Eetu Soronen – Emil Ålgars

Metroaseman simulaattori

Jono- ja palvelupisteiden simulointi metroasemalla

Metropolia Ammattikorkeakoulu
Tieto- ja viestintätekniikka
Ohjelmistotuotanto
Ohjelmointiprojekti
23.10.2022

Sisällys

1 Johdanto

Metroaseman simulaattori on diskreetti kolmivaihesimulaattori, joka on kirjoitettu Javalla käyttäen JavaFX-käyttöliittymäkirjastoa. Simulaattorin tuloksia voidaan lukea ja tallentaa Metropolian tarjoamasta MySQL-tietokannasta. Simulaattorin runko pohjautuu opettajamme Simo Silanderin Simu-frameworkiin.

Ohjelman tarkoitus on simuloida yksittäistä Metroasemaa ja erityisesti sen Palvelupisteitä ja niistä aiheutuneita ruuhkia. Aseman ja palvelupisteiden parametrejä voidaan muuttaa kuvaamaan useimpia tyypillistä metroasemia.

Tässä dokumentissa esittelemme simulaattorimme mallin suunnittelun ja sen teknisen toteutuksen. Dokumentti sisältää myös simulaattorin käyttöohjeen sekä yhteenvedon projektin kulusta ja simulaattorin lopullisesta muodosta.

2 Visio

Visiomme on saada aikaan simulaattori, joka voi toimia digitaalisena kaksosena useimpien metroasemien kanssa sen parametreja muuttamalla. Sen käyttötarkoitus on selvittää aseman sisäisiä ruuhkautumisia erilaisista syistä, ja miten niitä voidaan parantaa tai välttää kokonaan. Sovelluksella tulisi olla selkeä ja ymmärrettävä käyttöliittymä, jonka avulla simulaattorin asetuksia voidaan muuttaa ja tuloksia lukea tai tallentaa.

Kuva 1. Metroasema simulaattori (Ei vastaa lopullisen tuotteen visuaalista olemusta.)

3 Käsitteet, määritelmät

Asiakkaalla tarkoitetaan metroasemalle saapuvia asiakkaita. Jotka menevät palvelupisteiden läpi ja poistuvat metron kautta. Metron kautta saapuvia asiakkaita ei simuloida.

Palveluajalla tarkoitetaan yhden palvelupisteen käsittelyaikaa (palvelun aloituksesta palvelun loppuun), esim. lipunmyytiin kuluvaa.

Palvelupiste on piste, jossa asiakas käsitellään ennen kuin se liikkuu eteenpäin simulaattorissa. Esimerkiksi: Lipunmyynti, Lipuntarkastus.

Komponentti on Simulaattorin osa, jolla löytyy vastike reaalimaailmasta, kuten palvelupiste tai asiakas.

Simu-Framework on Simo Silanderin luoma ohjelmistokehys kolmivaihesimulaattoreita varten.

Komivaihesimuloinnissa ajan etenemistä edustaa A vaihe ja tapahtumat jaetaan tyypin mukaan kahteen ryhmään: B- ja C-tapahtumat. B-tapahtumat ajastetaan tietyille ajan hetkille ja C-tapahtumat suoritetaan ehtojen toteutuessa.

Jakarta Persistence eli JPA on rajapinta, jonka avulla Java-sovellukseen voidaan liittää tietokanta.

Hibernate on ohjelmistokehys joka toteuttaa JPA-rajapinnan. Sen avulla voidaan tallentaa ja lukea olioita tietokannasta.

MySQL on SQL-relaatiotietokanta jota käytämme sovelluksessame.

Singleton on luokan suunnitelumalli, joka takaa, että luokasta on vain yksi olio, johon päästään käsiksi mistä vain ohjelmassa.

4 Käsitteellinen malli

4.1 Tavoite

Simuloinnin tavoite on saada selville, kuinka aseman ja metron kapasiteetit, metrojen aikavälit, saapuvien asiakkaiden määrä, ja palvelupisteiden käsittelyajat vaikuttavat metrojen jonoihin. Tavoite on, näitä kaikkia parametreja voidaan muuttaa ja niiden vaikutusta simulaattorin eri osioissa tarkkailla.

4.2 Syötteet

Asiakkaiden saapumisia ja palvelupisteiden käsittelyaikoja kuvataan normaalijakaumilla, joissa odotusarvo on keskimääräinen kesto, ja varianssi kertoo miten paljon enemmän tai vähemmän kesto voi satunnaisesti olla. Jokaisen palvelupisteen sekä asiakkaiden luomista odotusarvoa ja varianssia voidaan säätää simulaattorissa. Saapuneet eli luodut asiakkaat asetetaan sisäänkäynnin jonoon.

Lisäksi simulaattorimme asemalla on maksimikapasiteetti, jonka täytyttyä sisäänkäynnin läpi ei pääse uusia asiakkaita sisään. Jokaisella metrolla on myös yhteinen maksimikapasiteetti, jonka täytyttyä asiakkaiden pitää jäädä odottamaan seuraavaa metroa. Näitä kapasiteetteja voidaan myös muuttaa.

Lipunmyynti-palvelupisteen ohittavat kaikki asiakkaat, joilla on mobiililippu, joka on säädettävä prosenttijakauma-arvo.

Viimeiseksi simulaattorin kestoa ja simuloinnin nopeutta (viivettä) voidaan myös säätää.

4.3 Tulosteet

Metroaseman yleisistä tilastoista pidetään kirjaa siitä, onko simulaattori käynnissä vai ei, kuinka paljon aikaa on kulunut, ja kuinka monta asiakasta on asemassa, sekä kuinka monta on poistunut.

Jokaisen palvelupisteen kohdalla pidetään kirjaa sen tilasta (vapaa / varattu), palvelluista asiakkaista, käsittelyn keskimääräisestä kestosta, jonon tämänhetkisestä pituudesta, ja jonotuksen keskikestosta. Erityisesti jonojen pituuksia ja keskikestoja tarkkailemalla voidaan selvittää mikä hidastaa asiakkaiden läpimenoja.

4.4 Sisältö

Mallissa otetaan huomioon, onko asema täynnä, onko saapuvalla asiakkaalla esiostettu lippu, ja kuinka kauan asiakkaalla kestää saapumisen, lipunmyynnin, lipuntarkastuksen läpimenossa, ja kuinka kauan hän joutuu odottamaan metroa. Metron saapumisväliajat ja kapasiteetit otetaan myös huomioon.

Taas asiakkaita, jotka saapuvat asemaan metron kautta ei oteta huomioon. Koska asemassamme ei ole kauppoja tai muita aseman toiminalle tarpeettomia palvelupisteitä, nämä asiakkaat eivät vaikuta aseman toimintaan vaan poistuvat heti asemasta. Siksi niitä ei kirjata minnekään.

4.5 Oletukset ja yksinkertaistukset

Oletamme että kaikki metroaseman asiakkaat ovat lainkuuliaisia kansalaisia, jotka eivät tee sariinikaasuiskuja tai muita järjestystä horjuttavia tekoja asemallamme. Myös muut kriisitilanteet ovat simulaattoristamme jätetty pois. Simuloimme vain asiakkaiden läpimenoja eri palvelupisteiden kautta metroaseman sisällä.

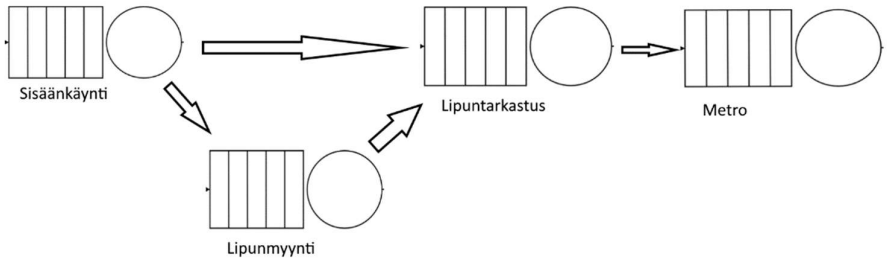
4.6 Mallin kuvaus

4.6.1 Komponenttilista

Komponentti	**Ominaisuuksia**

Asiakas	Saapumisväliaika normaalijakaumalla
Sisäänkäynti	Uudet asiakkaat luodaan sisäänkäynnin jonoon. Jos asemassa on tilaa, ne käsitellään ja asiakas menee joko lipunmyyntiin tai lipuntarkastukseen. Käsittelyaika on normaalijakauma.
Lipunmyynti	Asiakkaat joilla ei ole esiostettuja lippuja menevät tänne saavuttuaan asemaan. Käsittelyaika on normaalijakauma.
Lipuntarkastus	Kaikki asiakkaat menevät tämän läpi ennen kuin he pääsevät jonottamaan metroa. Käsittelyaika on säädettävä normaalijakauma.
Metro	Kapasiteetti muutettavissa. Saapumisaikavälit käyttävät normaalijakaumaa.

4.6.2 Prosessikaavio



Asiakkaat	Jono (∞)	Sisäänkäynti (palveluaikajakauma normaalijakauma)
Asiakkaat	Jono (∞)	Lipunmyynti palveluaikajakauma (normaalijakauma)
Asiakkaat	Jono (∞)	Lipuntarkastus (normaalijakauma)
Asiakkaat	Jono (∞)	Metro (normaalijakauma)

Kuva 2. Prosessikaavio kuvaus metroasema simulaattorista (kaikkien pisteiden määrä on dynaaminen ja käyttäjän määrittelemä)

Asiakkaiden saapumiset (eli luomiset) ja palvelupisteiden käsittelyajat noudattavat normaalijakaumaa, jonka odotusarvoa ja varianssia voidaan muuttaa.

5 Mallin ohjelmointitekninen toteutus

5.1 Käytetyt ohjelmointikielet ja kirjastot (ulkoiset API:t).

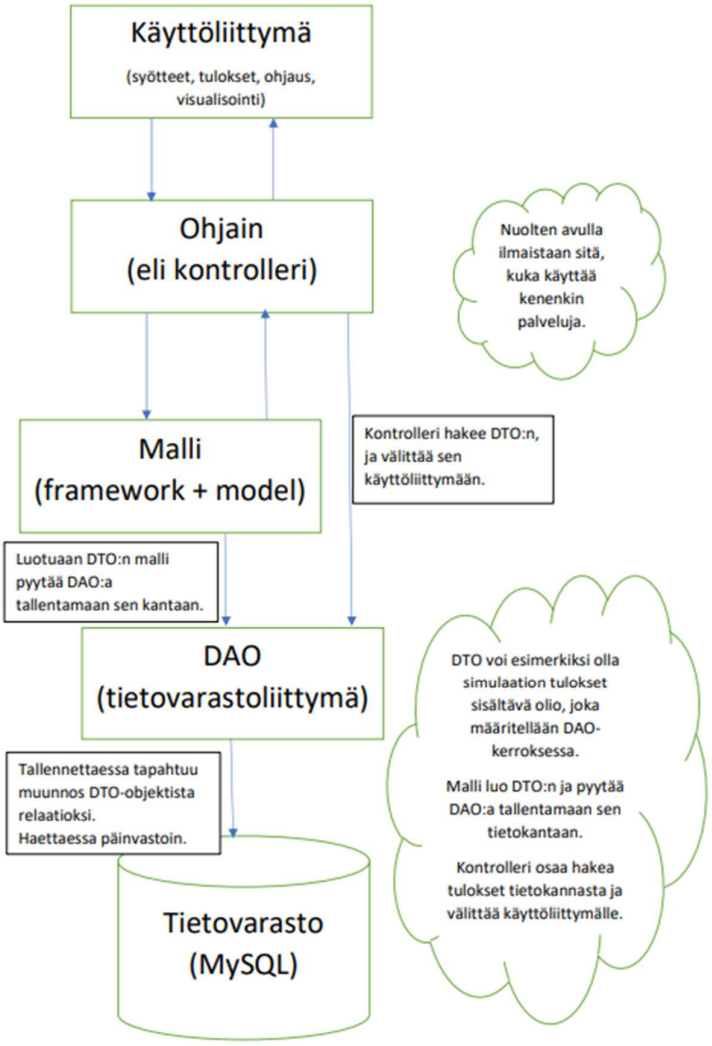
Koko simulaattori on kirjoitettu Javalla käyttäen JavaFX-ohjelmistoalustaa, jossa käyttöliittymä on kirjoitettu pääsääntöisesti fxml-merkintäkielellä käyttäen SceneBuilderia.

Käytettyjä Java-kirjastoja ovat Eduni distributions, joka tarjoaa erilaisia jakaumia ja joka toteuttaa saapumisessa ja palvelupisteissämme käyttämät normaalijakaumat.

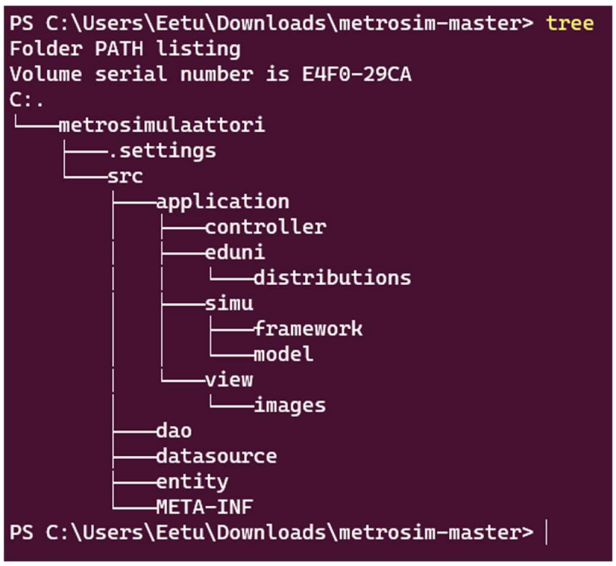
Tietokannan toteutukseen käytämme Jakarta Persistence ja Hibernate -kirjastoja, jotka tekevät ulkoisen MySQL-tietokannan ja sovelluksen välisesestä kommunikoinnista erittäin yksinkertaista.

5.2 Arkkitehtuuri

Simulaattorin kerrosarkkitehtuuri:



Kuva 3. Projektimme arkkitehtuurin kuvio.



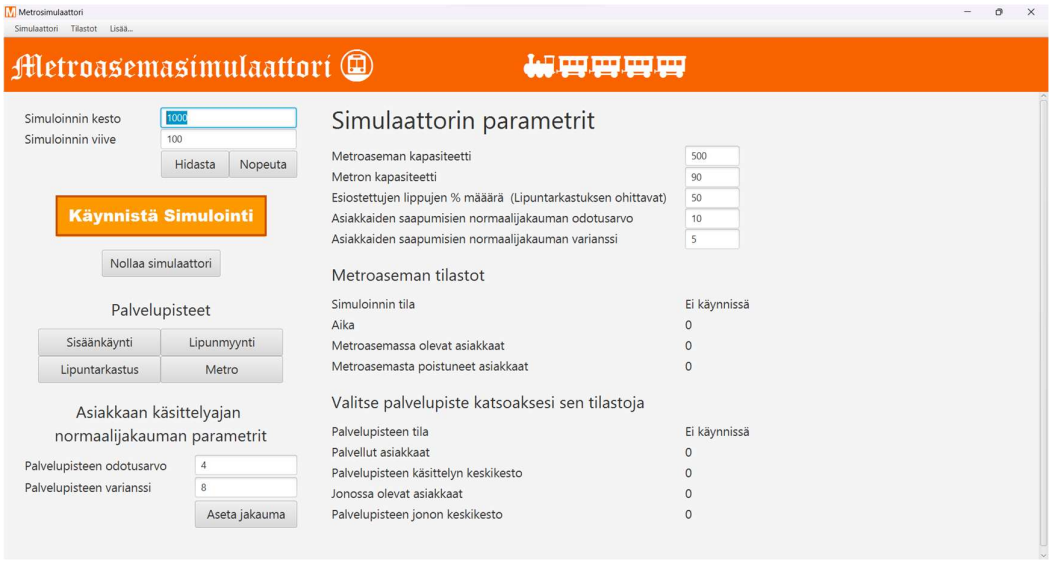
Kuva 4. Metrosimulaattorin hakemistorakenne

Projekti on suuniteltu MVC-arkkitehtuurin mukaisesti. Tämä tarkoittaa sitä, että malli ja näkymä eivät kommunikoi toistensa kanssa suoraan, vaan niiden välittäjänä toimii kontrolleri.

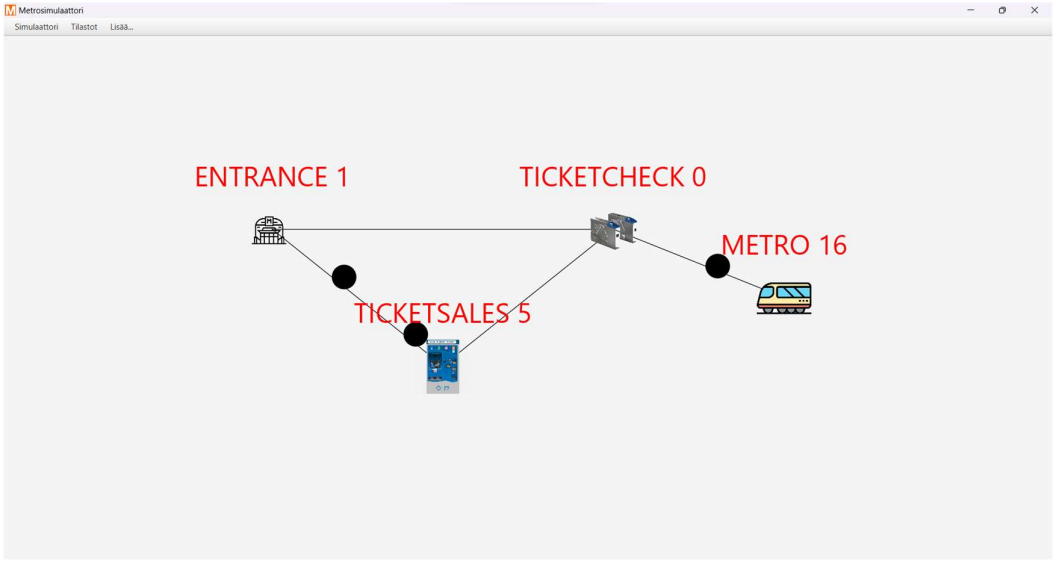
Itse sovellus sijaitsee application-pakkauksessa, joka sisältää neljä pakkausta. Näistä simu-pakkaus pitää sisällään varsinaisen simulaattorin mallin (asiakas, palvelupiste, kello, moottori yms.) ja eduni-kirjaston, jota käytetään normaalijakaumien muodostamisessa simun-sisällä. View-pakkaus pitää sisällään kaikki käyttöliittymän luomiseen ja sen ohjaamiseen liittyvät tiedostot, jotka taas kommunikoivat mallin kanssa käyttäen controller-luokkaa välittäjänä.

Application-pakkauksen ulkopuolella on myös dao, datasource, entity ja META-INF -pakkaukset, tekevät simulaattorin tuloksista olioita joita voidaan tallentaa ja lukea tietokannasta kontrollerin-avulla. Näiden luokkien toiminnallisuuden tarjoaa JPA ja Hibernate.

5.3 Käyttöliittymän kuvaus

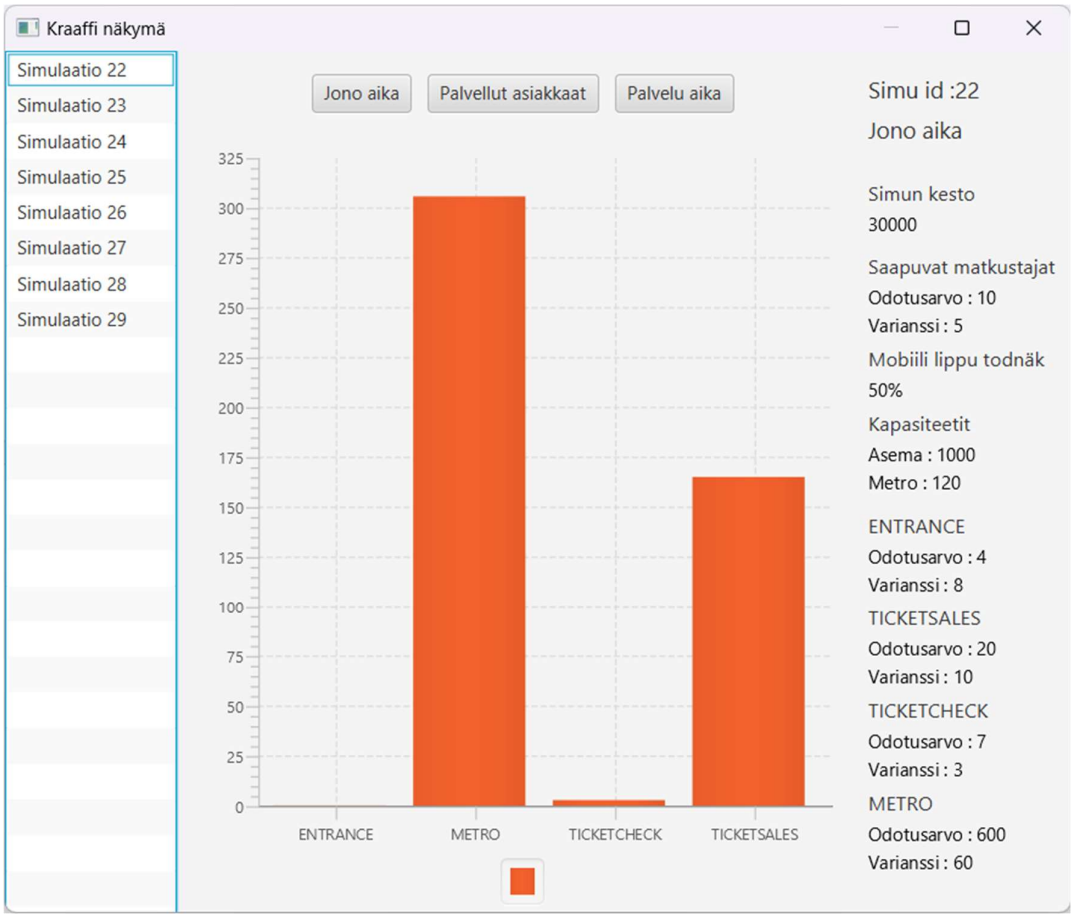


Kuva 5. Simulaattorin käyttöliittymä.



Kuva 6. Simulaattorin graafinen näkymä.

(18)

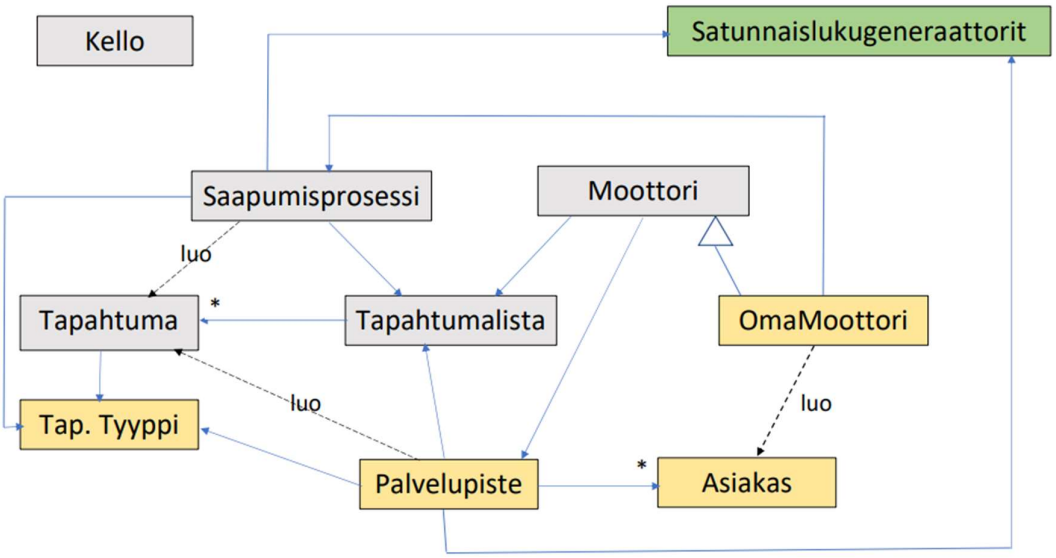


Kuva 7. Simulaattorin tilastot-näkymä.

Simulaattorin avautuessa näytetään tekstinäkymä, jossa simulaattorin kaikkia parametrejä ja tuloksia voidaan tarkastella. Täällä myös asetetaan simulaaattorin kesto, viive, ja käynnistetään se. Lisäksi simulaattorilla on graafinen näkymä, jossa kulkua voidaan tarkastella ja asiakkaiden kulkua ja jononpituuksia palvelupisteiden välillä on animoitu.

Vanhoja simulaatioita voidaan tarkastella tilastot-näkymästä, joka hakee kaikki simuloinnin parametrit ja tulokset tietokannasta. Näitä käytetään palveltujen asiakkaiden sekä palvelupisteiden käsittelyaikojen ja jonotusten kestojen havainnollistamiseen pylväsdiagrammeilla.

5.4 Sisäisen logiikan kuvaus



Kuva 8. Simulaattorin mallin luokat ja niiden suhteet. Harmaat luokat ovat simu/framework-pakkauksessa, oranssit simu/model-pakkauksessa. Satunnaislukugenerattorit ovat eduni/distributions-pakkauksessa.

Simulaattorimme malli on diskreetti tapahtumasimulaattori, joka sijaitsee simu-pakkauksessa. Se tarkoittaa sitä, että aika ei kulje simulaattorissamme lineaarisesti, vaan se hyppää aina seuraavaan hetkeen, jossa tapahtuma tapahtuu. Tapahtumat voivat olla asiakkaiden saapumisia tai käsittelyjä palvelupisteissä. Ajanpidosta huolehtii Kello-luokka, joka on singletoni. Siellä aikaa voidaan lukea tai siirtää eteenpäin.

Simulointi alkaa OmaMoottori-luokassa, joka on Moottori-luokan aliluokka. Se luo palvelupisteet, saapumisgeneraattorin, asettaa niiden parametrit ja pyörittää itse simulointia. Saapumisgeneraattori on luokka, joka luo uusia Asiakas-olioita Sisäänkäynti-palvelupisteen jonoon. Jokainen palvelupiste on siis Palvelupiste-luokan olio, joka käsittelee tiettyjä Tapahtuma-luokan olioita, jotka ovat aikajärjestyksessä Tapahtumalistassa.

OmaMoottori siis katsoo seuraavan tapahtuman ajankohdan, asettaa sen kellonajaksi, suorittaa sen. Saapumis-tapahtuman kohdalla tämä tarkoittaa asiakkaan luomista, käsit-

tely tapahtuman kohdalla taas se tarkoittaa asiakkaan asettamista palvelupisteen jonoon. Palvelupisteet käsittelevät asiakkaita tietyn ajan verran, eli ne lukevat kelloa, mutta ne eivät vie sitä eteenpäin. Tästä huolehtii yksinään OmaMoottori.

5.5 Ulkoisten tietovarastojen (tiedostot, tietokannat) kuvaukset

```
MariaDB [eetusoro]> describe ServicePoint;
+---------------------------+--------------+------+-----+---------+----------------+
| Field                     | Type         | Null | Key | Default | Extra          |
+---------------------------+--------------+------+-----+---------+----------------+
| id                        | int(11)      | NO   | PRI | NULL    | auto_increment |
| jononKeskikesto           | double       | NO   |     | NULL    |                |
| jonossaOlevatAsiakkaat    | int(11)      | NO   |     | NULL    |                |
| metronKapasiteetti        | int(11)      | NO   |     | NULL    |                |
| palvellutAsiakkaat        | int(11)      | NO   |     | NULL    |                |
| palvelunKeskiaika         | double       | NO   |     | NULL    |                |
| palvelupiste              | varchar(255) | YES  |     | NULL    |                |
| palvelupisteenOdotusarvo  | int(11)      | NO   |     | NULL    |                |
| palvelupisteenVarianssi   | int(11)      | NO   |     | NULL    |                |
+---------------------------+--------------+------+-----+---------+----------------+
9 rows in set (0.019 sec)

MariaDB [eetusoro]> describe Simulaattori;
+----------------+---------+------+-----+---------+----------------+
| Field          | Type    | Null | Key | Default | Extra          |
+----------------+---------+------+-----+---------+----------------+
| id             | int(11) | NO   | PRI | NULL    | auto_increment |
| simunKesto     | int(11) | NO   |     | NULL    |                |
| asema_id       | int(11) | YES  | MUL | NULL    |                |
| entrance_id    | int(11) | YES  | MUL | NULL    |                |
| metro_id       | int(11) | YES  | MUL | NULL    |                |
| ticketcheck_id | int(11) | YES  | MUL | NULL    |                |
| ticketsales_id | int(11) | YES  | MUL | NULL    |                |
+----------------+---------+------+-----+---------+----------------+
7 rows in set (0.017 sec)

MariaDB [eetusoro]> describe Station;
+--------------------------------+---------+------+-----+---------+----------------+
| Field                          | Type    | Null | Key | Default | Extra          |
+--------------------------------+---------+------+-----+---------+----------------+
| id                             | int(11) | NO   | PRI | NULL    | auto_increment |
| asemanKapasiteetti             | int(11) | NO   |     | NULL    |                |
| asemassaOlevatAsiakkaat        | int(11) | NO   |     | NULL    |                |
| asemastaPoistuneetAsiakkaat    | int(11) | NO   |     | NULL    |                |
| asiakkaidenSaapumisenOdotusarvo | int(11) | NO   |     | NULL    |                |
| asiakkaidenSaapumisenVarianssi | int(11) | NO   |     | NULL    |                |
| esiostettujenLippujenSuhde     | int(11) | NO   |     | NULL    |                |
+--------------------------------+---------+------+-----+---------+----------------+
7 rows in set (0.019 sec)
```

Kuva 9. Sovelluksen MySQL-tietokannan rakenne.

Sovelluksemme käyttää ulkoista Metropolian tarjoamaa MySQL tietokantaa. Sinne tallennetaan tietoa automaattisesti joka kerta kun simulaattorin on käynyt, ja sieltä luetaan

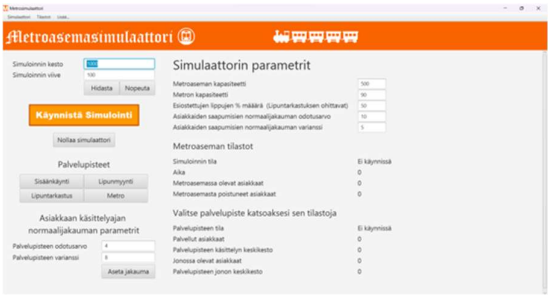
tietoa, kun ohjelmassa avataan tilastot-ikkuna. Sovelluksessa tietokantaan tallentaminen on toteutettu Kontrolleri-luokassa, joka hakee jokaisen simuloinnin päätyttyä hallitsemansa simuloinnin datan simu/model-pakkauksesta, luo sen pohjalta olion jokaiselle palvelupisteelle, asemalle, sekä simulaattorille. Nämä oliot tallennetaan omiin taulukoihinsa tietokantaan Hibernaten avulla, ja tietoa luetaan muuntamalla se taas olioksi Hibernaten avulla. Simulaattori-taulukko sisältää viittaukset jokaisen simulaation kohdalla muissa taulukoissa sijaitseviin tietoihin.

5.6 Testaus

Olemme ajaneet simulaattoria kehityksen jokaisessa vaiheessa ja saaneet sen sellaiseen kuntoon, että se antaa järkeviä tuloksia, eikä se kaadu, jos sille antaa virheellisiä parametrejä tai sen nollaa ennenaikaisesti. Simulaattorille ei voi antaa virheellisiä syötteitä, ja jos verkkotietokantaan ei saada yhteyttä, niin simulaattori toimii normaalisti lukematta tai tallentamatta tuloksiaan. JUnit-testejä meiltä ei vaadittu ryhmämme koon vuoksi.

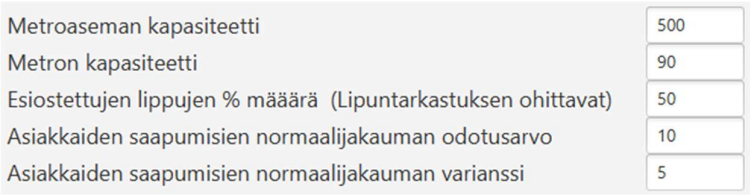
6 Simulaattorin käyttöohje

Sovelluksen käynnistettäessä asiakas asetetaan Simulaati näkymään .Tässänäkymässä käyttäjä voi ajaa simulaatioita ja muokatava parametrejä jotka vaikuttavat simulaation lopputulokseen. Näkymiä voi vaihtaa vasemmasta yläkulmasta simulaattori painikkeen alta, josta voi vaihtaa Simulaattori- ja Graafisen näkymän välillä. Jos käyttäjä haluaa nähdä simulointi historian voi hän mennä tilasto näkymään vasemassa yläkulmassa olevan tilasto näppäimen alta.



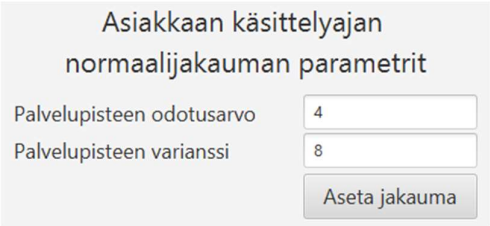
Kuva 10. Simulaattori näkymä

Simulaatio näkymän oikealla puolella tarjotaan käyttäjälle lukuisia muutettavia arvoja, jotka vaikuttavat koko simulaatioon. Kenttiin ei voi syöttää tekstiä, sillä simulaattori vaatii että näissä kentissä olevan numeroarvoja.



Kuva 11. Simulaatio näkymän oikean puolen parametrit oletusarvoilla

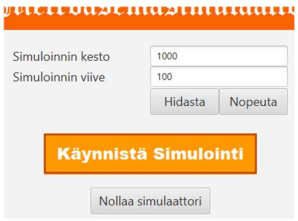
Simulaatio näkymän vasen alapuolisko on yksittäisten palvelupisteiden muuttujien säätelyyn. Palvelupisteen, jonka tietoja haluat muuttaa voit valita painamalla jotakin näppäintä, joka sijaitsee Palvelupisteet otsikon alla. Voit muuttaa, kuinka nopeasti ja millä varianssilla simulaattori käsittelee asiakkaita kyseisessä palvelupisteessä tekstikenttien kautta. Kun olet muuttanut arvot haluamaksisi niin Aseta jakauma näppäimen painaminen lukitsee arvot.



Kuva 12. Sisäänkäynti palvelupisteen oletusarvot otsikon ”Asiakkaan käsittelyajan normaalijakauman parametrit” alla.

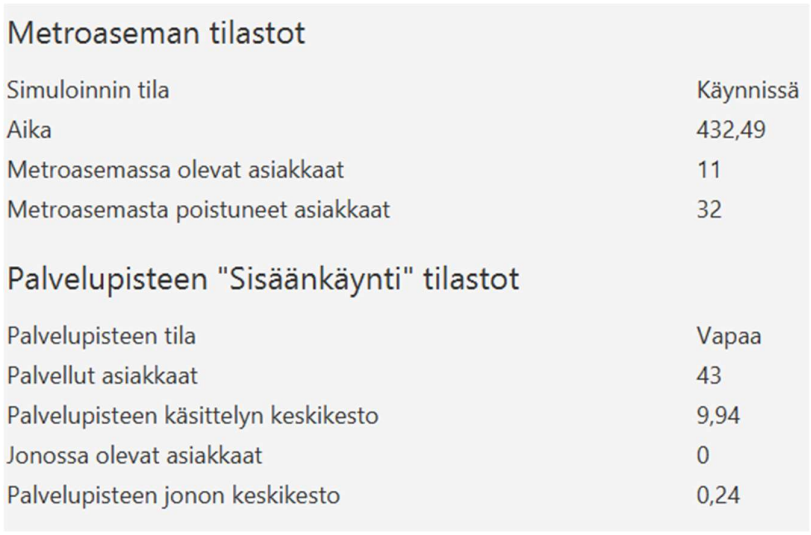
Simulaattori näkymän vasemmassa yläkulmassa on simulaattorin ”päänäkymä”, josta simulaattorin voi käynnistää painamalla oranssia ”käynnistä simulaattori” näppäintä. Simuloinnin voi päättää painamalla ”Nollaa simulaattori” näppäintä. Jos simulaattori on päättynyt itsestään, tai se on keskeytetty se pitää nollata, jonka jälkeen parametreja voi muuttaa ja uuden simulaation käynnistää. Simuloinnin keston voi muuttaa sitä vastaavasta tekstikentästä käynnistä simulaattori näppäimen yläpuolella. Simuloinnin viiveen voi muuttaa sitä vastaavasta tekstikentästä, jossa numero vastaa kuinka monta millisekunttia odotetaan kunnes simulaatio suorittaa seuraavaan tapahtuman. Kun simulaatio

on ajettu syöttää se tietokantaan simulaation tulokset. Nämä tulokset voi nähdä taulukkonäkymästä.



Kuva 13. Simulaatio näkymän pääkomponentit oletusarvoilla.

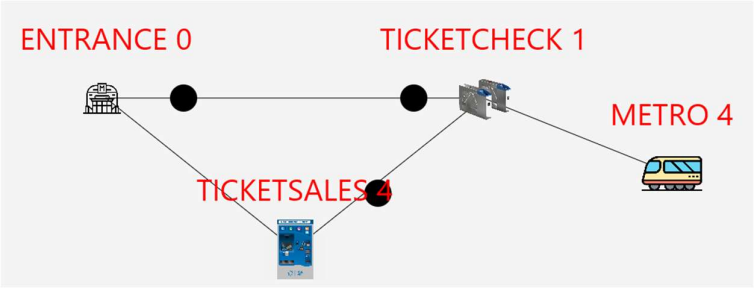
Simulaatio näkymän oikeassa alakulmassa näkyy simulaation tiedot suorituksen aikana. ”Aseman tilastot” alla olevat tiedot ovat riippumattomia siitä, mitä palvelupiste näppäintä on painettu, kun taas Palvelupisteen tiedot otsikon alla (tulee näkyviin vasta kun jokin palvelupiste on valittu) ovat riippuvaisia siitä mitä näppäintä on painettu.



Kuva 14. Esimerkki tilastoista, kun simulaattoria ajetaan oletusarvoilla.

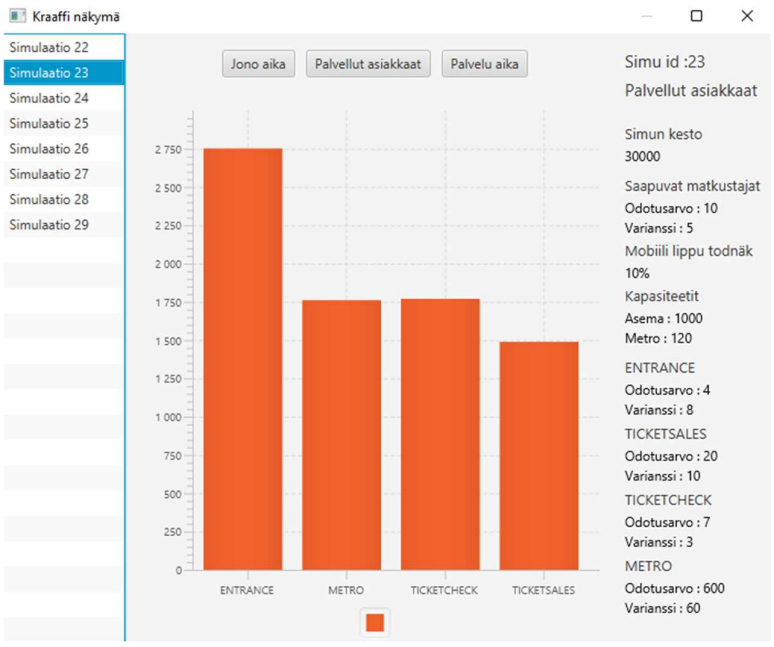
Graaffisessa näkymässä voidaan nähdä kuinka asiakkaat liikkuvat simuloinnin läpi. Asiakas kuvastetaan pallona joka liikkuu pisteestä toiseen. Asiakkiden näkyminen tässä näkymässä vaatii sen, että simulaatio on käynnistetty simulaattori näkymässä

(18)



Kuva 15. Graaffinen näkymä, kun simulaatio on päällä.

Tilasto näkymässä voi käyttäjä nähdä simulaatio historian, jossa on kaikki simulaatiot jotka ovat ajettu onnistuneesti (jos simulaatio nollataan kesken ajon se ei ilmesty tänne). Jos käyttäjä painaa DEL näppäintä kun jokin simulaatio on valittu, niin tämä simulaatio poistuu simulaatiohistoriasta. Taulukon yläpuolella on näppäimiä, joista voi vaihtaa mitä tietoja simulaatiosta näytetään. Oikealla näkyy simulaation tiedot / parametrit joilla simulaatio ajettiin.



Kuva 16. Taulukko näkymä.

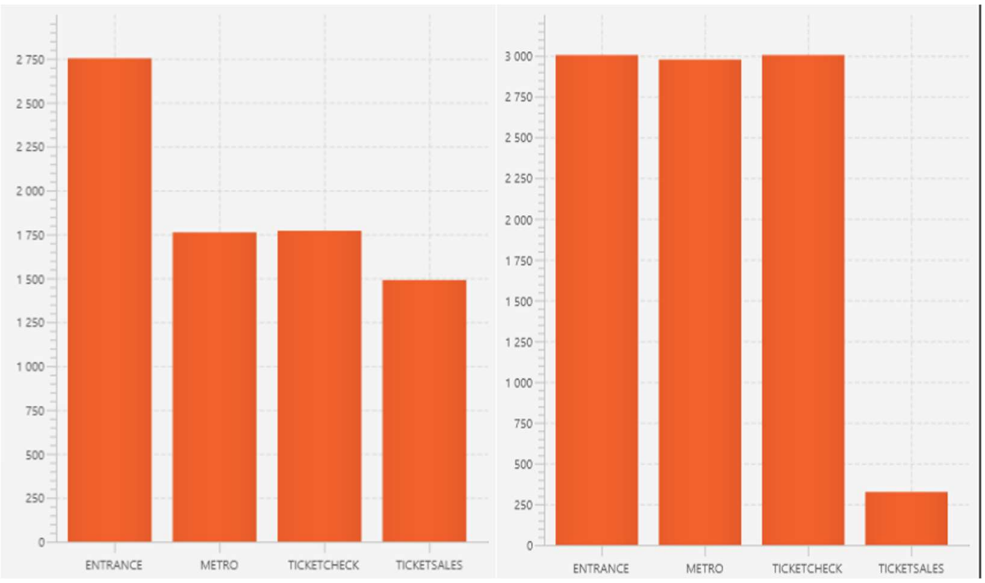
(18)

7 Tehdyt simulointikokeet

Kerrotaan, mitä kokeiltiin ja mitä saatiin selville.

Tähän osioon kannattaa panostaa esittämällä erilaisia simulointiajoja.

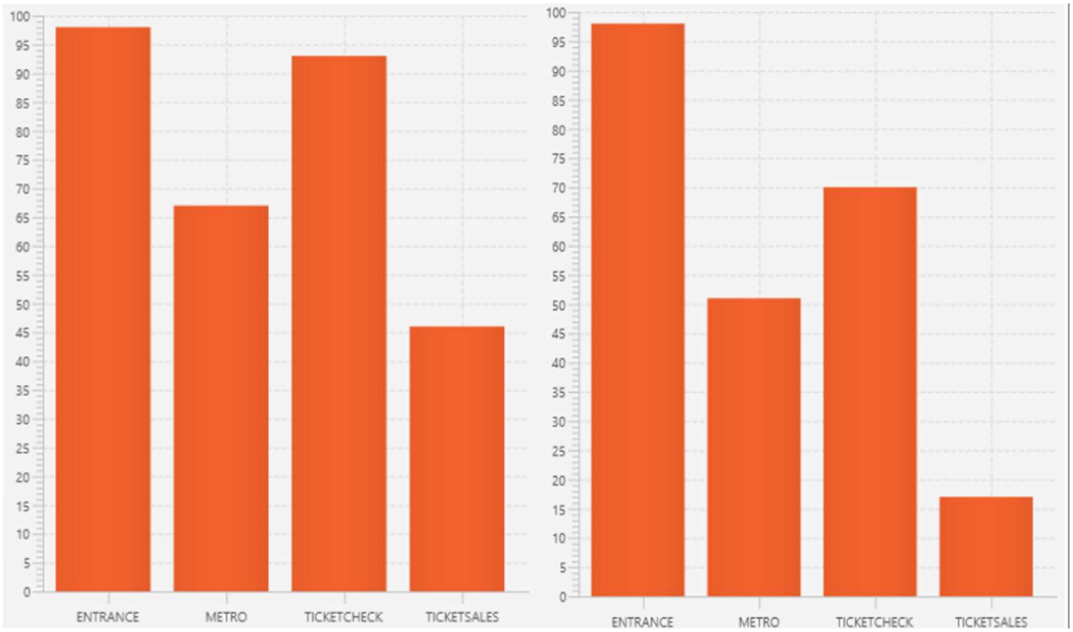
7.1 Mobiili lippujen määrän vaikutus asiakkaiden palveluun



Kuva 17. Vasen taulukko kuvastaa palveltujen asiakkaiden määrää, kun asiakkailla on 10% todennäköisyys olla mobiililippu. Oikea taulukko kuvastaa kun tämä todennäköisyys on 90%.

Kuvan 1 kuvastaa, että kun korkea määrä asiakkaita saapuu ilman mobiili lippuja, niin se aiheuttaa tungoksia metro aseman kulkuun. Tämän voi havaita katsomalla ENTRANCEN ja METRO tapahtumien asiakkaiden palvelumäärien erotusta, sillä tämä kuvastaa kuinka monta asiakasta on vielä järjestelmässä / ei palveltu.

7.2 Kuinka lipunmyynnin pituus vaikuttaa aseman tehokkuuteneen



Kuva 18. Kuvassa vasen taulukko on simulaatio oletusarvoilla. Oikeassa ainoa muutos on myyntiajan pidentäminen 40 sekunnilla. Oletus simulaatiossa puolilla asiakkaista on puhelin lippu.

Kuvasta 2 huomataan kuika suuren eron lipun myynnin pituus voi aiheuttaa asiakkaiden läpimenoon. 40s muutos aiheuttaa sen että palvelutjen asiakkaiden määrä noin 16.5 minuutin simulaatiossa vähenee noin 65 vain 51.

8 Yhteenveto

Simulaattoria lähdettiin rakentamaan ajatuksella, että se voisi mallintaa metroaseman kulkua tarpeeksi tarkasti, että sitä voisi hyödyntää metro asemien suunniteluun. Uskomme että hallinta, jonka annamme käyttäjälle riittää saavuttamaan funktionaalisuuden, ja visuaalisen selkeyden päämärämme. Simulaatiokokeissamme esittelimme esimerkkejä miten simulaaattoriamme voi käyttää metro asemien suorituskyvyn lisäämiseen.