Tampereen yliopisto

Kyykkäjengi: Eero Hakanen, Joona Nieminen & Tommi Poikolainen

DOKUMENTAATIO

COMP.CS.140 projektin dokumentaatio

SISÄLLYSLUETTELO

1. KÄYTTÖOHJE

Ohjelman käynnistyessä ruudulle aukeaa kirjautumisikkuna. Mikäli ikkunassa ilmenevä kieli ei miellytä, saa sitä vaihdettua napista, jossa lukee ”fi”. Syötä tietosi asianmukaisiin kenttiin, valitse oma tutkinto-ohjelmasi ja siirry ohjelmassa eteenpäin painamalla kielestä riippuen Save/Tallenna -nappia. Mikäli käyttäjä (tunnisteena käytetään opiskelijanumeroa) on aikaisemmin käyttänyt ohjelmaa haetaan käyttäjän aikaisemmin suorittamat kurssit ja ne näkyvät seuraavassa näkymässä.

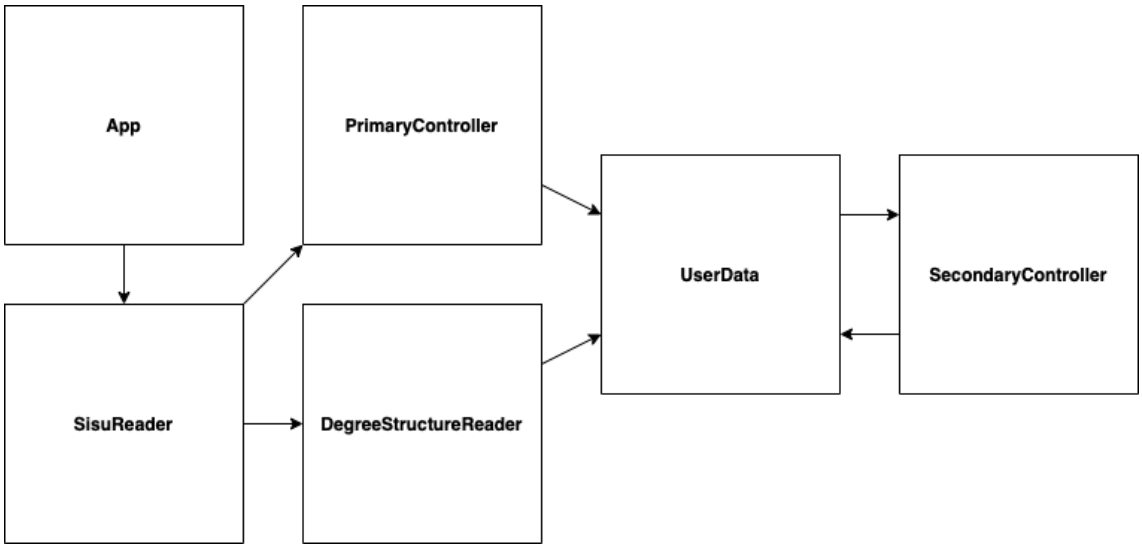
Hetken lataamisen jälkeen eteesi avautuu kurssivalintanäkymä. Näkymä koostuu puusta, jossa pääsee etenemään eteenpäin aina tuplaklikkaamalla haluttua osaa. Halutun moduulin/kokonaisuuden avattuaan kursseja voi merkata suoritetuksi tuplaklikkaamalla sitä. Kurssi ilmestyy oikealla olevaan suoritettujen kurssien ikkunaan. Vastaavasti suoritettua kurssia vasemmassa tai oikeassa ikkunassa tuplaklikkaamalla voi poistaa tämän kurssin suoritettujen listasta.

Kurssisuoritusten käsittelyn jälkeen voi palata kirjautumisikkunaan Back/Takaisin -napista. Takaisin kirjautumisikkunaan päästyä voit jälleen syöttää asianmukaisiin ikkunoihin jonkun muun opiskelijan tiedot tai kirjautua samoilla tunnuksilla sisään ja jatkaa omien tietojesi muokkaamista.

2. OHJELMAN RAKENNE

Ohjelman rakenne koostuu kuudesta luokasta. Alla oleva kuva havainnollistaa yksinkertaistetusti päällepäin luokkien suhteita. App-luokka käynnistää ohjelman ja hakee heti SisuReaderilla mahdolliset tutkinto-ohjelmat ensimmäisen ikkunan tutkinto-ohjelmavalikkoon. PrimaryController-luokka muodostaa ensimmäisen ikkunan ja tallentaa käyttäjän tiedot UserData-olion avulla.

DegreeStructureReader- ja SisuReader-luokkia käytetään hyväksi Sisun tietojen avaamiseen ja käsittelyyn. Luokkien avulla muodostetaan ohjelman toiseen näkymään asianmukainen tutkintopuu. SecondaryController-luokasta palatessa tallennetaan vielä UserData-olion tiedot ulkoiseen json-tiedostoon, jonka avulla tietoja on mahdollista käyttää myöhemmin.



Rakenne haluttiin jakaa tällä tavoin juuri näihin luokkiin, sillä tällä tavoin ohjelmassa tapahtuvat asiat pysyvät selkeinä. On helppoa tietää, että missä luokassa tapahtuu ja mitä asioita.

3. OHJELMAN TOIMINTA JA TOTEUTETUT LISÄOMINAISUUDET

Toteutettuja lisäominaisuuksia on kolme, lisäksi ohjelman grafiikan taso on vaatinut lisätyötä. Luettelo toteutuneista lisäominaisuuksista ja lisätöistä:

1. Kielen valinta: Ohjelman sekä kurssien ja tutkintojen kielet vaihtuvat toiseen tarvittaessa ensimmäisessä ikkunassa kielestä riippuen, joko ”fi” tai ”en” nappia painamalla. (Kaikille sisun tiedoille ei ole välttämättä saatavilla molempia kieliä, jolloin kyseisten tietojen kieltä ei voida muokata.)

2. Opiskelijadatan tallennus ja luku: Ohjelma tallentaa opiskelijan tiedot JSON-tiedostoon suljettaessa ja valikkoikkunasta kirjautumisikkunaan palatessa. Mikäli kirjautumiseen käytettyä opiskelijanumeroa vastaavan niminen tiedosto löytyy, tiedostosta luetaan suoritetut kurssit näkyviin ohjelmaan, jolloin niitä voidaan käsitellä lisää.
3. Automaattinen täydennys (Autocomplete): Kun tutkintonimikettä haetaan etusivulla valikosta (ComboBox-komponentti, ns. dropdown-menu), valikosta rajautuu vain mahdolliset hakutulokset merkin syöttämisen jälkeen. Esimerkiksi syöttämällä merkkijonon ”arkki” jättää hakuvaihtoehdoiksi vain arkkitehdin kandidaattitutkinnon ja arkkitehdin DI-tutkinnon. Haku toimii myös englanninkielisessä asetuksessa englanninkielisillä tutkinnoilla.
4. Ohjelman grafiikkaa on paranneltu hyödyntämällä css-tiedostoa. Toteutettu esimerkiksi nappien tyylin muuttuminen niiden tilan muuttuessa.

Ohjelman toiminta pohjautuu tietenkin emoluokkaan App. Luokan sisällä avataan/käynnistetään ikkuna ja tuodaan kaksi fxml-tiedostoa sekä yhden css-tiedoston mukaan koodin pariin. Ohjelma avaa ensin primary-ikkunan, eli kirjautumisikkunan. Tämän kautta ohjelman suoritus siirtyy luokkaan PrimaryController.

Luokan avulla käsitellään syötekenttiä, kielen vaihtamista sekä tutkintojen lisäämistä tutkintosyötteeseen. Tutkinnon syöttämistä helpottamaan on lisätty tekstin autocomplete-ominaisuus, joka myös pyörii tämän luokan sisällä. Syötteet annettuaan luokka muodostaa UserData-olion ja luokka tarkistaa, että onko tietokoneella jo olemassa vastaavalla opiskelijanumerolla varustettua json-tiedostoa, josta voisi suoritettuja kursseja lukea. Tämän tehtyään luokka antaa komennon ohjelmalle, joka siirtää ohjelman suorituksen toiseen ikkunaan.

SecondaryController-luokka luo käynnistyessään kaksi ikkunaa. Vasemmalle tulee näkyviin Sisusta haetusta json-tiedostosta tehty kurssipuu. Oikealla taas on ikkuna, josta näkee suoritetut kurssit. Edellä mainitun kurssipuun muodostus tapahtuu varsinaisesti viimeisessä mainitsemattoman luokan DegreeStructureReaderin ja SisuReaderin avulla. DegreeStructure-luokkaan on asetettu tietyt säännöt, joilla tietoja haetaan Sisusta Sisureaderin avulla rekursiivisesti. Tiedoista luodaan TreeItem-komponentteja, jotka asetetaan root-nimiseen TreeItemiin. SecondaryControllerissa annetaan luoduille kurssi-TreeItemeille ominaisuus, jolla niitä voidaan merkata joko käydyksi tai poistaa käydyistä kursseista.

Opiskelijan tiedot ovat tallennettuja suorituksen ajaksi UserData-olioon. Tämä UserData-luokka koostuu pääosin set- ja get-funktioista, eli toimii enimmäkseen tallennuskohteena ja tietojenkäsittelyn mahdollistajana. Ohjelman toisesta ikkunasta palatessa tai ohjelman vain suljettaessa kutsutaan App-luokassa olevaa saveUserData-funktiota. Tämän johdosta muodostuu json-muotoinen tiedosto, johon on tallennettuna opiskelijan tiedot ja tätä tiedostoa voidaan käyttää tietojen jatkokäsittelyyn.

4. SOVITTU JA TOTEUTUNUT TYÖNJAKO

Projektin alussa arvioidut tehtävät ja työt olivat jaettuna kolmeen osaan. Ensimmäisen osan eli Sisu-API:n toteutuksesta oli vastaamassa Joona ja suunnitelma pitikin loppupeleissä paikkansa, pois lukien Eeron pienen avun.

Seuraavaksi puhuttiin yleisesti kaiken taustalla tapahtuvan toteutuksesta (Back end) ja Eero tahtoi saada osakseen tämän työn. Todellisuudessa Eero sai hoidettua miltei kaiken tarvittavan, esim. kurssipuun luonnin Sisusta haettavasta datasta. Tommi sekä Joona auttoivat tarvittaessa pienempien toimintojen luonnissa.

Viimeisenä Tommi halusi ottaa vastuulleen ohjelman grafiikan ja sen muodostamiseen vaadittavan koodin, mutta todellisuudessa Eero sai tehtyä vahvan, miltei valmiin pohjan mitä Tommi lähti sitten muokkaamaan ja kehittämään loppuun asti.

Testaamisesta sovittiin, että jokainen tekisi testin aina luomalleen ominaisuudelleen, mutta todellisuudessa testauksesta on vastannut vain Joona sekä Eero. Graafisten käyttöliittymäkomponenttien testauksen teki Joona. Tommi on luonut tarvittavat rajapintadokumentit sekä koko ryhmä tämän käyttöohjeen. Työnjako siis sovittiin aina sitä mukaan, kun pääteltiin, että mitä kannattaisi seuraavaksi tehdä.

5. ONGELMAT JA PUUTTEET

Projektin aikana on tullut törmättyä muutamaan kielelliseen/taidolliseen rajoitteeseen kurssipuuta vasempaan ikkunaan kirjoittaessa. Ikoneita ei kyetty mitenkään luomaan kurssien suorituksen merkkaamiseksi. Sama asia kävi ilmi, kun ajatuksena oli luoda vain kursseista check boxeja ja muista tavallisia puun osia. Tästä syystä suoritusten merkkaamiseksi siirryttiin siihen, että ohjelmassa käytetään toista ikkunaa näyttämään suorituksia.