Jenkinsissä yhtä lukuun ottamatta (Kuva 18). Kuitenkin kyseinen testi meni läpi Android Studiossa ongelmitta.



Jenkins Search (CTRL+K) admin log out

Dashboard > AndroidUnitTests > #37 > Test Results

Test Result

1 failures (-1)

17 tests (±0)
Took 1 min 56 sec.
Add description

All Failed Tests

Test Name	Duration	Age
+ com.example.testi.NewSessionActivityInstrumentedTest.testStartNewSessionAndSaveToFirebase	1.9 sec	4

All Tests

Package	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
com.example.testi	1 min 47 sec	1	-1	0		16	+1	17	

Kuva 18. Kaikki sovelluksen testit ajettuna Jenkins-työkalulla.

7 Yhteenveto

Projektityö oli kokonaisuudessaan hyvin opettavainen kokemus kaikille projektiryhmän jäsenille. Uusina asioina projektikurssien aikana tulivat Figma ja Trello käyttö, Jenkinsin konfigurointi ja käyttö projektissa sekä UML-kaavioiden luominen. Lisäksi tutuksi tuli Scrum-kehitysmenetelmä, joka auttoi projektinhallinnassa ja aikatauluttamisessa. Projektityön syvensi ymmärrystämme ohjelmointiprojektien kokonaisprosessista ja huomasimme, miten tärkeä osa huolellinen suunnittelu on projektin kehitystä. Kiinnitimme myös testaukseen erityistä huomiota ja opimme siitä paljon uutta; testauksen aikana löytyi monia bugeja, joita emme välttämättä olisi huomanneet ilman testien luomista.

Saimme kaikki sovelluksen vaatimukset toteutettua, jotka määritettiin projektityön suunnitteluvaiheessa. Sovellus toimii niin kuin suunnittelimme ja testauksen myötä saimme parannettua sitä sekä korjaamaan useita löytyneitä bugeja. Jatkokehityksen näkökulmasta aloittaisimme lisäämällä tuen monelle muulle kielelle (esimerkiksi ruotsin tai norjan kielelle) sovelluksen käyttäjäkunnan laajentamiseksi. Lisäksi sovellus hyötyisi monista pienistä ominaisuuksista kuten esimerkiksi päivittäisen kokemuspistebonuksen saamisesta, tulosruudusta jokaisen pelin jälkeen ja käyttäjän palkitsemissysteemistä.

8 Liitteet

Trello: <https://trello.com/b/WvcGSa7y/sprint-1>

GitHub: <https://github.com/Jxkume/LanguageApp>

JavaDoc-esimerkki: <https://users.metropolia.fi/~valttku/rightDocumentation/com/example/testi/package-summary.html>