jamk

Fyysisten objektien simulointi kyberharjoituksessa

Teemu Lindström

Opinnäytetyö, AMK
Kesäkuu 2025
Tieto- ja viestintätekniikan tutkinto-ohjelma (AMK)

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Lindström, Teemu

Fyysisten objektien simulointi kyberharjoituksessa

Jyväskylä: Jyväskylän ammattikorkeakoulu. Kesäkuu 2025, 31 sivua.

Tieto- ja viestintätekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: kyllä

Tiivistelmä

Kyberharjoituksissa harjoitellaan todellisiin kyberuhkatilanteisiin perustuvia häiriötilanteita ja oikeaoppisia toimintatapoja niiden torjumiseksi. Yksi yleisimmistä harjoitusmuodoista on live-harjoitus, jossa harjoitusympäristö rakennetaan suljettuun virtuaaliympäristöön. Näissä ympäristöissä pyritään mallintamaan uhkatilanteita vastaavia tietojärjestelmiä, ja harjoituksen laadun sekä tehokkuuden kannalta merkittävä tekijä on realismi. Koska kyberuhkien vaikutukset voivat ulottua digitaalisen ympäristön ulkopuolelle, fyysiset tekijät on otettava huomioon myös harjoitusympäristön suunnittelussa.

Järjestetyssä logistiikan toimialan kyberharjoituksessa mallinnettiin logistiikka-alan toimitusketjuja, jotka sisälsivät fyysisiä kuljetuksia. Osana toimitusketjujen mallinnusta oli tarve kehittää simulaatiojärjestelmä kuljetuksien fyysisten objektien liikuttamiseen. Simulaatiojärjestelmä toteutettiin tutkimuksellisen kehittämistyön tuloksena. Tutkimuksessa selvitettiin myös fyysisten objektien simuloinnin mahdollisuuksia ja hyötyjä harjoituksen näkökulmasta.

Simulaatiojärjestelmän pääohjelmointikielenä käytettiin TypeScriptiä, ja kehittämisessä hyödynnettiin avoimen lähdekoodin teknologioita ja kirjastoja. Objektien liikkeen hallintaan käytettiin Graphile Worker -työjonoa yhteistyössä PostgreSQL-tietokannan kanssa. Objektien liikkuminen perustui kuvitteellisten toimijoiden runkolinjoihin ja reittilaskentaan, jota varten käytettiin Valhalla-reittihakumoottoria sekä GeoJSON Path Finder -kirjastoa. Palveluiden välinen tiedonsiirto toteutettiin PostGraphilen avulla kehitetyn API-rajapinnan kautta. Käyttöliittymä rakennettiin Vue.js-ohjelmistokehyksellä, ja karttanäkymät toteutettiin Leaflet-kirjastolla.

Tuloksena saatiin kehitettyä vaatimusmäärittelyä vastaava simulaatiojärjestelmä, jonka avulla toimitusketjujen kuljetuksia onnistuttiin mallintamaan. Tulokset osoittivat, että fyysisten objektien simulointi on mahdollista osana kyberharjoituksia, ja sen toteuttaminen tukee harjoituksen tilannekuvaa sekä päätöksenteko. Tuloksia voidaan hyödyntää mallipohjana tulevissa harjoituksissa, joissa tunnistetaan tarve fyysisten tekijöiden mallinnukselle.

Avainsanat (asiasanat)

Kyberturvallisuus, Kyberharjoitus, Paikkatietojärjestelmä, GIS, Paikkatieto, Simulointi

Muut tiedot (salassa pidettävät liitteet)

Lindström, Teemu

Simulation of physical objects in a cyber exercise

Jyväskylä: JAMK University of Applied Sciences, June 2025, 31 pages

Degree Programme in Information and Communication Technology. Bachelor´s thesis.

Permission for open access publication: Yes

Language of publication: Finnish

Abstract

Cyber exercises examine disruptions based on real-life cyber threat scenarios and provide training on appropriate actions to counter them. One of the most common types of exercises is the live exercise, where the training environment is built in a closed virtual environment without direct connection to the physical world. In these environments information systems are aimed to be modeled to match the threat scenarios, and realism is a significant factor in terms of the exercise's quality and effectiveness. Since the effects of cyber threats can extend beyond the digital environment, physical aspects must also be considered in the design of the exercise environment.

In a cyber exercise for logistics industry, logistical supply chains and their physical transports were modeled. As part of this modeling, there was a need to develop a simulation system for handling the movement of the physical objects involved in transportation. The simulation system was developed as a result of a research-based development. The research also examined the possibilities and benefits of simulating physical objects in a cyber exercise.

The main programming language used in the simulation system was TypeScript, and the development made use of opensource technologies and libraries. Objects movement was handled with Graphile Worker job queue in combination with a PostgreSQL database. The movement of objects was based on fictional actor's trunk lines and route calculations, using the Valhalla routing engine and GeoJSON Path Finder library. Communication between services was implemented via an API-interface developed with PostGraphile. The user interface was built using the Vue.js framework, and map views were created with the Leaflet library.

As a result of the research, a simulation system matching the requirements specification was successfully developed, allowing the simulation of transportation within supply chains. The results showed that simulating physical objects as part of a cyber exercise is feasible and that its implementation supports the overview and decision-making in the exercise. The results can be used as a model for future exercises involving physical elements.

Keywords/tags (subjects)

Cyber security, Cyber exercise, Geographic Information System, GIS, Geographic information, Simulation

Miscellaneous (Confidential information)

Sisältö

1 Johdanto 6
2 Tutkimusasetelma 7
3 Tietoperusta 8
3.1 Paikkatieto 8
3.1.1 Paikkatietojärjestelmä (GIS) 9
3.1.2 Reittihaku algoritmit 10
3.2 Kyberharjoitus 11
3.3 Käytetyt teknologiat ja kirjastot 12
3.3.1 TypeScript 12
3.3.2 Docker 13
3.3.3 Node.js 13
3.3.4 PostgreSQL 13
3.3.5 Reittihakumoottorit 14
3.3.6 Vue.js 15
3.3.7 PostGraphile 15
3.3.8 Graphile Worker 16
4 Simulaatiojärjestelmän toteutus 16
4.1 Vaatimusmäärittely 16
4.2 Reittihaku simulaatiossa 17
4.3 Simulaatio 20
4.4 Käyttöliittymän toteutus 22
4.5 API-rajapinta 26
5 Tulokset 28
6 Pohdinta 28
Lähteet 30

Kuviot

Kuvio 1. Valhalla reittihaku prosessi (Valhalla n.d.) 14
Kuvio 2. Esimerkki toimitusketjusta 18
Kuvio 3. Runkolinjaverkoston malli 19
Kuvio 4. Entiteetin tietomalli 20
Kuvio 5. Käyttöliittymän Entiteetit-välilehden päänäkymä 23
Kuvio 6. Entiteetin luonti käyttöliittymässä 23

Kuvio 7. Käyttöliittymän entiteettinäkymä 24
Kuvio 8. Käyttöliittymän entiteetin liikutustyökalu 25
Kuvio 9. Käyttöliittymän Kartta-välilehden päänäkymä 26
Kuvio 10. Palveluiden yhteydet rajapinnan kautta 27

1 Johdanto

Kyberturvallisuus on viime vuosina noussut yhä merkittävämpään rooliin yhteiskunnan turvallisuuden kannalta. Teknologian jatkuvan kehityksen sekä yhteiskunnan digitalisoitumisen myötä kyberuhkiin joudutaan varautumaan entistä laajemmin. Tämän lisäksi viimeaikainen horjuva maailmantilanne ja kiristyneet geopoliittiset jännitteet ovat tuoneet uusia haasteita kansainväliseen kyberturvallisuuteen. Suomen uusitussa kyberturvallisuusstrategiassa yksi keskeisimmistä tavoitteista on varautuminen ja siihen liittyen harjoitustoiminnan kehittäminen. (Suomen kyberturvallisuusstrategia 2024–2035 2024.)

Kyberharjoituksissa mallinnetaan todellisiin kyberuhkiin perustuvia häiriötilanteita, joiden juurisyitä ja vaikutuksia voidaan tutkia sekä harjoitella oikeaoppisia toimintatapoja niiden torjumiseksi. Yksi yleisimmistä harjoittelutavoista on live-harjoitus, jossa tilanteen mukaisia tietojärjestelmiä ja niihin kohdistuvia häiriöitä mallinnetaan suljetussa virtuaaliympäristössä. Harjoitukset keskittyvät pääsääntöisesti tietoturvaan ja digitaalisiin toimintaympäristöihin ilman suoraa yhteyttä fyysiseen maailmaan. (Karjalainen 2021; Kyberharjoitusohje 2019.)

Todellisuuteen perustuvien kyberuhkien vaikutukset voivat heijastua digitaalisten toimintaympäristöjen ulkopuolelle, kuten infrastruktuuriin, logistiikkaan tai energiantuotantoon (Kyberharjoitusohje 2019). Näiden fyysisten tekijöiden huomioiminen harjoitusympäristössä on harjoituksen tehokkuuden, monipuolisuuden ja realismin kannalta tärkeää.

Opinnäytetyön toimeksiantaja oli tunnistanut useita tarpeita fyysisten objektien simuloinnille harjoitusympäristöissä. Työn tavoitteena oli kehittää simulaatiojärjestelmä osaksi logistiikan toimialan kyberharjoitusta ja tutkia sen avulla fyysisten objektien simuloinnin mahdollisuuksia ja hyötyjä kyberharjoituksissa.

2 Tutkimusasetelma

Kyberharjoituksissa on tyypillistä, että ne toteutetaan suljetussa virtuaaliympäristössä. Näissä harjoituksissa suuri tekijä harjoituksen laadun kannalta on ympäristön todenmukaisuus, joten digitaalinen toimintaympäristö toteutetaan vastaamaan todellisuutta. Harjoituksissa mallinnettavien uhkatilanteiden vaikutukset voivat kuitenkin heijastua tämän digitaalisen toimintaympäristön ulkopuolisiin fyysisiin tekijöihin. Tästä syntyy tarve huomioida fyysiset tekijät osaksi kyberharjoitusympäristöä.

Tässä opinnäytetyössä tutkittiin näiden fyysisten tekijöiden mallintamista teknisen toteutuksen kautta. Työssä kehitettiin simulaatiojärjestelmä osaksi logistiikan toimialan kyberharjoitusta, jonka tavoitteena oli mallintaa logistiikan toimitusketjujen kuljetuksia simuloimalla fyysisten objektien, kuten ajoneuvojen, laivojen sekä pakettien liikettä paikkatietopohjaisesti. Tutkimus tehtiin teknisestä sekä hyödyllisyyden näkökulmasta seuraavien tutkimuskysymysten ohjaamana:

1. ”Voidaanko fyysisiä objekteja simuloida kyberharjoitusympäristössä?”
2. ”Voidaanko fyysisten objektien simuloinnilla tukea kyberharjoituksen tilannekuvaa ja päätöksentekoa?”

Tämän työn tutkimusmenetelmänä käytetään tutkimuksellista kehittämistyötä. Toimeksiantajalla on selkeä tarve kehitystyölle aiheeseen liittyen. Työssä yhdistetään aiheen teoria, konkreettinen tuotteen kehitys sekä tulosten analyysi. Kehittämistoimintaa ohjaavat tutkimuskysymykset ja tuotteen vaatimusmäärittely. Näillä ominaisuuksilla työssä tuodaan esille käytännönläheisyys, tutkimuksellisuus ja tiedontuotanto. (Toikko & Rantanen 2009.)

Opinnäytetyön tutkimuksen kohteena oli toteutettu simulaatiojärjestelmä, joten tutkimuskysymyksiä ja tuloksia tarkastellaan järjestelmän kehityksen kautta. Tämän vuoksi työn tietoperusta ja toteutus on rajattu sisältämään vain järjestelmän toiminnallisuuteen liittyvät aihealueet.

3 Tietoperusta

3.1 Paikkatieto

Paikkatiedolla tarkoitetaan dataa, jolla kuvataan minkä tahansa kappaleen, ilmiön tai tapahtuman sijaintia maapallon suhteen. Sijaintitieto voidaan ilmoittaa suorasti erilaisten koordinaattijärjestelmien avulla tai epäsuorasti, kuten osoitetiedolla tai paikantavalla tunnuksella. Sijaintitiedon lisäksi paikkatieto voi sisältää kohteen muita kuvaavia tai ajallisia ominaisuuksia. Esimerkiksi tiettyä kiinteistöä kuvaava paikkatieto voisi sisältää sijainnin lisäksi kiinteistötunnuksen, tai liikkuvalla kappaleella voisi olla aikaleima, joka kertoo viimeisimmän sijainnin päivityksen ajankohdan. (Geoinformatiikan sanasto 2018.)

Geokoodaus (engl. geocoding) on prosessi, jossa epäsuora sijaintitieto, kuten katuosoite, muunnetaan maantieteellisiksi koordinaateiksi. Prosessissa hyödynnetään tietokantoja, jotka sisältävät epäsuoran ja suoran sijaintitiedon välisen yhteyden. Geokoodausprosessille annetaan lähtötietona epäsuora sijaintitieto, jota se vertaa tietokantaan ja palauttaa parhaimman osuman. Geokoodausta hyödynnetään esimerkiksi analysoitaessa tutkimuksia, joissa tuloksina on lista katuosoitteita. Osoitteet voidaan yhdistää maantieteelliseen sijaintiin ja visualisoida kartalle. (What is geocoding? n.d.)

Paikkatieto jaetaan tyypillisesti kahteen pääryhmään: vektoridataan ja rasteridataan. Vektoridatassa kohteen sijaintia kuvataan koordinaatiston paikkavektoreilla, jotka voivat olla pisteitä, viivoja tai monikulmioita. Pisteellä voidaan kuvata tarkkoja pistemäisiä sijainteja, kuten puita, kylttejä tai pylväitä. Viivoilla voidaan kuvata reittejä tai kaksiulotteisia kohteita ilman leveyttä, kuten teitä, aitoja tai kaapeleita. Monikulmioita käytetään kuvaamaan erilaisia alueita ja pinta-aloja, kuten kiinteistöjä, vesistöjä tai rakennusten muotoja. (What is geospatial data? n.d.)

Rasteridatassa kohdealue jaetaan säännölliseen ruudukkoon, jossa yksi solu tai pikseli kuvaa tiettyä maantieteellistä aluetta. Solun x- ja y-koordinaatit toimivat sijaintitietona. Lisätietona solulle voidaan liittää erilaisia arvoja, kuten lämpötila, korkeus tai väri. Rasteridataa käytetään tyypillisesti vektoridatan taustakarttana tai peitekuvana, ja se soveltuu hyvin alueellisten muutosten visualisointiin. (What is raster data? n.d.)

3.1.1 Paikkatietojärjestelmä (GIS)

Paikkatietojärjestelmästä käytettävä lyhenne GIS tulee englanninkielisestä termistä ”Geographic Information System”. Se tarkoittaa yksinkertaisesti järjestelmää, joka käsittelee paikkatietoa. Järjestelmällä tarjotaan työkaluja paikkatiedon tallentamiseen, hallintaan, analysointiin ja esittämiseen. Paikkatietojärjestelmiä hyödynnetään laajasti eri aloilla tiedonhallintaan, ylläpitoon ja suunnitteluun. Sen käyttökohteita voivat olla esimerkiksi kaupunki- ja maankäytön suunnittelu, infrastruktuurin ylläpito, maanviljelyanalyysit tai reittihaku. (Geoinformatiikan sanasto 2018.)

Paikkatietojärjestelmän avulla erityyppiset paikkatiedot useista lähteistä voidaan keskittää yhteen digitaaliseen kokonaisuuteen, mikä mahdollistaa uusia ja monipuolisia tapoja käsitellä tietoa. Tiedonkäsittelyssä voidaan myös hyödyntää tietokoneiden laskentatehoa, jolloin suurten tietomäärien ja monimutkaisten analyysien käsittely nopeutuu. Näitä menetelmiä hyödyntämällä pystytään muun muassa visualisoimaan tietoa tarkemmalla tasolla, käsittelemään suuria määriä dataa, analysoimaan tuloksia monipuolisemmin ja tukemaan päätöksentekoa. (Fazal 2008.)

Esimerkkejä tunnetuista paikkatietojärjestelmistä ovat:

- **QGIS**. Avoimen lähdekoodin ohjelmisto, joka tarjoaa työkalut paikkatiedon visualisointiin ja päätöksentekoon.
- **ArcGIS**. Paikkatietoa käsittelevä alusta, joka mahdollistaa tiedon hallinnan, analysoinnin, kartoituksen ja jakamisen.
- **Karttapaikka**. Maanmittauslaitoksen palvelu, josta pystyy katsomaan Maanmittauslaitoksen paikkatietoaineistoja karttatasolla.

3.1.2 Reittihaku algoritmit

Edellä mainittuihin paikkatietojärjestelmän käyttökohteisiin kuuluu reittihaku. Reittihaulla tarkoitetaan palvelua, jonka avulla voidaan etsiä lyhyin tai kustannustehokkain reitti pisteiden välillä. Tyypillinen reittihaun käyttökohde on suunnistukseen käytettävät navigointi- ja paikannuslaitteet. Optimaalisen reitin laskentaan on kehitetty erilaisia graafiteoriaan pohjautuvia algoritmeja.

Graafilla tarkoitetaan joukkoa pisteitä ja niitä yhdistäviä viivoja. Graafin pisteitä kutsutaan solmuiksi (engl. vertices), ja toisinaan solmuista käytetään myös englanninkielistä termiä node. Solmujen välistä yhteyttä kuvaavaa viivaa kutsutaan kaareksi (engl. edge). Tyypillinen graafin käyttökohde on kuvata erilaisia verkostoja, kuten tietoliikenneverkkoja, kemiallisia kaavoja tai maantieteellisiä tieverkostoja. (Voloshin 2009.)

Graafien verkostot voivat olla painottamattomia tai painotettuja. Painottamattomissa verkoissa käytetyt algoritmit laskevat ainoastaan käytettyjen solmujen tai kaarien määrää. Reittilaskennassa käytetäänkin yleensä painotettua verkkoa, jolloin kaarille on määritelty kustannusta kuvaava painoarvo. Kustannusta voidaan ilmaista erilaisilla arvoilla, kuten matkan pituudella tai ajalla. Painotetun verkon reittilaskentaan on kehitetty useita algoritmeja eri käyttötarkoituksiin. Esimerkkejä painotetun verkon yksisuuntaisen reittilaskennan algoritmeista ovat Dijkstran algoritmi ja A*-algoritmi. (Voloshin 2009.)

Dijkstran algoritmi on Edsger W. Dijkstran vuonna 1959 julkaisemassa artikkelissa esittelemä reittilaskenta-algoritmi. Sitä pidetään laajasti ensimmäisenä tieteellisenä julkaisuna aiheesta. Dijkstran algoritmi on suunnattu painotetuille verkoille, joissa kaarien kustannukset ovat positiivisia lukuja. Algoritmin tehtävänä on etsiä lyhyin reitti valitusta solmusta kaikkiin muihin solmuihin. (Dijkstra 1959.)

A* on heuristinen reittilaskennan algoritmi, jonka vahvuutena on tarkkuuden ja tehokkuuden tasapaino. Se eroaa Dijkstran algoritmista hyödyntämällä kuljetun matkan lisäksi arviota jäljellä olevasta matkasta määränpäähän. Tämän lähestymistavan ansiosta se pystyy tilanteen mukaan priorisoimaan lupaavia reittejä ja välttämään huonolta vaikuttavia vaihtoehtoja. (Kumar 2024.)

3.2 Kyberharjoitus

Kyberharjoitus on tapahtuma, jossa osallistuvat organisaatiot harjoittelevat kyvykkyyttään vastata kyberhäiriöihin ja -uhkiin. Harjoittelemalla oikeita toimintatapoja, tunnistamalla riippuvuussuhteita ja riskikohtia sekä tutkimalla häiriöitä, organisaatiot voivat varautua todellisia kriisitilanteita varten. Kyberharjoitus keskittyy usein tietoturvaan ja tietojärjestelmiin, mutta myös viestintä ja johtaminen ovat merkittävässä roolissa. Yksittäisen järjestelmän häiriön vaikutus voi ulottua organisaation tuotantoketjussa pitkälle. Tällöin harjoitusympäristö ei välttämättä rajoitu vain tietoteknisiin järjestelmiin, vaan se voi myös huomioida fyysisen maailman ilmiöitä, kuten tavarantoimituksen ja vastaanoton. (Kyberharjoitusohje 2019.)

Erilaisia kyberharjoitustyyppejä on lukematon määrä (Kyberharjoitusohje 2019). Tämän työn kannalta olennaisin harjoitustyyppi on Karjalaisen (2021) väitöskirjan aiheena oleva ja kuvailema live-harjoitus. Live-harjoituksessa harjoitustilanne kehittyy jatkuvasti ennalta määritellyn harjoitusskenaarion käsikirjoituksen mukaan. Harjoitus rakennetaan osaksi eristettyä virtuaalista harjoitusympäristöä (engl. cyber range). Tämä ympäristö on interaktiivinen alusta, joka mallintaa organisaatioiden todellista ympäristöjä, kuten verkkoyhteyksiä, tietojärjestelmiä ja sovelluksi. Se on fyysisestä maailmasta eristetty kokonaisuus, joka mahdollistaa turvallisen ja riskittömän harjoittelun. (Karjalainen 2021; The Cyber Range: A Guide 2023.)

Harjoitusskenaario on todelliseen uhkaan perustuva kuvitteellinen tilanne. Se on harjoituksen tarinan juoni, jonka avulla kuvataan tapahtumien taustat ja lähtötilanne. Skenaario rakennetaan yleensä osallistuvien organisaatioiden tarpeiden tai nykytilanteen mukaiseksi. (Kyberharjoitusohje 2019.)

Live-harjoituksen osallistujat jaetaan tyypillisesti värikoodattuihin tiimeihin, joilla jokaisella on oma rooli harjoituksen kannalta. Kyberharjoitusohje (2019) ja Karjalainen (2021) määrittävät tiimit seuraavasti: harjoituksen järjestäjät ja harjoittelevien organisaatioiden yhteyshenkilöt muodostavat valkoisen tiimin (WT), jonka vastuulla on harjoituksen tapahtumien ylläpito ja hallinta. Varsinaiset harjoittelijat muodostavat sinisen tiimin (BT), jonka tehtävänä on toimia puolustavassa roolissa ja ratkaista häiriöistä ja hyökkäyksistä johtuvia ongelmia. Suuremmissa harjoituksissa sinisiä tiimejä voi olla useita. Punainen tiimi (RT) toimii uhkatoimijan ja hyökkääjän roolissa; sen tehtävä on toi-

mia vastapelaajana sinistä tiimiä vastaan toteuttamalla hyökkäyksiä. Vihreä tiimi (GT) vastaa harjoitusympäristön rakentamisesta ja ylläpidosta. Harjoituksen aikana vihreän tiimin tehtävänä on toimia teknisenä tukena muille tiimeille. (Kyberharjoitusohje 2019; Karjalainen 2021.)

Tarinan kulkua harjoituksessa ohjataan tapahtumien ja syötteiden avulla. Syötteet ovat tarinan käsikirjoituksen mukaisia viestejä harjoittelijoille, joilla kerrotaan tarinan kulusta ja tapahtumista. Syöte voidaan toteuttaa eri muodoissa, kuten sähköpostina, puheluna, kirjeenä tai uutisartikkelina. Syöte voi olla esimerkiksi tehtaan työntekijän sähköpostiviesti tukipalveluihin, jossa todetaan, että tietojärjestelmä ei toimi. Syötteen ajankohta, lähettäjä, vastaanottaja, viestimuoto ja sisältö suunnitellaan etukäteen, jotta harjoituksen johto voi seurata sujuvasti niiden lähetystä ja vaikutuksia. Syötteitä suunniteltaessa tulee huomioida niiden ajoitus ja aiheuttama työmäärä, jotta vältytään harjoittelijoiden ylikuormittamiselta. (Kyberharjoitusohje 2019.)

3.3 Käytetyt teknologiat ja kirjastot

3.3.1 TypeScript

TypeScript on Microsoftin kehittämä, ilmainen ja avoimeen lähdekoodiin perustuva ohjelmointikieli. Se pohjautuu JavaScript-ohjelmointikieleen ja laajentaa sen syntaksia vahvalla staattisella tyypityksellä. TypeScriptin tehtävänä on tarkistaa, että koodi on tyypitetty oikein ennen sen ajoa. (TypeScript n.d.; Handbook: The TypeScript Handbook 2025.)

JavaScriptissä tyypitys toteutetaan dynaamisesti, jolloin koodia ajettaessa JavaScript päättelee itse, miten arvoja käsitellään. Tämän vuoksi koodin toiminnasta ei voida olla varmoja ennen sen suorittamista. TypeScriptin staattisen tyypityksen avulla voidaan varmistaa koodin toimivuus halutulla tavalla ja samalla parantaa sen luettavuutta. (Handbook: The Basics 2025.) Verratessa staattista ja dynaamista tyypitystä voidaan päätellä, että staattisen tyypityksen oikeaoppinen käyttö vaatii pidemmän koodin kirjoitusta ja erityistä huomiota tyypityksiin. Koodin ylläpidon, luettavuuden ja toimivuuden kannalta tämä lisätyö on kuitenkin ajan myötä ja koodipohjan kasvaessa kannattavaa.

3.3.2 Docker

Docker on Go-ohjelmointikielellä kirjoitettu alusta, jonka avulla voidaan pystyttää virtuaalisia ympäristöjä järjestelmien kehitystä ja ajoa varten. Näitä virtuaaliympäristöjä kutsutaan konteiksi. Kontit ovat kevyitä, osittain isäntäkoneesta eristettyjä ympäristöjä, jotka sisältävät kaiken tarvittavan järjestelmän ajamista varten. Kontit ovat pääosin eristettyjä, mutta tieto- ja verkkoyhteyksiä isäntäkoneen ja muiden konttien välillä voidaan konfiguroida tarpeiden mukaan. Docker tarjoaa konttien hallintaan työkaluja, joilla voidaan muun muassa lisätä, poistaa, käynnistää tai pysäyttää kontteja. (What is Docker? n.d.)

3.3.3 Node.js

Node.js on Google Chromen V8 JavaScript -moottoria hyödyntävä avoimen lähdekoodin JavaScript-suoritusympäristö palvelimille. Sen toiminta perustuu yhdessä säikeessä toimivaan prosessiin, jossa koodin ei-estävä suorituskyky toteutetaan asynkronisilla I/O-toiminnoilla. Näillä toiminnoilla vältetään koodia suorittavan säikeen tukkeutuminen ja mahdollistetaan samanaikaisten pyyntöjen käsittely. (Introduction to Node.js n.d.)

3.3.4 PostgreSQL

PostgreSQL on avoimen lähdekoodin relaatiotietokantajärjestelmä, joka perustuu standardoituun SQL-kyselykieleen ja sen laajennuksiin. Se on tunnettu erityisesti kestävyydestään, luotettavuudestaan ja laajennettavuudestaan. PostgreSQL toimii kaikilla yleisimmillä käyttöjärjestelmillä, ja sen kehityksen taustalla on pitkäaikainen ja aktiivinen yhteisö. (About PostgreSQL n.d.)

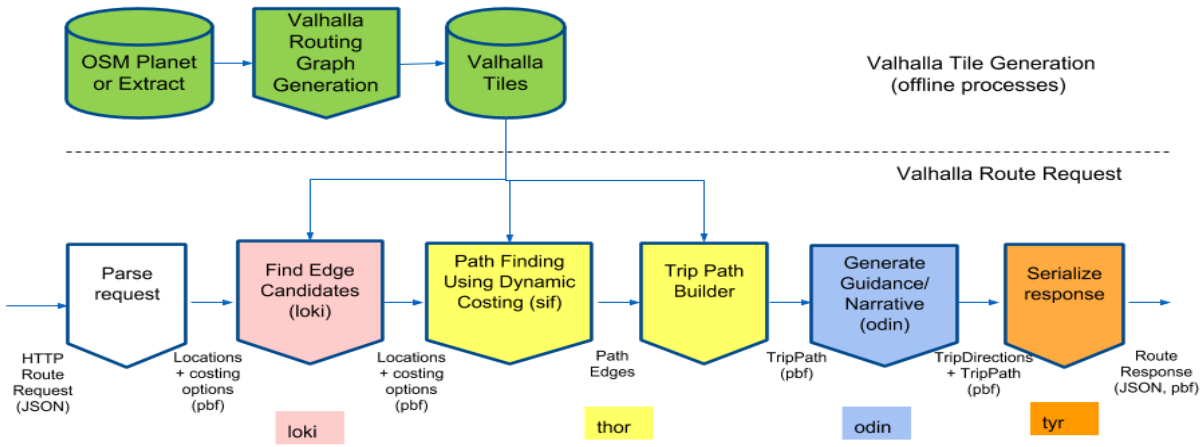
PostgreSQL tukee tiedon eheyttä noudattamalla ACID-periaatteita (About PostgreSQL n.d.). Termi ACID viittaa neljään keskeiseen ominaisuuteen, joilla tietokannan transaktioiden luotettavuus taataan. ACID muodostuu englanninkielisistä sanoista atomicity, consistency, isolation ja durability, jotka voidaan suomentaa muotoon atomisuus, eheys, eristyneisyys ja pysyvyys. (Haerder & Reuter 1983, 289–290.)

PostgreSQL:n vahvuuksiin kuuluu sen laajennettavuus. Se tarjoaa käyttäjille valmiita työkaluja ja toimintoja, joiden avulla on mahdollista rakentaa omia tietotyyppejä ja funktioita suoraan tietokannan sisällä. PostgreSQL tukee myös laajasti yhteisön kehittämiä lisäosia, kuten PostGIS. PostGIS-lisäosa laajentaa tietokannan ominaisuuksia paikkatiedon tallentamiseen, analysointiin ja käsittelyyn. (About PostgreSQL n.d.)

3.3.5 Reittihakumoottorit

Avoimeen lähdekoodiin ja paikkatietoaineistoon perustuvia reittihakumoottoreita on laaja valikoima. Nämä moottorit kuitenkin eroavat arkkitehtuuriltaan ja ominaisuuksiltaan, ja ne erottuvat edukseen omilla osa-alueillaan muihin moottoreihin verrattuna. Tässä luvussa esitetään tämän työn kannalta olennaiset reittihakumoottorit Valhalla ja GeoJSON Path Finder.

Valhalla on C++-ohjelmointikielellä toteutettu reittihakumoottori, joka hyödyntää OpenStreetMap-yhteisön tuottamaa paikkatietoaineistoa. Se tarjoaa REST-rajapinnan avulla perinteisen reittihaun lisäksi työkaluja esimerkiksi sijainnin geokoodaukseen sekä korkeus- ja metatietojen hakuun. Valhallan käyttämä laatoitettu (engl. tiled) reititysverkko jakaa verkon pieniin, hierarkkisiin alueisiin, mikä mahdollistaa kevyen muistinkäytön, offline-reitityksen sekä dynaamisen verkon päivityksen. Kuvio 1 esittää Valhallan reittihaun prosessin. (Valhalla n.d.)



Kuvio 1. Valhalla reittihaku prosessi (Valhalla n.d.)

GeoJSON Path Finder on JavaScript-kirjasto, joka on suunniteltu kevyempään reittihakuun. Reititysverkko määritellään kirjastolle GeoJSON-muodossa, jonka perusteella se rakentaa graafin reittihaun suorittamista varten. Reittihaun yhteydessä kirjastolle syötetään verkostoon sijoittuvat lähtö- ja päätepisteet koordinaatteineen GeoJSON-muodossa. Reittihaun tuloksena kirjasto palauttaa kuljettavan reitin koordinaatit sekä reitin kustannuksen. (GeoJSON Path Finder n.d.)

3.3.6 Vue.js

Vue.js on JavaScript-ohjelmistokehys käyttöliittymien rakentamiseen. Se tarjoaa käyttöliittymien kehitykseen komponenttipohjaisen mallin, jossa JavaScript-, HTML- ja CSS-koodi yhdistetään yhdeksi kokonaisuudeksi. Komponenttien avulla data sidotaan näytettäviin elementteihin, ja Vue:n reaktiivinen järjestelmä huolehtii siitä, että vain tarvittavat käyttöliittymän osat päivittyvät datan tilan muuttuessa. (Vue.js - Introduction n.d.)

3.3.7 PostGraphile

PostGraphile on avoimen lähdekoodin työkalu GraphQL-rajapintojen toteuttamiseen hyödyntämällä PostgreSQL-tietokannan rakennetta. Sen toiminta perustuu automaattiseen tietokannan analysointiin, joka jäljittää tietokannan rakenteen tauluja, suhteita ja funktioita myöten. PostGraphile luo analysoidun tietokannan rakenteen pohjalta GraphQL-rajapinnan ja dokumentaation. PostGraphile tunnistaa tietokannan muutokset ja päivittää rajapinnan rakenteen reaaliajassa ilman tarvetta uudelleenkäynnistykselle. Lisäksi se tarjoaa työkalut rajapinnan laajentamiseen omilla tietotyypeillä ja kyselyillä. (PostGraphile Instant GraphQL API n.d.)

GraphQL on avoimeen lähdekoodiin perustuva kyselykieli ja palvelinpuolen suoritusympäristö. Se käyttää vahvaa tyypitystä määrittelemään, millaista tietoa rajapinnasta voidaan hakea ja muokata. Tyypityksen ansioista rajapinnan käyttäjä voi vaikuttaa saadun tiedon määrään ja laatuun. GraphQL tukee tiedonkäsittelyn operaatioissaan tiedonhakua ja -manipulaatiota sekä reaaliaikaista päivitystä. (Introduction to GraphQL 2025.)

3.3.8 Graphile Worker

Graphile Worker on avoimen lähdekoodin JavaScript-kirjasto, joka suorittaa työjonoa (engl. job queue) PostgreSQL-tietokantaa hyödyntäen. Työjonon toiminta perustuu tietokantaan tallennettaviin tehtäviin, joita Graphile Worker lukee ja suorittaa. Tiedon eheys tehtäviä suorittaessa varmistetaan käyttämällä tietokannan transaktioita, tehtävien nykytilaa kuvaavia tiloja sekä automaattista tehtävien uudelleensuoritusta niiden epäonnistuessa. (Graphile Worker - Introduction n.d.)

Graphile Workerin monipuolisen muokattavuuden avulla voidaan hallita, miten ja milloin tehtävät suoritetaan. Konfiguraatioiden avulla voidaan vaikuttaa ohjelmiston suorituskykyyn esimerkiksi lisäämällä tehtävien rinnakkaista suorittamista. Tietokannassa voidaan myös suorittaa useita Graphile Worker -prosesseja samanaikaisesti, mikä mahdollistaa järjestelmän skaalattavuuden laitteistopohjaisesti. (Graphile Worker - Introduction n.d.)

4 Simulaatiojärjestelmän toteutus

4.1 Vaatimusmäärittely

Toimeksiantajan järjestämässä logistiikan toimialan kyberharjoituksessa mallinnettiin toimitusketjuja ja niihin liittyviä kuljetuksia, joissa esiintyi fyysisiä objekteja, kuten ajoneuvoja, laivoja ja paketteja. Näiden objektien simulointia varten suunniteltiin simulaatiojärjestelmä, jossa objekteja käsiteltiin entiteetteinä. Järjestelmän tarkoituksena oli mallintaa kuljetuksia paikkatietopohjaisesti simuloimalla entiteettien sijaintia ja liikettä reittien mukaisesti, mahdollistaa liikkeen hallinta sekä esittää tuotettua tietoa käyttöliittymässä. Suunnittelussa otettiin huomioon sekä järjestelmän että kyberharjoituksen vaatimukset, jotka ohjasivat arkkitehtuurin ja teknologioiden valintaa.

Kyberharjoituksen perusteella keskeisiksi vaatimuksiksi nousivat tarve mallintaa toimitusketjujen kuljetuksia paikkatietopohjaisesti sekä arkkitehtuuriin liittyvät näkökohdat, kuten käyttöönotto, hallinta, skaalautuvuus ja integraatio muiden järjestelmien kanssa. On tavallista, että harjoituksia varten kehitettyjä järjestelmiä käytetään uudelleen, jatkokehitetään tai hyödynnetään pohjana uusille järjestelmille. Tämä johtaa siihen, että järjestelmissä suositaan vakiintuneita arkkitehtuuriratkaisuja ja teknologioita. Toteutuksessa käytetyt teknologiat, kuten TypeScript-ohjelmointikieli ja PostgreSQL-tietokanta, olivat jo yleisesti käytössä toimeksiantajan järjestelmäkehityksessä, ja niitä

hyödynnettiin myös tämän järjestelmän toteutuksessa. PostgreSQL:n käyttöä puolsi myös sen tarjoama PostGIS-laajennus, joka sisältää työkaluja paikkatiedon käsittelyyn.

Liikkumisen haluttiin toimivan dynaamisesti ja realistisesti, riippumatta sijainnista tai objektista. Tämän vuoksi toteutuksessa tuli hyödyntää reittihakua. Pitkien ja monimutkaisten reittien tarkka laskenta edellyttää laajoja paikkatietoaineistoja ja tehokkaita taustajärjestelmiä. Koska laskentaprosessi voi olla erittäin kuormittava, reittilaskenta päätettiin erottaa omaksi palvelukseen.

Usean entiteetin samanaikainen liikkeen simulointi vaatii laskentatehoa. Järjestelmän skaalautuvuuden näkökulmasta oli tärkeää, että entiteettien määrän kasvaessa laskentatehoa voidaan lisätä. Skaalautuvuuden mahdollistamiseksi myös simulaatio eriytettiin omaksi palveluksi.

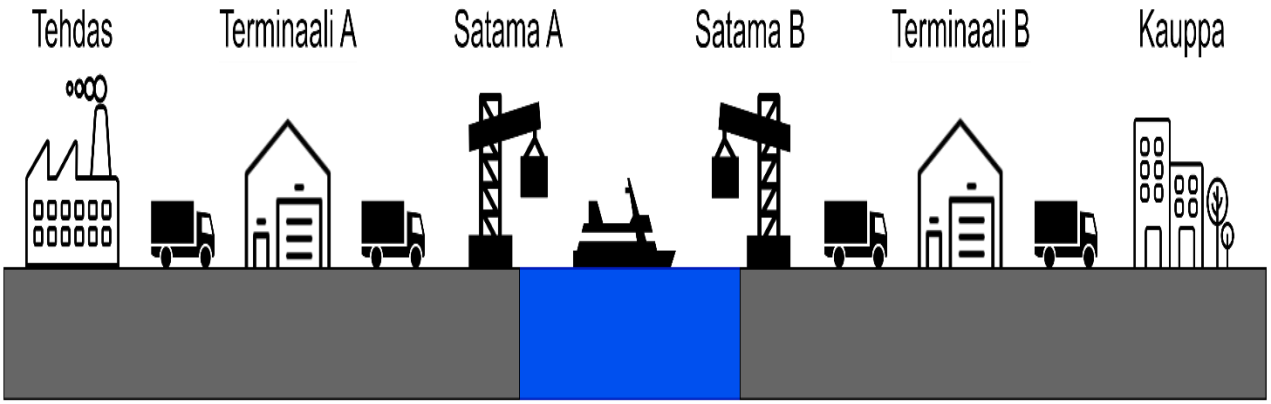
Näiden lähtötietojen pohjalta järjestelmän sisältö pystyttiin jakamaan viiteen toiminnalliseen osaan:

- **Käyttöliittymä** – Visualisointi ja hallintatyökalut
- **API (Application Programming Interface) -rajapinta** – Ohjaa tiedonsiirtoa komponenttien välillä
- **Tietokanta** – Sisältää paikkatiedot
- **Reittilaskenta** – Suorittaa reittilaskentaan liittyvät prosessit
- **Simulaatio** – Suorittaa entiteettien liikkumisen laskentaan liittyvät prosessit

4.2 Reittihaku simulaatiossa

Oman reittilaskentamoottorin kehitys olisi työlästä, joten rajoitetun ajan puitteissa vapaasti saatavilla olevat avoimeen lähdekoodiin perustuvat reittilaskentamoottorit todettiin toimivaksi vaihtoehdoksi. Moottorin valinnassa painotettiin erityisesti seuraavia ominaisuuksia: käytettävyys, laitevaatimukset, suorituskyky sekä tuki eri reitityksprofiileille, kuten auto, polkupyörä tai jalankulkija. Näiden kriteerien perusteella kevyellä testauksella ja ominaisuuksien vertailulla valittiin moottoriksi Valhalla. Valhalla tarjosi valmiit pohjat ja ohjeistuksen palvelun ajamiseen Docker-kontissa, mikä mahdollisti palvelun vaivattoman ja nopean käyttöönoton. Valhallan toiminta perustuu OpenStreetMap-yhteisön tuottamaan paikkatietoaineistoon (© OpenStreetMap contributors), joka on lisensoitu Open Database License (ODbL) -ehdoin.

Toimitusketjuja mallinnettaessa tuli huomioida ketjun alku-, väli- ja päätepisteet sekä pisteiden välillä vaihtuvat kuljetustavat. Esimerkiksi kuvion 2 mukaisesti tuotteen siirtyessä tehtaalta kaupan hyllylle toimitusketju voidaan jakaa usean toimijan yksittäisiin kuljetuksiin.

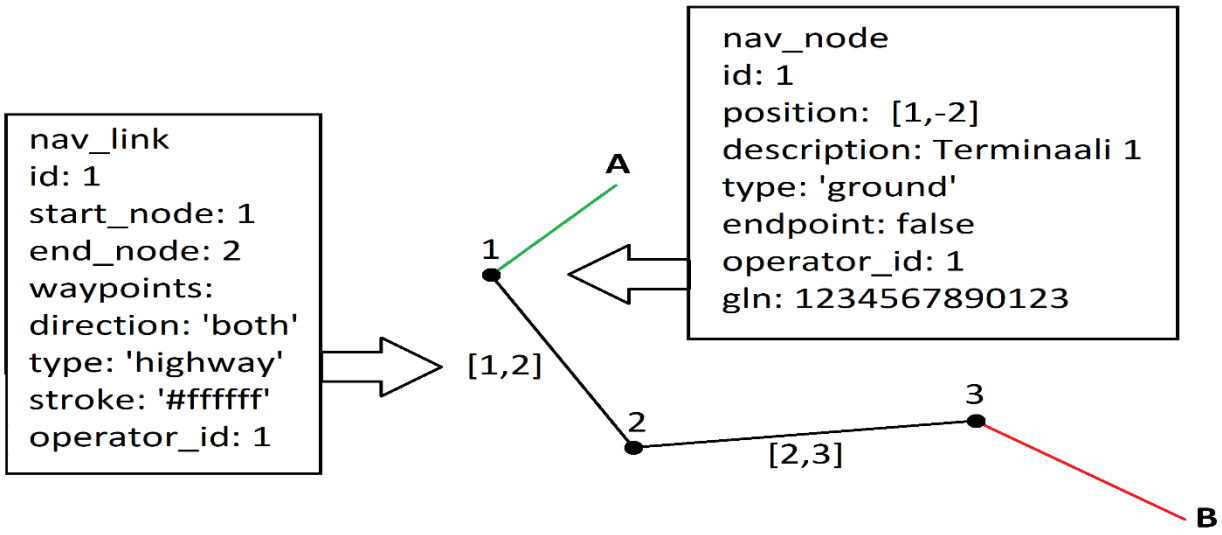


Kuvio 2. Esimerkki toimitusketjusta

Toimitusketjujen liikennöintiä varten kuljetuksia käsitteleville toimijoille jaettiin harjoitusympäristön mukaiset omat runkolinjat, jotka perustuivat toimijoiden logistiikkaterminaalien välisiin yhteyksiin. Toimitusketjujen kuljetukset painotettiin hyödyntämään näitä runkolinjoja.

Valhallan reittilaskenta perustuu OpenStreetMap-datan tieverkostoihin, joten se ei sellaisenaan sovellu runkolinjojen mallintamiseen eikä vesi- ja ilmaliikenteen reitteihin ilman erillistä koodipohjan ja paikkatietoaineiston muokkausta. Vesi- ja ilmaliikenteen reitit kahden pisteen välillä ovat yksinkertaisia tieverkostoihin verrattuna, ja ne voidaan usein kuvata muutamalla viivalla.

Runkolinjojen sekä vesi- ja ilmaliikenteen yksinkertaisuuden vuoksi päädyttiin käyttämään Valhallan lisäksi kevyempää toissijaista reittilaskentaa GeoJSON Path Finder -ohjelmointikirjastolla. Runkolinja-verkko rakennettiin tietokantaan kuvion 3 mukaisella mallilla, jossa määritellään erikseen logistiikkakeskuksia kuvaavat solmut ja niiden väliset yhteydet. Ennen reitin laskentaa verkko luettiin tietokannasta ja muunnettiin GeoJSON-muotoon kirjaston käyttöä varten. Näin toimitusketjujen rakentamisessa pystyttiin hyödyntämään runkolinjoja sekä laskemaan vesi- ja ilmaliikenteen reitit niistä, jolloin Valhallan tehtäväksi jäi yksityiskohtaisempi reititys maantieverkostoissa.



Kuvio 3. Runkolinjaverkoston malli

Runkolinjoja voidaan hyödyntää rakentamalla toimitusketjut niiden mukaisesti, mutta toimitusketjun todellinen alku- ja päätepiste ei aina sijaitse runkolinjalla. Näissä alku- ja loppumatkoissa, esimerkiksi kun tuote haetaan toimittajalta ja viedään varastoon tai toimitetaan varastosta loppuasiakkaalle, puhutaan niin sanotuista päätematkoista. Näille on vakiintuneet englanninkieliset termit first mile ja last mile (The Difference Between First, Middle and Last-Mile Logistics 2024). Harjoituksen yhteydessä toimitusketjujen alku- ja loppupisteet olivat aina teitse yhteydessä runkolinjaan, joten päätepisteet pystyttiin reitittämään Valhallan avulla lähimpään runkolinjan solmuun. Yhdistämällä päätematkat ja runkolinjoja pitkin kulkevat reitit saatiin rakennettua valmis toimitusketju GeoJSON-formaatissa. Tätä formaattia hyödynnettiin toimitusketjun visualisoinnissa ja sen pilkkomisessa pienempiin yksittäisiin reitteihin simulaatiota varten.

4.3 Simulaatio

Entiteettien tietomalli suunniteltiin siten, että simulaation logiikka pystyisi käsittelemään entiteettejä niiden ominaisuuksien, tilan ja tyypin perusteella. Kuvio 4 esittää tietomallin lopullisen rakenteen.

```
--- public.entity
create table public.entity (
    "id"                        uuid                    primary key default public.uuid_generate_v4(),
    "position"                  geometry(Point, 4326)   not null,
    "type"                      text                    not null,
    "self_propelled"            boolean                 not null,
    "state"                     text                    not null default 'IDLE',
    "contained_in"              uuid                    references public.entity (id) on delete set null,
    "metadata"                  jsonb                   not null default '{}'::json
);
create index entity_containedIn_index on entity USING btree (contained_in ASC);
```

Kuvio 4. Entiteetin tietomalli

Tietomallin tyyppi-sarakkeella kuvattiin, onko kyseessä esimerkiksi rekka, laiva tai lentokone. Simulaation logiikassa eri tyypeille voitiin määrittää tyyppikohtaisia ominaisuuksia, kuten nopeus. Entiteettien realistisen simuloinnin ja erityisesti logististen kuljetusten kannalta tunnistettiin myös tarve entiteettikohtaisille tiloille.

Ensimmäinen vaatimus tiloille oli entiteettien kyky olla toistensa sisällä. Tällä tavalla voitiin kuvata esimerkiksi tilanne, jossa yksittäinen paketti on rekan kyydissä. Simulaation logiikan kannalta tämä tarkoitti, että paketti on riippuvainen rekan sijainnista, eikä se pysty liikkumaan itsenäisesti ennen kuin se vapautetaan. Näin saatiin visualisointia varten rekursiivisesti haettua tieto siitä, mitä entiteetit sisältävät. Toinen vaatimus oli entiteetin kyky itsenäiseen liikkumiseen. Kaikki entiteetit eivät luonnollisesti pysty liikkumaan itsenäisesti, kuten esimerkiksi paketti, joten tämän tilan avulla simulaation logiikka voi ohittaa liikkeen laskemisen näiden entiteettien osalta. Tarvittaessa itsenäinen liikkuminen voitiin kuitenkin aktivoida tilaa vaihtamalla.

Entiteettien simulointiin käytettiin työjonoa hyödyntävää Graphile Worker -kirjastoa. Kirjaston valintaa tukivat sen yhteensopivuus muiden käytössä olevien teknologioiden kanssa sekä sen tarjoama laskentatehon skaalattavuus. Työjonon avulla simulointiin liittyvät laskutoimitukset toteutettiin tehtäväpohjaisesti, tallentaen ne tietokantaan. Simulaatio pystyttiin konfiguroimaan ympäristön laajuuden ja tarpeiden mukaan säätämällä Graphile Workerin rinnakkain ajettavien tehtävien määrää sekä ajamalla useampaa prosessia samanaikaisesti. Graphile Worker käyttää tehtävien hallinnassa tietokantatransaktioita varmistaakseen turvallisen tiedonsiirron. Prosessi asetettiin käsittelemään uusia tehtäviä säännöllisin väliajoin, jotta liikkeen laskennasta saatiin tasaista ja reaaliaikaista.

Liikkeen laskenta käynnistettiin antamalla API-rajapinnasta käsky, joka sisälsi entiteetin yksilöivän tunnisteen sekä kuljettavan reitin tai määränpään koordinaatit. Käskyn lisäasetuksilla liike voitiin pakottaa alkamaan reitin alusta tai lisätä nykyinen sijainti reitin alkuun. Ennen käskyn hyväksymistä tarkistettiin, onko liikkuminen mahdollista. Matkan tuli olla reititettävissä, Entiteetin täytyi olla reitin lähellä, entiteetillä ei saanut olla liikkeitä jonossa, eikä se saanut sijaita toisen entiteetin sisällä. Hyväksytyn käskyn jälkeen tietokantaan luotiin uusi tehtävä liikuttamista varten. Graphile Worker käsitteli tehtävän ja loi uuden tehtävän entiteetille. Tämä sykli toistui, kunnes entiteetti saapui määränpäähän.

Liikkeen laskenta perustui kuljettuun matkaan nopeuden ja ajan suhteen. Aikaa määritettäessä otettiin huomioon mahdolliset viiveet, kuten yhteysongelmat, laskutoimituksen monimutkaisuus tai simulaation ylikuormitus. Todellinen aika saatiin laskemalla aika tehtävän luonnin ja suorittamisen välillä. Kuljettu matka laskettiin kaavalla 1:

$$d = v \times \Delta t \qquad (1)$$

missä d = kuljettu matka

v = entiteetin nopeus

Δt = kulunut aika tehtävän luonnin ja suorittamisen välillä

Kun kuljettu matka oli selvillä, voitiin laskea piste reitillä kyseisen matkan päässä. Laskentaan käytettiin paikkatietoanalyyseihin erikoistunutta Turf.js-kirjastoa. Laskettu sijainti tallennettiin tietokantaan entiteetin uudeksi sijainniksi.

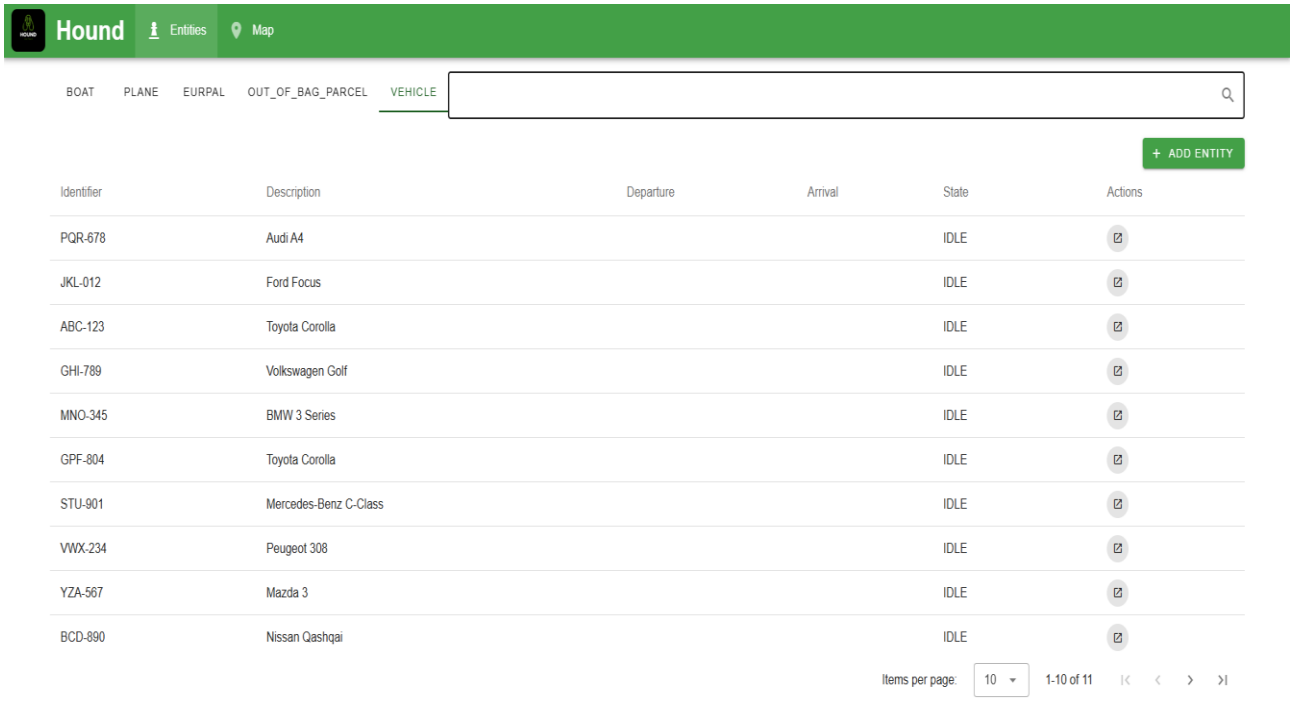
Logistiikan toimitusketjuja varten simulaation logiikkaan lisättiin eristettyjä alueita, joihin entiteetillä täytyi olla lupa päästäkseen sisään. Näillä alueilla mallinnettiin yksityisalueita, kuten lentokenttiä, satamia ja aidattuja yritysalueita. Kun entiteetti saapui alueen rajalle, se pysähtyi ja lähetti tiedot itsestään ja portista HTTP-kutsulla erilliseen portinvalvontajärjestelmään. Mikäli sisäänpääsy hyväksyttiin, entiteetti jatkoi matkaa normaalisti. Jos sisäänpääsy evättiin, entiteetti odotti hetken ja yritti uudelleen. Määritellyn yritysmäärän jälkeen entiteetti pysähtyi odottamaan uutta tehtävää. Alkuperäisen suunnitelman mukaan entiteetille oli tarkoitus toteuttaa palautuslogiikka, jos se jäisi jumiin, mutta tämä jätettiin aikataulusyistä toteuttamatta.

4.4 Käyttöliittymän toteutus

Käyttöliittymä toteutettiin Vue.js-ohjelmistokehyksellä, ja karttojen visualisointiin käytettiin avoimeen lähdekoodiin perustuvaa Leaflet-kirjastoa. Karttanäkymissä käytetty paikkatietoaineisto perustui OpenStreetMapin yhteisön tuottamaan aineistoon (© OpenStreetMap contributors), joka on lisensoitu Open Database License (ODbL) -ehdoin.

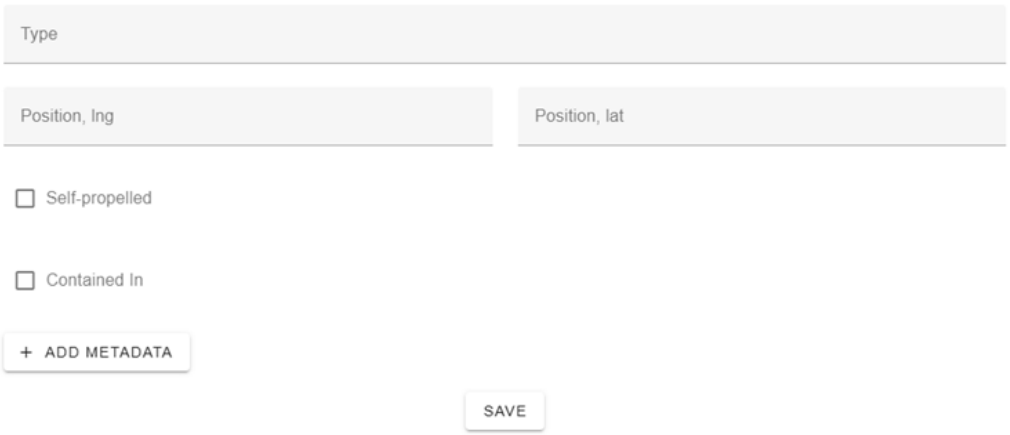
Tätä yhdistelmää oli käytetty aiemmin kehitetyissä järjestelmissä, joten yhteensopivuuden ja käytettävyyden vuoksi ne valittiin myös simulaatiojärjestelmän käyttöliittymään. Käyttöliittymän tehtävänä oli tarjota käyttäjälle työkalut entiteettien hallintaan sekä visualisoida entiteetit karttanäkymässä. Toiminnallisuus jaettiin kahteen välilehteen: Entiteetit ja Kartta.

Kuviossa 5 esitetään Entiteetit-välilehden päänäkymä. Näkymä sisältää taulukkomuotoisen listan, josta käyttäjä voi selata entiteettien perustietoja tyypin mukaan. Entiteetit haetaan tietokannasta API-rajapinnan kautta, ja tiedot päivittyvät reaaliajassa. Entiteetit voidaan järjestää sarakkeiden arvojen mukaan tai niitä voidaan hakea hakukentän avulla. Näkymässä on myös painike uuden entiteetin luomista varten.



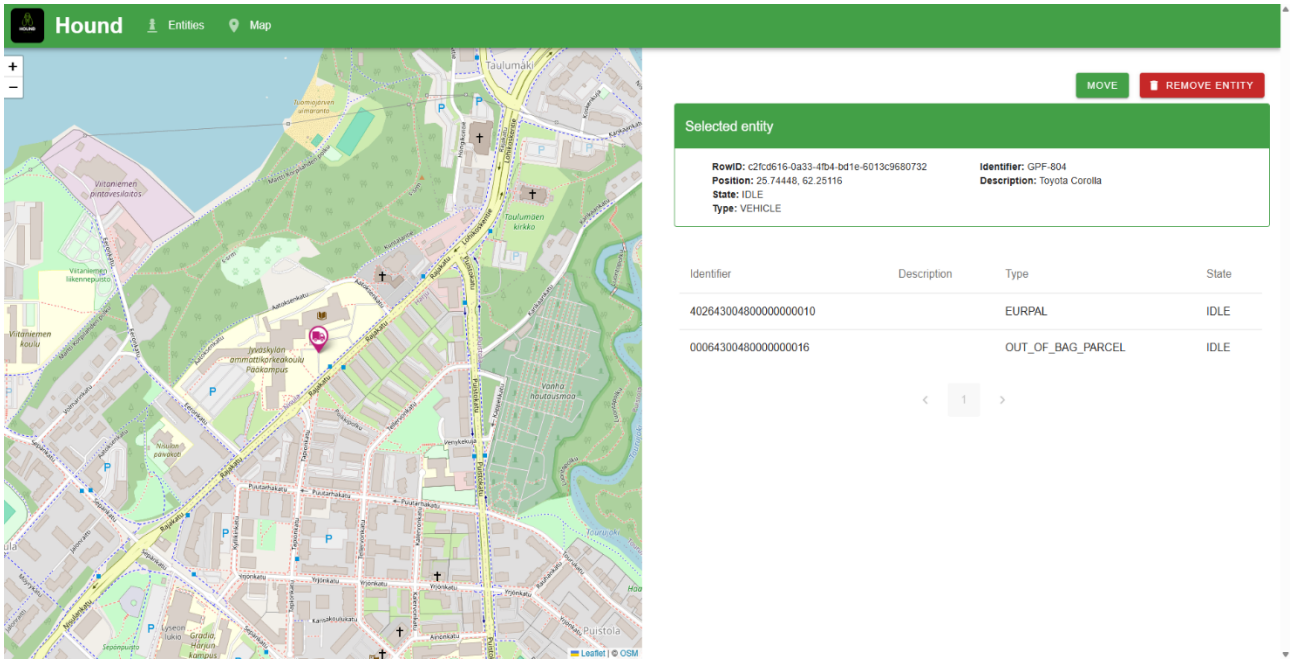
Kuvio 5. Käyttöliittymän Entiteetit-välilehden päänäkymä

Uuden entiteetin luonnin painikkeesta avautuu kuviossa 6 esitetty näkymä, jossa käyttäjä syöttää entiteetille tarvittavat tiedot ja hyväksyy ne. Hyväksynnän yhteydessä entiteetti lisätään tietokantaan ja se tulee heti käytettäväksi järjestelmän sisällä. Entiteetin luonnin jälkeen käyttäjä palautetaan takaisin päänäkymään.



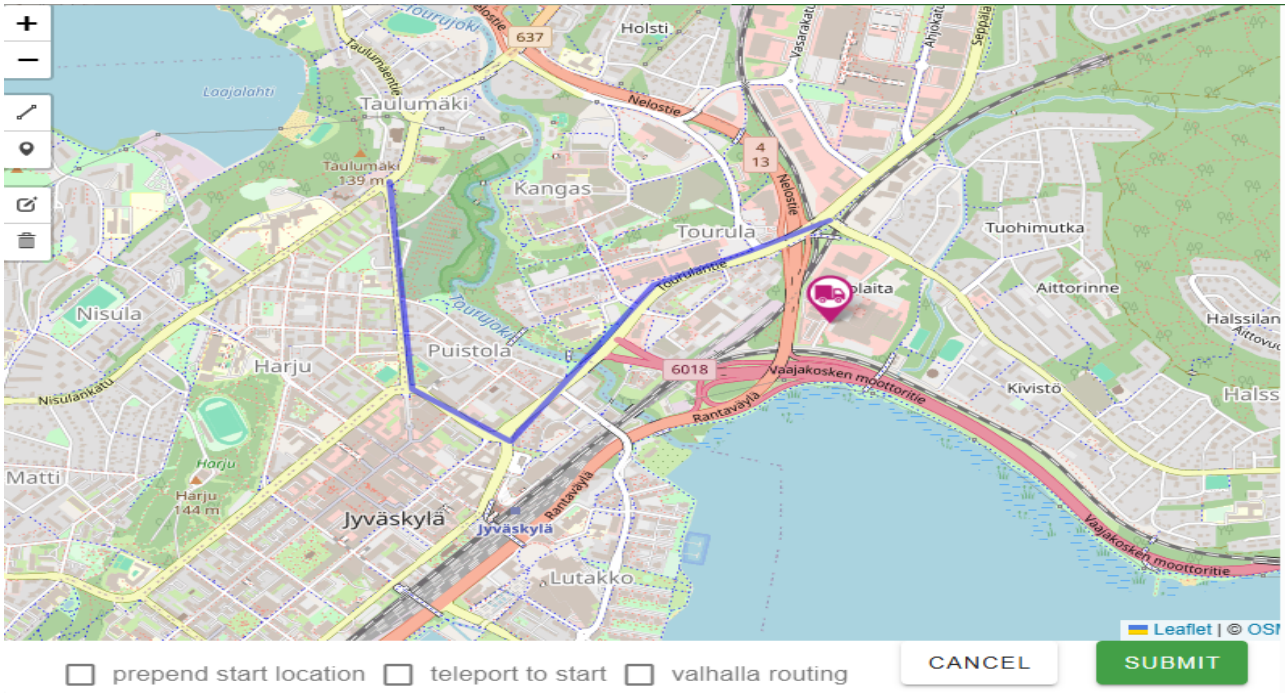
Kuvio 6. Entiteetin luonti käyttöliittymässä

Päänäkymästä käyttäjä voi myös avata yksittäisen entiteetin tarkemmat tiedot, jotka esitetään kuviossa 7. Tässä näkymässä esitetään perustietojen lisäksi entiteetin sijainti kartalla sekä sen sisältämät muut entiteetit. Sisäkkäisen entiteetin tietoihin voi siirtyä painamalla sitä listasta.



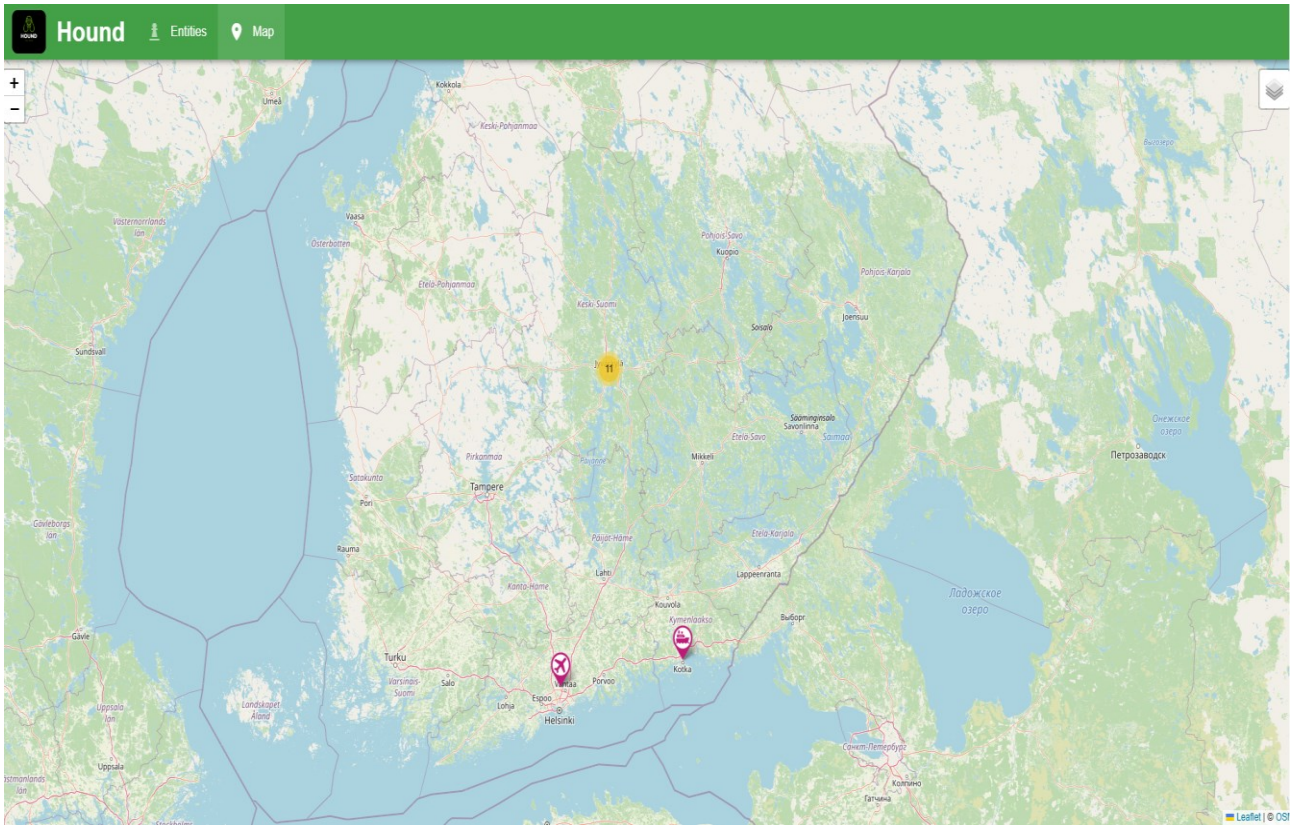
Kuvio 7. Käyttöliittymän entiteettinäkymä

Näkymä tarkemmista tiedoista sisältää myös työkalut entiteetin siirtoon ja poistoon. Entiteetin siirtotyökalu esitetään kuviossa 8. Entiteetille oli suunniteltu myös muita hallintatyökaluja, kuten entiteetin pysäytys tai siirto toiseen sijaintiin, mutta ne rajattiin aikataulusyistä pois tästä toteutuksesta.



Kuvio 8. Käyttöliittymän entiteetin liikutustyökalu

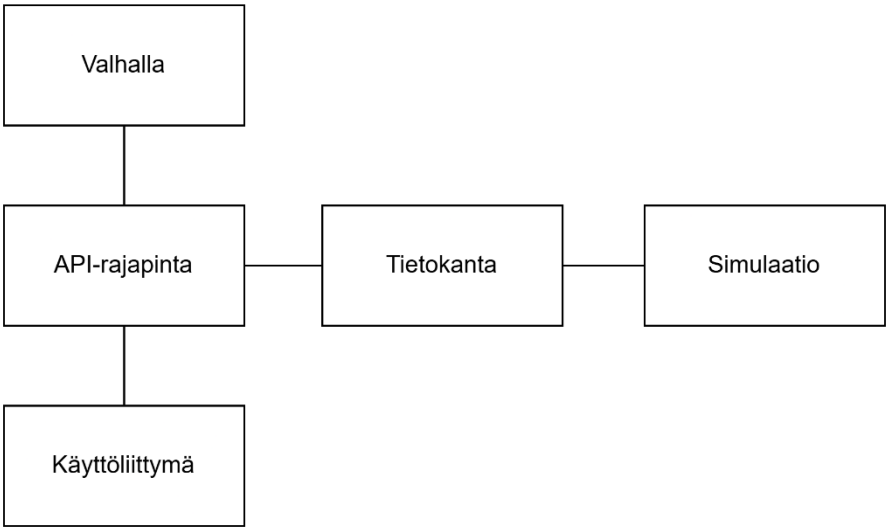
Kuviossa 9 esitetyssä Kartta-välilehden päänäkymässä käyttäjä voi seurata entiteettien ja eristettyjen alueiden sijaintia kartalla. Selkeyden vuoksi lähekkäiset entiteetit yhdistetään klustereiksi, joissa numero kertoo entiteettien lukumäärän. Karttanäkymässä on myös suodatusasetuksia, joilla voi piilottaa tietyn tyyppiset entiteetit kartalta. Entiteetin tiedot avautuvat klikkaamalla sitä kartasta, ja ne esitetään kuten kuviossa 7. Lisäksi kartalla näytetään entiteetin sijaintihistoria sekä sen nykyinen kuljettava reitti.



Kuvio 9. Käyttöliittymän Kartta-välilehden päänäkymä

4.5 API-rajapinta

Simulaatiojärjestelmän toiminta perustui useiden palveluiden yhteistoimintaan. Jotta palvelut voisivat toimia yhdessä, niiden tuli jakaa yhtenäiset tietotyypit ja pystyä kommunikoimaan keskenään. Tämän mahdollistamiseksi kehitettiin API-rajapinta, joka vastasi tiedonsiirrosta ja -käsittelystä palveluiden ja tietokannan välillä. Kuviossa 10 on esitetty palveluiden väliset yhteydet rajapinnan kautta.



Kuvio 10. Palveluiden yhteydet rajapinnan kautta

Palvelu rakennettiin Node.js-ympäristöön, ja sen rajapinta perustui PostGraphile-kirjaston muodostamaan GraphQL-rajapintaan. Valitut teknologiat olivat keskenään yhteensopivia ja vastasivat muissa järjestelmissä käytettyä arkkitehtuuria. PostGraphile loi tietokannan rakenteesta automaattisesti GraphQL-kyselyt, jotka tarjosivat perustyökalut tietokannan lukemiseen ja muokkaamiseen. Rajapinnan tehtävänä oli myös välittää tietoa palveluiden välillä sekä yhtenäistää niiden käyttämiä tietotyyppejä, minkä vuoksi rajapintaa laajennettiin omilla GraphQL-kyselyillä, PostGraphilen tarjoamien työkalujen avulla.

Luvussa 4.2 esitettiin toissijainen reititysjärjestelmä Valhallan rinnalle reitittämään runkolinjoja sekä vesi- ja ilmaliikennettä GeoJSON Path Finder -kirjaston avulla. Tämä toissijainen reititys oli luonteeltaan kevyt, ja Valhallan palveluympäristö haluttiin säilyttää alkuperäisessä muodossaan, joten kirjasto ja logiikka lopullisen reitin muodostamiseen päätettiin toteuttaa API-rajapinnan sisällä.

5 Tulokset

Työn tuloksena saatiin kehitettyä vaatimusmäärittelyn mukainen simulaatiojärjestelmä, jossa logistiikka-alan toimitusketjujen kuljetuksia pystyttiin mallintamaan paikkatietopohjaisesti reittihaun ja simulaatiologiikan avulla. Toimitusketjuja pystyttiin seuraamaan ja hallitsemaan sekä kokonaiskuvassa että yksittäisen kuljetuksen tasolla. Järjestelmä tarjosi myös työkalut simulaation skaalaukseen ja hallintaan, mikä mahdollisti sen soveltamisen eri ympäristöissä. Simulaation tuottama tieto toimitusketjuista ja niiden fyysisistä objekteista esitettiin reaaliaikaisesti käyttöliittymässä sekä tekstimuodossa että karttapohjaisesti.

Käytännön esimerkkinä tuloksien käytöstä toimii logistiikan toimialan kyberharjoitus, jossa kehitettyä simulaatiojärjestelmää käytettiin onnistuneesti fyysisten objektien simulointiin. Simuloinnin avulla tuotettu tieto tarjosi harjoituksen osapuolille entistä tarkempaa ja monipuolisempaa tietoa harjoituksen tapahtumista, jonka avulla voitiin tukea harjoituksen tilannekuvaa ja päätöksentekoa. Tuloksien perusteella voidaan todeta, että fyysisten objektien simulointi lisäsi harjoituksen laatua ja molempiin tutkimuskysymyksiin saatiin myönteinen vastaus.

6 Pohdinta

Työn tavoitteena oli tutkimuksellisen kehittämistyön kautta toteuttaa toimeksiantajalle tekninen simulaatiojärjestelmä ja tutkia sen avulla fyysisten objektien simulointia kyberharjoituksissa. Edellisessä luvussa esiteltiin työn tuloksena toteutettu simulaatiojärjestelmä, jolla fyysisiä objekteja simuloitiin onnistuneesti osana harjoitusympäristöä. Tuloksissa todettiin myös, että molempiin tutkimuskysymyksiin saatiin myönteiset vastaukset. Tulosten perusteella voidaan todeta, että työn tavoitteet on saavutettu.

Simulaatiojärjestelmää käytettiin onnistuneesti osana logistiikan toimialan kyberharjoitusta, ja se sisälsi tarvittavat toiminnot toimitusketjujen täysimittaiseen mallintamiseen. Kaikkia järjestelmään alun perin suunniteltuja toiminnallisuuksia ei kuitenkaan voitu toteuttaa aikarajoitteiden vuoksi. Järjestelmän prosessien optimointi jäi myös vähäiseksi, mikä vaikutti erityisesti käyttöliittymän käytettävyyteen ja nopeuteen sekä simulaation laskentatehoon.

Työssä yhdistyivät useat laajat aihealueet, kuten logistiikan käytänteet, simulaatioiden toiminta, paikkatiedon käsittely ja reittihakualgoritmit. Näihin aihealueisiin perehtyminen tapahtui pääosin työn aikana, mikä rajoitti mahdollisuuksia perusteelliseen suunnitteluun ennen toteutusta. Tämä johti siihen, että järjestelmän rakenne ja toiminnalliset vaatimukset tarkentuivat vähitellen työn edetessä.

Työssä käytetyt teknologiat soveltuivat pääosin hyvin käyttötarkoituksiinsa. Erityisesti reittilaskentaan erikoistuneet palvelut kehittyvät kuitenkin nopeasti ja tarjoavat monipuolisia vaihtoehtoja. Esimerkiksi tietokantapohjaiset reittihakuratkaisut ovat jatkokehityksen kannalta tutkimisen arvoisia. Tällaisen ratkaisun avulla reititys voitaisiin keskittää yhteen moottoriin, ja sen hallinta tietokannan kautta tukisi järjestelmän muuta rakennetta.

Kehityskohteina järjestelmässä ovat erityisesti vajaat ja puuttuvat toiminnallisuudet, kuten objektien liikuttamisen työkalut sekä palautuslogiikka jumiin jääville objekteille. Tärkeitä kehityskohteita ovat myös simulaation ja käyttöliittymän suorituskyvyn optimointi. Yksi kehitysidea on lisätä järjestelmään uusi entiteettityyppi, kuten junat, ja rakentaa niitä varten raideverkosto.

Toteutetun simulaatiojärjestelmän oli tarkoitus toimia myös pohjana tuleville järjestelmille, joissa vaaditaan fyysisten objektien simulointia. Järjestelmä tarjoaa toimivan rungon ja hyväksi todettuja käytänteitä muokkausta ja jatkokehitystä varten. Sovellettaessa järjestelmää muihin käyttötarkoituksiin on huomioitava, että sen logiikka pohjautuu tässä työssä esitettyyn kyberharjoitusympäristöön, ja tulosten hyödyntäminen vaatii tapauskohtaista arviointia.

Lähteet

About PostGIS. N.d. PostGIS-verkkosivusto. Viitattu 25.05.2025. https://postgis.net.

About PostgreSQL. N.d. PostgreSQL esittely sivusto. Viitattu 26.05.2025. https://www.postgresql.org/about/.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. Numerische Mathematik. 1, 269–271. Viitattu 15.4.2025. https://ir.cwi.nl/pub/9256/9256D.pdf.

Fazal, S. 2008. GIS basics. New Delhi: New Age International (P) Ltd., Publishers. Viitattu 14.4.2025. https://janet.finna.fi, Ebook Central Academic Complete.

Geoinformatiikan sanasto: TSK 51. 2018. Helsinki: Maanmittauslaitos. Viitattu 13.4.2025. https://sanastokeskus.fi/tiedostot/pdf/GeoinformatiikanSanasto.pdf?file=pdf/GeoinformatiikanSanasto.pdf.

GeoJSON Path Finder. N.d. GeoJSON Path Finder -verkkosivusto. Viitattu 25.5.2025. https://www.liedman.net/geojson-path-finder/.

Graphile Worker - Introduction. N.d. Graphile Worker -verkkosivusto. Viitattu 27.5.2025. https://worker.graphile.org/docs.

Haerder, T & Reuter, A. 1983. Principles of transaction-oriented database recovery. ACM Computing Surveys. 15, 4, 289–317. Viitattu 26.5.2025. https://doi.org/10.1145/289.291.

Handbook: The Basics. 2025. TypeScript-verkkodokumentaatio. Viitattu 13.5.2025. https://www.typescriptlang.org/docs/handbook/2/basic-types.html.

Handbook: The TypeScript handbook. 2025. TypeScript-verkkodokumentaatio. Viitattu 13.5.2025. https://www.typescriptlang.org/docs/handbook/intro.html.

Introduction to GraphQL. 2025. GraphQL esittely sivusto. Viitattu 27.5.2025. https://graphql.org/learn/.

Introduction to Node.js. N.d. Node.js-verkkosivuston opas. Viitattu 25.05.2025. https://nodejs.org/en/learn/getting-started/introduction-to-nodejs.

Karjalainen, M. 2021. Pedagogical Basis of Live Cybersecurity Exercises. Väitöskirja. Jyväskylän yliopisto, Informaatioteknologian tiedekunta. Viitattu 12.5.2025. https://urn.fi/URN:ISBN:978-951-39-8738-1.

Kumar, R. 2024. The A* Algorithm: A Complete Guide. Artikkeli DataCamp-verkkosivustolla. Viitattu 27.5.2025. https://www.datacamp.com/tutorial/a-star-algorithm.

Kyberharjoitusohje. 2019. Kyberturvallisuuskeskuksen käsikirja harjoituksen järjestäjälle. Viitattu 11.5.2025. https://www.kyberturvallisuuskeskus.fi/sites/default/files/media/file/Kyberharjoitus-opas.pdf.

PostGraphile Instant GraphQL API. N.d. PostGraphile-verkkosivuston etusivu. Viitattu 26.5.2025. https://www.graphile.org/postgraphile/.

The Cyber Range: A Guide. 2023. Opas National Institute of Standards and Technologyn verkkosivustolla. Viitattu 12.5.2025. https://www.nist.gov/system/files/documents/2023/09/29/The%20Cyber%20Range_A%20Guide.pdf.

The Difference Between First, Middle and Last-Mile Logistics. 2024. Artikkeli Amazon Freight -verkkosivustolla. Viitattu 18.4.2025. https://freight.amazon.co.uk/newsroom/first-middle-last-mile-logistics.

TypeScript. N.d. TypeScriptin virallinen kotisivu. Viitattu 13.5.2025. https://www.typescript-lang.org/.

Valhalla. N.d. Valhalla-verkkosivusto. Viitattu 25.5.205. https://valhalla.github.io/valhalla/.

Voloshin, V. I. 2009. Introduction to graph theory. New York: Nova Science Publishers. Viitattu 15.4.2025. https://janet.finna.fi, Ebook Central Academic Complete.

Vue.js - Introduction. N.d. Vue.js esittely sivusto. Viitattu 25.5.2025. https://vuejs.org/guide/introduction.html.

What is Docker?. N.d. Docker-verkkodokumentaatio. Viitattu 13.5.2025. https://docs.docker.com/get-started/docker-overview/.

What is geocoding?. N.d. Artikkeli ArcGIS Desktop -verkkodokumentaatiossa. Viitattu 13.4.2025. https://desktop.arcgis.com/en/arcmap/latest/manage-data/geocoding/what-is-geocoding.htm.

What is geospatial data?. N.d. Artikkeli IBM:n verkkosivustolla. Viitattu 13.4.2025. https://www.ibm.com/think/topics/geospatial-data.

What is raster data?. N.d. Artikkeli ArcGIS Desktop -verkkodokumentaatiossa. Viitattu 13.4.2025. https://desktop.arcgis.com/en/arcmap/latest/manage-data/raster-and-images/what-is-raster-data.htm.