Arttu Pennanen, Henri Vuento, Atte Räisänen, Perttu Vaarala

ImpulsePass toteutusdokumentti

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintäteknologian tutkinto-ohjelma

Toteutusdokumentti

5.5.2023

Sisällys

1 Johdanto

ImpulsePass on tapahtumalippujen ostamista helpottava sovellus, joka tuo kaikkien tapahtuman järjestäjien liput yhteen paikkaan. Sovellus noutaa tapahtumat eri myyjiltä ja esittää tuo ne yhteen paikkaan se myös esittää dataa kyseisiin tapahtumiin liittyen. Sovellus toimii apuvälineenä aikataulutukseen sekä siivittää käyttäjä tekemään tietoon perustuvia arvauksia tapahtumista tulevaisuudessa.

Sovelluksen aiempi nimi oli Kide.app analyysivehje, kuitenkin ohjelmistoprojekti 2 alkupuolella saimme Englannin kielen opettajalta idean kääntää sovelluksen suunta. Käännöksessä suljettiin idea siitä, että sovellus perustuisi yksinomaan Kide.app palveluntarjoajan palveluun, ja että keskittymispiste olisi sellaisissa tapahtumissa, jotka jäävät myymättä.

Sovelluksen toisena tavoitteena on siis nykyään auttaa käyttäjää pitämään kirjaa mielenkiintoisista tapahtumista ja auttaa muistuttaa niiden alkamisesta. Tämän kaiken lisäksi sovellus tarjoaa muille lipunmyynti yrityksille mahdollisuuden mainostaa juuri alkavia tapahtumia, joita ei vielä myyty loppuun

Oppimismielessä projekti kehittää projektin jäsenten ymmärrystä datan aggregoinnista. Sovellus on JavaFX -työpöytäsovellus.

Tässä dokumentaatiossa hyötykäytetään Ohjelmistotuotantoprojekti 1 kurssilla toteutettua sovelluksen dokumentaatioa, sillä sovelluksen arkkitehtuuri ei muuttunut merkittävästi Ohjelmistotuotantoprojekti 2 kurssin aikana.

2 Tuotteen vaatimukset

Tuote tarjoaa käyttäjille tavan analysoida ympärillä tapahtuvaa toimintaa. Sovellus tarjoaa paikan oppia kiinnostuksen lähteistä. Tuotteen omistajana toimii tuotteen kehittäjät.

2.1 Vaatimukset

Tuotteelle on asetettu vaatimuksia, jotta sen valmistuessa se tarjoaa käyttäjilleen arvokkaita kokemuksia.

Tuotteen on pystyttävä tallentamaan palveluntarjoajalta löytyvää dataa, jotta tuote pystyy muodostamaan historiaa tiedostoa, jota ei pidetä loputtomiin palveluntarjoajan palvelusta saatavilla.

Tuotteen on toimittava ketterästi, jotta käyttökokemus ei kärsi. Tuotteen on näytettävä edustavalta, ja pyrkiä olemaan tarpeeksi näyttävä ja asianmukainen.

Tuotteen on pystyttävä tallentamaan käyttäjän valintoja, jotta käyttäjä pystyy palaamaan tarkastelemaan käyttäjää kiinnostavien tapahtumien muutoksia.

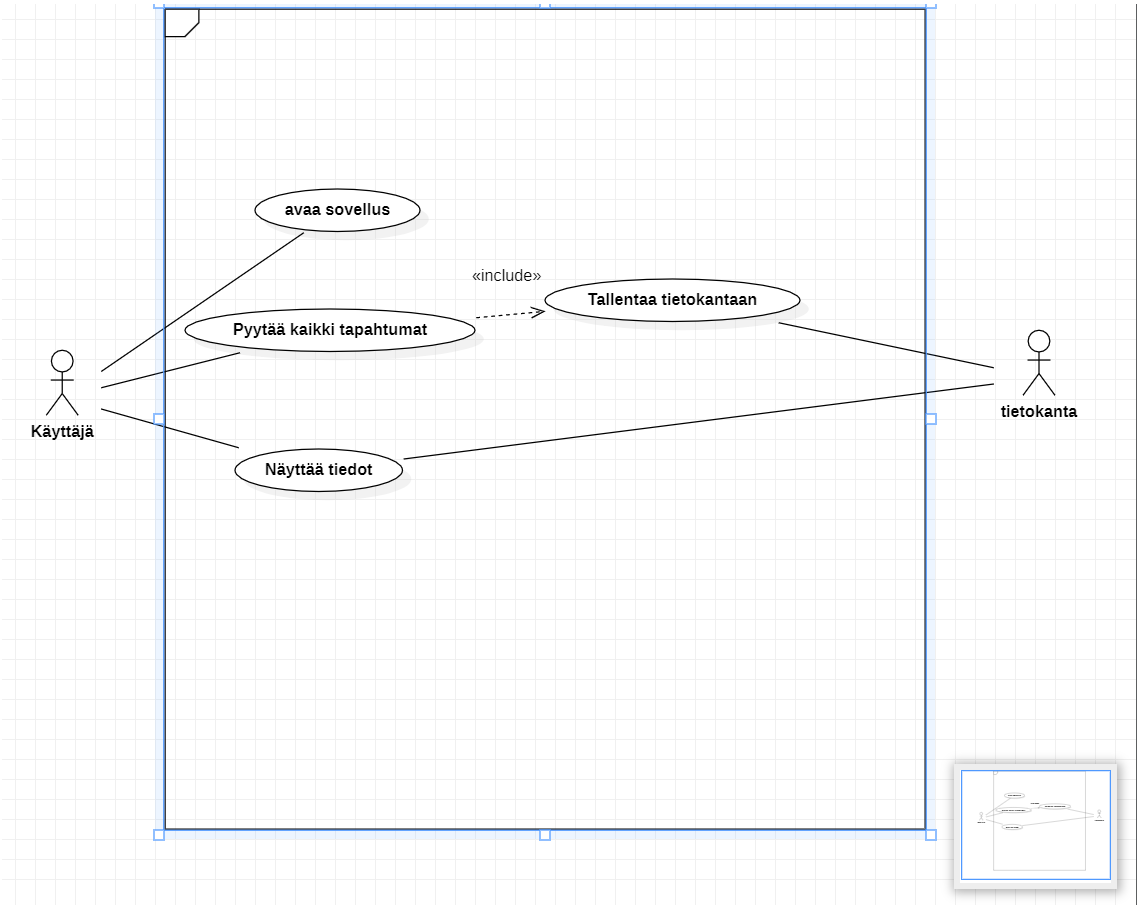
2.2 Käsitteet ja määritelmät

Tapahtuma
tapahtumalla kuvataan yksittäistä tapahtumantarjoajan tapahtumaa tiettynä ajanhetkenä. Yksittäinen tapahtuma voi ilmaantua tuotteen kaavioissa ja tietueissa useaan otteeseen, sillä tuote tallentaa tapahtumista tilannevedoksia (en. snapshot).

Tapahtumadatapiste
tapahtumadatapisteellä tarkoitetaan jonkin ajanhetken kokonaistilannevedosta palveluntarjoajan palvelun datasta. Yksittäinen datapiste sisältää kaikki sinä hetkenä tapahtumantarjoajan palvelusta löytyvät erikseen määritettyihin suodattimiin vastaavat tapahtumat.

3 Käyttäjäroolit ja käyttötapaukset

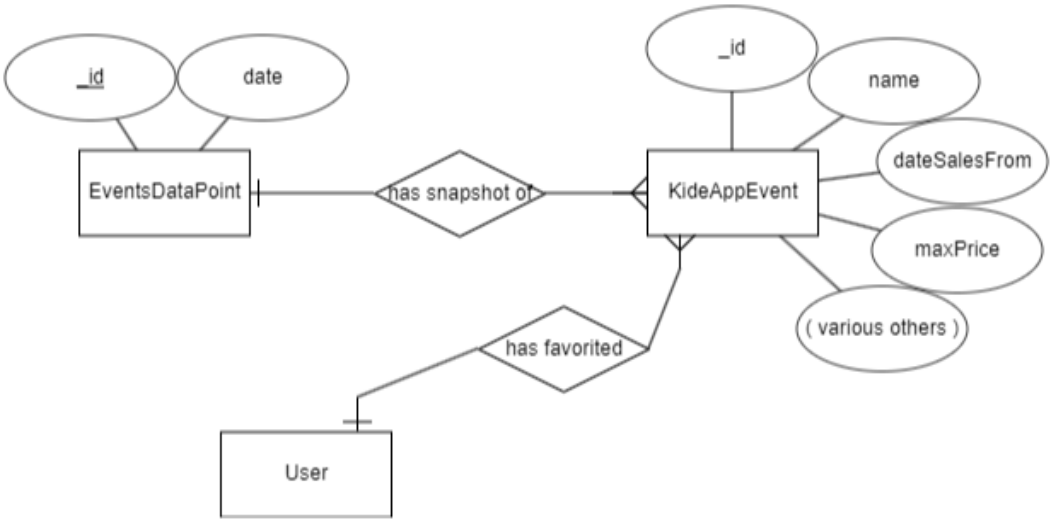
Sovelluksen käyttäjäroolit ovat kahdenlaiset käyttäjä, joka on kuka tahansa tapahtumia etsivä käyttäjä, luultavasti opiskelija, joka hakee tietoa tulevista tapahtumista ja lipuista koska haluaa itse osallistua kyseisiin tapahtumiin. Toisena roolina on tapatumien järjestäjät, joita taas kiinnostaa enemmän sovelluksen tarjoama statistiikka sitä, milloin tapatumia on eniten ja miten suosittuja ne ovat, jotta he voivat itse optimoida omien tapahtumien järjestämisen



Kuva 1 Käyttötapauskaavio, jossa käyttäjä hakee kaikki tapahtumat palveluntarjoajan palvelusta. Tapahtumat tallennetaan omaan tietokantaan

4 Ohjelmiston tietomalli

Ohjelmiston pääosaisena tietokantana toimii dokumenttikanta MongoDB. Dokumenttikanta valittiin helpottamaan tietojen uudelleentallentamista.



Kuva 2 Korkean tason ER-kaavio. Palveluntarjoajan tapahtumaa kuvataan KideAppEventtinä, sillä se on tapahtumamalli, jonka saimme sovitettua projektin tuotantoprosessiin.

Käytännössä tietokannasta haetaan tapahtumadatapisteitä (EventsDataPoint), jotka sisältävät aiemmin palveluntarjoajan sovelluksen omilta palvelimilta saatua tietoa.

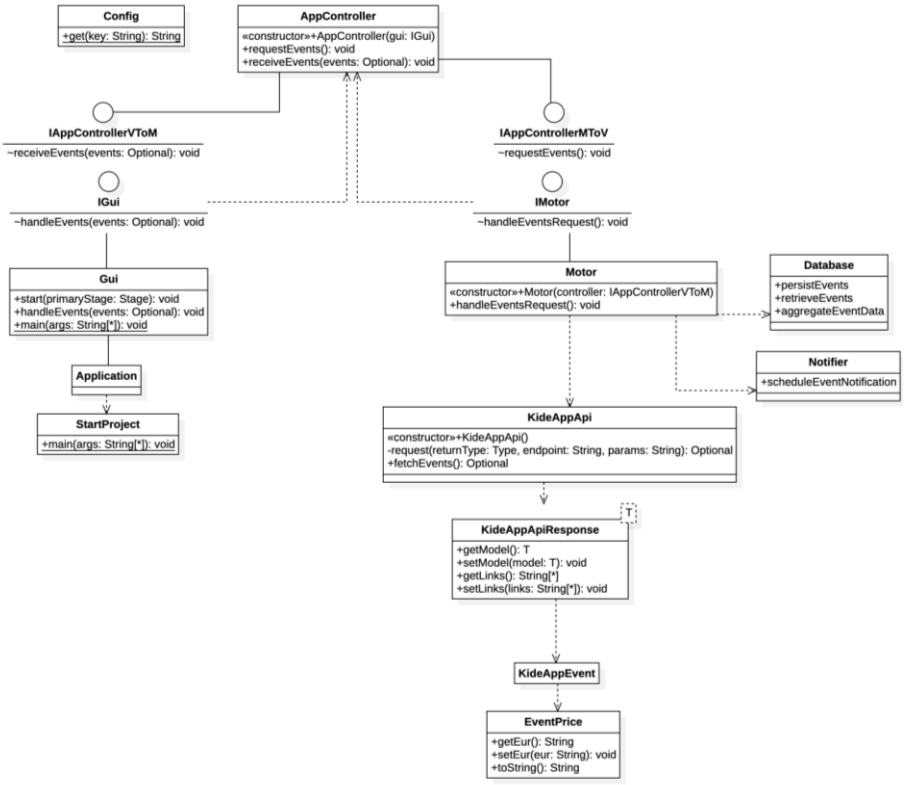
Sovelluksen käyttäjä voi tallentaa yksittäisiä tapahtumia omaan ”My Events” osioon.

Yksittäisen palveluntarjoajan sisältää monia kymmeniä eri kenttiä. ER-kaaviossa näitä ei ole mallinnettu (Kuva 2). Tapahtuman tarjoajan event entiteetin id kenttä on uniikki, mutta yhdestä tapahtuman tarjoajan tapahtumasta voi olla sovelluksessa useita eri versioita.

5 Ohjelmiston rakenne

Ohjelmisto on kehitetty käyttäen uudelleenmukailtua MVC-arkkitehtuuria (model, view, controller). Ohjelmistossa on kuitenkin otettu vapauksia MVC-arkkitehtuurin rajoihin liittyen sovelluskehityksen nopeuden kasvattamiseksi.

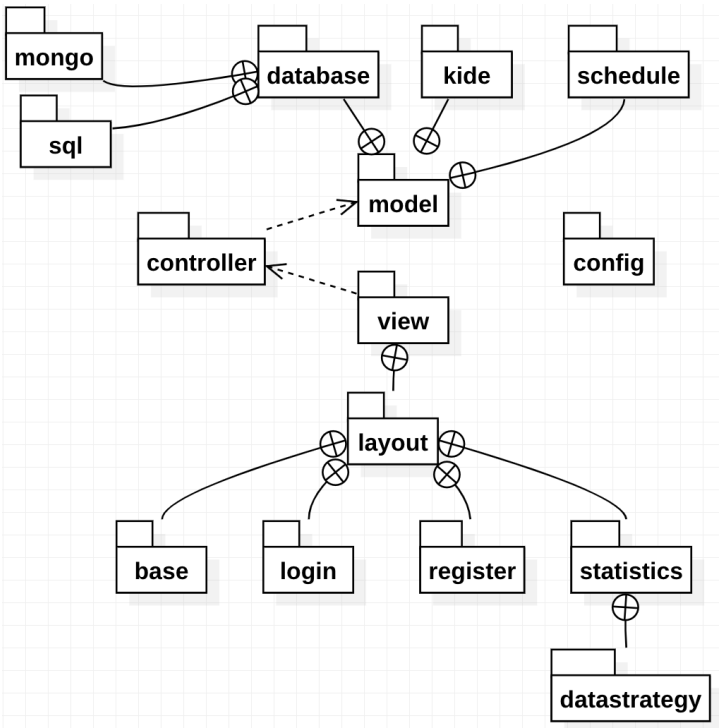
5.1 Luokkakaavio



Kuva 3 korkean tason luokkakaavio

Tieto liikkuu käyttöliittymästä alaspäin. Käyttöliittymässä tehdään pyyntö, controlleri delegoi pyynnön ja lähettää käyttöliittymälle vastauksen. Paikoitellen käyttöliittymä ohittaa controllerin. Näin on tehty sovelluskehityksen nopeuttamiseksi. (Kuva 3.)

5.2 Pakkauskaavio



Kuva 4 Ohjelmiston pakkauskaavio. Ohjelmisto noudattaa MVC-arkkitehtuuria.

Ohjelmisto noudattaa MVC (Model-View-Controller) -arkkitehtuurin periaatteita. Ohjelmisto on siis jaettu kolmeen pääpakkaukseen: Model, Controller ja View. Näiden lisäksi neljäntenä pakkauksena toimii Config, jota mikä tahansa pakkauksen komponentti voi tarvittaessa hyödyntää. Config-pakkaus tarjoaa ympäristömuuttujia, kuten tietokantojen osoitteita, joiden avulla ohjelmisto voi mukautua eri ympäristöihin ja konfiguraatioihin.

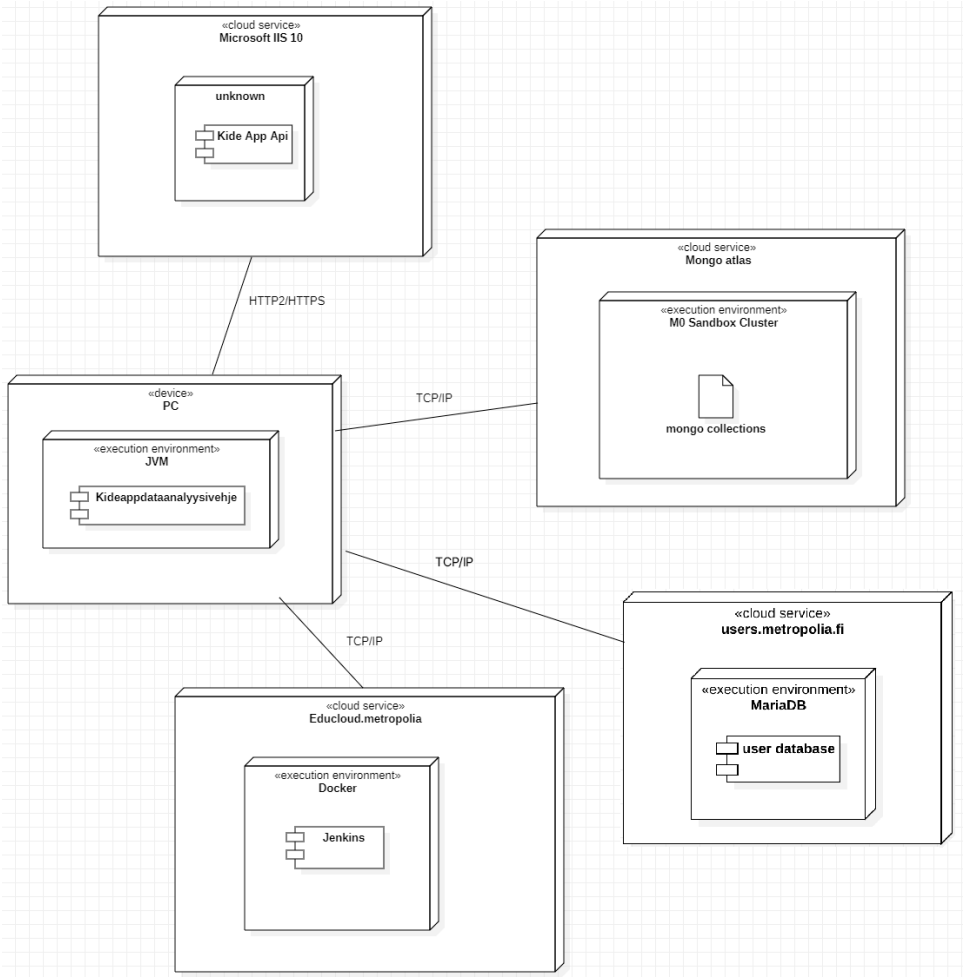
Model-pakkaus sisältää sovelluksen liiketoimintalogiikan ja tietojen käsittelyyn liittyvät komponentit. Tämä voi sisältää esimerkiksi tietokantayhteyden hallinnan, tietomallit ja niiden käsittelyyn liittyvät toiminnot.

Controller-pakkaus vastaa sovelluksen toiminnallisuuksien ohjaamisesta ja käyttäjän vuorovaikutuksesta. Se vastaanottaa käyttäjän syötteet ja välittää ne

sopiville Model-komponenteille. Controller myös käsittelee tietomallien päivitykset ja viestittää ne View-komponenteille näytettäväksi.

View-pakkaus huolehtii käyttöliittymän esittämisestä ja käyttäjän kanssa tapahtuvasta kommunikaatiosta. Se vastaa tietojen esittämisestä käyttäjälle ja käyttäjän syötteiden vastaanottamisesta. View-komponentit voivat hyödyntää Controllerin tarjoamia tietoja ja toiminnallisuuksia käyttöliittymän päivittämiseksi.

Yhdessä nämä pakkausryhmät muodostavat kokonaisvaltaisen rakenteen, joka helpottaa ohjelmiston ylläpitoa, laajentamista ja testaamista. MVC-arkkitehtuurin avulla eri vastuualueet eriytetään selkeästi, mikä edistää modulaarisuutta ja uudelleenkäytettävyyttä ohjelmiston kehityksessä.



Kuva 5 Tuotantosovelluksen sijoittelukaavio

Sovellus ajetaan aina omalla päätelaitteella. Sovellus voidaan ajaa millä tahansa JVM ja JavaFX teknologioiota tukevaan ympäristöön.

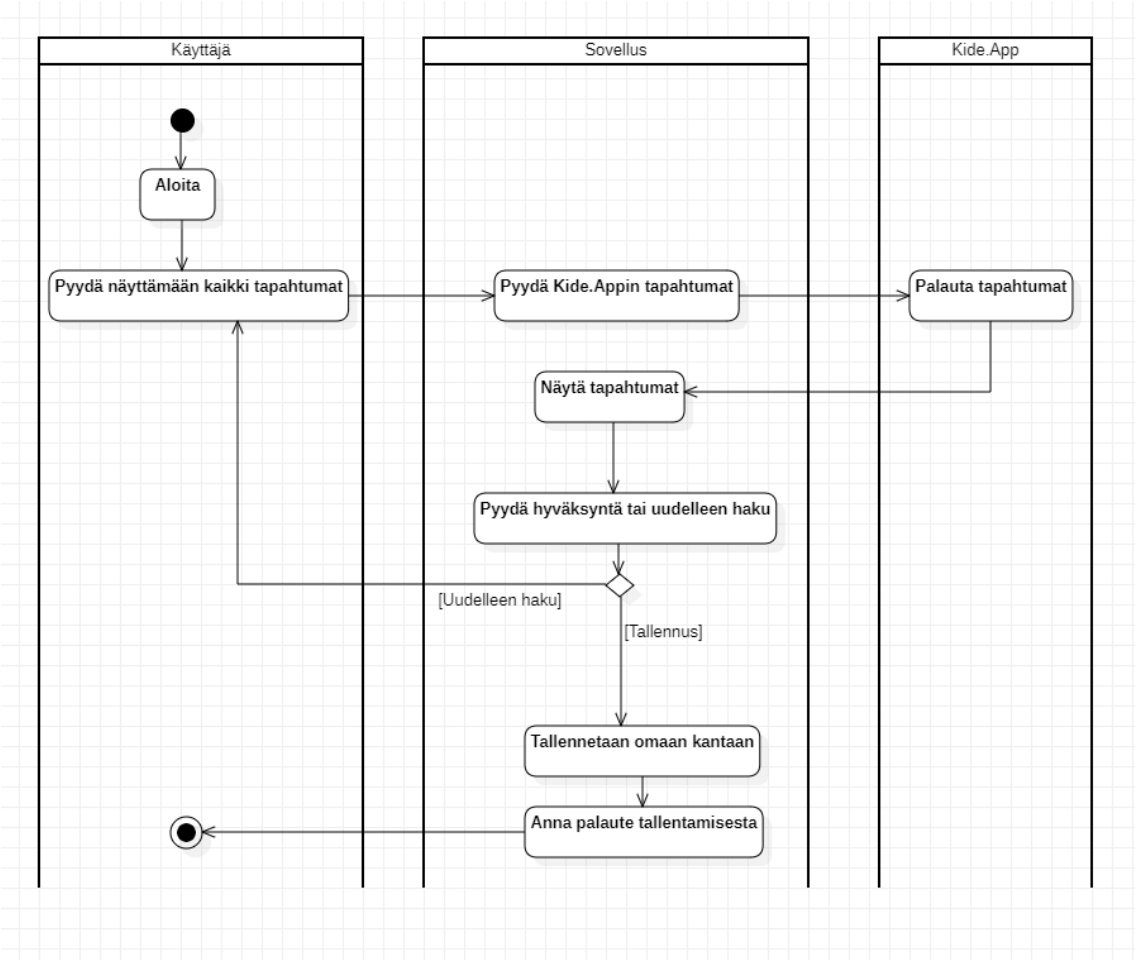
Tuotantosovellus on käytön aikana yhteydessä kolmeen eri palvelimeen. Mongo dokumenttitietokantaan, SQL relaatiotietokantaan sekä palveluntarjoajien palvelun http rajapintaan (Kuva 5).

Tuotantosovelluksesta uusia versioita tarkastaessa otetaan myös yhteys Metropolian palvelimilla olevaan Jenkins instanssiin, joka tarkastaa, että sovelluksen testit menevät lävitse.

6 Ohjelmiston toiminta

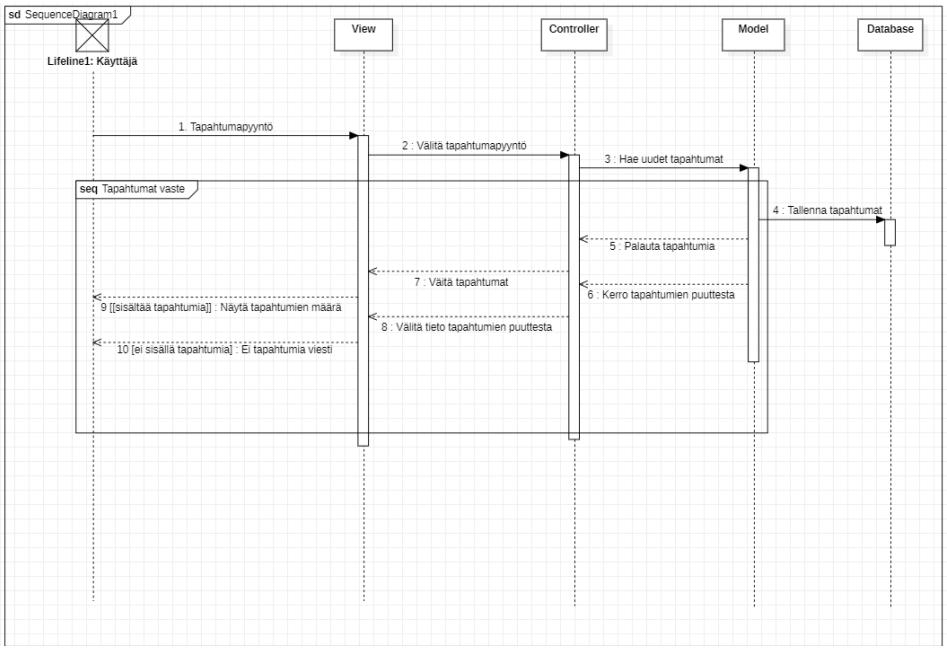
6.1 Käyttötapaus

Ohjelmiston toimintaa käsitellessä raporttiin on valittu yksi käyttötapaus, jolla saa yleisen kuvan siitä, miten tuote toimii, ja miten sitä käytetään.



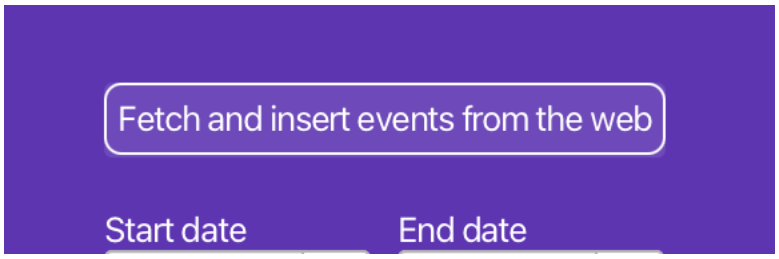
Kuva 6 Aktiviteettikaavio käyttötapaukselle, jossa käyttäjä tahtoo persistoida uusimmat tiedot tietokantaan

Tarkastelunkohteeksi on valittu käyttötapaus, jossa käyttäjä tahtoo tallentaa sillä hetkellä Kide.app palvelusta löytyvät tapahtumat tuotteen omaan tietokantaan (Kuva 6).



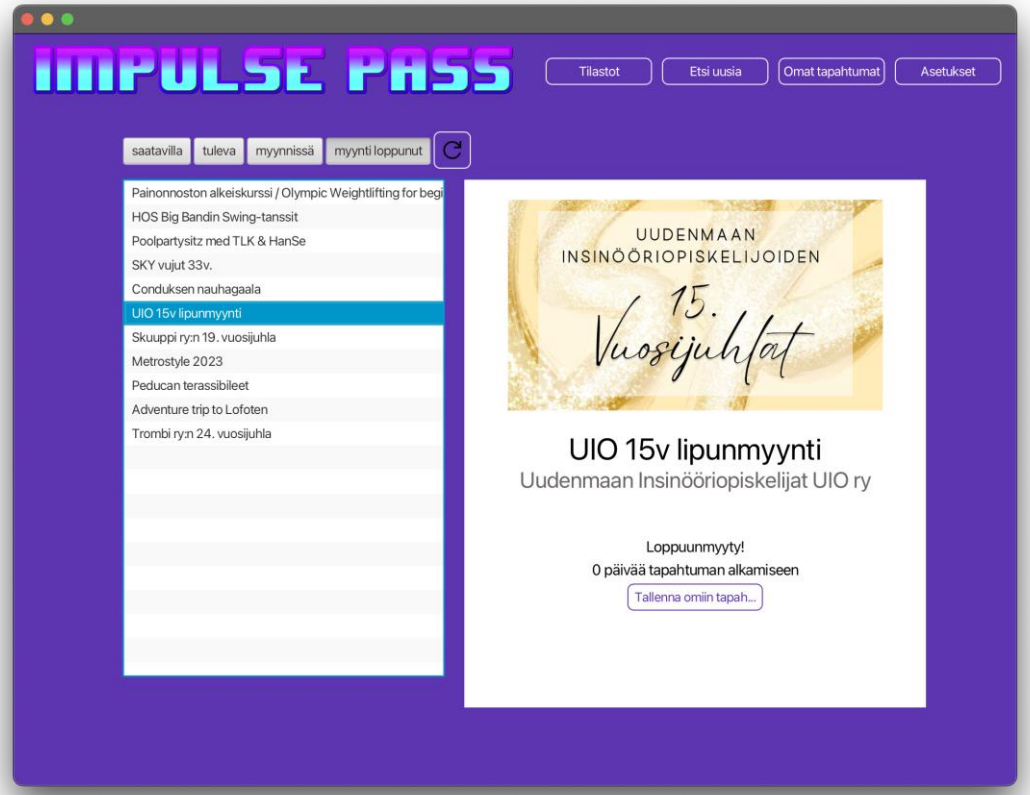
Kuva 7 Sekvenssikaavio samalle käyttötapaukselle kuin kuvassa 6

Yksi käyttötapaus johtaa pyyntöjä useaan eri palveluun. Siksipä on kehittämisen kannalta olennaista, että eri palvelut toimivat tuotteessa harmonisesti.



Kuva 8 Käyttötapauksen mahdollistava nappi käyttöliittymässä

Käyttötapauskokemus optimoitiin mahdollisimman pitkälle ohjelmiston kehityksen aikana. Yksi napinpainallus riittää (Kuva 8).

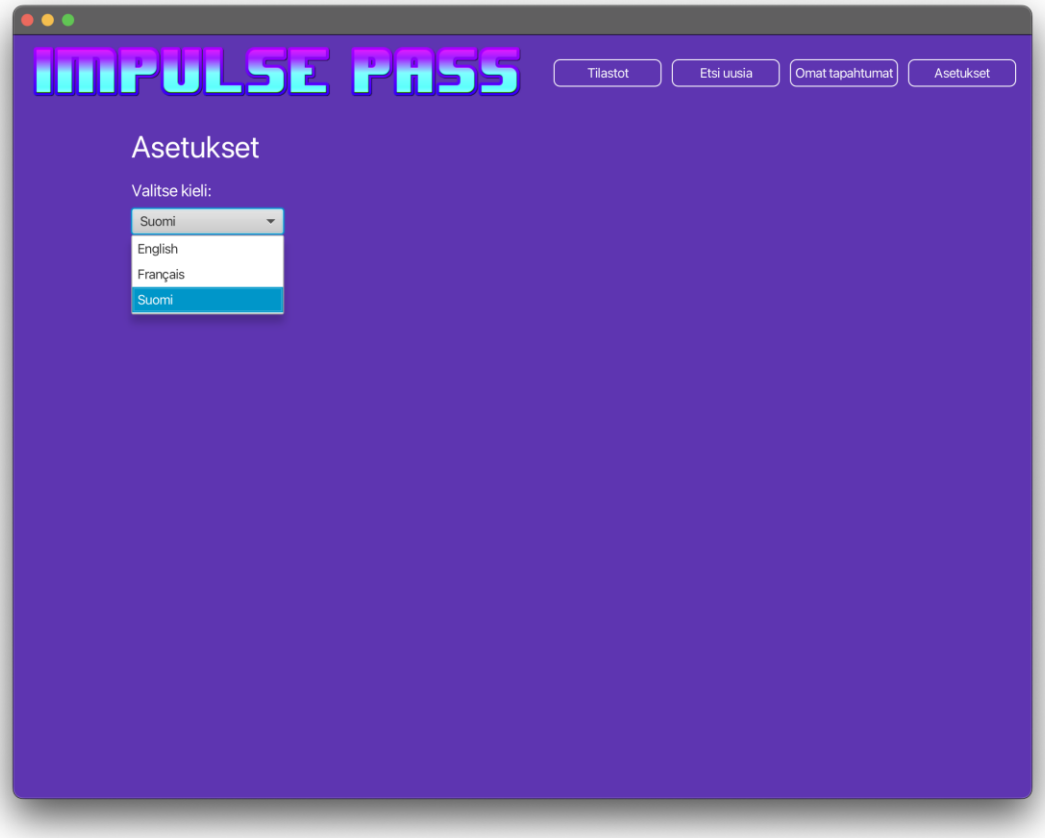


Kuva 9 uusien tapahtumien tallentamisen jälkeen tapahtumat ovat pysyvästi käytettävissä

Tapahtumat ovat käyttötapauksen mukaisesti saatavilla (Kuva 9).

6.2 Lokalisointi

Ohjelmistoprojekti 2 -kurssilla projektin jakokehitykseen valittiin yhdeksi tärkeimmistä muutoskohteista sovelluksen lokalisointi. ImpulsePass sovellukselle annettiin kuvitteellinen rahoittaja Kanadasta, jonka takia sovellukselle tärkeää, että sovellus toimii Suomen kielen lisäksi ainakin Englanniksi, sekä Ranskaksi.



Kuva 10 Kielen vaihtaminen sovelluksessa

Sovellukseen kehitettiin sisäinen ratkaisu lokalisoinnille. Luotiin asetussivu, johon lisättiin kielenvaihto-ominaisuus. Kielen vaihtaminen muuttaa koko sovelluksen kielen valittuun kieleen. Kielet asuvat projektissa bundletiedostoissa, jotka mahdollistavat yksinkertaisen kielen vaihdoksen. (Kuva 10)

Valittu kieli asuu myös Singleton luokassa, mikä mahdollistaa valitun kielen huomaamisen, ja sen käyttämisen esimerkiksi päivämäärien lokalisointiin sekä relevantin valuutan valitsemiseen.

Yksi herännyt ongelma lokalisoinnin kanssa on se, että tapahtumien tarjoajat eivät takaa tapahtumistaan tietoa ImpulsePass sovelluksen vaatimille kielille. Tätä seikkaa ei lähdetty tarkemmin tutkimaan kurssin aikarajoitteiden takia, ja myöskin siitä syystä, että jos esimerkiksi Kanadassa ImpulsePassia käyttää, sovellus tarjoaisi tapahtumia pääosassa Kanadalaisilta tapahtumantarjoajilta.

7 Kehitysprosessi ja kehitysvaiheen tekniikat

Projekti toteutettiin noudattamalla ketterää kehitysmenetelmää kahden kurssin ajan. Ensimmäisellä kurssilla työ jakautui neljän kahden viikon sprintin ajalle, ja toisella kurssilla kolmen kahden viikon mittaisen sprintin aikana. Jokaisen sprintin seurantaan hyödynsimme Nektion palvelua, ja jokaisessa sprintissä oli nimetty "scrum master", joka vastasi agiilin ohjelmistokehityksen toteuttamisesta. Scrum masterin rooli vaihtui jokaisen sprintin aikana, jotta jokainen kehittäjä sai mahdollisuuden kokea erilaisia ohjelmistokehityksen rooleja ja vastuita.

Versionhallintaan valitsimme gitin, joka toimi GitHub-palvelun avulla. Git mahdollisti tehokkaan ja joustavan yhteistyön kehittäjien välillä, kun taas GitHub tarjosi keskitetyn alustan koodin hallintaan ja yhteisen työskentelytilan. Ohjelmointiympäristönä käytimme Eclipseä, joka tarjosi monipuoliset työkalut ja ominaisuudet ohjelmistokehityksen tueksi.

Yksikkötestit ohjelmistolle toteutettiin käyttäen JUnit-kehystä, joka tarjosi tehokkaat työkalut testien kirjoittamiseen, suorittamiseen ja tulosten analysointiin. Jatkuvan testauksen osalta hyödynsimme Jenkins-palvelinta, jossa koontiversion tilaa pystyi seuraamaan reaaliaikaisesti. Jenkinsin avulla voimme automatisoida testauksen ja varmistaa, että uudet muutokset eivät aiheuta hajoamisia tai virheitä järjestelmään.

Kehitysvaiheen tekniikoiden avulla pystyimme varmistamaan ohjelmiston laadun ja toimivuuden projektin eri vaiheissa. Ketterä kehitysprosessi yhdessä käytettyjen työkalujen ja tekniikoiden kanssa mahdollisti tehokkaan ja joustavan projektin toteutuksen, joka vastasi asiakkaan tarpeita ja odotuksia.

8 Yhteenveto

Yhteenvetona voidaan todeta, että projektin tavoitteena oli luoda sovellus, joka tarjoaa käyttäjille helpon tavan löytää edullisia ja mielenkiintoisia tapahtumia samalla kun tarjoaa tapahtumien järjestäjille mahdollisuuden seurata trendejä ja kerätä dataa. Sovelluksen oli tarkoitus tarjota käyttäjille myös mahdollisuus tallentaa kiinnostavia tapahtumia ja pysyä ajan tasalla tapahtumien sekä lipunmyynnin alkamisesta.

Olemme onnistuneet saavuttamaan suunnitellut tavoitteet projektin puitteissa. Kuitenkin huomasimme, että käyttökokemuksen parantaminen tarjoaa vielä paljon potentiaalia. Projektin aikarajoitteiden vuoksi emme pystyneet toteuttamaan kaikkia ideoitamme, mutta jätämme ne tulevaisuuden mahdollisuuksiksi ja jatkokehityksen kohteiksi.

Tulevaisuudessa voisimme keskittyä parantamaan sovelluksen käyttöliittymää ja responsiivisuutta. Voisimme myös kehittää lisää ominaisuuksia, kuten suosittelujärjestelmän tai integroida sosiaalisen median jakamistoiminnallisuuden. Lisäksi voimme harkita käyttäjäpalautteen keräämistä ja analysointia, jotta voimme jatkuvasti parantaa sovellusta käyttäjien tarpeiden mukaan.

Kaiken kaikkiaan olemme tyytyväisiä saavutettuihin tuloksiin ja uskomme, että sovelluksella on potentiaalia tarjota arvoa käyttäjille ja tapahtumien järjestäjille. Projektin myötä olemme oppineet paljon ja saaneet arvokasta kokemusta ohjelmistokehitysprosessista ketterän menetelmän avulla.