



Ryhmä 5:

Oliver Hamberg, Elmo Vahvaselkä ja Ai Van Vo

Shakkisovellus - Tekninen dokumentaatio

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintäteknologian tutkinto-ohjelma

1.5.2022

Sisällys

| | | |
|--------|-----------------------------------|----|
| 1. | Johdanto | 3 |
| 2. | Tuotteen vaatimukset | 4 |
| 3. | Käyttäjäroolit ja käyttötapaukset | 5 |
| 3.1. | Pelaajatunnuksen hallinnointi | 5 |
| 3.2. | Pikapeli | 6 |
| 3.3. | Tilastoitu peli | 6 |
| 3.4. | Pelaajakohtaiset tilastot | 6 |
| 3.5. | Leaderboard | 6 |
| 3.6. | Kielen vaihtaminen | 6 |
| 4. | Kehitysympäristö | 7 |
| 5. | Ohjelmiston tietomalli | 8 |
| 6. | Ohjelmiston rakenne | 9 |
| 6.1. | Pakkaustason rakenne. | 9 |
| 6.2. | Luokkatason rakenteet | 10 |
| 6.2.1. | Nappulat | 10 |
| 6.2.2. | Nappulan siirtäminen | 11 |
| 6.2.3. | Tallentaminen | 12 |
| 6.3. | Käyttöliittymän rakenne | 13 |
| 7. | Testaus | 14 |
| 7.1. | Model | 15 |
| 7.2. | Dao | 17 |
| 7.3. | View | 17 |
| 8. | Yhteenveto | 18 |

1. Johdanto

Tässä dokumentissa esitellään OTP1 ja OTP2 -kursseilla kehitettyä shakkisovellusta. Sovellus on toteutettu Javalla.

Tämä dokumentaatio on suunnattu henkilöille, joilla ei ole aiempaa tietoa kyseisestä projektista ja sen on tarkoitus opastaa lukijaa ymmärtämään miten sovellus on toteutettu ja kuinka sovellus toimii.

Erillisenä dokumenttina on kirjoitettu sovelluksen käyttöohje, jossa kerrotaan, miten ohjelman saa toimimaan ja kuinka sitä käytetään. Tämä dokumentti pyrkii välttämään päällekkäisyyksiä käyttöohjeen kanssa, joten käyttöohjeen lukemista suositellaan yleiskuvan saamiseksi sovelluksesta.

Sovelluksen tarkoitus on tarjota kahdelle henkilölle mahdollisuus pelata shakkia samalta päätteeltä. Käyttäjät tekevät siirrot samalla hiirellä vuorotellen.

Luvussa kaksi esitellään tuotteen vaatimukset. Kolmannessa luvussa havainnollistetaan käyttäjärooleja sekä käyttötapauksia.

Luvussa neljä esitellään SQL-tietokannan rakennetta sanallisesti sekä kaavion avulla. Tämän jälkeen luvussa viisi esitellään ohjelmiston toteutusta pakkaus- ja luokkakaavioiden avulla.

Seuraavaksi luvussa kuusi esitellään ohjelmiston toimintaa käyttötapauksien ja kuvakaappauksien avulla. Luvussa seitsemän esitellään sovelluksen kehitystä ja siihen liittyviä tekniikoita kuten JUnit-testejä ja Jenkins-palvelinta. Lopuksi kaikki edellä mainittu tiivistetään vielä yhteenvedossa.

2. Tuotteen vaatimukset

Lopullisen tuotteen on tarkoitus tarjota kahdelle henkilölle mahdollisuus pelata shakkia paikallisesti. Pelissä ei ole verkko-ominaisuutta, joten pelaajat pelaavat samalta päätteeltä käyttäen samaa hiirtä.

Pelejä on mahdollista pelata tilastoimattomasti eli ns. pikapelinä tai kilpailullisesti tilastoituna pelinä. Tilastoidun pelin tiedot tallennetaan pelin päätteeksi tietokantaan.

Tilastoitu peli edellyttää pelaajatunnusta, joka on mahdollista luoda nopeasti tilastoidun pelin alussa. Koska peliä pelataan vain paikallisesti yhdeltä laitteelta, ei tunnuksia suojata salasanoilla. Oletuksena on, että käyttäjät tuntevat toisensa ja heidän välillään on riittävästi luottamusta. Salasanattomuudella on tarkoitus tehdä käyttökokemuksesta sulavampi.

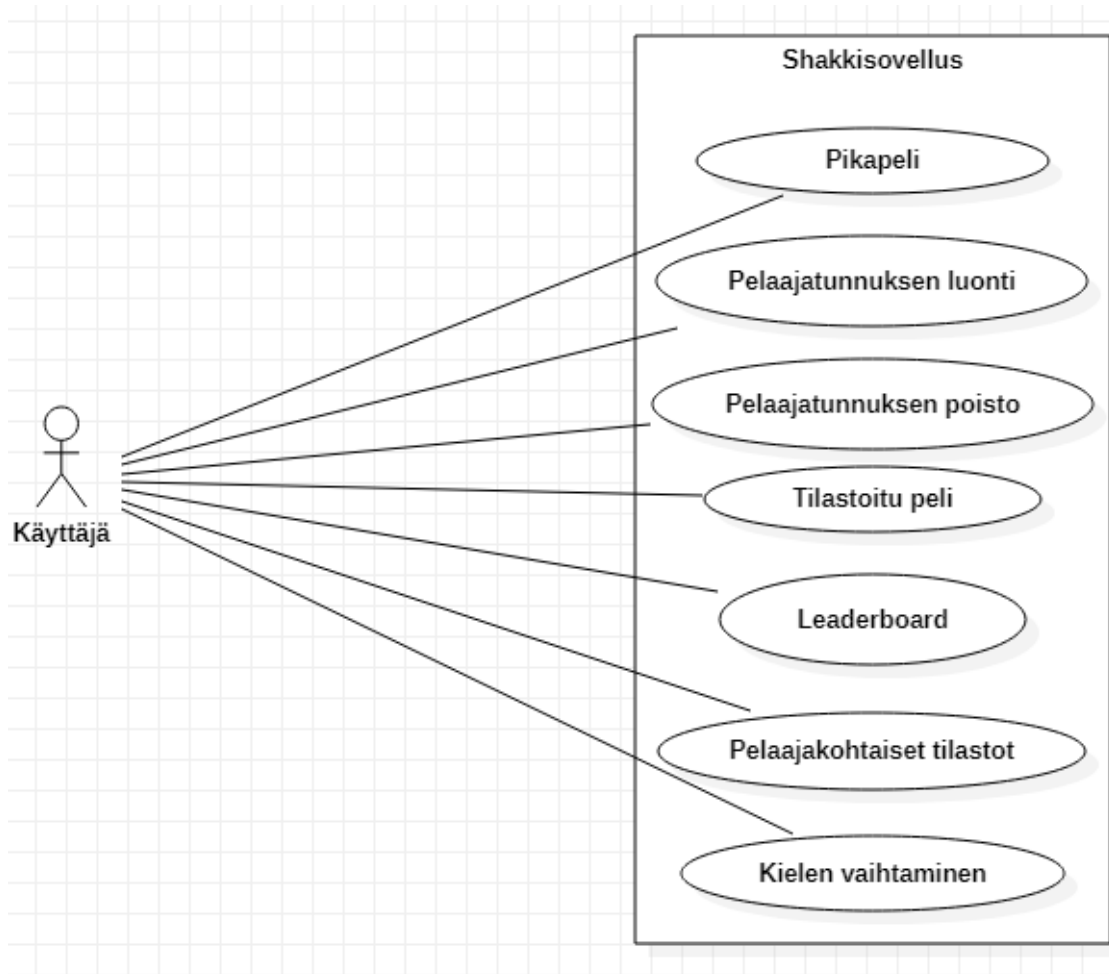
Käyttäjällä on mahdollisuus tarkastella leaderboardia, josta näkee parhaiten menestyneet pelaajat joko voittoprosentin tai voitettujen pelien määrän mukaan.

Lisäksi on mahdollista tarkastella yksittäisen pelaajan tilastoja, josta näkee kyseisen pelaajan voittojen ja tappioiden määrän sekä voittoprosentin.

Yksittäisen pelaajan tiedoissa näkyy myös tämän pelihistoria ja jokaisesta pelatusta pelistä näkee vastustajan, pelatun värin, voittiko pelaaja pelin, siirtojen määrän, peliin kuluneen ajan ja päivämäärän, jolloin peli on pelattu.

3. Käyttäjäroolit ja käyttötapaukset

Käyttäjän kannalta keskeisimmät käyttötapaukset on esitelty kuvassa 1.



Kuva 1. käyttötapaukset.

3.1. Pelaajatunnuksen hallinnointi

Kun pelaaja pelaa ensimmäistä kertaa tilastoitua peliä, hän voi luoda pelaajatunnuksen. Pelaajatunnuksen luominen ei edellytä salasanaa. Halutessaan käyttäjä pystyy poistamaan pelaajatunnuksensa pelaajakohtaisista tilastoista.

3.2. Pikapeli

Pikapelissä kaksi pelaajaa pääse heti aloittamaan pelin heti, kun pikapeli vaihtoehto on valittu päävalikosta. Pelin tietoja ei tallenneta tietokantaan.

3.3. Tilastoitu peli

Tilastoitu peli aloittaminen edellyttää, että molemmat pelaajat omaavat pelaajatunnukset. Tilastoidun pelin tiedot tallennetaan tietokantaan.

3.4. Pelaajakohtaiset tilastot

Tarkasteltava pelaaja valitaan näkymässä olevasta pudotusvalikosta.

Tilastoista näkee tarkasteltavan pelaajan pelaamien pelien määrän, voittojen määrän ja voittoprosentin. Lisäksi näytetään pelaajan pelihistoria, jos on listattuna kaikki pelaajan pelaamat pelit.

Pelihistoriasta ilmenee, millä värillä pelaajan on pelannut, ketä vastaan hän on pelannut sekä voittiko vai hävisikö pelaaja kyseisen pelin. Lisäksi nähtävissä on myös tehtyjen siirtojen määrä, päivämäärä koska peli pelattiin ja pelin kesto.

Pelaajatunnuksen poistopainike löytyy myös tästä näkymästä.

3.5. Leaderboard

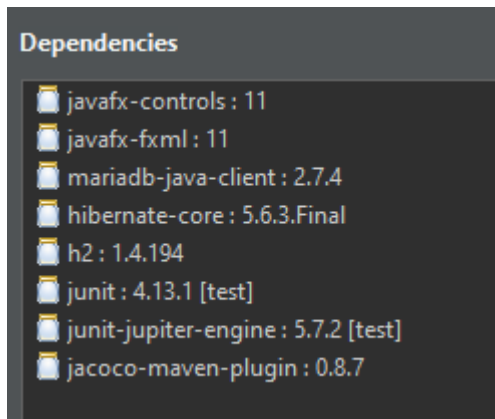
Leaderboardissa pelaajien menestystä voi verrata toisiinsa. Näkymässä näytetään pelaajat, joilla on paras voittoprosentti ja eniten voittoja. Näiden alapuolella on pelaajalista, jossa näkyy pelaajatunnus, voitot, pelattujen pelien määrä ja voittoprosentti.

3.6. Kielen vaihtaminen

Käyttäjä pystyy vaihtamaan kielen päävalikosta. Tällä hetkellä tuettuina kielinä ovat englanti ja suomi.

4. Kehitysympäristö

Sovellusta on työstetty tietokoneella Java kehitysympäristössä ja se on toteutettu Maven-projektina, jonka riippuvuudet on esitelty kuvassa 2. Javasta on käytetty versiota 11. Sovelluksen tiedot tallennetaan paikalliseen MariaDB SQL-tietokantaan.



Kuva 2. Maven-projektin riippuvuudet.

Sovelluksen lähdekoodin versionhallintaan on käytetty GitLabia. Projektin repositorio löytyy osoitteesta: <https://gitlab.metropolia.fi/elmov/otp-shakkisovellus>.

Koonti on toteutettu Educloud pilvipalvelussa olevaan virtuaaliseen Linux-palvelimeen asennetulla Jenkins-ohjelmalla. Koontia varten luotiin myös oma verkkotietokanta Educloudiin.

Koontia varten Maven-projektissa oli myös seuraavat pluginit:

- Maven compiler v. 3.8.1
- Maven surefire v. 3.0.0-M5
- Javafx maven v. 0.0.7
- Jacoco maven v. 0.8.6

5. Ohjelmiston tietomalli

Ohjelmisto hyödyntää paikallista SQL pohjaista MariaDB-relaatiotietokantaohjelmaa. Tietokantaan tallentaminen on toteutettu Hibernate ORM:in avulla. Hibernate luo tietokannan Java-olioiden pohjalta. Tietokannan rakennetta on havainnollistettu kuvassa 2.



Kuva 2. Sovelluksen relaatiotietokantamalli.

Tietokantaan tallennetaan pelaajien tiedot Pelaaja-olioina. Jokaisella pelaaja-oliolla on integer tyyppinen id ja String tyyppiä olevan pelaajaTunnus.

PelinTiedot-olioon tallennetaan tiedot yksittäisestä pelistä. Se sisältää pelin id:en integerinä, pelin keston sekunteina long integerinä, päivämäärän Date-oliona ja siirrot talletettuna listaan Siirto-olioina. Mustalla ja valkoisella pelanneet pelaajat sekä voittaja talletetaan viiteavaimina Pelaaja-olioihin.

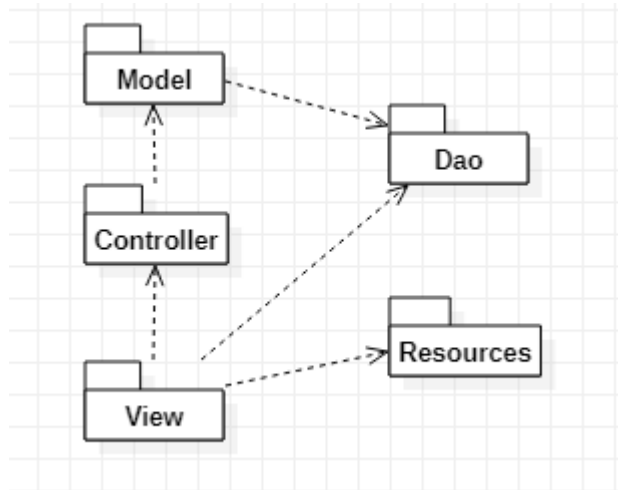
Mustalla ja valkoisella pelaajalla sekä voittajalla voi olla vain yksi Pelaaja-olio. Yksi Pelaaja-olio voi taas pelata monta peliä valkoisena tai mustana sekä olla useamman pelin voittaja.

Siirto-olio sisältää tiedot pelissä tehdystä yksittäisestä siirrosta. Siirrolla on integer muotoa oleva id, tiedot siirron lähtöruudun x- ja y-koordinaateista sekä kohderuudun x- ja y-koordinaateista. Lisäksi mahdollinen korotus talletetaan NappulanTyyppi enumina.

6. Ohjelmiston rakenne

6.1. Pakkaustason rakenne.

Sovelluksen toteutuksen pohjana on toiminut MVC-malli (Model View Controller). Kuvassa 3 havainnollistetaan sovelluksen loogista rakennetta pakkaustasolla.



Kuva 3. sovelluksen pakkausrakenne.

Sovelluksessa on viisi pakkausta. Model sisältää shakkipelin toiminnallisuuden ja sen luokat pitävät kirjaa pelitilanteesta. Se varmistaa, että pelin sääntöjä noudatetaan ja pitää huolta siitä, että pelinappulat ovat siellä missä pitää. Model on yhteydessä Dao-pakkaukseen kun se pelin päätteeksi haluaa tallentaa pelin lopputuloksen tietokantaan.

Controller-pakkaus toimii välikätenä modelin ja viewin välillä. Se välittää käyttäjän käyttöliittymässä tapahtuvat toimet modeliin ja modelissa tapahtuvat muutokset viewiin.

View pakkaus edustaa käyttäjälle näkyvää osaa eli käyttöliittymää ja on vastuussa syötteiden keräämisestä käyttäjältä. View on yhteydessä dao-pakkaukseen, kun se haluaa hakea tietoa pelaajista tai peleistä sekä kun se haluaa luoda uuden pelaajatunnuksen.

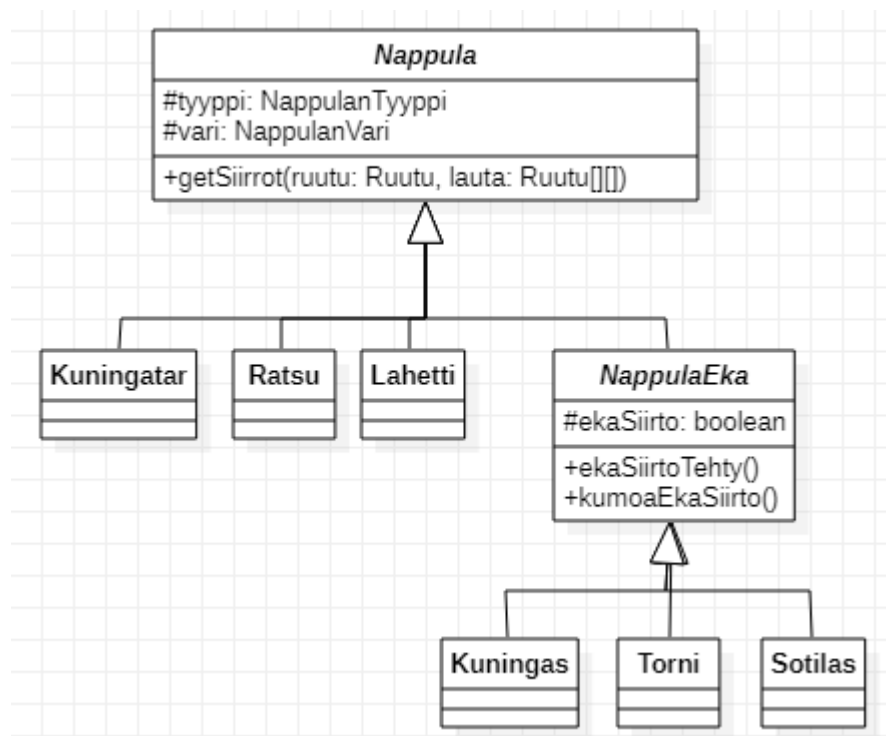
View on myös yhteydessä resources pakkaukseen, jossa säilötään erilaisia ohjelman käyttämiä resursseja kuten teksti-, ääni- ja kuvatiedostoja.

Dao-pakkaus toimii ohjelman ja tietokannan välikätenä. Kaikki tietojen tallentamiseen liittyvä toiminnallisuus on tässä pakkauksessa.

6.2. Luokkatason rakenteet

6.2.1. Nappulat

Kuvassa 4 esitellään miten nappulat on toteutettu. Ohjelmassa.



Kuva 4. Nappuloiden toteutus luokkatasolla.

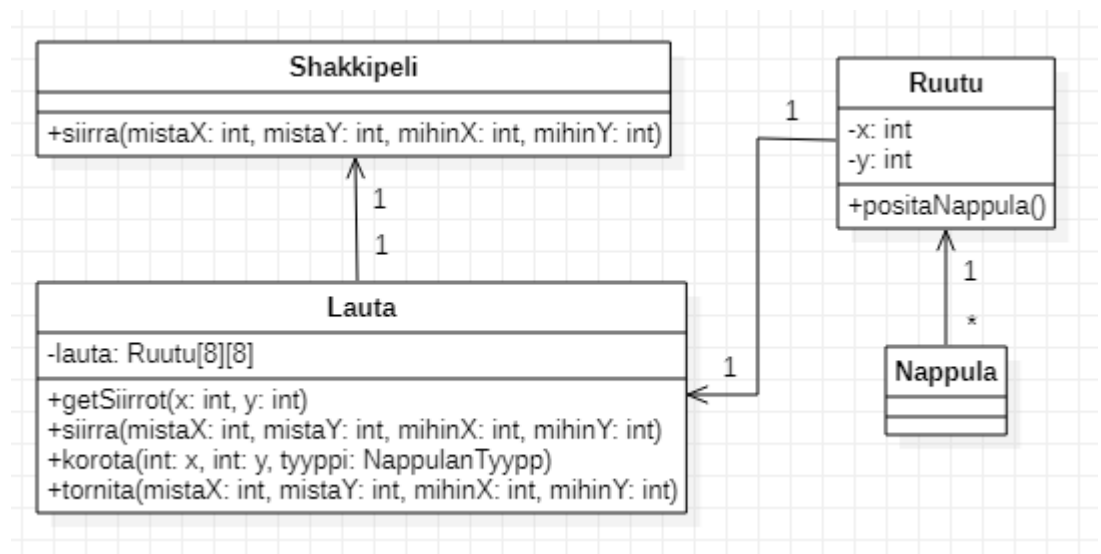
Abstraktiluokka Nappula määrittää, että jokaisella nappulalla on oltava tyyppi ja väri. Lisäksi jokaisen nappulan on palautettava lista siirroista, jotka kyseisellä nappulalla voi tehdä sen mukaan, missä ruudussa nappula sijaitsee ja mikä on pelitilanne.

Kuningatar, Ratsu ja Lahetti luokat perivät suoraan Nappulan.

Abstraktiluokka NappulaEka perii Nappulan ja määrittää lisätoiminnallisuuksia. Joidenkin nappuloiden tulee olla tietoisia onko niiden ensimmäinen siirto tehty, joten se kirjataan NappulaEkaan. Lisäksi ensimmäinen siirto pitää pystyä kuittaamaan tehdyksi ja tekemättömäksi.

6.2.2. Nappulan siirtäminen

Kuvaan 5 on kerätty model tason luokista oleelliset tiedot, jotka liittyvät nappulan siirtämiseen.



Kuva 5. Siirtoon liittyvä toiminnallisuus modelissa.

Kun pelaaja viewissä valitsee nappulan, jota hän haluaa liikuttaa, pyytää view kontrollerin kautta välittämään kaikki kyseisen nappulan siirrot. Kun pelaaja yrittää tehdä siirron, välitetään tieto siirrosta Shakkipeli-luokalle.

Shakkipelin siirrä-metodi tarkistaa, että siirto on varmasti sallittu, tarvitseeko mahdollinen korotus tehdä ja onko siirto tornitus. Siirron sallittavuuteen vaikuttaa mm. onko pelaajan kuningas shakattu,

Mikäli korotus on tehtävä, pyydetään pelaajaa määrittämään mihin nappulaan sotilas korotetaan. Lisäksi siirrä-metodi huomioi, että onko kyseessä tornitus. Mikäli siirto on sallittu se välitetään Lautaluokalle.

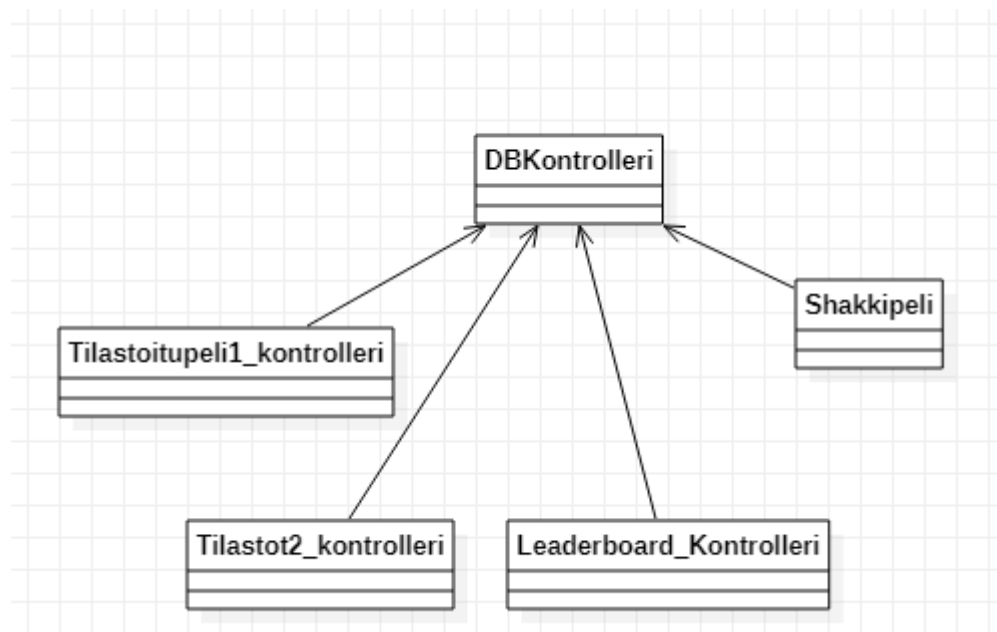
Lauta-luokka sisältää Ruutu-olioista koostuvan kaksiulotteisen taulukon, jonka koko on 8x8. Jokaisella Ruudulla on tieto, sijainnistaan laudalla ja mahdollisesta ruudulla sijaitsevasta nappulasta.

Siirtokäskyn saatuaan Lauta-poistaa nappulan sen lähtöruudusta ja siirtää sen kohderuutuun. Mikäli kohderuudussa on nappula, korvataan ruudussa oleva nappula siirretyllä eli syödään. Lauta-luokalla on oma erityinen tornita metodi, jota Shakkipeli-luokka kutsuu, mikäli siirto on tornitus.

Lauta-luokka suorittaa myös Shakkipeli-luokalta saamansa korotuksen, korvaamassa ruudussa olevan sotilaan luomalla siihen uuden samaa väriä olevan kuningattaren, ratsun, lähetin tai tornin.

Lopuksi Shakkipeli-luokan siirrä-metodi vielä tarkistaa, että aiheuttiko pelaajan siirto shakin tai matin.

6.2.3. Tallentaminen



Kuva 6. Tietokannan toiminnallisuus koodissa

Pelaajan valitessa tietokantaan liittyvä elementti, tämän elementin luokka kutsuu tietokanta kontrollerin tietokanta funktioita.

Kun pelaaja luo tunnustaan tilastoidun pelin näkymässä, hän kirjoittaa nimensä tekstilaatikkoon, jonka sisältö sen jälkeen lähetetään DBKontrollerille. DBKontrolleri sitten kutsuu funktioita luodakseen uuden Pelaaja-olion käyttäjätunnuksen perusteella. Pelaaja-olio tallennetaan sitten tietokantaan. Kun pelaaja haluaa käyttää aikaisemmin luotua käyttäjätunnusta, tietokanta kontrolleri hakee kaikki luodut Pelaaja-oliot listasta.

Tilastoja katsellessa tietokannasta haetaan kaikki käyttäjät, jonka jälkeen Pelaaja-olijoilta haetaan kaikki tarpeellinen data, kuten voittoprosentti, voitot, jne. Tämä data asetellaan sitten tilasto kontrollerin puolella oikeaan muotoon ja laitetaan esille.

Leaderboard näkymässä haetaan kaikki käyttäjät jälleen kerran tietokannasta, jonka jälkeen luomme niistä PelaajaMuutos-olioita, jotta käsittely olisi mahdollisimman helppoa JavaFX elementeillä.

Jos kyseessä on tilastoitu peli, shakkipeli-luokka lähettää PelinTiedot-olion tietokanta kontrollerille, joka taas tallentaa kaikki pelin tiedot tietokantaan. Pelin tiedoista tallentuu esimerkiksi päivämäärä, kesto, osallistujat ja voittaja. Pelaajan ID toimii usein avaimena näissä tietokannoissa.

6.3. Käyttöliittymän rakenne

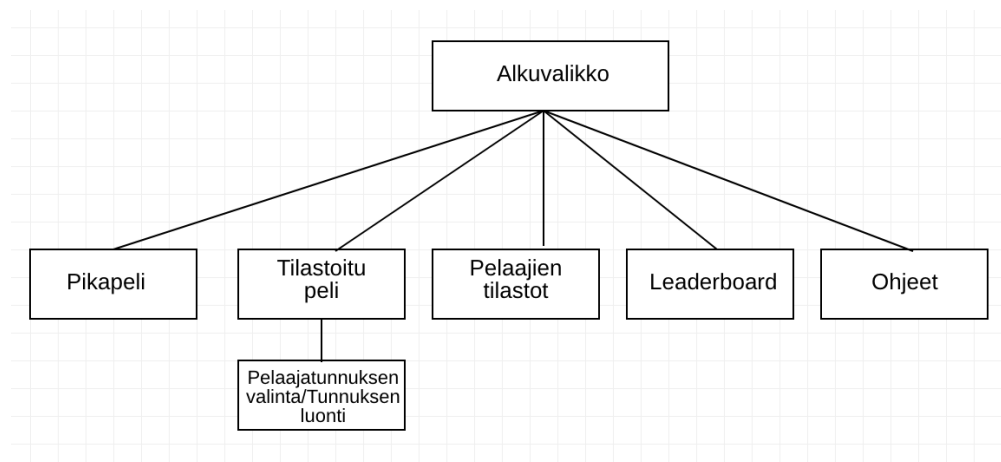
JavaFX käyttöliittymä on luotu Scenebuilder -asettelutyökalua käyttäen. Scenebuilderilla luotuihin fxml tiedostoihin lisätään jokaisella tiedostolle oma Controller niminen luokka, jossa määritellään käyttöliittymässä esiintyvien elementtien toiminnot.

Sovelluksen avautuessa, näkyy ensimmäisenä alkuvalikko. Näkymän oikeassa yläkulmasta voi vaihtaa sovelluksen kieltä suomeen tai englantiin. Sovelluksen oletuskieli on englanti. Alkuvalikosta pääsee vastaavaa nappia painamalla sen

kuvaamaan näkymään. Vaihtoehtoina ovat pikapeli, tilastoitu peli, josta aukeaa pelaajatunnuksen valintanäkymä sekä tunnuksen luontinäkymä, pelaajien tilastot, leaderboard, peliohje näkymä sekä sovelluksen sulkemisnappula.

Käyttäjän napsauttaessa käyttöliittymän elementtiä, esim. pikapeli nappulaa, avautuu kontrollerin metodissa ilmoitettu näkymä.

Kuvassa 7 näkyy käyttöliittymän rakenne alkuvalikosta katsottuna.










Kuva 7. Käyttöliittymän rakenne.

7. Testaus

Tavoitteena oli kehittää ohjelmisto TDD-ohjelmointitavalla (Test-Driven Development). Mutta ajan rajallisuuden takia tämä ei aina toteutunut. Lisäksi aina ei oltu varma miten joitakin uusia tekniikoita, kuten kansainvälistämiseen ja lokalisointiin liittyvät asiat, tulisi testa ennen kuin niiden toteutus oli valmis.

Ohjelmiston testaus toteutettiin JUnit-testeillä. Kuvassa 6 esitellään ohjelmiston testikattavuutta.

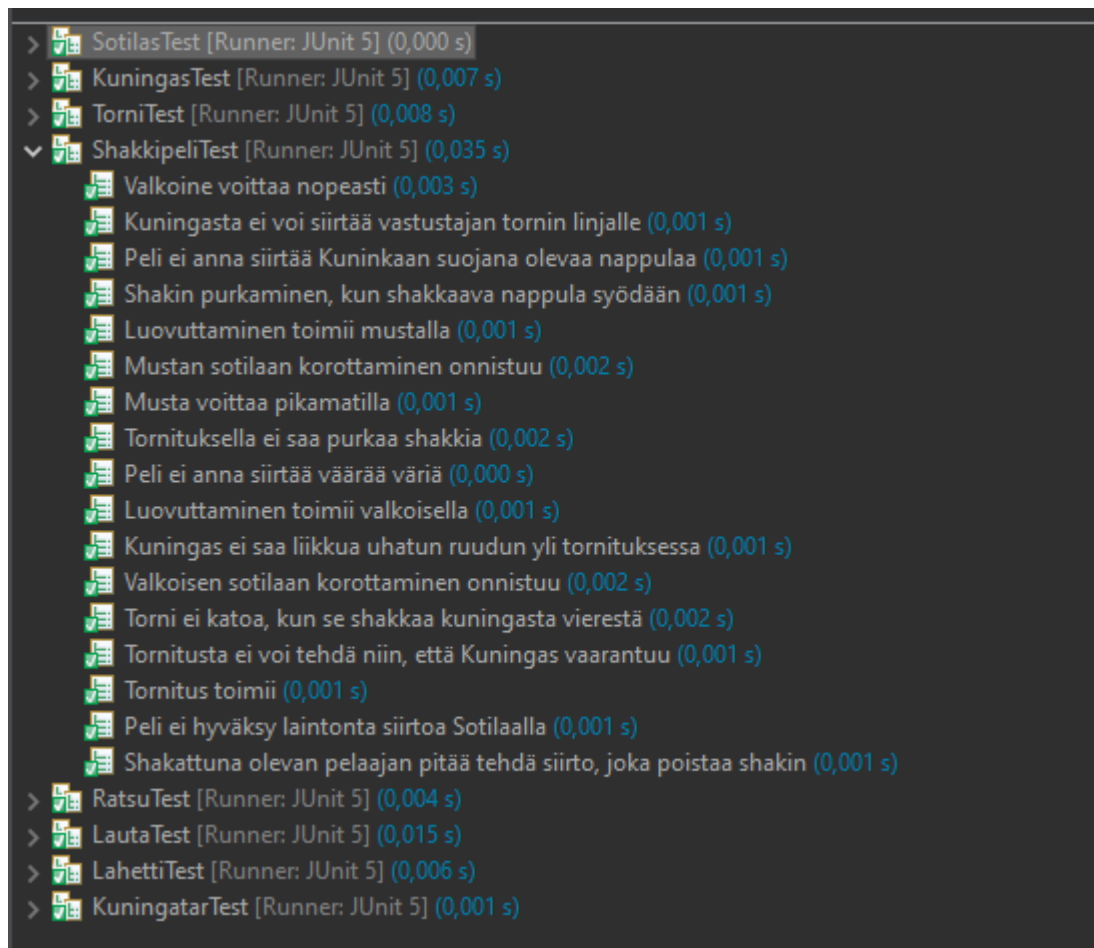
| Element | Coverage | Covered Instructions | Missed Instructions | Total Instructions |
|------------------------|--|----------------------|---------------------|--------------------|
| ▼ Shakkisovellus |  77,1 % | 12 359 | 3 679 | 16 038 |
| ▼ src/main/java |  57,4 % | 4 884 | 3 629 | 8 513 |
| > view.src.application |  5,9 % | 215 | 3 436 | 3 651 |
| > model |  98,3 % | 4 341 | 75 | 4 416 |
| > controller |  0,0 % | 0 | 62 | 62 |
| > dao |  85,4 % | 328 | 56 | 384 |
| > src/test/java |  99,3 % | 7 475 | 50 | 7 525 |

Kuva 6. Testikattavuus.

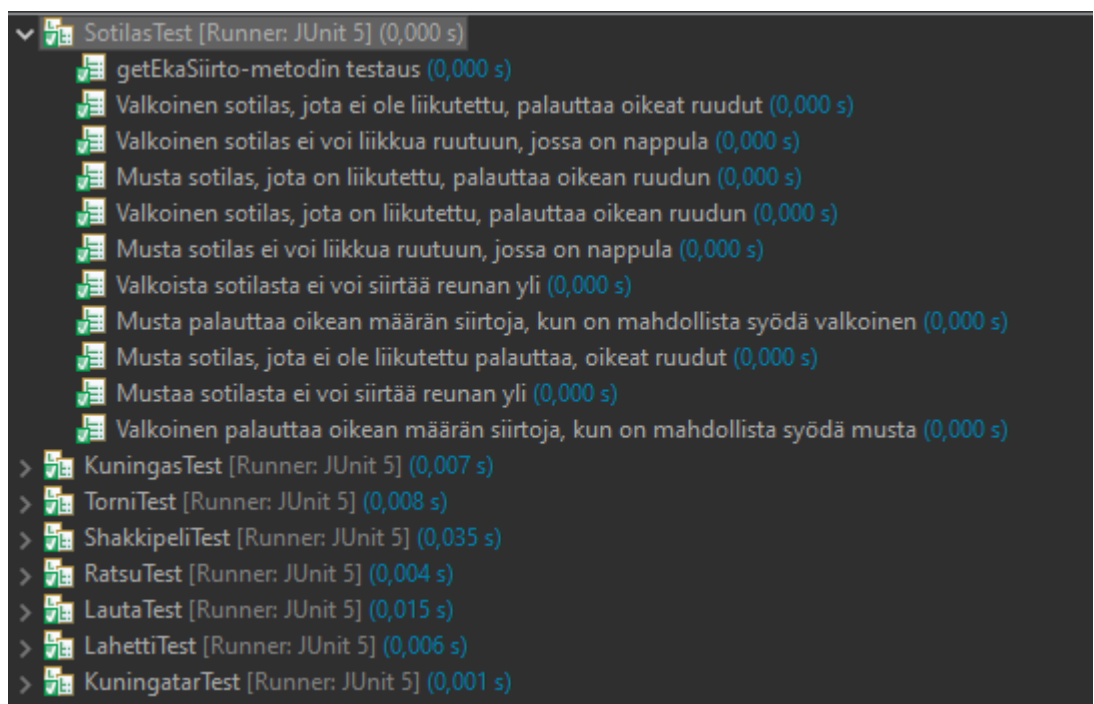
Kuvasta voi havaita, että viewiä ei ole juuri testattu, sillä käyttöliittymän testaaminen ei ollut osa vaatimuksia. Käyttöliittymän joillekin apuluokille on kirjoitettu testejä ja lisäksi aina kun uusi ominaisuus on lisätty käyttöliittymään se on pyritty testaamaan manuaalisesti.

7.1. Model

Modelin testeissä pyrittiin testaamaan mahdollisimman monenlaisia pelitilanteita pelilaudalla sekä kaikkien nappuloiden toiminnallisuus erikseen. Kuvassa 7 esitellään Shakkipeli-luokkaan liittyvät testit ja kuvassa 8 Sotilas-luokan testejä.



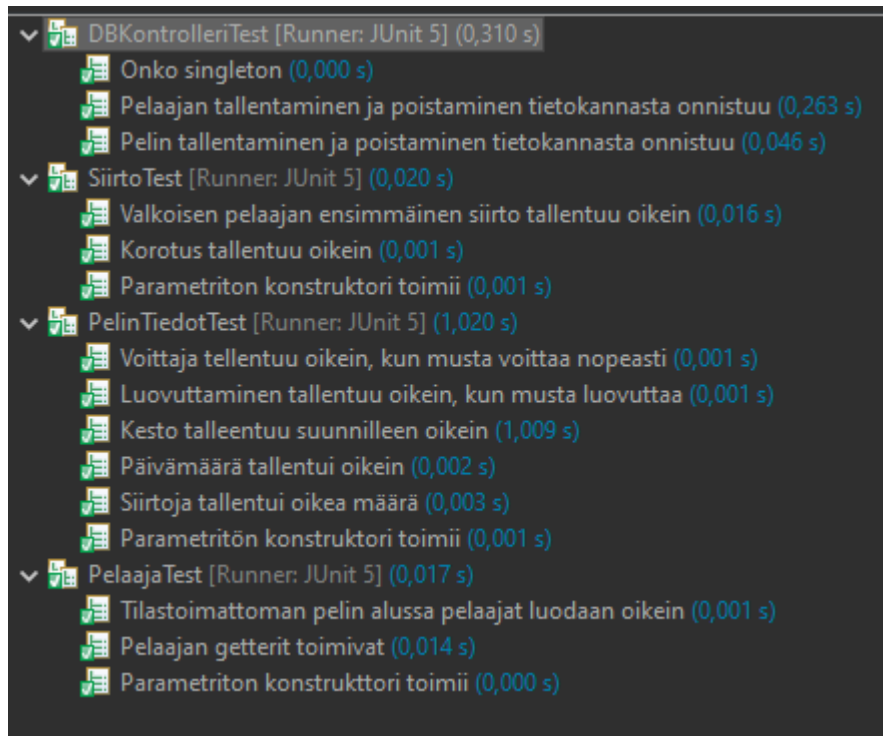
Kuva 7. Shakkipeli-luokan testit.



Kuva 8. Sotilas-luokan testit.

7.2. Dao

Kuvassa 9. esitellään dao-pakkaukseen ja tietokantaan liittyvät testit.

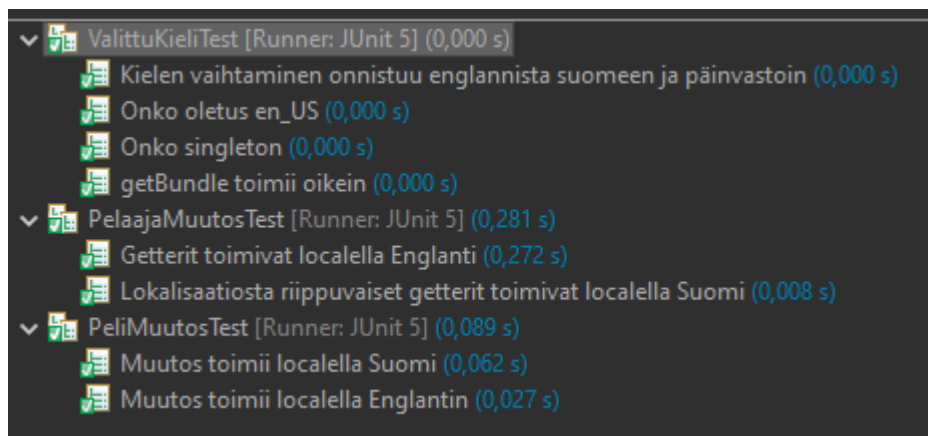


Kuva 9. Dao-pakkauksen testit.

Testeillä on pyritty varmistamaan, että pelin tiedot ja uudet pelaajat tallentuvat oikein tietokantaan ja ne saadaan haettua oikein tietokannasta.

7.3. View

View-pakkaukseen liittyvät testit on esitelty kuvassa 10.



Kuva 10. View-pakkauksen testit.

Lokalisoinnissa on tarkastettu muun muassa sovelluksen oletuskieltä ja textresourcen haun onnistumista. Käyttöliittymässä puolestaan on testattu Oma tilastot -näkyssä ja Leaderboard -näkyssä olevien taulukkotietojen muunnosten oikeellisuutta.

8. Yhteenveto

Projektin alkuperäinen tarkoitus oli saada suomenkielinen paikallisesti omalla tietokoneella pyöritettävä shakkipeli toimimaan.

Ryhmä on tyytyväinen edistymiseen. Shakin pelaamisen kannalta merkittävimmät toiminnallisuudet on onnistuttu toteuttamaan. Halutessaan pelaaja voi luoda pelaajatunnuksen, jolloin tilastoitujen pelien tiedot tallentuvat tietokantaan ja omaa menestystään pystyy tarkastelemaan myöhemmin. Myös lokalisointi on onnistuttu toteuttamaan sovellukselle.

Jatkokehitysideoita sovellukselle:

- Logiikka pattitilanteiden huomioimiselle.
- Tietokannan siirto pilvipalvelimelle paikallisen tietokannan sijasta dataturvallisuuden takaamiseksi.
- Tekoälyn luonti peleille, jotta peliä voisi jatkossa pelata myös yksin.
- Omien pelien tarkempi tarkastelu, niin että jokaista pelissä olevaa siirtoa pääsisi tarkastelemaan.