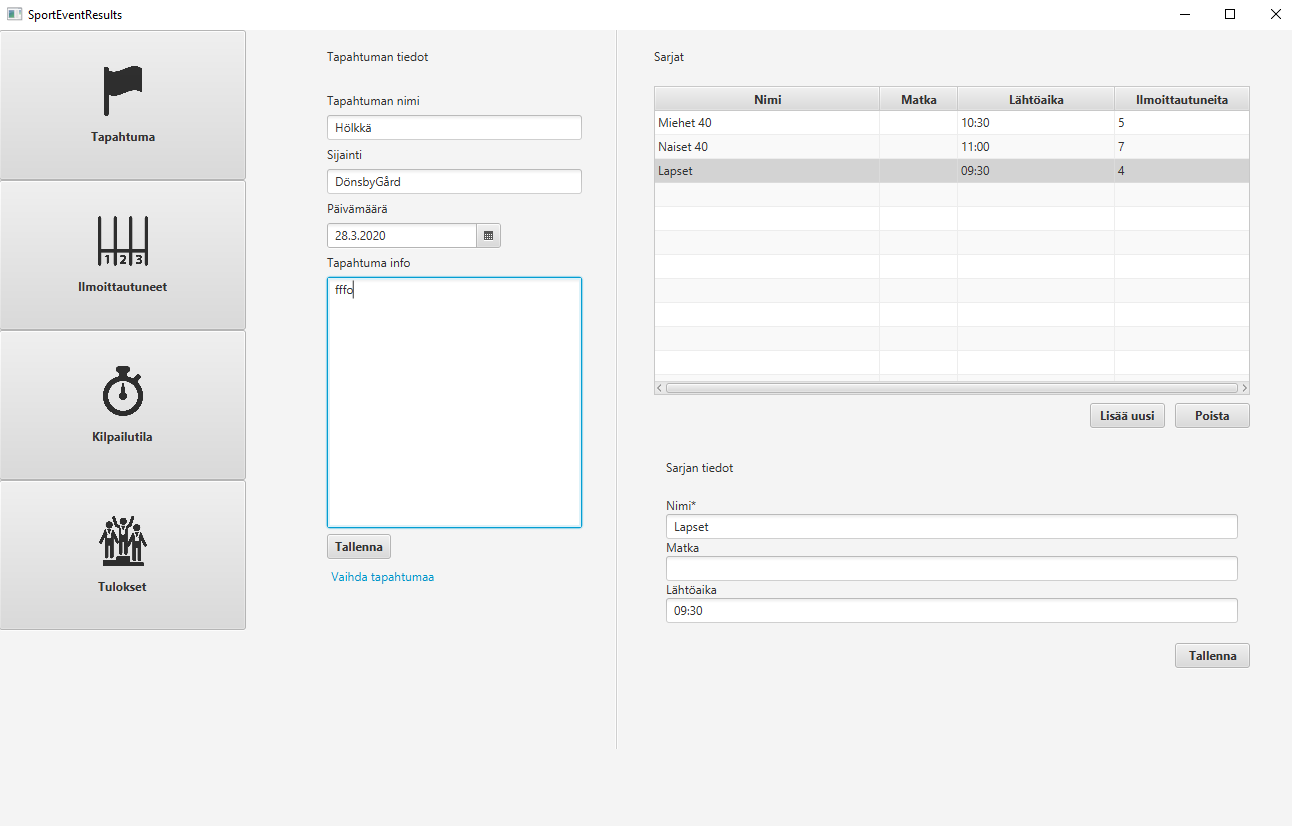
# Ohjelmiston rakenteen ja toiminnan kuvaus

SportEventResults on tulospalveluohjelma urheilutapahtumiin. Ohjelma on tarkoitettu käytettäväksi lähinnä massakuntotapahtumien osanottajien ja tulostietojen hallintaan. Ohjelmassa on kaksi tilaa. Valmistelutilaa käytetään, kun kilpailut eivät ole käynnissä. Tällöin voidaan syöttää ja tarkastella kilpailuiden tietoja, kuten osanottajatietoja. Kilpailutilassa taas ohjelmaan syötetään maaliin tulevien kilpailijoiden numeroita ja heille saatuja aikoja. Maalintulleista osallistujista voidaan muodostaa tuloslistoja sarjoittain.



## Toiminnan kuvaus yleisellä tasolla

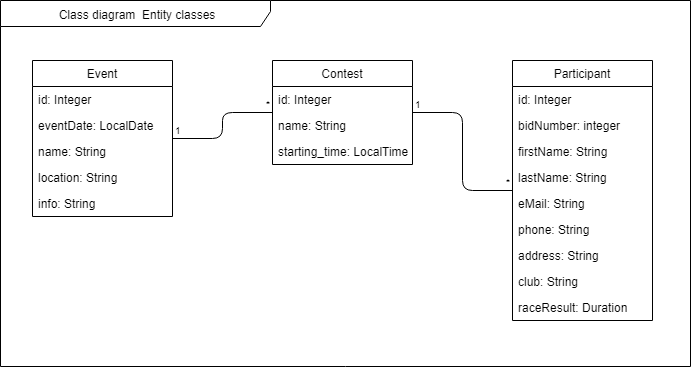
Ohjelman käynnistyessä käyttäjä valitsee urheilutapahtuman, jota haluaa muokata tai luo uuden tapahtuman. Tapahtumilla on nimi, päivämäärä, alkamis- ja loppumisajat, tapahtumapaikka, sekä vapaamuotoista lisätietoa. Tapahtumassa on tyypillisesti useita kilpailusarjoja. Uusia sarjoja voi lisätä ja muokata tapahtumanäkymässä. Sarjalla on nimi, alkamisaika ja tapahtuma, johon sarja liittyy (+ ehkä matka ja reitti).

Jokainen tapahtuman osanottaja liittyy aina johonkin tapahtuman sarjaan. Osanottajia hallitaan näkymässä, jossa osanottajat listataan sarjoittain. Käyttäjä voi valita näytettävän sarjan. Sarjaan liittyvät osanottajat näytetään taulussa, jossa näkyy myös tärkeimmät osanottajaan liittyvät tiedot. Tarkemmat tiedot saa näkyviin valitsemalla yhden osanottajan muokattavaksi. Sarjakohtaiset osanottajalistat voi myös tulostaa.

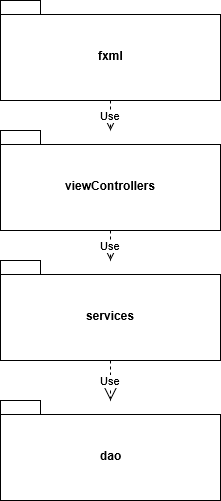
Osanottajaan liittyy nimen ja yhteystietojen lisäksi kilpailunumero. Yhdessä kilpailussa samaa numeroa ei voi olla kahdella kilpailijalla. Yhden osanottajan muokkaus tapahtuu uudessa ikkunassa, jossa tiedot voi tallentaa tai kilpailijan voi poistaa, samaa ikkunaa käytetään myös uuden kilpailijan luomiseen.

Kilpailutilassa tapahtumaan liittyvät osanottajat jaotellaan maaliin tulleisiin ja vielä maaliin saapumattomiin. Käyttäjä syöttää maaliin saapuvan kilpailunumeron, sekä ajan, joka osanottajalla on kestänyt suorituksessa. Tällöin osanottaja siirretään maaliin tulleisiin. Näkymässä voi myös siirtää osanottajia takaisin saapumattomiin, mikäli tietojen syöttämisessä on esimerkiksi tapahtunut virhe. Näkymässä näkyy myös saapumattomien lukumäärä.

Lopulta tapahtuman tuloksia voidaan tarkastella, tulostaa ja viedä ohjelmasta ulos tulosnäkymässä.



## Ohjelmiston rakenne

Kyseessä on Windows-työpöytä sovellus. Ohjelman arkkitehtuuri noudattaa MVC-mallia, ja näkymät, niiden kontrollerit ja muut ohjelman toiminnallisuudet, kuten tallennusominaisuudet on erotettu toisistaan. Kerrosarkkitehtuurimaisesti kukin toiminnallisuus on eriytetty omiin kerroksiinsa, jotka ovat entiteettiluokkia lukuun ottamatta yhteydessä vain viereisiin kerroksiin.

Käyttöliittymä on toteutettu JavaFX:llä ja käyttöliittymän näkymät on määritetty FXML-pohjaisesti. Jokaista näkymää vastaa yksi ohjausolio, joka vastaa kyseiseen näkymään liittyvästä toiminnallisuudesta, kuten tapahtumien käsittelystä, ja joka ohjaa pyyntöjä Service-luokille, jotka puolestaan ovat yhteydessä tietokantaa käsitteleviin luokkiin.

Ohjelman service-oliot tarjoavat käsiteolioihin liittyviä palveluita. Service- ja Utility -luokilla on pyritty keventämään kontrolleriluokkien työkuormaa, sekä mahdollistamaan sujuvampi siirtyminen mahdollisiin muihin käyttöliittymiin, mikäli ohjelmaa haluttaisiin kehittää esimerkiksi toimimaan web-käyttöliittymän kautta.

Tietokantatoiminnat on eristetty CRUD toiminnallisuuden tarjoaville dao-luokille. Ohjelma käyttää H2-relaatiotietokantaa JDBC-rajapinnan ja -ajureiden avulla. Tietokannan taulut vastaavat ohjelman entiteettiluokkia.

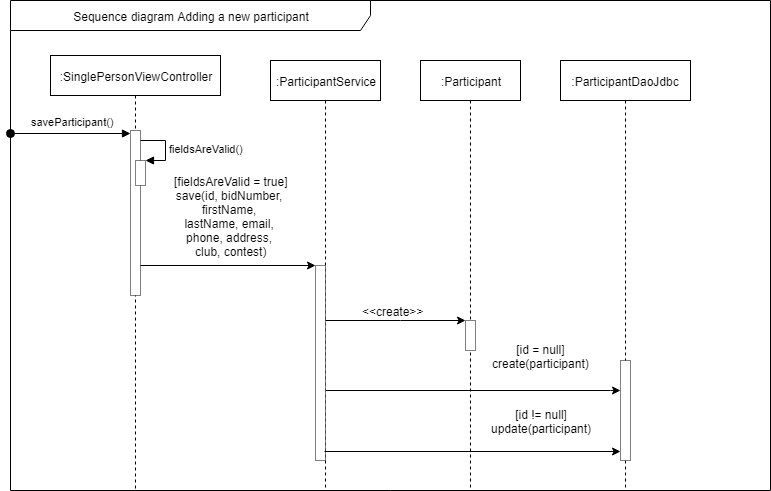
## Ohjelman toiminta

Ohjelman toiminta keskittyy tietokannan käytön ympärille. Käyttäjä tallentaa tietoja kantaan ja ohjelma hakee kannasta oleelliset tiedot kuhunkin näkymään. Muu toiminnallisuus enimmäkseen tukee tietokannan käyttöä ja syötteiden vastaanottoa. Keskeisimpiä tehtäviä ovat sarjaan liittyvien osanottajien haku, osanottajien haku numeron tai kilpailunumeron perusteella, kilpailijoiden lisääminen, sekä tapahtumien ja sarjojen hallinta. Nämä toiminnot ohjataan service-luokkien kautta.

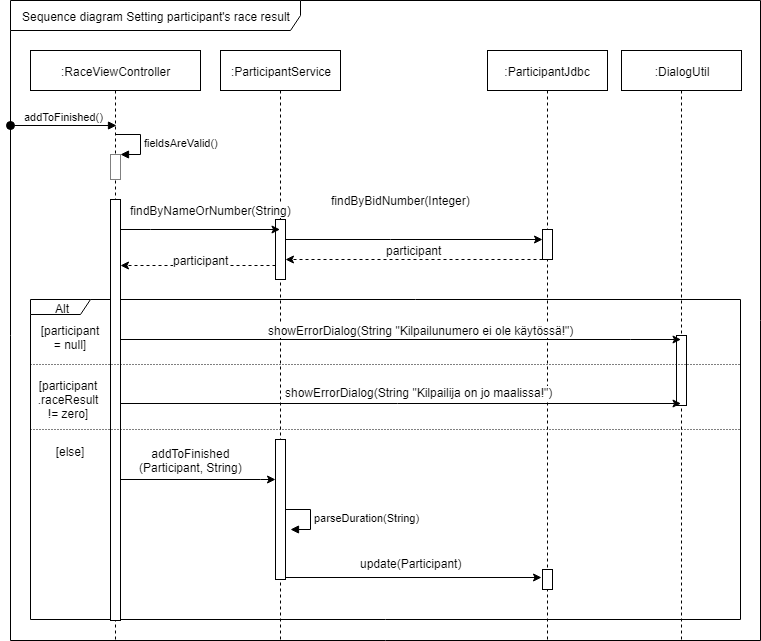
Käyttäjän antamat syötteet validoidaan tarkastamalla muodon oikeellisuus, sekä mm. varmistamalla, ettei annettu kilpailunumero ole käytössä tai kilpailija ole jo maalissa. Syötteiden muodon oikeellisuus tarkistetaan kussakin kontrolleriluokassa olevassa fieldsAreValid()-metodissa, joka palauttaa boolean-tyyppisen vastauksen. Epäkelvoista arvoista ilmoitetaan käyttäjälle lähettämällä virheteksti yleiskäyttöisellä DialogUtil-luokan showErrorDialog(String)-metodille.

Jos käyttäjä on poistamassa jotain, kuten osanottajaa, varmistetaan asia käyttäjältä DialogUtil-luokan promptDelete(String)-metodissa luotavalla popup-ikkunassa, joka palauttaa true, mikäli käyttäjä haluaa toteuttaa muutoksen ja muuten false.

Esimerkiksi uutta kilpailijaa luodessa tarkistetaan ensin kenttien arvot kontrolleriluokassa, lähetetään sitten arvot kilpailijasta vastaavalle ParticipantService-luokalle ja sen metodille save, joka luo arvoja vastaavan osanottajan. Mikäli osanottajalla ei ole vielä id-attribuuttia, eli sitä ei ole tietokannassa, tallennetaan tietokantaan uusi osanottaja. Muussa tapauksessa päivitetään olemassa oleva osanottaja saatujen arvojen mukaiseksi.



Toisena erimerkkinä osanottajan tullessa maaliin käyttäjä syöttää osanottajan kilpailunumeron ja tälle kellotetun ajan ja painaa ”Lisää” nappia. Ensin tarkistetaan kenttien syötteiden oikeellisuus fieldsAreValid()-metodissa. Seuraavakasi haetaan participantServiceltä syötettyä numeroa vastaava osanottaja. Varmistetaan, että kilpailunumero on käytössä ja ettei osanottaja ole jo merkattu maaliin, muuten kutsutaan ParticipantServicen addToFinished-metodia. Seuraavaksi koetetaan muuttaa aikakentän String-muotoinen arvo Duration-muotoiseksi. Jos muunnos onnistuu, asetetaan osanottajan kilpailuaika vastaamana sitä ja kutsutaan dao-kerroksen metodia tietojen tallentamiseksi tietokantaan. Lopulta päivitetään näkymä vastaamaan uutta tilannetta.



Sekä lähtölista, että lopputulokset on mahdollista tulostaa tai tuoda ohjelmasta ulos. Molemmissa tapauksissa tietokannassa oleva tieto valmistellaan ensin html-muotoon, ja sen jälkeen joko tallennetaan tiedostoon tai annetaan tulostimelle hoidettavaksi.

Etsi toiminnallisuus lähtölistoissa hakee ParticipantServicen findByNameOrNumber(String)avulla listan hakuehdot täyttävistä kilpailijoista ja täyttää taulunäkymän näillä tuloksilla.

## Kehitys ja itsereflektio

Ohjelman nykyisessä muodossa maaliin saapuvat syötetään käsin kirjoittamalla maaliin saapuvan kilpailunumero, sekä erillisellä kellolla saatu kesto. Tämä voi tapahtumasta riippuen olla ihan varteenotettava lähestymistapa, mutta jos ohjelmaa lähtisi kehittämään, tulisi toiminto muuttaa siten, että halutessa nämä tiedot tulisivat automaattisesti ajanottolaitteista ja kilpailijan tunnistus tapahtuisi RFID-tekniikan avulla.

Toinen selkeä kehityskohde olisi siirtää painopistettä nettiin niin, että tapahtuman esivalmisteluita pystyisi tekemään selaimen kautta. Samoin tulosten julkaiseminen olisi hyvä saada virtaviivaiseksi ja reaaliaikaiseksi.

### Työnkulku ja pohdintaa

Työ sujui varsin sujuvasti suunnitelman mukaan. Koodia tuli aika paljon järjesteltyä uudelleen työn aikana. Vaikka ohjelma on periaatteessa yksinkertainen ja järjestys aika perus, niin niillä taidoilla ja tiedoilla, mitä projektia aloittaessa oli, en usko, että olisin tarkemmallakaan suunnittelulla osannut välttää muodon jatkuvaa hakemista.

Kun ensimmäiseen iteraatioon valmistuessa käsitteiden suhteet ja ohjelman perustoiminnallisuus oli mietittynä, tuntui, että ohjelma alkaakin olla jo valmis. Lopulta aika paljon oli kuitenkin tekemistä ennen kuin ohjelman käyttö oli saatu luontevaksi ja mm. kaikki virhemahdollisuudet käytyläpi.

Iteraatioittain lisättävät ominaisuudet muodostivat mukavia kokonaisuuksia, joita oli selkää lisäillä ohjelmaan. Toiseen iteraatioon sain käyttöön graafisen käyttöliittymän, mikä toisaalta auttoi itseäkin hahmottamaan paremmin mitä tästä on tulossa. Hieman alkuperäisestä suunnitelmasta poiketen tuntui luontevimmalta aloittaa osanottajia koskevista toiminnallisuuksista ja rakentaa sarja- ja tapahtuma- runkoa sitten siihen ympärille.

Kun osanottajien syöttäminen ja sarjojen käsittely oli valmis, olikin aika helppo lähteä lisäämään tulosten syöttöominaisuutta ja tuloslistojen käsittelyä. Lopulta keskeiset alkuperäiset toiminnalliset vaatimukset sain toteutettua ja jotain kaavailemiani lisäominaisuuksiakin, kuten jonkinlaisen tulostusmahdollisuuden ja tietojen viemisen ohjelmasta ulos.

Aika paljon lueskelin projektin aikana käytännön juttuja JavaFX:n käytöstä, esimerkiksi miten taulukoita täytetään, miten ikkunoita suljetaan jne. Samoin koetin lueskelle ohjelmien rakenteesta myös luentomonisteen ulkopuolelta. UML-kaavioissa oli opeteltavaa myös, mutta niiden piirtely kyllä tuntui hyödylliseltä myös omien ajatusten jäsentelyssä.

Jos lähtisin tekemään projektia uudelleen, ottaisin testaamisen kiinteästi ohjelman kehittämiseen mukaan heti alusta lähtien. Tämä oli tarkoitus nytkin, mutta asia tahtoi jäädä jatkuvasti tekemättä. Huomasin, että testien kirjoittaminen auttaisi kyllä selkeiden metodien kirjoittamisisessa ja toisaalta varmistamaan, että kaikki edelleen toimii, vaikka ohjelmaan tekee muutoksia.

Moni asia jää mietityttämään. Service-taso tässä jää aika ohueksi, joten sen mielekkyys tässä vaiheessa mietityttää. Toisaalta jos hommaa lähtisi jatkokehittämään sen merkitys varmasti kasvaisi. Jonkin sovelluskehyksen käyttöönotto selkeyttäisi ja tehostaisi varmasti joitain asioita. Näkymistä jäin miettimään taas taulukoiden kokonaan uudelleen lataamisen vaihtoehtoja niitä päivittäessä.

Ohjelma on lopulta siinä kunnossa, että uskon sillä voisi periaatteessa olla käytännön käyttöäkin jossain pienemmissä tapahtumissa. Oikeasti mielenkiintoinen tästä kyllä tulisi kuitenkin vasta kun web-puoli ja ajanoton yhdistäminen olisivat mukana.