



Kerkko Kyyrö, Ville Haapamäki, Olli Ruuskanen, Saku Myyryläinen

Kouluhallinnointisovellus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintäteknologian tutkinto-ohjelma

Tekninen dokumentti

6.5.2022

Sisällys

1.....	Johdanto	3
2.....	Käyttöönotto ja konfigurointi	3
3.....	Tuotteen vaatimukset	3
4.....	Käyttäjäroolit ja käyttötapaukset	3
5.....	Tietokanta	5
6.....	Ohjelmiston rakenne	6
8.....	Testit	12
9.....	Kehitysprosessi ja kehitysvaiheen tekniikat	13
10.....	Jatkokehitys	14
11.....	Yhteenveto	14

1 Johdanto

Dokumentin tavoitteena on kuvata koulunhallinnointisovelluksen idea ja sen toimintaan liittyvät tekniset asiat. Dokumentti on suunnattu varsinaisesti uusille kehittäjille ja asiakkaille, jotka ovat kiinnostuneita ohjelmistosta.

Ohjelmisto on suunnattu ala-asteiden käyttöön ja sen käyttäjinä toimii ensisijaisesti oppilaiden huoltajat, sekä heidän opettajansa, mutta myös muu kouluhenkilökunta voi käyttää sovellusta esimerkiksi sen viestintäominaisuuden takia. Ohjelmiston tavoitteena on ensisijaisesti helpottaa opiskelijoiden koulunkäyntiin liittyvien asioiden käsittelyä.

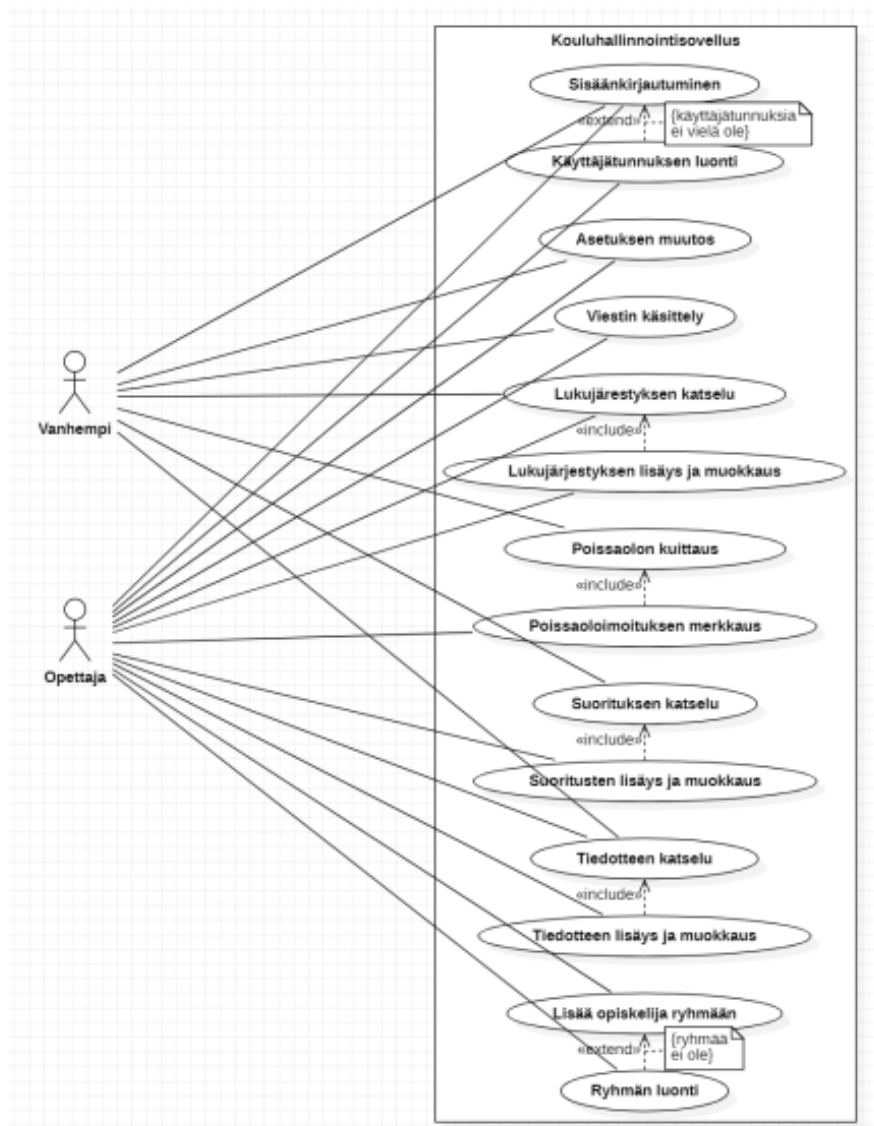
2 Käyttöönotto ja konfigurointi

Sovelluksessa tulee olla aina yksi opettaja käyttäjä, joka on "pääkäyttäjä". Kun sovellus luovutetaan asiakkaalle, sen mukana on jo valmiiksi luotu opettajan kirjautumistunnukset. Tämän jälkeen opettajan on mahdollista lisätä uusia käyttäjiä opettajille ja huoltajille. Opettajat ja huoltajat aloittavat sovelluksen käyttöönoton kirjautumalla sisään saaduillaan tunnuksilla.

3 Tuotteen vaatimukset

Koulunhallinnointisovelluksen tarkoituksena on helpottaa oppilaan koulunkäyntiin liittyvää prosessia. Sovelluksen avulla opettajat ja huoltajat pystyvät käsittelemään oppilaisiin liittyviä asioita, kuten suorituksia, läsnäoloja ja lukujärjestyksiä. Sovelluksesta löytyy myös viestittelyyn ja tiedotteisiin liittyvät toiminnot, jotka mahdollistavat kommunikoinnin opettajien ja huoltajien välillä. Sovelluksen käyttäjien tulee kirjautua sisään omilla tunnuksillaan, joita pystyy luomaan vain opettajat.

4 Käyttäjäroolit ja käyttötapaukset



Kuva 1: Sovelluksen käyttötapauskaavio.

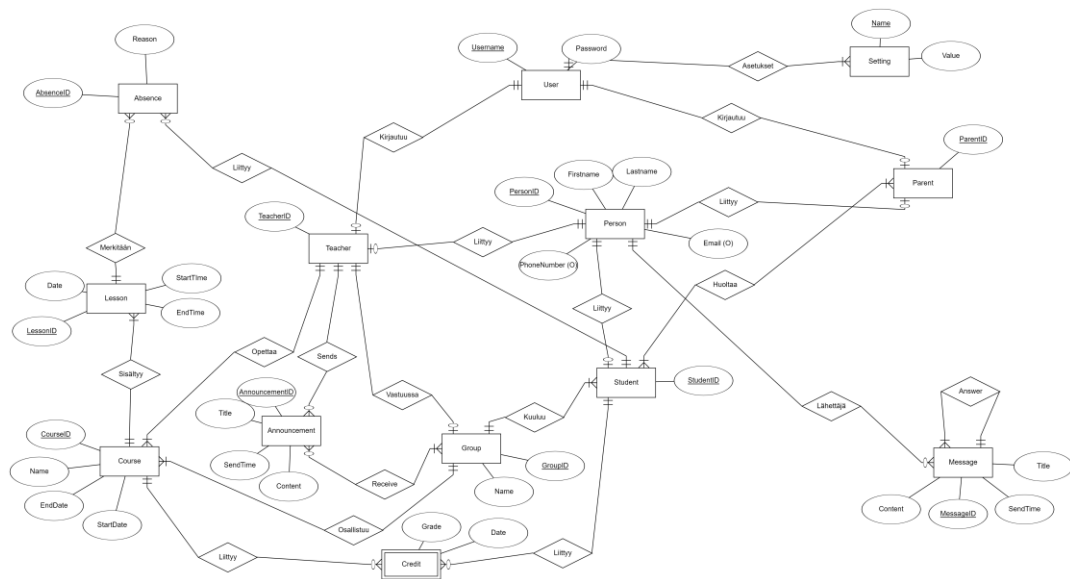
Ohjelmassa on kaksi käyttäjääroolia: opettaja ja huoltaja. Opettaja pystyy luomaan uusia tunnuksia huoltajille ja toisille opettajille. Huoltajat eivät pysty luomaan tunnuksia.

Huoltajat ovat oppilaiden vanhempia. Ne saavat käyttäjätunnuksensa opettajalta, joka luo tunnuksia vanhemmille samalla, kun lisää oppilaan koulun tietoihin. Huoltajat pystyvät seuraamaan lastensa suorituksia, vastaanottamaan tiedotteita, lähettää viestejä opettajille ja merkkamaan oppilaan poissaolon syyn.

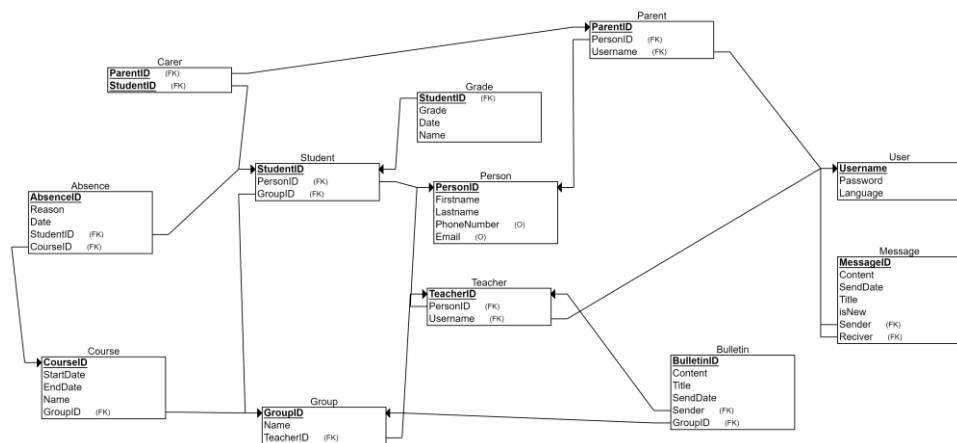
Opettaja pystyy myös luomaan/ muokkaamaan kursseja, lähettämään viestejä huoltajille tai toisille opettajille, luomaan uusia ryhmiä, lähettää tiedotteita omille ryhmilleen, lisätä opiskelijoille suorituksia ja merkitsemään opiskelijoita poissa-oleviksi.

5 Tietokanta

Mallinsimme tietokannan kuvien tyylisellä tavalla:

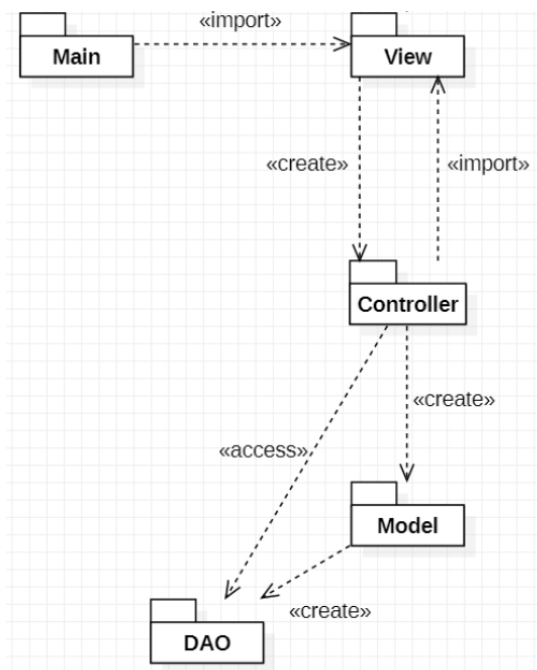


Kuva 2: ER-kaavio ohjelman tietokannasta.



6 Ohjelmiston rakenne

Ohjelmisto on koodattu MVC-arkkitehtuuria noudattaen. Lähdekoodi on jaettu Model, View ja Controller –pakkauksiin kuvassa 5 kuvatulla tavalla. Lisäksi default-paketissa (kuvassa Main) on sovelluksen käynnistävä Main.java tiedosto.

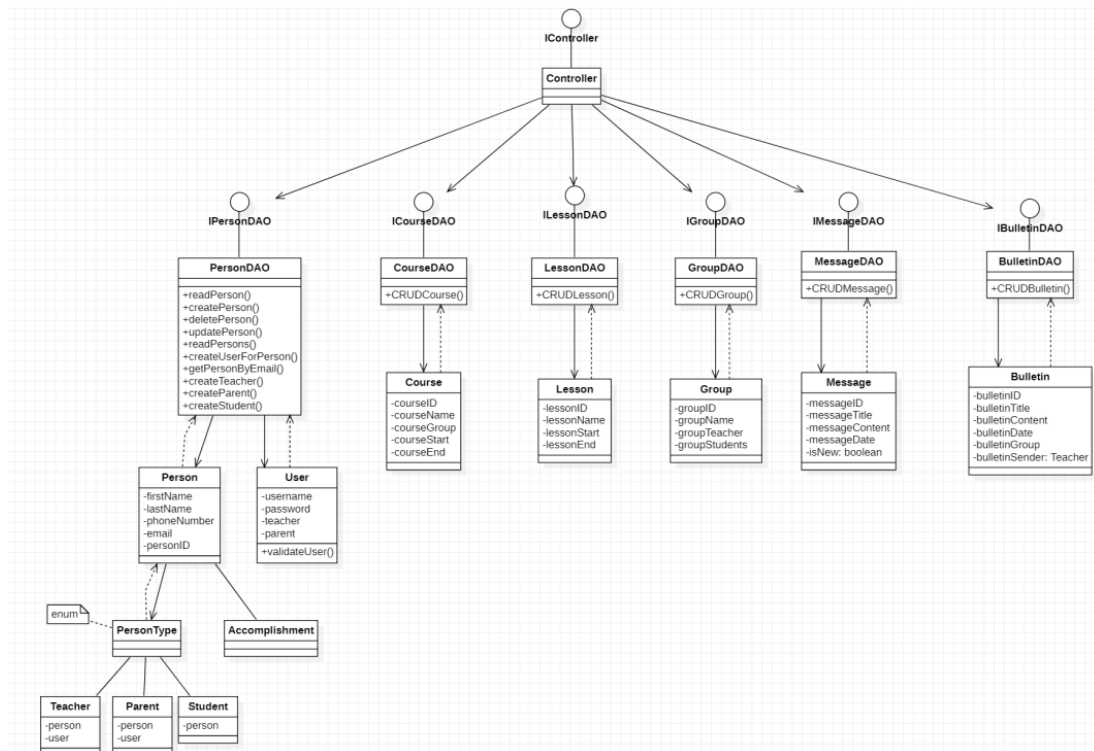


Kuva 5: Sovelluksen pakkauskaavio.

View-tasolla on JavaFX:llä koodatut näkymät ja niihin liittyvät kontrollerit, sekä GUI.java, joka vastaa käyttöliittymän (ja koko sovelluksen) käynnistämisestä. Näkymien kontrollerit kommunikoivat modelin kanssa.

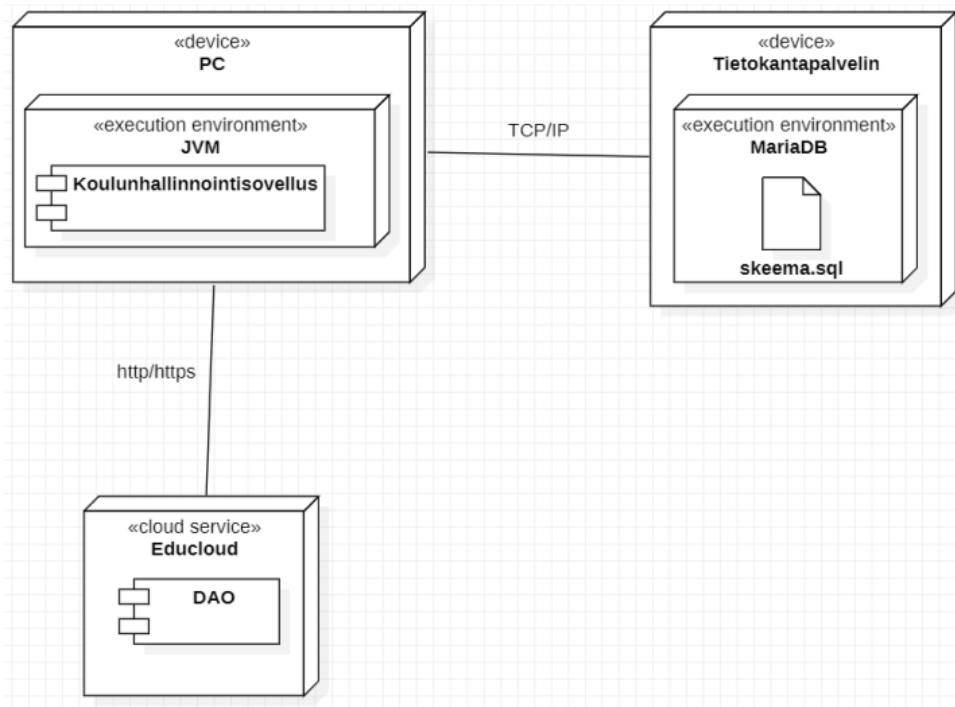
Controller-tasolla on pääkontrolleri Controller.java, joka vastaa mm. käyttäjän kirjautumisesta.

Model-tasolle on toteutettu sovelluksen eri oliotyyppien määrittelyt (mm. henkilöt, kurssit, ryhmät, poissaolot), sekä niihin liittyvät tietokannan CRUD-toiminnot omiin DAO-luokkiinsa (kuva 6). DAO-luokat käsittelevät ja palauttavat omaa oliotyyppiään vastaavia olioita kontrollereiden kautta näkymien näytettäväksi.



Kuva 6: Ohjelmiston model-tason luokkakaavio.

Tuotannossa ollessaan ohjelmisto käsittelee dataa, joka on tallessa Educloud-palvelimelle asennetussa MariaDB-relaatiotietokannassa (kuva 7). Itse ohjelma on käynnissä käyttäjän koneella paikallisesti. Kehityksessä kehittäjillä on omat lokaalit tietokannat sovelluksen testaamista varten.

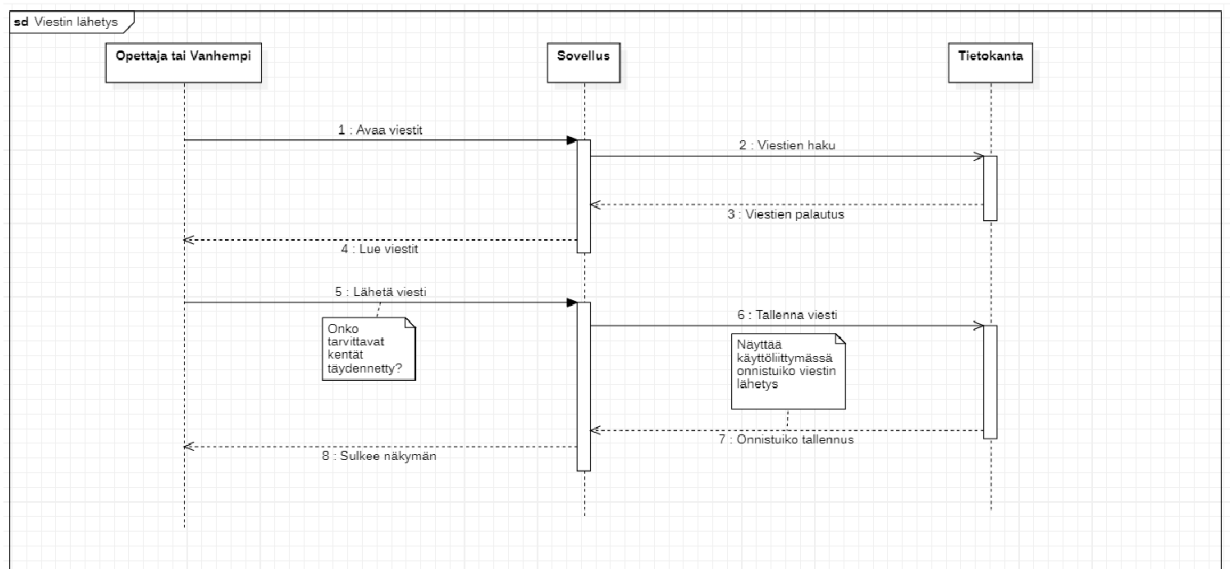


Kuva 7: Sovelluksen sijoittelukaavio.

Ohjelmiston toiminta

Sekvenssikaavio

Kaavion tavoitteena on kuvata opettajan/huoltajan viestittely toiminnon käyttöä



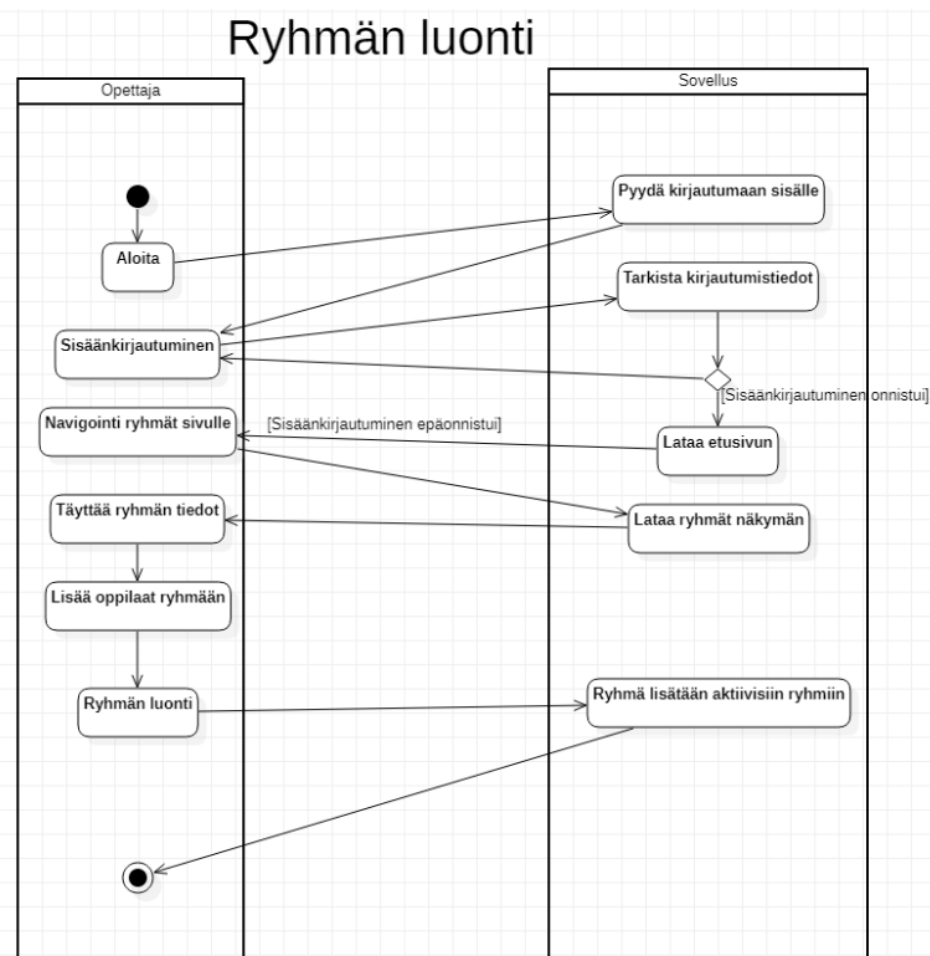
Kuva 8: Sekvenssikaavio opettajan ja huoltajan välisestä viestittelystä.

Tässä kaaviossa käydään läpi, kuinka viesti lähetetään.

Käyttäjätyypillä ei ole väliä, sillä vanhempi ja opettaja pystyvät kummatkin lähettämään viestejä. Oletamme alussa käyttäjän olevan jo valmiiksi kirjautunut sisään sovellukseen. Käyttäjä avaa viestit näkymän ja sovellus hakee tietokannasta viimeisimmät keskustelut. Käyttäjä avaa viestikeskustelu näkymän, joko painamalla olemassa olevaa viestikeskustelua tai aloittamalla uuden viestiketjun haluamansa vastaanottajan kanssa.

Viestiketju näkymässä käyttäjä voi kirjoittaa tekstikenttään haluamansa tekstin ja painaa lähetys nappia lähettääkseen viestin. Viestin lähetyksessä tarkastetaan, onko viesti vaaditussa muodossa, jonka jälkeen sovellus tallentaa viestin tietokantaan. Tietokanta ilmoittaa onnistuiko viestin tallennus, jotta pystymme ilmoittamaan tiedon käyttäjälle käyttöliittymään.

Aktiviteettikaavio



Kuva 9: Aktiviteettikaavio uuden ryhmän luomisesta.

Aktiviteettikaaviossa käymme läpi, kuinka ryhmän luonti toimii sovelluksessa. Aina toimintojen alussa käyttäjän tarvitsee kirjautua sisään sovellukseen, jonka jälkeen pystytään erottelemaan mahdolliset toiminnot käyttäjälle.

Ryhmä sivu on näkyvillä vain opettaja käyttäjätyypille. Sivulta opettaja pystyy katsomaan kaikkia aktiivisia ryhmiä jo valmiiksi olevia ryhmiä ja luomaan uusia halutessaan. Aktiivisia ryhmiä voi myös poistaa opettajan halusta.

Tiedotteet kohdassa vanhempi tyyppinen käyttäjä voi katsella opettaja käyttäjien lähettämiä tiedotteita. Opettaja tyyppinen käyttäjä voi lähettää tiedotteita ryhmille ja myös katsella lähettämiään tiedotteita. Opettajan lähettäessä tiedotetta, opettajan pitää täyttää lähettämiseen vaaditut kentät, jotta tiedote tallennetaan tietokantaan. Tietokanta ilmoittaa onnistuiko viestin tallennus, jotta pystymme ilmoittamaan tiedon käyttäjälle käyttöliittymään.

Kurssit kohdassa vanhempi pystyy katselemaan omia kurssejaan ja opettaja pystyy luomaan ja muokkaamaan kursseja. Opettajan luodessa tai muokatessa kurssia käyttäjän pitää täyttää vaaditut tiedot kurssille, jotta kurssi tallennetaan tietokantaan. Tietokanta ilmoittaa kurssin onnistuneesta luonnista tai muutoksesta, josta sovellus ilmoittaa käyttäjälle käyttöliittymään.




Lisää käyttäjä sivu näkyy vain opettaja tyyppille. Sivulla on mahdollista luoda uusia opettajia ja huoltajia. Huoltajalle pitää teon aikana ilmoittaa hänen lapsensa. Tulevaisuudessa ideana on, että huoltajia on myös mahdollista muokata ja lisätä lapsia tarvittaessa.

Poissaolot kohdassa näkyy vanhemmille valitun oppilaan poissaolot ja poissaoloille on mahdollista antaa syy, jonka opettaja pystyy näkemään. Opettajalle näkyy kaikkien oman ryhmänsä kurssien oppilaiden poissaolot ja syyt. Opettaja pysty myös lisäämään poissaolot kurssien opiskelijoille ja antamaan syyt, jos näkee sen tarpeelliseksi.

Arvosanat kohdassa vanhemmille näkyy oppilaan suoritusten tiedot, kuten nimi, saatu arvosana ja ajankohta, jolloin opettaja on arvosanan antanut. Vanhemmat pystyvät vaihtamaan sivupalkin alasvetovalikosta kenen oppilaan arvosanat näkyvät. Opettajat pystyvät lisäämään, muokkaamaan, poistamaan ja katselemaan yhden oppilaan suorituksia kerrallaan. Opettaja pystyy alasvetovalikosta vaihtamaan oppilaan.

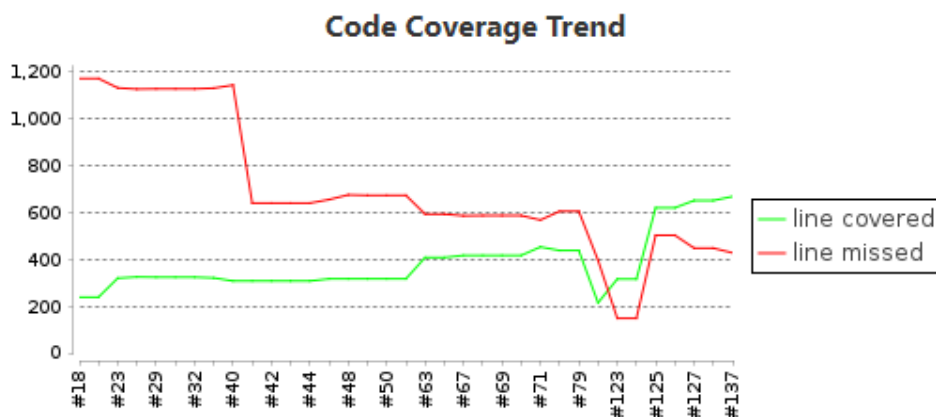
7 Testit

Sovelluksessa teimme automaattiset testit model, dao ja controller paketeille.

name	instruction
controller	M: 106 C: 393 79% 
model	M: 105 C: 763 88% 
model.dao	M: 910 C: 1160 56% 

Kuva 10: Testien kattavuusyhteenveto paketeittain

DAO luokissa on paljon virheentarkastus kohtia (jos tietokantaan yhdistäminen ei toimi), jonka ansiosta kattavuus saattaa näyttää pienemmältä muihin paketteihin verrattuna. Myös kaikki funktiot controller luokassa käyttävät jotain DAO:a, joten jotkut funktiot saavat tupla testauksen testeissä.



Kuva 11: Jenkinsin tuottama sovelluksen koodikattavuus trendi

Kuvassa 11 voi nähdä alussa ohitettujen rivien määrän 1200. Tässä vaiheessa emme olleet vielä poistaneet näkymä paketteja Jenkinsin kattavuus trendistä.

Teimme manuaalista testausta käyttöliittymään. Testasimme toiminnot syöttämällä haluttuja arvoja ja arvoja, joiden tulisi ilmoittaa virheestä käyttäjälle. Testasimme myös kielen vaihdon ja sen muutokset näkymiin, mutta emme huanneet katsoa vaihtuuko päivämäärien arvot riippuen valitusta kielestä.

8 Kehitysprosessi ja kehitysvaiheen tekniikat

Sovellus on MVC-hierarkiaa noudattava Java-ohjelma, joka on ohjelmoitu Eclipse IDE:ssä Mavenilla alustetun Java-projektin päälle. Sovelluksen versionhallinta on toteutettu Gitillä, ja sovellus on puskettu Metropolian GitLabissa olevaan julkiseen repositorioon.

Käyttöliittymän ohjelmointi on tehty JavaFX:llä, eli sovellus on Java-työpöytäsovellus. Käyttöliittymien näkymät on tehty SceneBuilder-ohjelmalla .fxml-tiedostoina.

Sovellus käyttää tietokantanaan MariaDB-relaatiotietokantaa, joka pyörii Educloud-palvelimella virtuaalikoneessa. Kehitysvaiheessa tietokantana käytettiin myös ryhmäläisten omilla koneilla pyöriviä tietokantainstansseja.

Sovelluksen ORM on tehty Hibernatea käyttäen, johon liittyvä logiikka on koodattu DAO-luokkiin MVC-arkkitehtuurin model-tasolle.

Ohjelmiston yksikkötestit on tehty JUnit 5:llä. Lisäksi JUnitilla testataan DAO-luokkien toimivuus tietokannan kanssa.

Ohjelmiston koonti on tehty Mavenilla. Jatkuvaan integrointiin käytetään Docker-alustalla pyörivää Jenkinsiä, joka on asennettu samalle Educloud-palvelimelle kuin tietokantakin. Jenkin koostaa Mavenilla ja ajaa automaattisesti JUnitin testit GitLabiin pushatusta koodista tasaisin väliajoin. Lisäksi Jenkinsissä on käytössä Build Monitor View jobien tilan seurannan helpottamiseksi.

9 Jatkokehitys

Sovelluksessa on tällä hetkellä tärkeimmät haluamamme toiminnot, mutta osa ominaisuuksista jouduttiin jättämään pois ajanpuutteen takia.

Yksi tärkeimmistä pois jätetyistä ominaisuuksista oli lukujärjestys ja tunnit. Lukujärjestys auttaisi sekä opettajia, että huoltajia. Tunneilla pystyisimme Esimerkiksi. Poissaoloja voidaan vain merkata kurssin ja päivän kanssa. Tunneilla pystyisimme näyttämään tarkasti, milloin opiskelija on ollut myöhässä.

Päänäkymä. Tällä voitaisiin helpottaa huoltajan ja opettajan käyttökokemusta. Päänäkymässä näkyisi kaikki uudet viestit, tiedotteet, poissaolot ja sen päivän lukujärjestys. Moni huoltaja haluaa tarkastaa lapsensa tiedot mahdollisimman vaivattomasti ja tällä voisimme helpottaa heidän käyttökokemustansa.

Huoltajien muokkaus. Tällä hetkellä huoltajia ja heidän lapsiaan ei voi muokata. Tässä tulee ongelma, jos huoltajalla on monta lasta ja toinen lapsi tulee eri aikaan kouluun.

10 Yhteenveto

Yhteenvetona sovelluksen tarkoitus on helpottaa oppilaan koulunkäyntiin liittyvien seikkojen seurantaa ja ylläpitoa. Sovellus on tarkoitettu ala-aste käyttöön, missä oppilailla ei ole pääsyä sovellukseen. Sovelluksen pääkäyttäjät ovat oppilaiden huoltajat ja heidän opettajansa. Sovellusta pääsee käyttämään, kun opettaja luo uudet tunnukset uudelle käyttäjälle.

Sovelluksen kautta huoltajat ja opettajat voivat viestitellä keskenään. Opettajat pystyvät lähettämään tiedotteita, jotka ovat suunnattu tietyn opiskelijaryhmän huoltajille. Opettajat voivat luoda kursseja, sekä lisätä niille opiskelijaryhmiä. Opettajat pystyvät myös antamaan arvosanoja ja merkitsemään poissaolot. Huoltajat pystyvät katselemaan arvosanoja ja merkitsemään poissaoloille syitä.