# Sisuprojektin dokumentaatio

Ronja Koivisto, Essi Pietilä

## Toteutetut ominaisuudet

Tässä projektissa toteutettuun opintorekisteriin on toteutettu seuraavat perusominaisuudet:

- Tutkintojen tietojen hakeminen Sisun API:sta
- Tutkinnon valitseminen ja muuttaminen myöhemmin
- Tutkintorakenteen esittäminen Sisu API:n tietojen mukaan
- Tutkinnon tilanne, eli suoritetut opintopisteet suhteessa valitun tutkinnon minimivaatimuksiin esitetään ja suoritetut kurssit saa halutessaan nähtäville
- Virheiden hallittu käsittely
- Yksikkötestejä jokaiselle luokalle
- Graafinen käyttöliittymä on toteutettu itse

sekä nämä lisäominaisuudet:

- Opiskelijan tietojen syöttö ohjelmaa käynnistettäessä, niiden ja suoritusten tilanteen hakeminen ohjelmasta sekä tallennus JSON-tiedostoon ohjelman päättyessä. Opiskelijan tietoja on mahdollista myös muuttaa sulkematta ohjelmaa.
- Oma lisäominaisuus: Valitun tutkinto-ohjelman rakenne avautuu uuteen ikkunaan. Ikkunasta on mahdollista siirtyä edelliseen näkymään vaihtamaan tutkinto-ohjelmaa. Näkymässä on toteutettu myös edistymisen seuranta, josta käyttäjä näkee listauksen suorittamistaan kursseista sekä kertyneiden opintopisteiden suhteen tutkinnon vaatimiin opintopisteisiin.

## Ohjelman sisältämät luokat

Ohjelma sisältää neljä varsinaista luokkaa sekä testiluokat näistä jokaiselle.

### App

Ohjelman pääluokka, jossa määritellystä Application-luokasta luodaan App-niminen olio. App-luokassa toteutetaan ohjelman käyttöliittymä, tutkinto-ohjelmien luku sekä tietojen tallennus json-tiedostoon ohjelman suorituksen päättyessä. Luokka tallentaa tiedon kaikista tutkinto-ohjelmista TreeMapiin, jossa avaimena on tutkinto-ohjelman nimi String-tyyppinä ja arvona tutkinto-ohjelmalle luotu Degree-olio. Luokka myös tallentaa nykyisen opiskelijan Student-olion currentStudent-attribuuttiin, opiskelijan valitseman tutkinto-ohjelman Degree-olion currentDegree-attribuuttiin sekä opiskelijan etenemiselle luodun Progress-olion currentProgress-attribuuttiin. Lisäksi luokan attribuutteina on ohjelman kolmatta ikkunaan tarvittavat Scene, TreeView ja Label. Pääluokalla on lisäksi Type-niminen enum-tyyppinen yksityinen attribuutti, jonka avulla tutkintorakenteen tietoja luettaessa on mahdollista kohdistaa erilaiset toimenpiteet tutkinnon, modulin ja kurssin luvulle.

Luokka myös funktiota datan lukemiseen ja tallentamiseen. ReadUrl-funktiolla yliopiston tutkintorakenteet haetaan funktion parametrinaan saamasta URL-osoitteesta, tiedot luetaan ja tallennetaan String-tyyppiseksi. Funktio palauttaa tämän String-tyyppisen muuttujan, joka sisältää Sisun API:sta saadut tiedot. ReadJson-funktio lukee Json-muodossa olevan datan ja palauttaa Json objectin, joka sisältää luetun tiedon. Funktio ottaa parametreinaan ID:n, joka modulia tai kurssia luettaessa lisätään verkkokutsun perään, jotta saadaan juuri kyseisen modulin tai kurssin tiedot, sekä Type-tyyppisen muuttujan, jonka avulla tunnistetaan, milloin luettava kohde on tutkinto, milloin tutkinnon moduli ja milloin yksittäinen kurssi. Funktio ottaa parametrinaan tunnuksen, jonka takaa

kyseisen kurssin tiedot löytyvät Json-muodossa. Kurssin luomisen yhteydessä, sen puurakenteessa esiintyvän valintaruudun klikkaamiseen lisätään tässä vaiheessa myös toiminnallisuus. Kun kurssi valitaan, lisätään se edistymistä seuraavaan progress-olioon ja vastaavasti poistettaessa valinta, kurssi myös poistetaan tästä progress-oliosta. GetSubModules-funktio lukee tutkinto-ohjelman sisältämiä moduleja ja luo kullekin kohdan kolmannen näkymän puurakenteeseen. Funktio ottaa parametrinaan ID:n, josta kyseinen moduli löytyy sekä CheckBoxTreeItem<String> tyyppisen yläotsikon, jonka alle kyseinen moduli kuuluu. Funktio lisää kyseisen modulin mukaisen otsikon (CheckBoxTreeItem<String>) puurakenteeseen ja tarkistaa rekursiota hyödyntäen modulin sisältämät alimodulit. ReadToDegree-funktio ottaa parametrinaan Json Objectin, joka sisältää yliopiston tutkinto-ohjelmat. Tutkinnoista luetaan ja otetaan talteen tutkinnon nimi, tutkinnon ID sekä opintopisteiden maksimi- ja minimimäärä. Näitä tietoja käytetään uuden Degree-luokan olion luomiseen. Luotu Degree-olio tallennetaan arvona TreeMapiin tutkinto-ohjelman nimen (String) mukaisen avaimen alle.  WriteJsonFile-funktio tallentaa opiskelijan tiedot sekä opiskelijan etenemisen Json-tiedostoon. Tiedoston nimi on opiskelijan asetusikkunassa syöttämä nimi. Lisäksi kolme viimeisimpänä mainittua funktiota ovat void-tyyppisiä eivätkä näin ollen palauta mitään.

Lisättävää?

### Student

Student-luokka sisältää yksinkertaisen toteutuksen opiskelijalle. Opiskelijalle määritetään nimi, opiskelijanumero, aloitusvuosi sekä tavoiteltu valmistumisvuosi. Luokan kaikki attribuutit ovat yksityisiä. Attribuuteille ei ole tässä luokassa asetettu rajoituksia, sillä luokka on haluttu pitää yleispätevänä ja siksi projektin kannalta oleelliset rajoitukset opiskelijan tiedoille määritetään App.java tiedostossa. Opiskelijan nimen saa ulkopuolisista luokista String-arvona selville julkisella getName -metodilla, opiskelijanumeron niin ikään String-arvona getStudentNumber-metodilla ja aloitus ja lopetusvuodet int-arvona getStartYear- ja getEndYear-metodeilla.

### Degree

Degree-luokka tallentaa tutkinnon nimen, opintopisteiden vähimmäisvaatimuksen, enimmäisvaatimuksen sekä sen yksilöivän id:n Sisu API:ssa, kaikki String-attribuutteina. Myös nämä ovat kaikki yksityisiä, mutta ne saa kutsuttua. Tutkinnon nimen, opintopisteiden vähimmäisvaatimuksen, enimmäisvaatimuksen sekä yksilöivän id:n saa ulkopuolisista luokista selville String-arvoina selville julkisilla metodeilla: tutkinnon nimen getDegreeName-metodilla, opintopisteiden vähimmäisvaatimuksen getMinCredits-metodilla, enimmäisvaatimuksen getMaxCredits-metodilla ja yksilöivän id:n getDegreeId-metodilla.

### Progress

Progress-luokka sisältää toteutuksen opiskelijan edistymiselle. Luokka ylläpitää TreeMapin avulla tietoa käydyistä kursseista. TreeMapissa avaimena on käydyn kurssin nimi String-tyyppisenä ja arvona kurssin laajuus opintopisteinä. Lisäksi luokka tallentaa tiedon tutkinto-ohjelman vaatimista opintopisteistä sekä pitää yllä tietoa opiskelijan kerryttämistä opintopisteistä int-attribuutteina. Nämä kaikki ovat luokan yksityisiä attribuutteja. AddCompletedCourse-metodilla lisätään suoritettu kurssi TreeMapiin ja lisätään kurssin opintopistemäärä jo kertyneihin opintopisteisiin. RemoveCompletedCourse-metodilla haluttu kurssi poistetaan TreeMapista ja vähennetään kurssin opintopistemäärä kertyneistä opintopisteistä. GetProgress-metodi palauttaa pair-tyyppinä kertyneet opintopisteet ja tutkinnon vaatiman opintopistemäärän. GetCompletedCourses-metodi palauttaa kopion TreeMapista, joka sisältää kaikki suoritetut kurssit. Kaikki edellä kuvatut metodit ovat julkisia.

## Käyttöohje

Ohjelman ensimmäisessä ikkunassa käyttäjä syöttää nimensä, opiskelijanumeronsa, opintojen aloitusvuotensa sekä valmistumisen tavoitevuotensa. Nimi ja opiskelijanumero saavat sisältää kirjaimia ja/tai numeroita rajoituksetta. Opintojen aloitusvuosi on oltava numero. Lisäksi sen on oltava vähintään vuosi 1960, mutta korkeintaan kuluva vuosi. Valmistumisen tavoitevuoden on niin ikään oltava numero. Sen täytyy olla suurempi kuin aloitusvuosi, mutta korkeintaan 7 vuoden päässä aloitusvuodesta.

Kun käyttäjä on syöttänyt tietonsa ja painanut Valmis-nappia siirrytään näkymään, jossa on listattuna kaikki tutkinto-ohjelmat. Vasemmalla olevasta sivupalkista löytyvät Takaisin alkuun – ja Lopeta-napit. Sivupalkin saa näkyville painamalla klikkaamalla ohjelman vasemmasta reunasta löytyvää kolmea viivaa. Vastaavasti sen voi piilottaa painamalla nyt samasta kohdasta löytyvää nuolta. Takaisin alkuun- nappia painamalla käyttäjä pääsee muuttamaan aiemmin antamiaan tietoja (nimi, opiskelijanumero, jne.), ja Lopeta-napista ohjelman suoritus päättyy. Näkymään on listattu tutkinto-ohjelmat nappeina, joita klikkaamalla käyttäjä pääsee tarkastelemaan tarkemmin kyseisen tutkinto-ohjelman rakennetta. Klikkaamalla tutkinto-ohjelmaa käyttäjälle siirtyy uuteen ikkunaan, jossa tutkinnon osat ja osien sisältämät kurssit esitetään puurakenteessa. Puurakenteesta käyttäjä voi kyseisen osan tai kurssin valintaruutua klikkaamalla merkitä kyseisen osan suoritetuksi. Näkymän oikeassa reunassa kuvataan tutkinnon edistymistä. Kun suoritettuja kursseja on lisätty ja käyttäjä painaa Näytä suoritetut kurssit -nappia, päivitetään näkymään listaus suoritetuista kursseista sekä opintopistemäärä, joka tutkintoa on suoritettu. Jos suoritettuja kursseja vielä lisätään tai poistetaan, saa listauksen ja kertyneiden opintopisteiden päivittymään painalla Päivitä suoritetut kurssit -nappia. Tämänkin näkymän vasemmassa reunassa on sivupalkki, josta löytyvät edellisessäkin näkymässä olleet Takaisin alkuun- ja Lopeta-napit. Näiden lisäksi sivupalkissa on Vaihda tutkinto -nappi, josta käyttäjä pääsee takaisin edelliseen näkymään, jossa tutkinto-ohjelmat ovat listattuna.

Käyttäjä voi lopettaa ohjelman suorituksen missä vaiheessa tahansa sulkemalla ikkunan oikean ylänurkan rastista tai klikkaamalla sivupalkista löytyvää Lopeta-nappia. Tiedostoon tallennetaan ne tiedot, jotka käyttäjä on antanut enne ohjelman suorituksen päättymistä.

## Ohjelman toiminta

Ensimmäisessä ikkunassa käyttäjä syöttää nimensä, opiskelijanumeronsa, opintojensa aloitusvuoden sekä valmistumisen tavoitevuoden. Klikattuaan Valmis-nappia luodaan uusi Student-olio, johon tallennetaan opiskelijan tiedot. Kyseinen Student-olio tallennetaan pääluokan currentStudent-attribuuttiin. Kuten käyttöohjeessa jo mainittiin, opiskelijan nimelle ja opiskelijanumerolle ei ole rajoitteita, sillä ne tallennetaan String-muuttujina Student-olioon. Aloitusvuoden ja valmistumisen tavoitevuoden on sen sijaan oltava numeroita, koska ne tallennetaan integer-tyyppeinä Student-olioon. Lisäksi aloitusvuodelle on rajoite, että sen on oltava välillä 1960–2022. Vastaavasti valmistusvuosi ei voi olla pienempi tai yhtä suuri kuin aloitusvuosi, ja se saa olla korkeintaan 7 vuotta aloitusvuotta myöhempi. Valmis-nappia painamalla ohjelma myös hakee yliopiston tutkintorakenteet ja tutkinto-ohjelmien nimet Sisun API:sta. Tässä vaiheessa jokaiselle tutkinto-ohjelmalle luodaan Degree-olio, jotka kaikki tallennetaan TreeMapiin arvoina kyseisen tutkinto-ohjelman nimen (String) mukaisen avaimen alle.

Toista näkymää varten jokaiselle tutkinto-ohjelmalle luodaan nappi, jota painamalla käyttäjä siirtyy uuteen ikkunaan, jossa kyseisen tutkinnon rakenne esitetään puurakenteena. Toiseen näkymään luodaan lisäksi sivupalkki, josta löytyvät tässä vaiheessa napit ”Alkuun” ja ”Lopeta”. Alkuun-nappi asettaa sceneksi ohjelman ensimmäisen scenen, jossa käyttäjästä kerätään tietoja ja näin ollen

käyttäjä voi nyt muuttaa aiemmin antamiaan tietoja. Lopeta-nappia painamalla ohjelman suoritus päättyy. Ohjelman suorituksen päättyessä luodaan tiedosto, johon tallennetaan opiskelijan siihen mennessä antamat tiedot.

Klikkaamalla jonkin tutkinto-ohjelman nappia käyttäjä siirtyy ohjelman kolmanteen ikkunaan. Tässä vaiheessa valittu tutkinto-ohjelma tallennetaan myös pääluokan currentDegree-attribuuttiin. Käyttäjän edistymiselle luodaan uusi Progress-olio, joka tallennetaan pääluokan currentProgress-attribuuttiin. Nyt ohjelma myös hakee Sisun API:sta tiedot valitun tutkinto-ohjelman sisältämistä osista ja kursseista, ja nämä esitetään kolmannessa näkymässä hierarkkisesti puurakenteessa. Edellisen näkymän tapaan myös tämän näkymän vasemmasta reunasta löytyy sivupalkki, josta löytyy nyt Vaihda tutkinto -nappi Alkuun- ja Lopeta-nappien lisäksi. Tutkinnot-nappia painamalla sceneksi asetetaan ohjelman toinen scene, jossa käyttäjä pääsee siis vaihtamaan tutkinto-ohjelmaa. Aiemmassa tutkinnossa suoritetuiksi merkittyjä tutkinnon osia tai kursseja ei tallenneta, jolloin tutkinnon vaihtaminen nollaa edistymisen aiemmin valitussa tutkinto-ohjelmassa. Alkuun- ja Lopeta napit toimivat samalla tavalla kuin aiemmassa näkymässä. Kuten käyttöohjeessakin jo mainittiin, Tutkinnot-nappi ei näy sivupalkissa heti, jos sen on pitänyt auki siirtyessä toisesta näkymästä kolmanteen, vaan näppäimen päivittyminen näkymään vaatii sivupalkin sulkemisen ja avaamisen uudestaan.

Kolmannen näkymän oikeasta reunasta löytyy seuranta tutkinnon edistymiseen. Kun käyttäjä klikkaa suorittamansa kurssin tai tutkinnon osan valintaruutua puunäkymässä, valitut kurssit ja niiden laajuudet tallennetaan currentProgress-olioon. Käyttäjän eteneminen päivitetään näkymään Näytä suoritetut kurssit-nappia painamalla. Tällöin näkymään avautuu lista suoritetuista kursseista sekä tieto siitä, kuinka suuri osa tutkinnon vaatimista opintopisteistä on suoritettu. Mikäli kurssisuorituksia muutetaan, päivittyy edistyminen Näytä suoritetut kurssit-napin tilalla olevaa Päivitä suoritetut kurssit- nappia painamalla.

Mikäli käyttäjä lopettaa ohjelman suorituksen aloitusnäkymässä painamalla näkymän ylänurkassa olevaa rastia, mitään tietoja ei tallenneta. Jos käyttäjä on syöttänyt tietonsa ja siirtynyt tutkinto-ohjelmat sisältävään näkymään, tämän jälkeen lopetettaessa ohjelman suorituksen, luodaan json-tiedosto. Mikäli käyttäjä on ohjelman toisessa näkymässä eli on pelkästään syöttänyt omat tietonsa, mutta ei ole valinnut tutkinto-ohjelmaa tai suoritettuja kursseja, ja lopettaa ohjelman suorituksen painamalla näkymän ylänurkassa olevaa rastia tai sivupalkista löytyvää Lopeta-nappia, tallennetaan json-tiedostoon pelkästään käyttäjän nimi, opiskelijanumero, opintojen aloitusvuosi ja valmistumisen tavoitevuosi. Mikäli käyttäjä sulkee ohjelman kolmannessa näkymässä ja on valinnut suoritettuja kursseja, tallennetaan json-tiedostoon aloitusnäkymässä annettujen tietojen lisäksi suoritettujen kurssien nimet ja niiden laajuudet opintopisteinä. Lisäksi tiedostoon tallennetaan tieto kertyneistä opintopisteistä sekä tutkinnon vaatimasta opintopistemäärästä. Luodun Json-tiedoston nimi on sama kuin käyttäjän antama nimi, johon on lisätty .json-pääte.

## Tiedossa olevat ongelmat ja puutteet

Ohjelmassa on muutama tiedossa oleva ongelma ja/tai puute, jotka rajoittavat ohjelman toimintaa. Käyttäjä ei voi esimerkiksi lisätä kursseja toisesta tutkinto-ohjelmasta, vaan kerralla pystytään seuraamaan ja tallentamaan edistymistä yhdessä tutkinto-ohjelmassa. Mikäli tutkinto-ohjelmaa vaihdetaan, kurssivalinnat ja tutkinnon eteneminen nollautuvat ja seuranta aloitetaan alusta valitun uuden tutkinto-ohjelman kohdalla.

Toinen ohjelmassa esiintyvä ongelma liittyy Sisu API:n moduleihin, joista ei anneta lisätietoja. Joissakin tutkinto-ohjelmissa Sisu API näyttää modulin, mutta modulista ei ole lisätietoja, joten ohjelma ei tiedä mitä osia ja kursseja kyseiset modulit sisältävät, ja näin ollen ei myöskään tiedetä modulin laajuutta opintopisteinä. Tällaiset modulit esitetään puurakenteessa otsikolla ”Sisu API näyttää tässä modulin ilman lisätietoja”. Näiden otsikoiden alla ei ole tarkentavia alaotsikoita tai kursseja, mutta modulit voi merkitä suoritetuiksi valintaruudusta kuin minkä tahansa muun kurssin tai tutkinnon osan.

Ohjelmassa esiintyy myös kertyneiden opintopisteiden seurantaan ja niiden suhteeseen tutkinnon vaatimiin opintopisteisiin liittyvä ongelma. Vaikka käyttäjä merkitsee suoritetuksi kaikki tutkinnon sisältämät osat ja kurssit, seuranta voi joidenkin tutkinto-ohjelmien kohdalla näyttää, ettei opintopisteitä ole vaadittua määrää. Tämä aiheutuu tutkinnon sisältämistä vapaasti valittavista opintokokonaisuuksista sekä edellä kuvatuista moduleista, joille Sisu API ei anna lisätietoja. Nämä osat eivät sisällä kursseja, jotka käyttäjä voisi merkitä suoritetuksi, mutta toisaalta eivät myöskään omaa tiettyä laajuutta eivätkä näin ollen kerrytä opintopisteitä käyttäjän merkittyä niitä suoritetuiksi.

Lisäksi lisäominaisuutena oleva sivuvalikko hukkuu näkyvistä, kun kurssien valinnasta palaa takaisin tutkinnon valintaan.

## Työnjako

Projektin alussa oli vaikea hahmottaa kaikkea sen vaatimia osia ja kunkin osuuden toteuttamiseen vaadittavaa aikaa, joten lähdimme perehtymään ja työstämään sitä ensin yhteisesti. Tiedossa oli, että tulemme todennäköisesti työstämään projektia saman aikaisesti samassa paikassa, joten tarkka työnjako ei mielestämme ollut tarpeen. Tällainen toimintatapa toimi hyvin myös aiemmalla ohjelmointikurssilla. Projektin osista ja niiden toteuttamisesta oli helppo käydä keskustelua ja tehdä yhteisiä päätöksiä ryhmän ollessa samassa paikassa kirjoittamassa koodia.

Koska selvää työnjakoa ei ennakolta ollut, työnjako ja vastuualueet päivittyivät ikään kuin iteratiivisesti projektin edetessä. Aluksi päätettiin, että toinen lähtee työstämään ohjelman aloitusikkunaa ja toinen tutkintorakenteiden hakua Sisun API:sta. Tämän jälkeen toimittiin siten, että toisen saadessa osuutensa valmiiksi keskusteltiin yhdessä, mikä on projektin toteuttamisen seuraava vaihe, jota tämä osuutensa valmiiksi saanut lähti seuraavana työstämään. Työmäärä jakautui tällä keinoin tasaisesti.