Kalenterisovellus

PROJEKTITYÖN DOKUMENTTI

YLÖNEN OIVA

SISÄLLYSLUETTELO

Henkilötiedot *2*

Yleiskuvaus *2*

Käyttöohje *2*

Käynnistys 2

Navigointi 2

Tapahtuman lisääminen 3

Tapahtuman poistaminen 4

Tapahtumien selaaminen 4

Raporttien haku 6

Tapahtumien suodattaminen 6

Ohjelman rakenne *7*

Algoritmit *8*

Tietorakenteet *9*

Tiedostot ja verkossa oleva tieto *9*

Testaus *10*

Ohjelman tunnetut puutteet ja viat *11*

3 parasta ja 3 heikointa kohtaa *11*

Parhaat 11

Heikoimmat 12

Poikkeamat suunnitelmasta, totetunut työjärjestys ja aikataulu *12*

Kokonaisarvio lopputuloksesta *13*

Viitteet ja muualta otettu koodi *13*

Käytetyt lähteet 13

Tekoälyn käyttö 13

Liitteet *13*

Henkilötiedot

Otsikko: Kalenteri

Tekijän nimi: Oiva Ylönen

Opiskelijanumero: 100788387

Koulutusohjelma: Informaatioverkostot

Vuosikurssi: 2

Päiväys: 21.4.2025

Yleiskuvaus

Toteutin ScalaFX -pohjaisen työpöytäkalenterin, jonka keskeiset ominaisuudet löytyy sovelluksen yläpalkista. Yläpalkista pystyy siirtymään näkymien välillä haluamaansa näkymään.

Kalenterista pystyy muun muassa lisäämään ja poistamaan tapahtumia, sekä tallentamaan muutokset pysyvästi iCalendar-muotoiseen tiedostoon. Tämä mahdollistaa kalenterin tapahtumien yhteensopivuuden muiden kalenterisovellusten kanssa. Tapahtumat näytetään kolmessa eri näkymässä, päivä, viikko ja kuukausi. Näiden välillä selaaminen onnistuu kalenterisovelluksen yläpalkista. Tapahtumia on myös mahdollista suodattaa kategorioiden mukaan, jolloin näkymien tapahtumat päivityvät. Kategoria myös määrää minkä värisen reunan tapahtuma saa viikko ja päivä näkymässä. Kalenterista löytyy myös raportti näkymä josta voi tarkastella eri kategorioiden tapahtumien määrää ja kestoa eri ajanjaksoilla.

Työn toteutus on tehty vaativan ja keskivaikean välimaastossa. Tein kolme vaikeaa ominaisuutta neljästä, mutta muutama helpompi ominaisuus jäi toteuttamatta.

Käyttöohje

Käynnistys

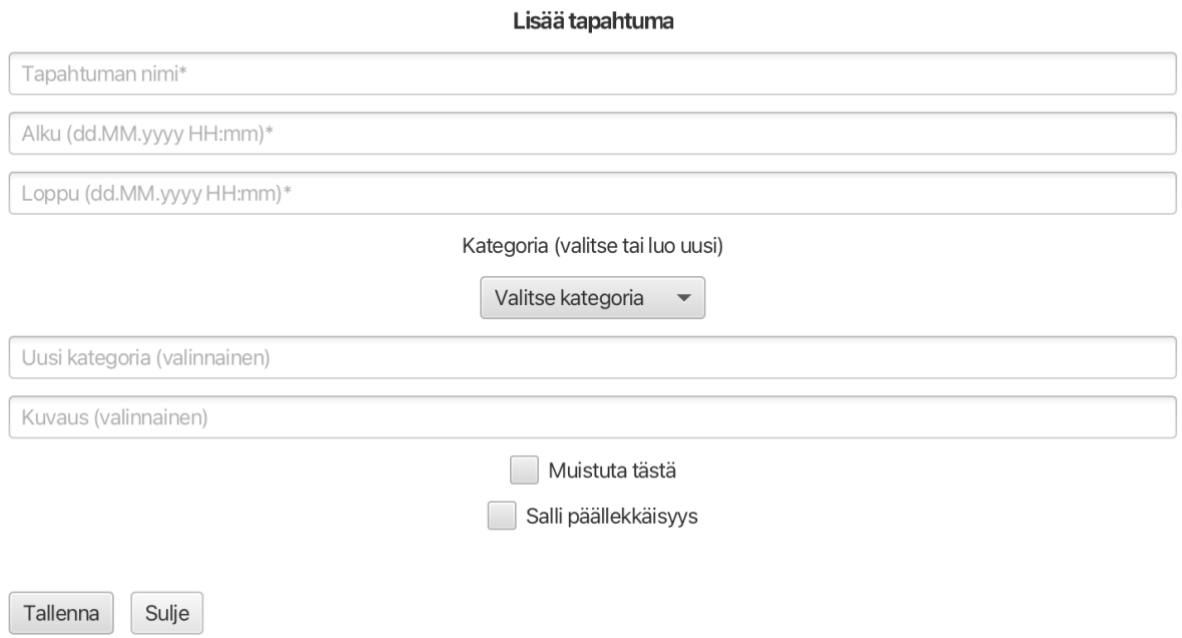
Sbt run -komennolla sovellus otetaan käyttöön. Tämä avaa oletusnäkymän joka on määritetty nyt kuukausinäkymäksi.

Navigointi

Kalenterin Yläpalkki

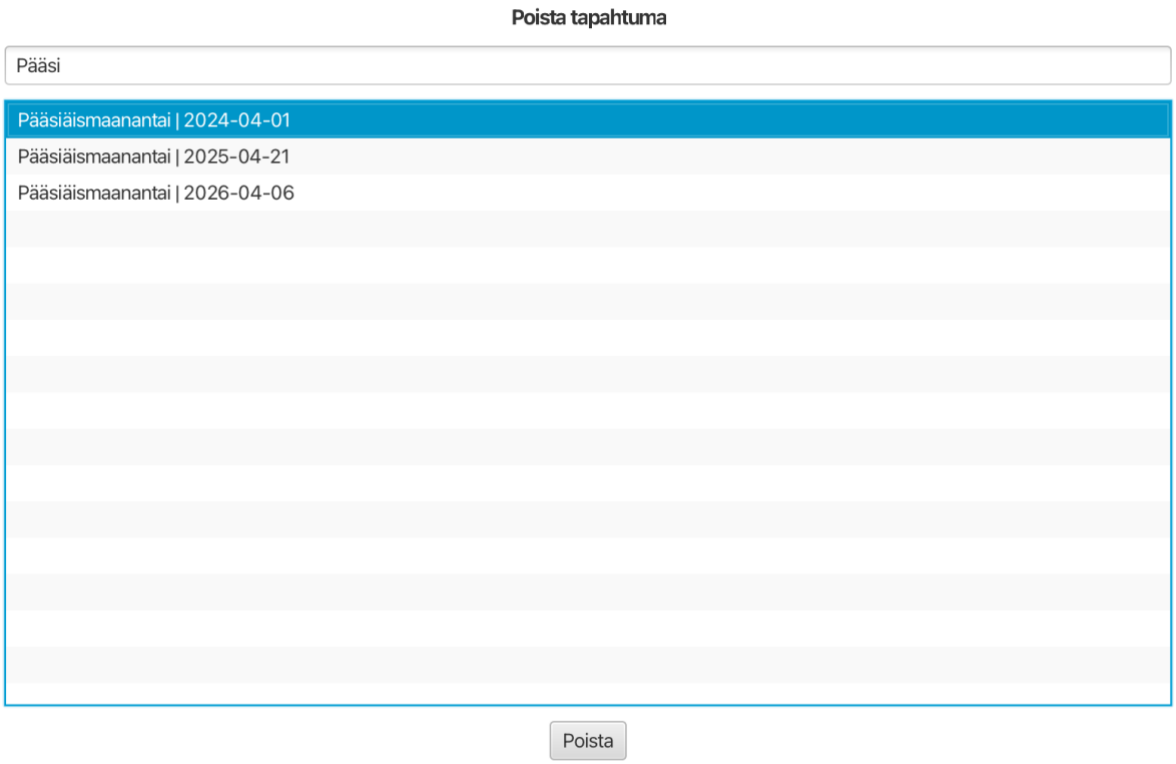
Navigaation käyttö on hyvin yksinkertainen ja intuitiivinen. Kalenterin toiminnallisuuksien navigoinnissa toimii yläpalkki. Klikkaamalla siirrytään näkymästä toiseen. Päivä, viikko ja kuukausi näkymistä pidetään statusta yllä. Esimerkiksi jos lisätään tapahtuma, niin tämän jälkeen automaattisesti siirrytään takaisin viimeksi käytettyyn päivä/viikko/kuukausi -näkymään.

Tapahtuman lisääminen



Tapahtuman lisääminen onnistuu valisemalla Lisää -näkymä ja täyttämällä lomakkeen pakolliset tiedot oikein (kolme ensimmäistä kenttää). Tapahtumalle voidaan määrittää, myös kategoria ja sekä mahdollisia lisätietoja. Muistutsta ja päällekkäisyyden sallimista seurataan chekboksilla true/false arvoin. Tapahtuma tallennetaan Tallenna napista, jolloin tapahtuma tallentuu kalenteriin pysyvästi ja se päivittyy välittömästi kaikkiin aikanäkymiin. Tallenna napista myös siirrytään edelliseen aikanäkymään. Tallennuksessa virheellisen syötteen antaminen ei onnistu, ja sovellus ilmoittaa tästä. Poistu napista siirrytään takaisin aikanäkymään tallentamatta tapahtumaa.

Tapahtuman poistaminen

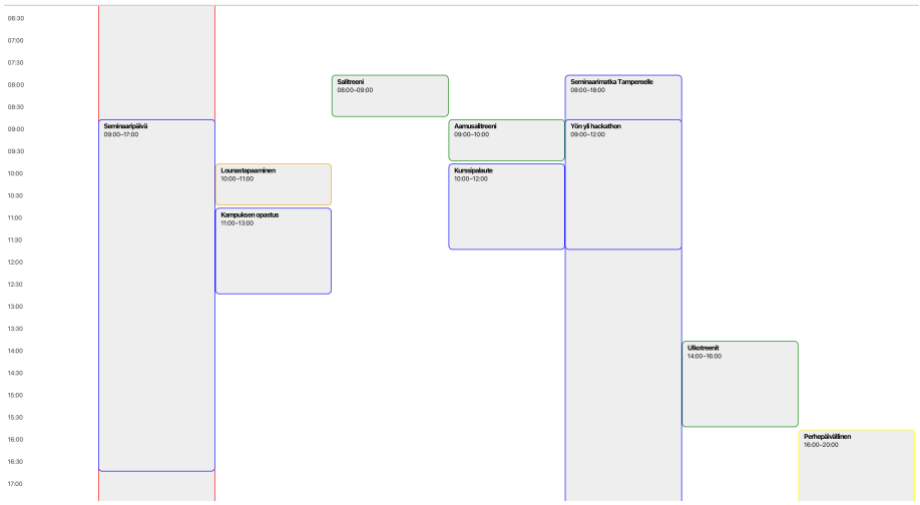


Tapahtumat voidaan poistaa pysyvästi poista ikkunasta. Valisemalla haluttu tapahtuma listasta. Poista ikkuna mahdollistaa myös tapahtumien selaamisen. Poista nappi aktivoituu kun tapahtuma valitaan klikkaamalla kentästä. Nappia painamalla tapahtuma poistuu pysyvästi kalenterista. Kalenteri päivittyy heti ja siirtyy edelliseen aikanäkymään.

Tapahtumien selaaminen

Tapahtumien selaaminen onnistuu kolmessa eri näkymässä. Päivä, viikko ja kuukausi. Kuukausi näkymä näyttää vain 3 ensimmäistä tapahtumaa päivälle kun taas viikko ja päivä näyttää kaikki tapahtumat, sekä värikoodaukset eri tapahtumatyypeille. Kaikissa näkymissä on mahdollista liikkua ajassa eteen tai taaksepäin navigaationapeista.

< Huhtikuu 2025 >

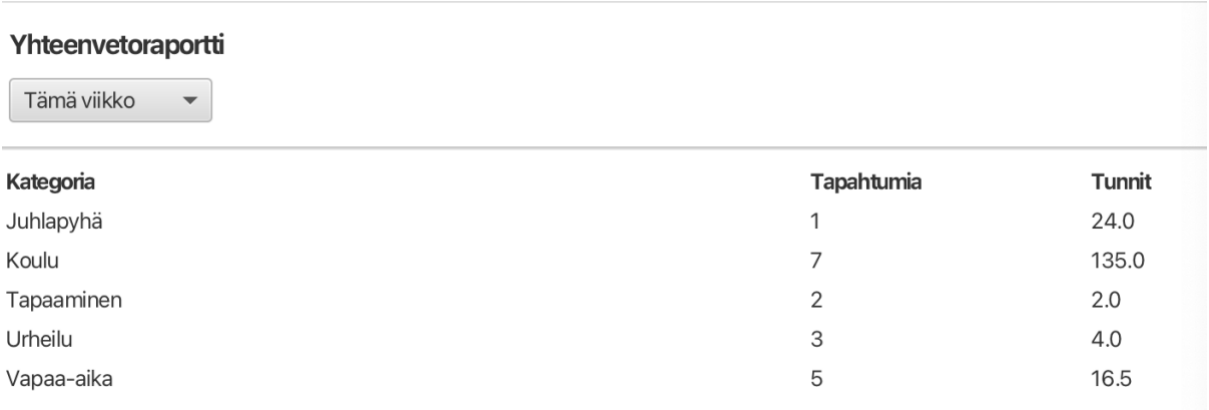


< MA 21.4.2025 >

< Huhtikuu 2025 >

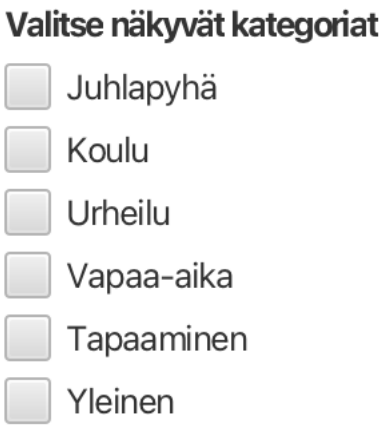
MA	TI	KE	TO	PE	LA	SU
	1	2	3	4	5	6
7	8	9	10	11	12	13
14 Luennot: Ohjelmoinnin per... Treenit: Salibandy Elokuvailta ystävien kanssa	15 Tapaaminen opettajan kan... Opiskeluryhmän työskentely Joogaharjoitus	16 Projekti: Ryhmätyön työstä... Kahvitreffit vanhan ystävä... Workshop: Tiedon visualis...	17 Aamulenkki Oppimistehtävän palautus Aamujumppa	18 Pitkäperjantai Verkkopalaveri harjoittelup... Piknik puistossa	19 Jalkapalloturnaus Illanviietto ystävien kanssa Jalkapallopeli	20 Koodiklinikka Discordissa testi Tapahtuma
21 Pääsiäismaanantai Opintoreissu museoon Oivan testi	22 Lounastapaaminen mentor... Oivan testi Lounastapaaminen	23 Lautapeli-ilta Salitreeni Lautapeli-ilta kaverin luona	24 Lautapeli-ilta Aamusalitreeni Lautapeli-ilta kaverin luona	25 Seminaarimatka Tamperee... Yön yli hackathon	26 Seminaarimatka Tamperee... Rentoutumista luonnossa Yön yli hackathon	27 Seminaarimatka Tamperee... Perhepäiväillinen
28 Seminaarimatka Tamperee... Seminaarin päätöspäivä	29	30				

Raporttien haku



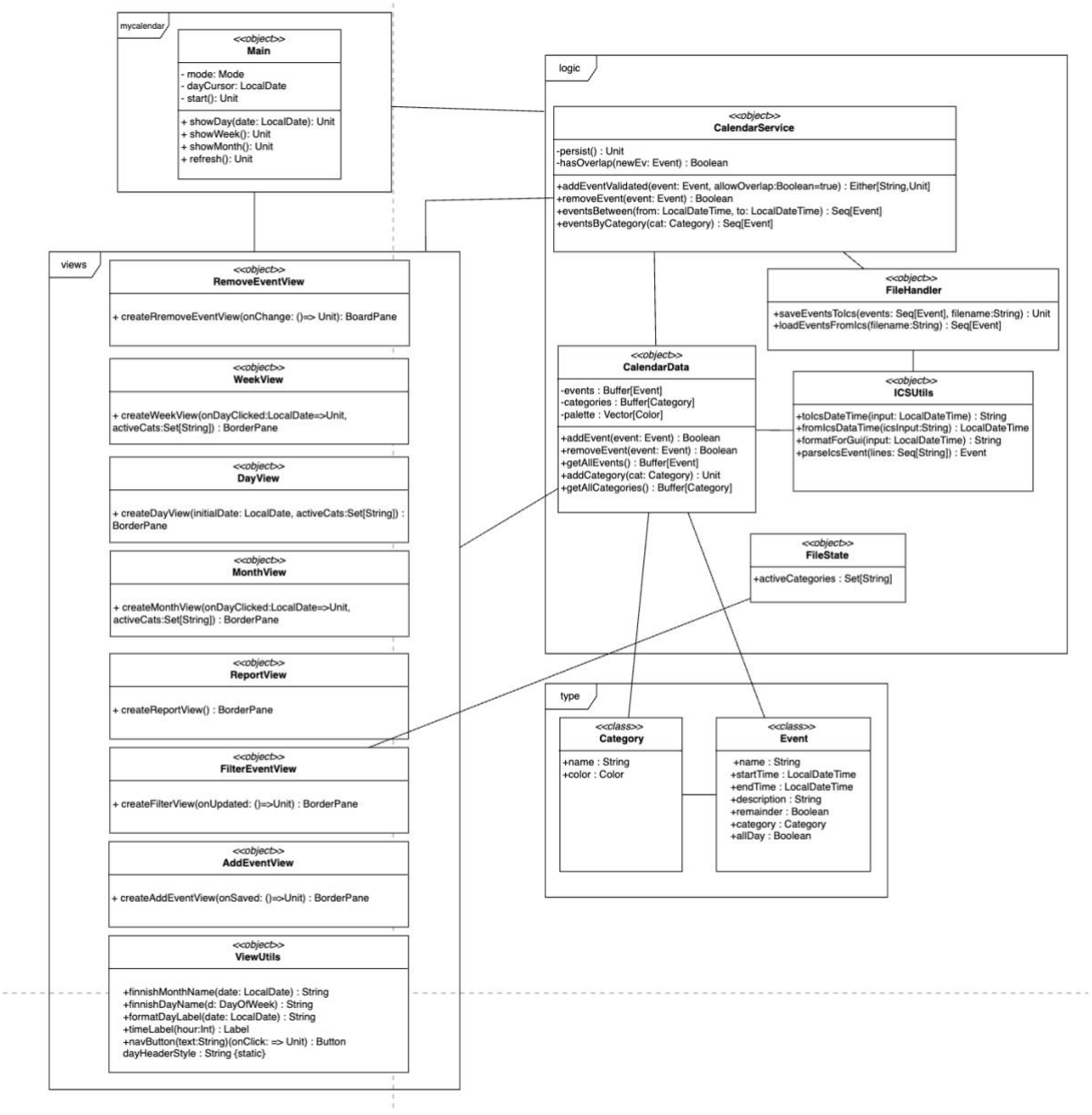
Tapahtumista voidaan saada enemmän tietoa raporttinäkymästä. Näkymässä on kategorioiden mukaan tapahtumien määrä ja tunnit. Tapahtumia voidaan ottaa mukaan nykyisen viikon ajalta, nykyisen kuukauden ajalta tai kaikki kalenterin tapahtumat.

Tapahtumien suodattaminen



Tapahtumien suodattaminen onnistuu suodata näkymästä. Jos mitään ei ole valittu, niin näytetään kaikki tapahtumat. Muuten valisemalla näytetään näkymissä kaikki valittujen kategorioiden tapahtumat. Klikkaus päivittää heti näkymät ja siirtyy edelliseen aikanäkymään. Eli jos haluaa valita useamman kategoria, joutuu mennä yläpalkista uudelleen valisemaan seuraavan kategorian.

Ohjelman rakenne



Tässä lyhyesti avattuna osakokonaisuuksien keskeisimmät tehtävät

Mycalendar:

Main käynnistää sovelluksen ja luo navigaation näkymien välillä siirtymiseen

Type:

Event kuvaa yksittäistä kelenterimerkintää

Category on tapahtuman luokitus ja määrittää värin

Logic:

CalendarData on yksinkertainen muistivarasto jonne tapahtumat ja kaategoriat kerätään

CalendarService toimii **CalendarDatan** päällä ja tarkistaa datan oikeellisuutta esim ettei loppu ole ennen alkua tai tapahtumalla ei ole päällekkäisyyttä

Filehandler tallettaa ja lukee ICS tiedostosta

Views:

WeekView, **DayView** ja **MonthView** näyttävät dataa omina näkyminä

AddEventView ja **RemoveEventView** tarjoaa lomakkeen lisäämiseen tai poistamiseen

FilterView pitää kirjaa käyttäjän valitsemista kategoriasuodatuksista yhdessä **FileState**:n kanssa

ReportView kokoaa raportin tapahtumiin käytetystä ajasta ja määrästä

Algoritmit

Sovelluksen algoritmit ovat käytännöllisyys edellä tehtyjä, enkä projektissa keskittynyt algoritmien tehokkuuteen pienen datamäärän vuoksi. Keskeiset algoritmit liittyvät erityisesti aikavälien käsittelyyn, suodattamiseen sekä käyttöliittymän päivittämiseen. Käyn läpi muutaman esimerkin sovelluksen erilaisista algoritmeistä.

Yksi keskeinen algoritmi littyy tapahtumien lisäämisen käsittelyy. Virheitä käsittelee funktio **addEventValidated**. Kun käyttäjä lisää uuden tapahtuman, niin ohjelma tarkistaa, että tapahtuman aloitusaika on ennen lopetusta. Jos ei synny virhettä niin tarkastetaan päällekkäiset tapahtumat jos päällekkäisyyttä ei sallita. Jos tästäkään ei synny virhettä tapahtuma lisätään kalenteriin ja kalenteri päivitetään.

Toinen olleellinen algoritmi liittyy tapahtumien näyttämiseen. Käyttöliittymässä tapahtumien näyttäminen pohjautuu ruudukkoihin. Päivä ja viikkonäkymät jaetaan ennalta aikaslotteihin. Päivä 15min väleille ja viikko 30min väleille. Tapahtumat sijoitetaan ajan perustella oikeaan kohtaan ja oikean pituisiksi. Tämä käsitellään **refresh** funktion sisältä löytyvästä for-loopista. Algoritmi selvittää tapahtuma alkamisen ja päättymisen. Sitten lasketaan aloitus ja lopeturuudut jonka jälkeen saadaan tapahtuman korkeus ruuduissa. Sitten tapahtuma lisätään uutena tapahtumaoliona oikeaan kohtaan kalenterin ruudukkoa.

Raporttinäkymässä käytetään ryhmittelyalgoritmia, joka kokoaa tapahtumat kategorioittain ja laskee näille tapahtuimen lukumäärän ja kokonaiskeston tunneissa. Tässä hyödynnetään groupBy-rakennetta. Kokonaiskesto lasketaan Duration.between(start, end) -metodilla joka muunnetaan vielä tunneiksi.

ICS-tiedostonkäsittelyssä hyödynnetään rivien tunnistamista. Lukualgoritmi kokoaa rivit BEGIN:VEVENT ja END:VEVENT välistä ja etsii tunnettujen kenttien kuten SUMMARY tai DTSTART/DTEND arvot rakentaen niistä Eventolion. Vastaava kirjoitusalgorimi käy taphtumat läpi ja muodostaa ne oikeaan .ics tiedostorakenteeseen.

Tietorakenteet

Kalenterisovelluksen tietorakenteet ovat kohtuullisen yksinkertaisia. Toisaalta käytännölisiä ja tehokkaita valitussa käytössä. Tapahtumat tallenetaan mutable.Buffer[Event] rakenteeseen ja sitä hallitaan CalendarDatassa. Hyvää tässä on nopea lisäys ja poista ja mahdollisuus säilyttää tapahtumat järjestyksessä. Myös käytössä olleet suodattimet kuten filter, group, find toimii hyvin. Koska oletuksena on että tapahtumamäärä ei ole kovin suuri tehokkaammille tietorakenteille ei ole tarvetta. Tapahtumia käsitellään monipuolisesti koodissa, sillä lähes kaikki ominaisuudet, jossain määrin nojaavat kalenterin tapahtumiin. Samaan tappaan myös kategoria on tallennettu mutable.Buffer[Category] rakenteeseen. Kategorialle generoidaan automaattisesti väri valmiista värialetista lisäyksen yhteydessä.

Myös suodatettavista näkymistä pidetään kirjaa tietorakenteessa joka on Set[String] muodossa. Tätä tietorakennetta hyödynnetään suodatuksessa, kun halutaan määrittää mitkä kaikki kategoriaryhmien tapahtumat näytetään.

Tiedostot ja verkossa oleva tieto

Kalenterisovelluksen keskeinen ominaisuus on se, että kaikki tapahtumat ja kategoriat säilyvät pysyvästi tiedostossa, vaikka ohjelma sammutetaan. Tätä varten käytetään iCalendar tiedostomuotoa, joka mahdollistaa myös yhteensopivuuden muiden yleisten kalenterisovellusten kanssa.

Aluksi käsittelin useampaa tiedostoa samaan aikaan resources kansiossa Tämä oli hyvä ominaisuus jos uusia kalentereita halusi tuoda lisää sovellukseen. Kuitenkin tapahtumien poistaminen ja tallentaminen on huomattavasti yksinkertaisempaa toteuttaa yhden kalenterin sisältä, joten liitin samaan tiedostoon kaikki tapahtumat. Käytännössä tämä ei kuitenkaan rajoita paljon tapahtumien lisäämistä muista kalenterista. Riittää, että VEVENT oliot kopioi userEvents.ics tiedoston loppuun. Tällöin ne luetaan käynnistyksen yhteydessä. Käyttöliittymän puolella ei kuitenkaan ole mahdollisuutta lisätä uusia kalentereita.

Kalenterin testidatasta esimerkki:

```
BEGIN:VEVENT
UID:50
DTSTART:20250428T090000
DTEND:20250428T170000
SUMMARY:Seminaarin päätöspäivä
DESCRIPTION:Esityksiä ja yhteenvetoja
CATEGORIES:Koulu
END:VEVENT
```

Kalenteriin helpoin tapa lisätä uutta dataa on käyttöliittymästä. Kalenterissa on jo tällähetkellä hyvä määrä testidataa tehtynä valmiiksi. Lisäksi tapahtumista löytyy suomalaisen kalenterin juhlapyhät valmiina.

Ohjelmalla ei tällä hetkellä ole mitään verkkoyhteyksiä, eikä se hae tietoa ulkopuolisista palveluista. Kaikki tallennus tapahtuu paikallisesti käyttäjän konella.

Testaus

Toteutin ohjelman testausta iteratiiviesti ohelmoinnin rinnalla. Testaus sisälsi vähän yksikkötestausta sekä manuaalista käyttöliittymän testausta. Tavoitteena oli pitää huoli, että kaikki keskeiset toiminnallisuudet toimivat odotetusti oikella että virheellisillä syötteillä.

Alkuperäisessä suunntelmassa ajattelin tekeväni enemmän yksikkötestausta kriittisille metodeille, kuten addEvent ja saveData. Käytännössä yksikkötestaus jäi pääosin omaan testailuun REPL-ympäristössä, sekä muutamaan manuaalisesti ajettavaan testiluokkaan, jotka eivät päätynyt lopulliseen palautukseen. Tämä on projektissa pieni heikkous. Testaus oli projektissa tarkoitus aina tehdä, lopulta kuitenkin kun tein projektin yksin tunsin projektin kohtuullisen hyvin ja pystyin paikantamaan virheet riittävän nopeasti, joko komentoriviltä tai ilman testaamista. Tämä lähestyminen ei tietenkään toimi kun tehdään tätäkin suurempia projekteja isommassa kehitystiimissä.

Testaus lopulta painottui käyttöliittymäpuolelle. Eli testaus toimii nyt enemmän käyttäjän näkökulmasta. Esimerkiksi tapahtuman lisäämisessä on käsitelty useita virhetilanteita ja käyttäjä saa virheellisistä syötteistä palautetta.

Virhesyötteet tapahtuman lisäämiselle ovat:

“Lopetusaika ei voi olla ennen aloitusaikaa”

“Tapatuma menee päälleikkäin olemassa olevan varauksen kanssa”

“Loppu ennen alkua”

“Aikojen oltava 15min välein”

“Virheellinen syöte”

Virhetekstien perusteella käyttäjän on helppo päätellä mikä virhe syötteessä on. Tämä myös estää virheellisen tapahtuman lisäämisen kalenteriin. Myös pientä virheentestausta tehdään ICS tiedoston lukemisen yhteydessä, jos aloitusaika puuttuu. Tämä on tehty hyödyntämällä Option rakennetta.

Ohjelman tunnetut puutteet ja viat

Kalenerisovelluksen kriteereistä jätettiin muutama toteuttamatta kokonaan:

- Vaikeista kriteereistä toistuvia tapahtumia ei tehty ollenkaan.
- Muistutukselle pystyy tallentamaan true/false arvon, mutta muistutusta ei käsitellä tämän enempää ohelmassa, eli käytännössä muistutusten toteutus puuttuu kokonaan.
- Keskivaikeista kriteereistä maalaamalla tapahtuman lisäämistä ei tehty ollenkaan. Eli kaikki lisääminen täytyy tehdä lisää tapahtuma lomakkeesta.

Muut puutteet:

- Jossainmäärin ICS yhteensopivuus on puutteellinen eli kaikkea mahdollista tietoa ei pystytä lukemaan kalenteriin.
- Tapahtumien tunnistaminen perustuu nimeen. On kuitenkin mahdollista että tapahtumia on useita samalla nimelle, esim jos tapahtumia nimeää tyylillä “Luento”
- Tapahtumia lisäämällä samalle ajalle aikaisemmin lisätyt piiloutuu vanhojen taakse. Tässä voi hyödyntää kuitenkin suodattamista tai vaihtoehtoisesti välttää päällekkäisen lisäämisen

3 parasta ja 3 heikointa kohtaa

Parhaat

1. Visuaalisesti selkeä ja helppokäyttöinen käyttöliittymä

Ohelman käyttöiittymään käytin paljon aikaa ja tein sen huolella. Sovellus on minimalistisella teemalla yhtenäinen kaikissa näkymissä ja tapahtumat on selkeästi lisätty kalenteriin. Värikoodaus on selkeä ja fontit yms muu tyyli on valittu järkevästi. Tässä auttoi aikaisempi osaaminen CSS kielestä ja nettisivujen suunnittelusta.

2. Skaalautuvuus eri näyttökoihin

Vaikka tämä ei ollut erityisesti tehtävän vaatimuksena kalenterisovellusta on helppo käyttää eri kokoisissa ikkunoissa. Tämä varmistetaan siten, että horisontaalinen

asteikko skaalataan prosenttien mukaan ja vertikaaliin mahhdollistetaan skrollaus ominaisuus.

3. Laajennettava ja selkeä arkkitehtuuri

Koodi on jaettu loogisiin kokonaisuuksiin. Logiikka löytyy omasta paketista ja samoin näkymät muodstavat yhdessä selkeän kokonaisuuden. Uusien ominaisuuksien lisääminen onnistuu kohtuullisen helposti rikkomatta muuta koodia.

Heikoimmat

1. Puutteellinen eventin käsittely

Muistutuksia ei käsitellä kunnolla projektissa vaikka niiden tekemistä on aloitettu. Samoin esimerkiksi allday, jäi hieman turhaksi tiedoksi tallentaa. Alkuperäinen suunnitelma tälle oli tehdä kokopäivätapahtumille oma sarake, jossa tämä olisi ollut hyödyllisempää. Näiden lisäksi myös esimerkiksi lisätietoja ei käsitellä kalenterissa muuten kuin keräämällä se käyttäjältä. Lisätietoa ei voi nähdä kalenterissa mistään, vaikka säilytetään ISC tiedostossa tallessa.

2. Rajoittuneet ICS tiedostot

Tiedoston luku onnistuu nyt vain tavallisimmille rakenteille. Uuden kalenteritiedoston lisäämisen yhteydessä osa tiedoista poistuu käynnistyksen yhteydessä koska kaikkea tietoa ei osata uudelleen kirjoittaa.

3. Testaus

Projektissa ei ole erisllisiä testiluokkia tai yksikkötestejä, vaan kaikki testaus toteutettiin käsin tai käyttöliittymänkautta. Testien puute ei vaikuta projektin toimivuuteen, mutta jatkokehityksen kannalta ne olisi tärkeä lisä.

Poikkeamat suunnitelmasta, totetunut työjärjestys ja aikataulu

Alkuperäisessä suunnitelmassa tavoitteeni oli toteuttaa projekti keskivaikean ja vaikean välimaastosta. Mielestäni pääsin tähän tavoitteeseeni. Etenin myös suunnitelemissani välivaiheissa. Eli ensin tein loogiset toiminnallisuudet valmiiksi, jonka jälkeen siirryin tekemään käyttöliittymää. Työmääräksi arvioin noin 100 tuntia ja karkeasti arvioituna tämä määrä ylittyi hieman. Suurimman työn tein projektin lopussa pääsiäisloman aikana, jolloin viimeisellä viikolla tunteja tuli noin 50.

Alkuperäinen aikataulutus ei täysin toteutunut, mutta projekti mielestäni eteni järkevästi aina sprinttien välillä. Toisinaan committien tekemminen hieman kesti

todellisesta koodaamisajankohdasta, mutta tämä ei nyt varmaan suuri virhe ollut, sillä projekti oli yksilötehtävä.

Kokonaisarvio lopputuloksesta

Olen toedella tyytyväinen lopputulokseen. Lopputuloksena on toimiva ja graafisesti miellyttävä kalenteri. Opin projektissa lisää käyttöliittymän rakentamisesta ScalaFX:n avulla, vaikka osittain tämä oli jo tuttua. Hyvä rakenteen suunnittelu auttoi selvästi projektissa ja suuria muutoksia rakenteeseen ei tarvinnutkaan tehdä projektin aikana.

Viitteet ja muualta otettu koodi

Käytetyt lähteet

Scala ja ScalaFX

https://docs.scala-lang.org

https://scalafx.org/docs/home/

CSS tyylit

https://docs.oracle.com/javafx/2/css_tutorial/jfxpub-css_tutorial.htm

Java.time

https://docs.oracle.com/javase/8/docs/api/java/time/package-summary.html

https://www.baeldung.com/scala/date-time

iCalendar

https://icalendar.org

https://datatracker.ietf.org/doc/html/rfc5545#section-8.3.2

Näiden lisäksi tietysi A+ matriaalit ja tehdyt tehtävät valmistivat projektiin

Tekoälyn käyttö

Tein lopussa testdataa kalenteriin chatGPT:n avulla. Eli userEvents.ics tiedoston osa tapahtumista on tehty chatin avulla hieman säästääkseni aikaa ja samalla saada kalenteriin täytettä, jotta ei olisi aivan tyhjä.

Liitteet

Kuvat käyttöliittymästä osana jo selityksiä