Raideliikenteen avoimen datan visualisointi matkustajalle

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Riihimäki, Tieto- ja viestintätekniikka

Syksy, 2018

Petri Salo

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tieto- ja viestintätekniikka
Riihimäki

Tekijä Petri Salo **Vuosi** 2018

Työn nimi **Raideliikenteen avoimen datan visualisointi matkustajalle**

Työn ohjaaja/t Teemu Järvenpää

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa responsiivinen, sekä työpöytä-, että mobiilikäyttöön soveltuva web-sovellus Suomen junaliikenteen datan visualisointiin matkustajalle. Tavoitteena oli tehdä oikean junan tietojen löytämisestä nopeaa ja mahdollisimman mutkatonta. Tietojen näyttämiseen oli tavoitteena näkymä, josta olisi mahdollista yhdellä vilkaisulla nähdä kaikki tärkeimmät tiedot junan kulkuun liittyen.

Tarve aiheena olleelle palvelulle on muodostunut kirjoittajan empiirisistä kokemuksista säännöllisesti junalla matkustavana. Säännöllisesti junalla matkustaessa poikkeustilanteet ja usein toistuvat myöhästymiset ovat luoneet tarpeen palvelulle, josta tiedot junien liikkeistä ja sen hetkisestä tilanteesta pystyisi tarkastamaan nopeasti ja mutkattomasti.

Palvelu toteutettiin pilvessä ajettavana web-sovelluksena, jolloin sitä on mahdollista käyttää mahdollisimman monella eri päätelaitteella ja vältetään tarve erillisen sovelluksen asentamiselle. Toteutuksessa hyödynnettiin monipuolisesti eri ohjelmointikieliä ja rajapintoja, jotta käyttäjälle pystyttäisiin tarjoamaan mahdollisimman selkeä, matalalatenssinen ja tietoturvallinen palvelu junatietojen tarkasteluun.

Avainsanat avoin data, raideliikenne, älyliikenne

Sivut 28 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Information and Communication Technology
Riihimäki

Author Petri Salo **Year** 2018

Subject Visualization of open data for rail traffic to the passenger

Supervisors Teemu Järvenpää

ABSTRACT

The goal of the thesis project was to design and implement a responsive web-application, suitable for both desktop and mobile use, which would visualize information on rail traffic in Finland. The goal was to make finding the right train and the information related to it as easy and seamless as possible. The information should be presented in a manner, where it would be possible to see the most important information at a glance.

A need for this kind of service was formed from the author's own experiences as a daily commuter by train. The exceptional situations and recurring delays created a need for a service, where information on train locations and the status of the train could be checked quickly and seamlessly.

The service was implemented in the cloud as a web application, so that it could be used with as many different devices as possible and to avoid the need to install a native application. Different programming languages and interfaces were used in the implementation to achieve a clear, secure and low latency service for viewing information on train traffic.

Keywords Open data, rail traffic, smart traffic

Pages 28 pages

SISÄLLYS

1 JOHDANTO

Opinnäytetyön aiheena on junalla säännöllisesti matkustaville kehitetty Junassa-websovellus, jonka avulla matkustajan on mahdollista nopeasti ja vähällä vaivalla tarkastaa oman junavuoronsa ajankohtainen tilanne, koskien junan aikataulua, sijaintia, nopeutta ja etäisyyttä seuraaviin asemiin.

Tarve sovelluksen kehittämiseen on syntynyt kirjoittajan omista kokemuksista päivittäisenä junamatkustajana. Säännöllisesti matkustaessa on muodostunut tarve reaaliaikaiselle tiedolle juna-aikataulujen poikkeuksista ja junien sijainnista.

Markkinoilla on olemassa myös muita sovelluksia ja palveluja, jotka tarjoavat osittain samoja tietoja ja toiminnallisuutta, kuin tässä työssä kehitetty sovellus. Tässä työssä toteutetussa sovelluksessa on kuitenkin pyritty muita samankaltaisia sovelluksia ajantasaisempaan tiedonsaantiin, monipuolisempaan esitystapaan ja alustariippumattomuuteen.

Kehitettävä sovellus poikkeaa muista markkinoilla olevista sovelluksista ja palveluista tarjoamalla säännöllisesti sovellusta käyttävälle matkustajalle nopean pääsyn omaa junavuoroaan koskeviin ajantasaisiin tietoihin. Tiedoista muodostetaan helppolukuinen koostenäkymä, josta kaikki olennainen tieto on mahdollista yhdellä silmäyksellä nähdä. Käyttäjälle tarjotaan mahdollisuus tallentaa käyttämänsä junavuorot palveluun, jolloin hänet pystytään automaattisesti ohjaamaan oikean junavuoron tietoihin hänen saapuessaan web-sivulle.

Tämän opinnäytetyön tarkoitus on tutkia ja kehittää ratkaisua ongelmaan, miten raideliikenteen saatavilla olevaa avointa dataa voidaan esittää junamatkustajalle matkakokemuksen parantamiseksi.

2 TIETOPERUSTA

Työn pohjana käytettiin laajaa valikoimaa nykyisin web-sovelluskehityksessä käytössä olevista tekniikoista ja käytänteistä. Työssä esitellään aiheen kannalta olennaisimmat työkalut ja tekniikat. Työn teoriaosuus on jaettu käsittelemään hyödynnettävää dataa, palvelintekniikkaa ja ohjelmistoja, tiedon tallennukseen käytettyjä menetelmiä, sekä työn kannalta olennaisimpia ohjelmointitekniikoita ja ohjelmointikieliä.

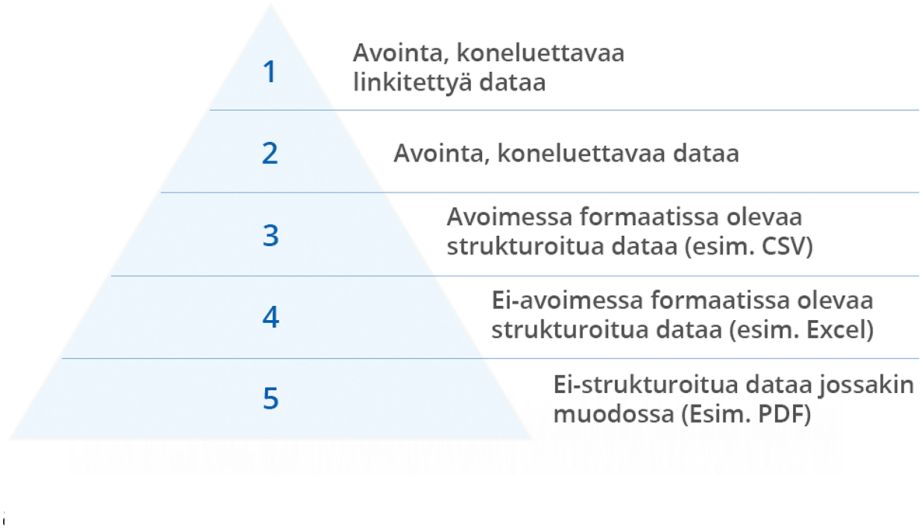
Tietoperustassa käydään läpi toteutettuun web-sovellukseen olennaisesti liittyvät tekniikat ja teoriat, teoriapohja etenee käsitellen ensin tiedon lähteenä käytettyä rajapintaa ja siinä käytettyä tietomuotoa, edeten sitten

käytettyihin palvelinympäristöihin, ohjelmointiratkaisuihin ja lähdekoodin versionhallintaan. Tekstissä käsitellään myös palvelimen konfiguraation ja palvelinpuolen ohjelmoinnin osalta, mitä asioita näissä tulisi tietoturvan näkökulmasta huomioida.

2.1 Avoin data

Avoimella datalla tarkoitetaan tietoa, joka on saatettu saataville julkisesti, koneluettavassa muodossa ja on lisensoitu vapaaseen käyttöön ilman erillistä maksua. Kaikki avoin data ei kuitenkaan täytä kaikkia näistä kriteereistä. (Helsinki Region Infoshare, 2017)

Avoimelle datalle on määritelty kuvassa 1 näkyvä viisiportainen luokittelumalli, jonka ylimmässä ykkösluokassa edellytetään, että data on avointa, koneluettavaa ja linkitettyä. Alimmassa viitosluokassa ei sen sijaan aseteta tarkempia vaatimuksia datan muodolle, vaan data voi olla mitä tahansa ei-strukturoitua dataa ei-koneluettavassa muodossa. (Avoindata.fi, 2018)



Kuva 1. Avoimen datan viisiportainen luokittelumalli (Avoindata.fi, 2018)

Tässä työssä on hyödynnetty Liikenneviraston ylläpitämää avointa Digitraffic-rajapintaa, joka ei edellytä käyttäjältä rekisteröitymistä tai muutakaan tunnistautumista, vaan kaikki data on saatavilla koneluettavassa JSON-muodossa. Digitraffic-rajapinnasta on saatavilla ajankohtaista dataa Suomen tie-, meri- ja rautatieliikenteestä. (Liikennevirasto, 2018)

JSON on lyhenne sanoista JavaScript Object Notation ja on tekstipohjaisen tiedon välittämiseen tarkoitettu tiedostomuoto. Nimensä mukaisesti JSON-datasta on mahdollista parsia JavaScript-ohjelmointikielen olio, joka saa sisäänsä JSON-datassa määritellyn tietorakenteen. JSON-muotoista da-

taa voidaan kuitenkin käyttää myös muissa ohjelmointikielissä ja useimmissa moderneissa ohjelmointikielissä onkin sisäänrakennettu JSON-parseri. (W3Schools, n.d.)

JSON-tietorakenne koostuu avaimista ja niitä vastaavista arvoista. Rakenne muistuttaa siis sanakirjaa, jossa on aina avainsana ja sitä vastaava merkitys. JSON:ssa avainta vastaava arvo voi olla merkkijono, taulukko tai toinen sanakirjarakenne, jolloin tietueista rakentuu moniulotteinen laajeneva puurakenne. .


```
[ ▼
  { ▼ 13 properties, 36 KB
    "trainNumber": 1,
    "departureDate": "2018-10-19",
    "operatorUICCode": 10,
    "operatorShortCode": "vr",
    "trainType": "IC",
    "trainCategory": "Long-distance",
    "commuterLineID": "",
    "runningCurrently": false,
    "cancelled": false,
    "version": 241744722258,
    "timetableType": "REGULAR",
    "timetableAcceptanceDate": "2018-05-28T09:11:08.000Z",
    "timeTableRows": [ ▼ 134 items, 35 KB
      { ▼ 13 properties, 372 bytes
        "stationShortCode": "HKI",
        "stationUICCode": 1,
        "countryCode": "FI",
        "type": "DEPARTURE",
        "trainStopping": true,
        "commercialStop": true,
        "commercialTrack": "8",
        "cancelled": false,
        "scheduledTime": "2018-10-19T04:09:00.000Z",
        "actualTime": "2018-10-19T04:09:16.000Z",
        "differenceInMinutes": 0,
        "causes": [],
        "trainReady": { ▼ 3 properties, 73 bytes
          "source": "KUPLA",
          "accepted": true,
          "timestamp": "2018-10-19T04:02:30.000Z"
        }
      },
      { ▼ 12 properties, 284 bytes
        "stationShortCode": "PSL",
        "stationUICCode": 10,
```


Kuva 2. Esimerkki JSON-muotoisesta datasta

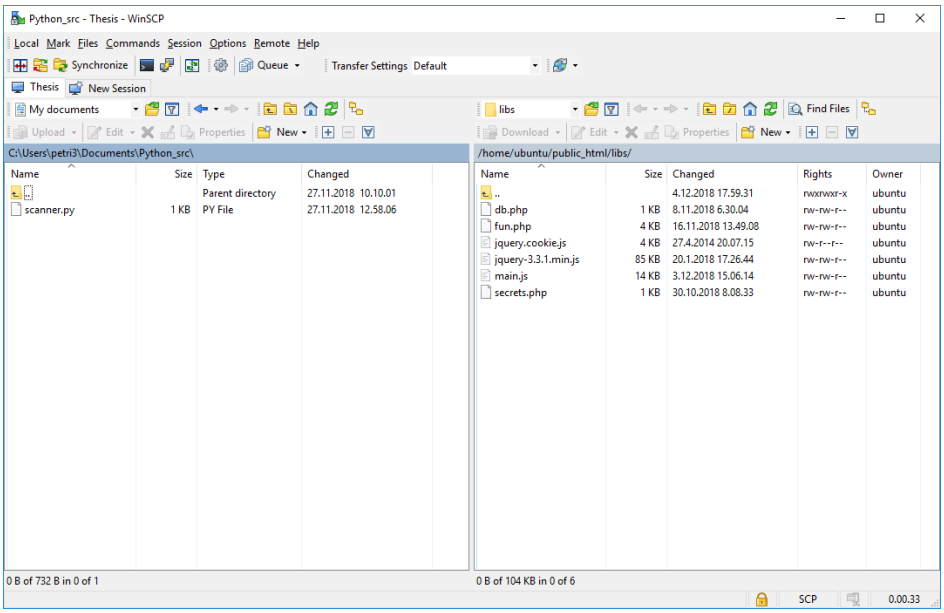
nähdään Liikenneviraston Digitraffic-rajapinnasta haettua JSON-muotoista dataa, jonka juurena on numeerisesti indeksoitu taulukko. Jokainen taulukon solu sisältää sanakirja-rakenteen, joka käsittää yksittäisen junan tiedot. Rakenteessa timeTableRows-avaimen sisälle on määritelty numeerisesti indeksoitu taulukko junan aikatauluun merkityistä juna-asemista, jonka jokainen solu sisältää jälleen uuden sanakirja-rakenteen.

2.2 Palvelin

Palvelimella voidaan tarkoittaa joko muita tietokoneita verkossa palvelevaa tietokonetta tai pelkästään tietokoneelle asennettua yksittäistä sovellusta, joka palvelee paikallisesti samalla tietokoneella tai verkon yli toisilla tietokoneilla olevia sovelluksia.

Nykyisin palvelintietokone on usein ulkoisen palveluntarjoajan palvelinkeskuksessa ajettava virtuaalikone. Yhdellä fyysisellä palvelintietokoneella on mahdollista ajaa useita virtuaalikoneita. Tällaista palvelua kutsutaan pilvipalveluksi. Pilvessä olevia virtuaalikoneita on mahdollista ohjata etähallinnalla, joka tyypillisesti tehdään salatun SSH-yhteyden läpi. (Ranger, 2018)

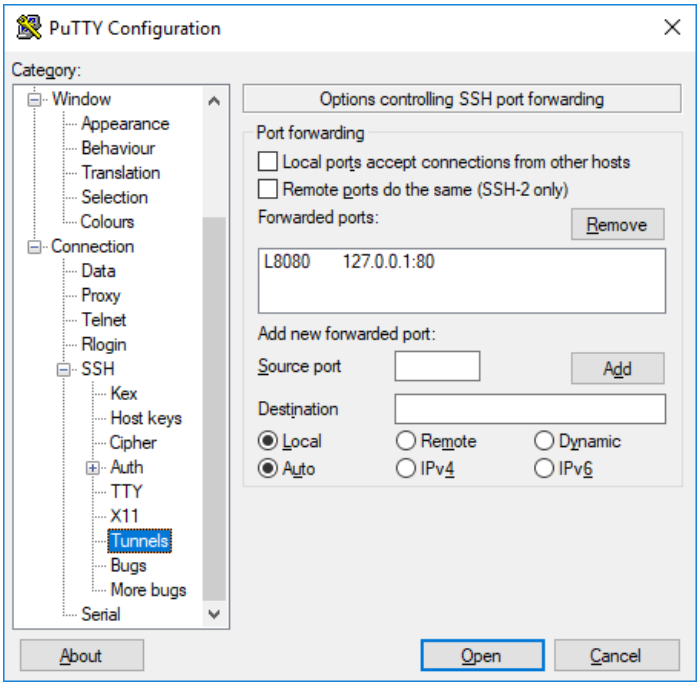
SSH, eli Secure Shell verkon yli etähallintaan tarkoitettu sovellus, joka muodostaa epäsymmetrisillä avaimilla varmennetun salatun yhteyden kahden tietokoneen välille (Ellingwood, 2014). SSH mahdollistaa myös tiedostojen siirtämisen salatun yhteyden yli, joka onnistuu käyttäen SCP-ohjelmaa Linux-käyttöjärjestelmässä tai kuvassa 3 näkyvää WinSCP-ohjelmaa Windows-käyttöjärjestelmässä.



Kuva 3. WinSCP-ohjelman ikkuna

SSH-ohjelmiston kehitti vuonna 1995 Suomalainen Tatu Ylönen työskennellessään tutkijana Helsingin Yliopistolla. Myöhemmin Ylönen perusti SSH Communications Security -yrityksen hallinnoimaan Secure Shellin kehitystä. SSH:n ensimmäiset versiot olivat avointa lähdekoodia, mutta myöhemmin ohjelmiston lähdekoodi on suljettu. Suljetun lähdekoodin SSH:n rinnalle on OpenBSD-projektissa kehitetty OpenSSH. (Geerling, 2014)

SSH-yhteyden läpi on mahdollista tunneloida muuta verkkoliikennettä, jolloin verkkoliikenne kulkeutuu salatun yhteyden läpi, josta kuvassa 4 esimerkki Putty-ohjelmasta. Tunnelointi mahdollistaa turvallisen pääsyn verkkoympäristöihin tai hallintapaneeleihin, jotka ovat muuten suojattu palomuurilla julkisen verkon ulkopuolelle. (SSH Communications Security, 2017)



Kuva 4. Tunnelointiasetukset Putty-ohjelmassa

Amazon on yksi maailman suurimmista pilvipalveluntarjoajista (Evans, 2017). Amazonin asiakaskuntaan kuuluu niin suuria monikansallisia pörssiyhtiöitä, kuin muutaman hengen startup-yrityksiä ja harrastajia (Wootton, 2017). Asiakaskunnan monimuotoisuuden on mahdollistanut palveluiden tarjoaminen useammalla eri hinnoittelumallilla, joista ”On demand” hinnoittelu on pienille toimijoille houkuttelevin, sillä se ei sisällä asiakkaan puolelta sitoutumista, vaan asiakas maksaa palveluista käytön mukaan (Nguyen, 2018).

Amazon Web Services on Amazonin tarjoama kokoelma pilvipalveluja, joihin kuuluu erilaisia virtuaalikoneita, tietokanta- ja tiedostopalvelimia, sekä sisällönvälitysjärjestelmiä ja analytiikkapalveluja.

EC2 (Elastic Cloud Computing) on Amazonin palvelu, jossa asiakas pystyy luomaan tarpeisiinsa sopivia instansseja tai virtuaalikoneita. EC2-instansseja on saatavilla lukuisilla eri ohjelmisto- ja laitekonfiguraatioilla, sekä hinnoittelumalleilla. EC2-instansseja voidaan hallinnoida etänä SSH-yhteyden tai web-käyttöliittymän kautta. Asiakas pystyy valitsemaan palvelimensa käyttöjärjestelmän useista eri esiasennettavista vaihtoehdoista.

Suosituin vaihtoehto palvelimen käyttöjärjestelmäksi maailmanlaajuisesti on Ubuntu Linux (W3Techs, 2018).

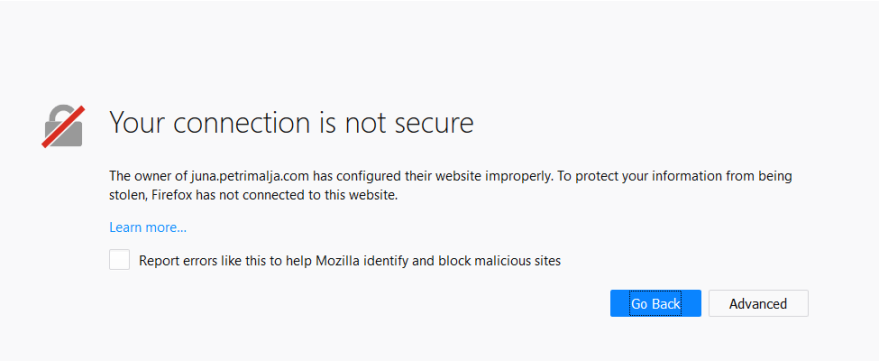
Ubuntu on Canonical-yhtiön ylläpitämä Linux-distribuutio, joka on noussut työpöytäkäytössä maailman suosituimmaksi Linux jakeluksi. Ubuntu on noussut suosituimmaksi Linux-jakeluksi myös palvelimien käyttöjärjestelmänä. Marraskuussa 2018 Ubuntua käytti 37,9 prosenttia Linux-käyttöjärjestelmää ajavista palvelimista (W3Techs, 2018).

Ubuntu Server on käyttöjärjestelmän palvelimiin tarkoitettu versio, joka tavallisesta työpöytäversiosta poiketen ei sisällä lainkaan graafista työpöytäjärjestelmää, vaan on tarkoitettu käytettäväksi kokonaisuudessaan komentorivin kautta. Jakelussa on myös oletusarvoisesti mukana ohjelmapaketteja, joita normaalisti palvelinkäytössä tarvitaan. Ubuntu Server asennuksessa on myös mahdollista jo asennuksen aikana valita haluttuja palvelinohjelmistoja mukaan asennukseen.

Apache on maailman suosituin web-palvelinohjelma, joka vastaa palvelimelle tuleviin HTTP/HTTPS-pyyntöihin. Apachea käyttää 45,1 prosenttia tilastoiduista web-palvelimista (W3Techs, 2018).

HTTP, eli HyperText Transfer Protocol on WorldWideWebiä varten luotu tiedonsiirtoprotokolla, joka on tarkoitettu hypertekstin ja median siirtoon. Nykyisin käytössä oleva HTTP/1.1 on määritelty alun perin vuonna 1999 julkaistussa RFC2616-standardissa, mutta sen on lähitulevaisuudessa korvaamassa uusi vuonna 2015 esitelty HTTP/2-standardi, joka mahdollistaa useiden resurssien siirron yhden palvelinpyynnön aikana ja palvelimelta asiakkaalle suunnatut push-viestit. (Fielding, ym., 1999; Belshe, Peon & Thomson, 2015)

HTTPS, eli HyperText Transfer Protocol Secure on nimensä mukaisesti turvallinen versio HTTP-protokollasta, jossa liikenne palvelimen ja asiakkaan välillä on salattu. Ennen tiedonsiirron aloittamista asiakaspääte ja palvelin vaihtavat keskenään salausavaimia, joilla verkkoliikenne tullaan salaamaan. Kummallakin on erikseen olemassa yksityinen avain, jolla salattu tieto pystytään purkamaan. (Comodo CA, n.d.)



Kuva 5. Firefox-selaimen näyttämä varoitus sertifikaatista

Kaikki modernit web-selaimet edellyttävät nykyisin HTTPS-yhteyttä käyttävältä web-palvelimelta sertifikaattia, jonka myöntäjätahon tulee olla tunnettu (Strauss, 2017). Sertifikaatteja myyvät yritykset tarjoavat myös mahdollisuuden lisätä tarkempia varmistettuja asiakkaan tunnistetietoja sertifikaattiin, joiden avulla palveluiden käyttäjät voivat varmistua myös palveluntarjoajan identiteetistä. Let's Encrypt on suosittu ilmainen sertifikaattien myöntäjätaho, jota voidaan käyttää verkkopalvelun yhteyden salaamiseen, silloin kun omistajan identiteetin varmistaminen ei ole välttämätöntä (Gebhart & Schoen, 2018). Se käyttää juurisertifikaattinaan IdenTrustin sertifikaattia, joka on maailman yleisin sertifikaattien myöntäjätaho (W3Techs, 2018). Jos palvelussa käytetään niin kutsuttua itsemyönnettyä sertifikaattia, jonka myöntäjätaho ei ole tunnettu, selain varoittaa sivustolle saapuvaa käyttäjää kuvan 5 mukaisesti (Nicholson, 2016).

2.3 **Relaatiotietokanta**

Relaatiotietokannassa eri tauluilla on keskenään relaatioita, joissa kahden tai useamman taulun sisältämät tiedot liittyvät toisiinsa. Tauluun määriteltyä perusavainta voidaan käyttää toisen taulun tiedoissa viiteavaimena, jolloin taulujen sisältämät tiedot saadaan yhdistettyä toisiinsa tämän yhteisen nimittäjän kautta. Taulun perusavaimen tulee aina olla uniikki, eli se saa esiintyä taulussa ainoastaan kerran. Sama viiteavain sen sijaan voi esiintyä taulussa useilla eri riveillä. Viiteavaimen avulla voidaan liittää kahden tai useamman taulun sisältämät tiedot yhteen, vertaamalla yhden taulun perusavainta ja toisten taulujen viiteavaimia. (Hovi, Huotari & Lahdenmäki, 2003, s. 9)

Tässä työssä luodussa junien tietoja sisältävässä taulussa junanumero on perusavain, yhtä junanumeroa siis vastaa aina vain yksi junavuoro. Aikataulutiedot sisältävässä taulussa junanumero esiintyy viiteavaimena, jolloin yhdellä junavuorolla voi taulussa olla useita aikataulutietoja. Taulujen sisältämiä rivejä pystytään yhdistelemään kyselyissä avainten perusteella.

Tietokannan tietoturvan kannalta on tärkeää, että mikään kolmas osapuoli ei pysty yhdistämään tietokantapalvelimeen suoraan verkosta. Yhdistäminen verkosta voidaan estää joko koko tietokantaan, tai yksittäisille käyttäjille erikseen. Estämällä etäyhteyden muodostaminen tietokantaan, on tietokanta suojattuna, vaikka hyökkääjä olisikin saanut tietokannan salasanat haltuunsa.

Tietokannan suunnittelussa on myös tarpeen huomioida, että tietokannan käyttäjille ei anneta laajempia oikeuksia, kuin on välttämätöntä. Jos useampi eri sovellus käyttää tietokantaa, jokaiselle sovellukselle kannattaa luoda erillinen tunnus tietokantaan ja määrittää jokaiselle tunnukselle sovelluksen toiminnan kannalta välttämättömät käyttöoikeudet. Rajoitta-

malla yksittäisten käyttäjien oikeuksia vastaamaan niiden normaalisti suorittamia toimintoja, karsitaan hyökkääjän mahdollisuuksia ajaa mielivaltaisia komentoja tietokannassa SQL-injektiohyökkäyksessä.

2.4 **Palvelinpuolen web-ohjelmointi**

Palvelinpuolen ohjelmoinnilla tarkoitetaan web-sovelluksen palvelimella suoritettavan komponentin ohjelmointia. Web-palvelin suorittaa ohjelmakoodin palvelimella ennen asiakkaan lähettämään HTTP-pyyntöön vastaamista. Asiakkaalle palautetaan ainoastaan ohjelman tuloste, eikä varsinaista lähdekoodia, toisin kuin asiakaspuolen web-ohjelmissa.

Palvelinpuolen web-ohjelmoinnissa yleisimmät ohjelmointikielet ovat PHP, C# ja Java (W3Techs, 2018). Myös muut ohjelmointikielet ovat viime vuosina kasvattaneet suosiotaan palvelinpuolen web-ohjelmoinnissa, näistä mainittakoon Javascript ja Python, joista Pythonia on käytetty myös tässä työssä datan käsittelyssä (Garbade, 2017).

PHP, jota tässä työssä on käytetty pääasiallisena palvelinpuolen ohjelmointikielenä, on Rasmus Lerdorfin vuonna 1994 kehittämä ohjelmointikieli (The PHP Group, n.d.). PHP-ohjelmakoodia suoritetaan palvelimella PHP-tulkilla, kun käyttäjän selain lähettää palvelimelle pyynnön PHP-tiedostosta. Käyttäjälle palautetaan PHP-ohjelman antama tuloste.

PHP on dynaamisesti tyypittyvä ohjelmointikieli, joka tarkoittaa, että muuttujien tietotyyppiä ei tarvitse PHP:ssä erikseen määritellä, vaan tulkki osaa ohjelman suorituksen aikana automaattisesti tehdä tarvittavat tyyppimuunnokset (Pillai, 2004). PHP on alun perin kehitetty proseduraaliseksi ohjelmointikieleksi, mutta siihen on myöhemmin lisätty tuki olio-ohjelmointiin (The PHP Group, n.d.).

Toinen tässä opinnäytetyössä palvelimella käytetty ohjelmointikieli on Python, joka myös on dynaamisesti tyypittyvä tulkattava ohjelmointikieli. Muuttujille ei siis tarvitse niitä luodessa määrittää tietotyyppiä, vaan Python-tulkki määrittää muuttujalle tietotyypin sen sisällön mukaan. Pythonin ja PHP:n erona on kuitenkin se, että Python on vahvasti tyypitetty ohjelmointikieli, kun taas PHP:ssä on heikko tyypitys. (Pillai, 2004)

2.5 **Asiakaspuolen web-ohjelmointi**

Asiakaspuolen web-ohjelmoinnissa kehitetään niitä web-sovelluksen osia, jotka suoritetaan vasta käyttäjän selaimessa. Nykyisin asiakaspuolen web-ohjelmoinnissa alan standardiksi on muodostunut Javascript-ohjelmointikieli (W3Techs, 2018). Javascriptin rinnalla käytetään HTML- ja CSS-kuvauskieliä web-sovelluksen käyttöliittymän rakenteen ja ulkoasun määrittämiseen.

Javascript on tulkattava olio-ohjelmointikieli, jossa on myös dynaaminen tyypitys. Se ei nimestään huolimatta ole yhteydessä Java-ohjelmointikieleen, joka on myös ollut aiemmin suosittu ohjelmointikieli web-sovelluskehityksessä ja on edelleen jokseenkin suosittu palvelinpuolen ohjelmoinnissa (Byrne, 2016).

Javascript mahdollistaa web-sivun sisällön muuttamisen asynkronisesti reaaliajassa ja se mahdollistaa reagoinnin käyttäjän toimintaan web-sivulla, tehden näin web-sivusta interaktiivisen web-sovelluksen. HTML-elementteihin voidaan liittää eri tapahtumien seurauksena aktivoituvia käsittelijöitä, jolloin selain suorittaa määritellyn Javascript-funktion käyttäjän suorittaessa tietyn toiminnon web-sivulla.

Tapahtuman käsittelijä voidaan liittää web-sivulla olevan lomakkeen tekstikenttiin tai lähetysnappiin, jolloin lomakkeen sisältö voidaan validoida ennen kuin se lähetetään palvelimelle. Validoinnilla ennen lomakkeen lähettämistä säästetään palvelimen resursseja, kun web-sovellus asiakkaan päässä pystyy tarkistamaan, että lomake on täytetty oikein ennen sen lähettämistä palvelimelle.

Asiakaspuolen web-ohjelmoinnissa on tarjolla suuri määrä kolmannen osapuolen kirjastoja, joilla ohjelmointikieleen saadaan lisättyä uusia toiminnallisuuksia ja vähennettyä kirjoitettavan ohjelmakoodin määrää käyttämällä kirjastojen valmiita funktio ja oliorakenteita. (Walsh, 2007)

2.6 **Versionhallinta**

Sovelluskehityksessä on tärkeää käyttää hyödyksi versionhallintatyökaluja, joilla pidetään kirjaa ohjelmakoodiin tehdyistä muutoksista ja mahdollistetaan palaaminen takaisin lähdekoodista vanhempaan versioon, jos ohjelman toiminta häiriytyy jonkin tehdyn muutoksen seurauksena. (McQuaid, 2014)

Versionhallintatyökalun käyttämisestä tulee erityisen tärkeää silloin, kun saman ohjelmistoprojektin parissa työskentelee useampi ohjelmoija. Versionhallintatyökalulla heidän erillään toisistaan kehittämänsä versiot pystytään yhdistämään, jopa täysin automaattisesti. Versionhallintatyökalu mahdollistaa myös kaikkien muutosten jäljitettävyyden. (McQuaid, 2014)

Nykyisin yhdeksi suosituimmista versionhallintatyökaluksi on noussut avoimen lähdekoodin hajautettu versionhallintatyökalu Git, jonka on alun perin kehittänyt Linux-käyttöjärjestelmänkin luojana tunnettu Linus Torvalds. Git eroaa monista muista versionhallintatyökaluista siinä, että se on hajautettu, jolloin jokainen kehittäjä pystyy työskentelemään itsenäisesti ja yhdistämään sitten tekemänsä muutokset yhteiseen koodivarantoon. (Brown, 2018)

Gitin käytön yleistyessä on syntynyt useita verkkopalveluja tarjoamaan palvelintilaa, sekä hallintapalveluja Git-koodivarannoille. Näistä yleisimmin käytetty on Github, jota myös Linux-käyttöjärjestelmän ytimen kehitystiimi nykyisin käyttää (McMillan, 2012).

Ulkoisten koodivarantojen kanssa toimiessaan, kehittäjät päivittävät muutoksiaan yhteiselle palvelimelle. Jotta kehittäjä voisi siirtää omat muutoksensa palvelimelle, hänen oma versionsa lähdekoodista täytyy olla uudempi, kuin palvelimella oleva versio. Kehittäjän versio voi olla palvelimen versiota vanhempi, jos joku toinen kehittäjä on tehnyt ohjelmakoodiin muutoksia sen jälkeen, kun kehittäjä on ladannut projektin lähdekoodin itselleen. Ollessaan palvelimen versiota jäljessä, täytyy kehittäjän ennen omien muutoksiensa siirtämistä palvelimelle, ladata uusimmat muutokset itselleen, sekä yhdistää ne omaan versioonsa lähdekoodista. Näin eri versioiden mahdolliset ristiriidat eivät koskaan siirry palvelimella olevaan yhteiseen versioon.

2.7 **Tietoturva**

Web-sovelluskehityksessä on erityisen tärkeää huomioida tietoturva, sillä kehitettävä sovellus on aina yhteydessä verkkoon ja alttiina hyökkäyksille. Web-sovelluksen ei välttämättä tarvitse edes olla erityisen tunnettu, sillä hyökkääjillä on nykyisin käytössään automatisoituja työkaluja, joilla voidaan etsiä tunnettuja tietoturva-aukkoja verkossa olevista web-sovelluksista. (Bashah Mat Ali, Shakhatreh, Abdullah & Alostad, 2011, ss. 455-456)

Erityisen tärkeää web-sovelluksen kehittämisessä on olla luottamatta käyttäjältä tulevan syötteen aitouteen. Käyttäjäsyötteen kautta hyökkääjä voi onnistua saamaan pääsyn palvelimen tietokantaan tai onnistua syöttämään web-sivulle omaa haitallista ohjelmakoodia. (ENISA, 2016)

2.7.1 Palvelimen tietoturva

Palvelin, joka on yhteydessä julkiseen verkkoon tai jonne on pääsy ulkopuolisilla henkilöillä, on tarpeen suojata. Palvelimen tietoturvan kannalta on tärkeää, että liikenne palvelimelle on suojattu palomuurilla ja vain ne portit, joita palvelinohjelmat käyttävät, ovat avoinna.

Jos palvelimelle yhdistetään etäkäyttöyhteydellä, kuten SSH:lla, on tärkeää, että palvelimelle tunnistaudutaan salausavaimella joka on suojattu salasanalla. Salausavaimia on tärkeää säilyttää niin, että ne eivät joudu vääriin käsiin ja ne tulisi myös vaihtaa säännöllisesti uusiin. Jokaisella palvelimeen yhdistävällä laitteella tulisi myös olla oma käyttäjäkohtainen avain. Näin pystytään hallitsemaan palvelimen käyttöoikeuksia käyttäjä- ja laitekohtaisesti. (SSH Communications Security, 2018)

Palvelimella ajettavilla palvelinohjelmilla tulee olla rajatut käyttöoikeudet järjestelmään, jotka on määritelty siten, että sovellus pystyy suorittamaan vain niitä toimintoja, joihin se on suunniteltu. Rajoittamalla sovellusten käyttöoikeuksia, pystytään suojaamaan palvelimen muita järjestelmiä hyökkäykseltä, yhden sovelluksen tietoturvan murtuessa.

Palvelimen tietoturva tulee ottaa huomioon myös palvelinohjelmien konfiguroinnissa, joka saattaa oletusmuodossaan olla tarkoitettu vain kehityskäyttöön ja sallia potentiaalisesti vaarallisia toimintoja (Panda Security, 2017; ENISA, 2016). Kehittäjän kannattaa kytkeä pois käytöstä kaikki sellaiset ominaisuudet, joita kehitettävässä sovelluksessa ei käytetä. Myös virhetietojen näyttäminen käyttäjälle, voi paljastaa hyökkääjälle hyödyllistä tietoa sovelluksen tai palvelimen konfiguraatiosta (Geerling, 2016).

2.7.2 Tietoturva web-sovelluksissa

Palvelinpuolen web-ohjelmoinnissa suurin riski keskittyy sovelluksen käyttäjältä vastaanotettavan syötteen käsittelyyn (ENISA, 2016). Jos käyttäjän syötettä välitetään edelleen tietokantaohjelmalle, saattaa palvelin tällöin altistua SQL-injektiohyökkäykselle. SQL-injektiohyökkäyksessä hyökkääjä onnistuu ajamaan mielivaltaisia tietokantakyselyjä palvelimelle (Porup, 2018). Tämän vuoksi on tärkeää huomioida tietokantaa suunniteltaessa, että tietokantakäyttäjillä ei ole laajempia oikeuksia ajaa käskyjä tietokantaan, kuin on tarpeen sovelluksen normaalissa toiminnassa.

Toinen mahdollinen riski muodostuu, jos palvelinohjelma suorittaa komentoja tai muita ohjelmia suoraan palvelimen käyttöjärjestelmässä. Tämä riski voi realisoitua myös, vaikka sovellusta ei olisi suunniteltu tällaisia käskyjä ajamaan, mikäli ohjelmakoodia suorittavassa tulkissa on tietoturva-aukko. Palvelinohjelmien tietoturva-aukkoja hyväksikäyttämällä hyökkääjä voi päästä ajamaan mielivaltaisia komentoja palvelimen käyttöjärjestelmässä. On siis tärkeää, että palvelinohjelmia ei suoriteta järjestelmässä pääkäyttäjän oikeuksilla, ellei sovelluksen toiminta edellytä laajoja oikeuksia järjestelmään.

Mahdollinen tietoturvariski on myös, jos käyttäjiltä vastaanotettua syötettä sijoitetaan takaisin web-sivulle. Silloin hyökkääjä voi käyttää tätä syöttääkseen web-sivulle haitallista ohjelmakoodia. Tätä hyökkäystä kutsutaan XSS-hyökkäykseksi, joka tulee sanoista Cross Site Scripting. (Kaur & Kaur, 2017, s. 154)

XSS hyökkäyksessä hyökkääjä pyrkii saamaan hyökkäyksen kohteena olevalle web-sivulle omaa Javascript-ohjelmakoodiaan. Tavoitteena on suorittaa hyökkääjän ohjelmakoodia sivustolla vierailevien käyttäjien selaimessa. (Kaur & Kaur, 2017, s. 155)

Hyökkäys mahdollistaa sivun sisällön vaihtamisen, näppäinsyötteiden nauhoittamisen tai käyttäjän ohjaamisen kokonaan toiselle web-sivulle. Sivustolla jossa on sisäänkirjautuminen tai verkkokauppa, hyökkääjä voi onnistua kaappaamaan käyttäjän salasanoja tai luottokorttitietoja. Hyökkäys mahdollistaa myös haittaohjelmien asentamisen käyttäjän koneelle. (Kaur & Kaur, 2017 ss. 156-157)

XSS-hyökkäysten estämiseksi kaikki käyttäjiltä peräisin oleva syöte tulee puhdistaa HTML-erikoismerkeistä, ennen tulostamista web-sivulle. (Kaur & Kaur, 2017, s. 155)

3 TAVOITE JA TARKOITUS

Tämän opinnäytetyön tavoitteena oli kehittää web-sovellus, jolla säännöllisesti junalla matkustava voisi nopeasti ja mutkattomasti tarkastaa oman junavuoronsa tilanteen. Tietojen tuli olla mahdollisimman reaaliaikaisia ja ne tuli esittää käyttäjälle mahdollisimman selkeässä ja helppolukuisessa muodossa.

Säännöllisesti sovellusta käyttävän matkustajan tulisi saada oman junavuoronsa tiedot esille ilman, että hänen tarvitsisi erikseen hakea omaa junavuoroansa järjestelmästä. Tämä tarkoittaa, että sovelluksen pitää muistaa käyttäjän säännöllisesti käyttämät junavuorot ja tarjota näin käyttäjälle oikean junavuoron tiedot, ilman erillisiä toimenpiteitä käyttäjältä.

Sovelluksen tulisi esittää selkeästi yhdessä näkymässä kaikki oleelliset junavuoroa koskevat tiedot, helposti ymmärrettävässä muodossa ja reaaliaikaisesti. Reaaliaikaisuuden tulee myös ilmentyä käyttäjälle siten, että hän pystyy varmistumaan tietojen ajantasaisuudesta ja paikkansapitävyydestä.

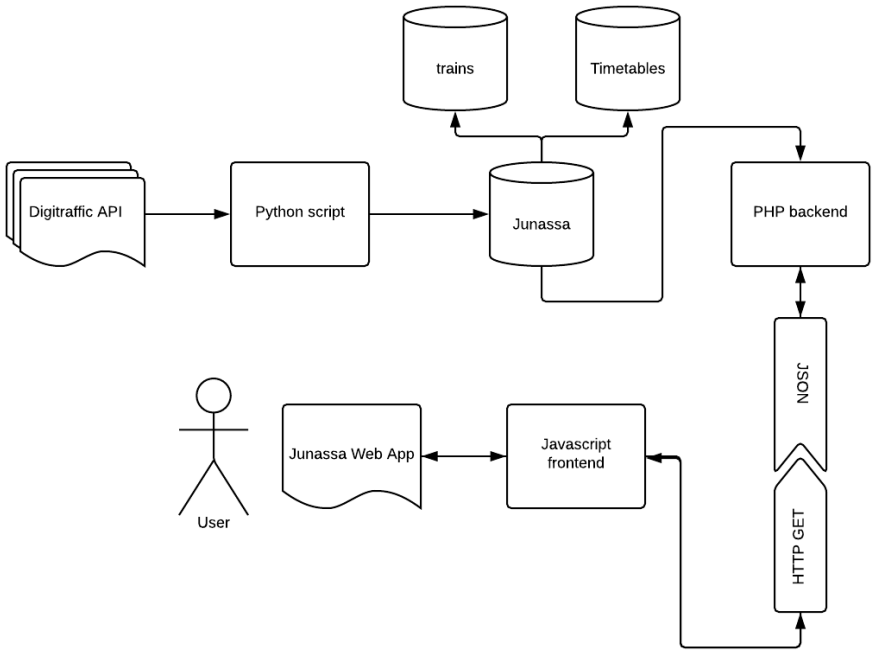
Kehitettävän sovelluksen tulisi olla alustariippumaton, responsiivinen ja suorituskykyinen kaikilla kohteena olevilla päätelaitteilla.

4 SUUNNITTELU JA TOTEUTUS

Sovelluksen suunnittelun lähtökohtana oli suunnitella sovellus siten, että saatavilla olevaa rautatieliikenteen avointa dataa pystyttäisiin esittämään helppolukuisesti, kompaktissa muodossa ja siten, että tieto olisi mahdollisimman ajantasaista. Suunnittelussa tuli huomioida myös, että sovelluksen eri komponentit olisivat riittävän suorituskykyisiä, eivätkä vaatisi suuria resursseja toimiakseen tehokkaasti.

Käytettävien tekniikoiden tulisi olla saatavilla olevan tutkimustiedon ja yleisten mielipiteiden mukaisesti käyttötarkoitukseen parhaiten sopivia.

Teknisiin valintoihin vaikuttaa myös tekijän taustaosaaminen käytettävistä järjestelmistä ja tekniikoista. Kuvassa 6 näkyvässä kaaviossa on hahmoteltu sovelluksen eri komponentteja ja niiden toimintaa yhtenä kokonaisuutena.



Kuva 6. Kaavio sovelluksen rakenteesta ja toiminnasta

4.1 **Suunnittelu**

Sovelluksen teknisen toteutuksen suunnittelussa lähtökohtana oli pohtia mitkä ovat ne tärkeimmät tiedot, joita junamatkustaja kaipaa ennen junamatkaa ja matkan aikana. Samanaikaisesti tarkasteltiin, millaista dataa on saatavilla palvelun pohjana olevassa Liikenneviraston Digitraffic-rajapinnassa ja kuinka ajantasaisia nämä tiedot ovat.

Seuraavaksi suunnittelussa pohdittiin, miten dataa tulisi esittää, jotta se olisi mahdollisimman helppolukuista ja mahtuisi pieneen tilaan mobiililaitteen näytölle. Datan esitystavan selkiytyessä alettiin myös pohtia sovelluksen rakennetta ja miten tiedot saataisiin välitettyä rajapinnasta aina loppukäyttäjän selaimeen.

Rajapinnan toimintaan tutustuessa muodostui hyvin nopeasti selkeäksi, että tiedot olisi välttämätöntä kerätä toteutettavan sovelluksen omaan tietokantaan. Digitraffic-rajapinnassa ei ole sovelluksen toiminnan kannalta riittävän monipuolisia mahdollisuuksia rajata tietohakuja, eikä rajapinnan suorituskyky ole riittävän hyvä, jotta loppukäyttäjän selain voisi hakea datan suoraan rajapinnasta käyttökokemuksen kärsimättä merkittävästi.

Tarve omalle tietokannalle tarkoitti myös, että sovellukselle tarvittaisiin oma palvelinympäristö, johon tietokantaa päivittävä ohjelma sitten rakennettaisiin.

4.1.1 Avoimen datan rajapinta

Rajapinnan hyödyntämisen suunnittelu aloitettiin tutustumalla Liikenneviraston Digitraffic APIn toimintaan ja sieltä saatavan JSON-muotoisen datan rakenteeseen ja sisältöön. Ennen minkään muun toteutuksen osan tarkempaa suunnittelua tai prototyyppien kehitystä, testattiin datan hakemista ja käsittelyä. Tämä ohjelmallinen, automaattiseen tiedonkäsittelyyn liittyvä testaus päätettiin toteuttaa käyttäen Python-ohjelmointikieltä. Pythonilla tiedonkäsittely onnistuu yksinkertaisen ohjelmarakenteen ansiosta nopeasti ja pienellä määrällä ohjelmakoodia.

Rajapinnan toimintaan ja sen antamaan dataan tutustuessa kävi nopeasti selväksi, että suunniteltu sovellus tarvitsisi oman tietokannan ja palvelimella ajettavan taustajärjestelmän päivittämään palvelimen omaa tietokantaa rajapinnasta. Oma tietokanta tarvittiin, sillä rajapinnasta ei pystytä rajaamaan kyselyjä sovelluksen edellyttämällä tavalla, jolloin sovellus joutuisi lataamaan loppukäyttäjälle paljon ylimääräistä dataa ja hoitamaan datan käsittelyn ja analysoinnin loppukäyttäjän päässä, jolloin sovelluksen toiminta hidastuisi merkittävästi.

Tiedonhakua oli aluksi testattu Python-ohjelmilla ja näitä päätettiin käyttää myös varsinaisessa sovelluksessa pohjana tiedon hakemiseen, käsittelyyn ja syöttämiseen tietokantaan. Pythonin käyttöä voidaan perustella sen tehokkuudella nimenomaan tietojenkäsittelyssä, kehitystyön nopeudella vähemmän ohjelmakoodin ansiosta, sekä tästä seuraavalla helpommalla koodikannan ylläpidolla.

Palvelimelle toteutettu Python-ohjelma lukee datan Digitraffic-rajapinnasta, erittelee sieltä palvelun toimintaan tarvittavat tiedot, käsittelee ne ja tallentaa edelleen palvelimella olevaan tietokantaan. Tietokannasta erillinen käyttäjää palveleva palvelinohjelma pystyy hakemaan tietoja tehokkaasti, hyödyntäen kyselyissä relaatiotietokannan kykyä yhdistellä ja hakea tietoja nopeasti.

4.1.2 Käytettävät palvelinpuolen tekniikat

Palvelinpuolella käytettävien tekniikoiden osalta sovelluksen kehityksessä oli useita käyttötarkoitukseen sopivia vaihtoehtoja, joista käytettävät tekniset ratkaisut valittiin suosion ja yhteensopivuuden kannalta. Valinnoissa vaikutti myös aiempi tuntemus valittujen teknisten ratkaisujen käytöstä.

HTTP-palvelimeksi valittiin Apache2, joka onkin tähän suosituin vaihtoehto ja mahdollistaa monipuolisten lisäosien asentamisen ilman monimutkaisia konfiguraatiomuutoksia (W3Techs, 2018). Apacheen on myös mahdollista konfiguroida SSL-salaus nopeasti kolmannen osapuolen työkalulla, jossa koko sertifikaatin asennus ja Apache-konfiguraation muuttaminen pystytään tekemään ohjelmallisesti.

Palvelinpuolen ohjelmoinnissa käytettiin PHP:tä, joka on tällä hetkellä ylivoimaisesti yleisin käytössä oleva palvelinpuolen ohjelmointikieli (W3Techs, 2018). PHP on myös hyvin helppo asentaa toimimaan Apachen rinnalle ja se mahdollistaa tietokannan hallinnoinnin PHPMyAdmin-työkalulla, jolla tietokantaa pystytään hallinnoimaan suoraan selaimessa toimivalla graafisella käyttöliittymällä.

Suunnitellun sovelluksen toiminta edellyttää toimiakseen relaatiotietokannan, jolla pystytään tekemään monipuolisia hakuja yhdistellen tietoja useista eri tietokantatauluista. Tietokannan valinnassa täytyi luonnollisesti huomioida myös yhteensopivuus muiden valittujen teknisten ratkaisujen kanssa, joten MySQL, joka on myös yksi suosituimmista tietokantajärjestelmistä, valittiin sovelluksen tietokannaksi (Arsenault, 2017).

4.1.3 Käytettävät asiakaspuolen tekniikat

Asiakaspuolen osalta sovelluksen tekninen ratkaisu on alan standardiksi muodostunut HTML5, CSS ja Javascript, jossa hyödynnettiin myös jQuery-kirjastoa. JQuery on yleisin web-sivuilla käytettävä Javascript-kirjasto (W3Techs, 2018).

Suurin osa sovelluksen ohjelmakoodista on kirjoitettu Javascript-ohjelmointikielellä, joka tukee tavoitetta, että mahdollisimman suuri osa sovelluksen toimintaan liittyvästä laskennasta tehtäisiin loppukäyttäjän selaimella. Palvelimella olevan PHP:n pääasiallinen tehtävä on tarjoilla Javascriptille dataa tietokannasta JSON-muotoisena ja välttää palvelinpäässä paljon järjestelmäresursseja kuluttavia toimintoja.

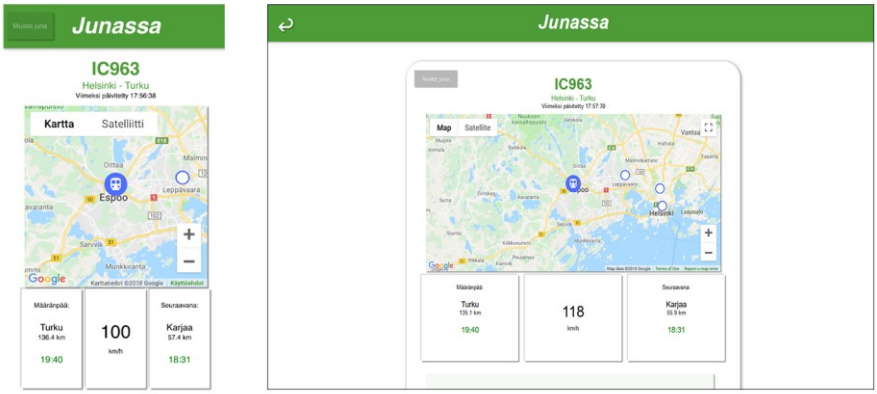
Javascriptin kanssa käytetään jQuery-kirjastoa, jolla saadaan merkittävästi vähennettyä tarvittavan ohjelmakoodin määrää ja näin nopeutettu kehitystyötä (Walsh, 2007). JQuerylla JSON-datan hakeminen asynkronisesti palvelimelta ja parsiminen Javascript-olioksi saadaan tehtyä yhdellä rivillä koodia, jonka lisäksi datan läpikäyminen onnistuu tehokkaasti jQueryn each-metodilla.

Dynaamisen karttanäkymän toteuttamisessa päätettiin hyödyntää Googlen karttapalveluita, johon on helppo liittää omaan web-sovellukseen liittyviä karttamerkkejä ja toimintoja. Google Maps APIn käyttö edellyttää rekisteröitymistä Google Cloud Platform -palveluun ja API-avaimen luomista.

4.1.4 Käyttöliittymäsuunnittelu

Web-sovelluksen käyttöliittymää suunniteltaessa lähtökohtana oli, että sen tulee olla täysin käytettävissä älypuhelimen kapealla kosketusnäytöllä. Koska kaiken näytettävän datan tuli mahtua älypuhelimen näytön kapeaan tilaan, täytyi harkita tarkkaan, mitkä tiedot ja toiminnot ovat ensisijaisia ja tulisi mahtua näkymään näytöllä ilman vierittämistä. Saman käyttöliittymän pitäisi kuitenkin olla miellyttävä käyttää myös huomattavasti suuremmalla tietokoneen näytöllä ja hiirellä ohjaten.

Ratkaisuna oli aloittaa käyttöliittymän hahmottaminen mobiilikäyttäjän näkökulmasta, eli niin että kaikki tärkein mahtuu mobiililaitteen näytölle samanaikaisesti. Mobiilikäyttäjällä sivuston sisältö on koko näytön leveydellä, kun taas suuremmalla näytöllä sisältö on rajattu erillisen laatikon sisään, kuten näemme kuvassa 7. Näin pystytään tarjoamaan miellyttävä käyttökokemus kaikenkokoisten päätelaitteiden käyttäjille. Elementtien ja niiden sisältämän sisällön tulisi myös skaalautua näytön koon mukaisesti, asettelun pysyessä samana.



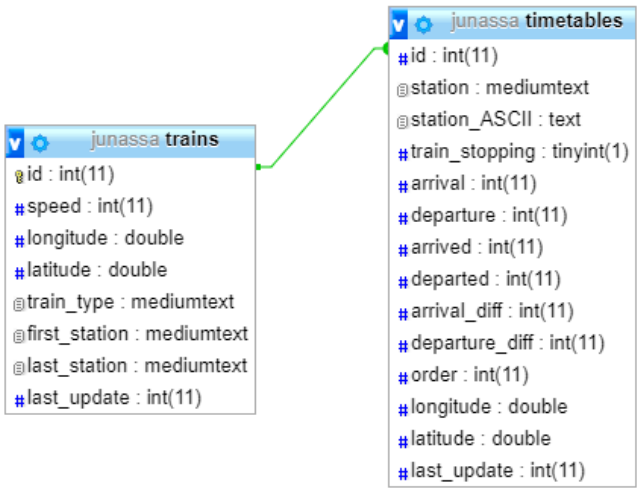
Kuva 7. Mobiili ja työpöytäkäyttöliittymä rinnakkain

4.1.5 Tietokantarakenne

Palvelun tietokanta muodostuu kuvassa 8 näkyvän rakenteen mukaisesti kahdesta taulusta, jotka linkittyvät toisiinsa id-arvon kautta ja muodostavat näin yksinkertaisen relaatiotietokannan. Tämä mahdollistaa tietokantakyselyt, joissa hyödynnetään dataa kummastakin taulusta.

Relaatiota käytetään hyödyksi erityisesti haettaessa junia käyttäjän antamalla hakusanalla, jolloin kyselyssä haetaan vastaavuuksia kummastakin tietokantataulusta. Palautettavat arvot tulevat kuitenkin ainoastaan toisesta taulusta. Tämä on toteutettu LEFT JOIN -kyselyllä, joka palauttaa

kaikki trains-taulun hakuehtoja vastaavat kentät, mutta sisältää ehtolauseita myös timetables-taulun arvoille.



Kuva 8. Tietokannan rakenne ja relaatiot

Trains-taulu sisältää tiedot yksittäisistä junista.

- *id*, junan numero, uniikki tunniste
- *speed*, junan nopeus kilometriä tunnissa
- *longitude*, junan koordinaatti pituussuunnassa
- *latitude*, junan koordinaatti leveyssuunnassa
- *train_type*, junan tyyppi, lähijunissa lähijunan tunnus
- *first_station*, junan lähtöasema
- *last_station*, junan pääteasema
- *last_update*, unix-aikakoodi, milloin tiedot on viimeksi päivitetty

Timetables-taulu sisältää jokaisen junan aikatauluun merkityt asemat.

- *id*, junan numero, foreign key -viittaus trains-tauluun
- *station*, aseman nimi
- *station_ASCII*, aseman nimi ilman ääkkösiä
- *train_stopping*, pysähtyykö juna reitille merkityllä väliasemalla
- *arrival*, aikataulun mukainen saapumisaika
- *departure*, aikataulun mukainen lähtöaika
- *arrived*, todellinen saapumisaika
- *departed*, todellinen lähtöaika
- *arrival_diff*, ero aikataulun mukaiseen saapumisaikaan
- *departure_diff*, ero aikataulun mukaiseen lähtöaikaan
- *order*, aseman järjestysnumero junan reitillä
- *longitude*, aseman koordinaatti pituussuunnassa
- *latitude*, aseman koordinaatti leveyssuunnassa
- *last_update*, unix-aikakoodi, milloin tiedot on viimeksi päivitetty

4.2 **Toteutus**

Web-sovelluksen kehitys toteutettiin syksyn 2018 aikana ja siinä käytettiin pohjana edellisenä keväänä luotuja Python-ohjelmia, joilla testattiin junan aikataulutietojen hakemista Liikenneviraston Digitraffic rajapinnasta. Tiedon hakemista rajapinnasta lukuun ottamatta, kaikki sovelluksen ohjelmakoodi kirjoitettiin opinnäytetyöprosessin aikana alusta loppuun saakka.

Sovelluksen toteutusalustaksi valittuun Amazon Web Servicesiin luotiin prosessin alussa uusi käyttäjätunnus Petrimalja-toiminimelle. Amazon tarjoaa uusille asiakkailleen vuoden määräajaksi osaa palveluistaan veloituksetta, tietyin rajoituksin. Yksi tähän ns. ”Free Tier” –tarjoukseen liitetty palvelu on EC2:n t2.micro-tyypin instanssi, joka valittiin tähän toteutukseen palvelinalustaksi. (Amazon, n.d.)

Palvelulle haluttiin helppokäyttöinen verkko-osoite, jossa päätettiin hyödyntää olemassa olevaa petrimalja.com-domain-osoitetta ja luoda siihen palvelua varten alidomain-osoite. Oman domain-osoitteen käyttö mahdollisti myös ilmaisen SSL-sertifikaatin saamisen Let’s Encrypt -palvelusta, Amazon AWS:n osoite on turvallisuussyistä estetty palvelusta.

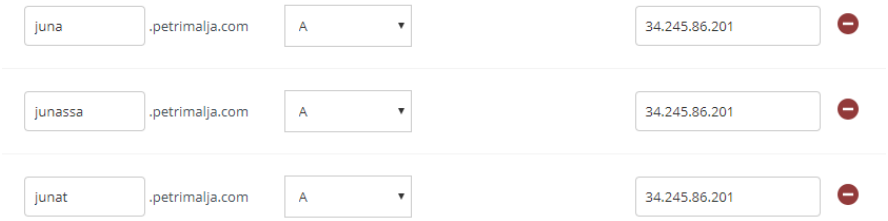
4.2.1 Palvelimen konfiguraatio

Palvelinpuolen osalta tekninen toteutus aloitettiin luomalla Amazon Web Services -pilveen t2.micro-tyyppinen EC2-instanssi, johon valittiin käyttöjärjestelmäksi Ubuntu Server 16.04 LTS -linuxjakelu. Virtuaalikonetta luotaessa AWS:ään ei ollut vielä tullut saataville uusinta 18.04 LTS -versiota, joka sisältää uudempia versioita tässä työssä käytetyistä ohjelmapaketeista. Uutta instanssia luotaessa, sille luodaan myös samalla SSH-avain, jolla virtuaalikoneelle kirjaudutaan. Uudelle instanssille luodaan myös Security Group, jolla määritellään avoimet portit, joista palvelimeen pystytään yhdistämään julkisesta internet-verkosta (Amazon Web Services, n.d.). Tässä työssä ainoat avattavat portit olivat HTTP-portti 80 ja HTTPS-portti 443. SSH-yhteyden käyttämä portti 22 on oletusarvoisesti auki.

Palvelimen esiasennus kestää muutamia minuutteja, jonka jälkeen palvelimelle muodostetaan etähallintayhteys. SSH-yhteyden kautta palvelimelle pystytään asentamaan muut valitut tekniset ratkaisut komentorivikomennoilla pakettienhallintatyökalua hyödyntäen.

Ensimmäisenä palvelimelle asennettiin Apache2-palvelin, PHP7 ja MySQL-tietokanta, eli niin sanottu ”LAMP stack”. Seuraavaksi asennettiin PHPMyAdmin-tietokantahallintatyökalu, sekä viimeisin saatavilla oleva Python3 ja tarvittavat Python-kirjastot. Apachen konfiguraatiossa muutettiin palvelimen juurikansion sijainti */var/www/html* -polusta */home/ubuntu/public_html* –hakemistoon, tässä ei kuitenkaan käytetty Apachen Userdir-modia, sillä tässä työssä palvelimelle ei ole tarkoitus luoda useampaa web-sivustoa.

Palvelimelle haluttiin selkokielinen domain-osoite, joka olisi helppo muistaa ja jolle olisi mahdollista saada SSL-sertifikaatti. Palvelulle luotiin olemassaolevalle *petrimalja.com*-domainille alidomainit *juna.petrimalja.com*, *junassa.petrimalja.com* ja *junat.petrimalja.com.* Kuvassa 9 nähdään One.com-webhotellipalvelun hallintasivujen DNS-asetukset, jossa sovelluksen käyttämät alidomainit on määritelty. A-tyypin DNS-asetuksella domain-nimi ohjataan palvelimen ip-osoitteeseen (dnsimple, n.d.).



Kuva 9. DNS-määrittelyt

4.2.2 Datan hakeminen rajapinnasta

Tietojen hakeminen Liikenneviraston Digitraffic-rajapinnasta aloitetaan hakemalla tiedot kaikista järjestelmässä olevista juna-asemista, niiden nimistä, lyhenteistä ja sijainneista. Haettavista tiedoista tärkeimmät tässä yhteydessä ovat asemien nimet ja niiden lyhenteet, sillä myöhemmin haettavissa aikataulutiedoissa asemiin viitataan ainoastaan niiden lyhenteillä. Juna-asemien sijaintitietoja sovellus käyttää sijoittaessaan junan pysähdyspaikkoja kartalle ja laskiessaan etäisyyttä seuraaville asemille.

Seuraavaksi ladataan tiedot kuluvan päivän junista, tämä vastaus on kooltaan päivästä riippuen jopa kymmeniä megatavuja. Koko päivän junien aikataulutiedot ladataan kuitenkin ainoastaan kerran, kun palvelimella oleva Python-ohjelma käynnistetään. Myöhemmin sovellus hakee vain muuttuneet aikataulutiedot 30 sekunnin välein.

Lopuksi haetaan vielä reaaliaikaisen seurannan kannalta tärkein, eli tieto junien sijainneista ja nopeuksista. Sijainti- ja nopeustiedot päivitetään aikataulutietoja useammin, viiden sekunnin välein.

Kaikkea rajapinnasta haettavia tietoja ei syötetä tietokantaan, vaan niistä parsitaan ainoastaan sovelluksen käyttöön olennaiset tiedot talteen syötettäväksi tietokantaan. Järjestelmä muuntaa aikataulutiedoissa olevat kellonajat ISO 8601 formaatista unix-aikakoodeiksi, helpompaa vertailua ja käsittelyä varten. Rajapinnan aikataulutiedoissa junan saapuminen ja lähteminen asemalle ovat erillisillä riveillä, joten ohjelma käy aikataulutiedot läpi ja yhdistää tietoja siten, että kaikki tarvittavat tiedot asemasta saadaan sisällytetyksi yhdelle riville tietokannassa.

4.2.3 Datan syöttäminen tietokantaan

Python-ohjelma, joka hakee junien tiedot rajapinnasta, parsii niistä olennaisimmat tiedot sopivaan rakenteeseen ja muuttaa aikakoodit unix-aikakoodeiksi. Käydessään läpi junan aikataulutietoja, ohjelma muodostaa jokaisen väliajan saapumis ja lähtöajasta yhden rivin tietokantaan. Riville on merkitty viiteavaimena junan tunnus, saapumis- ja lähtemisaika aikataulun mukaan, todelliset ajat junan jo saavuttua tai lähdettyä asemalta, sekä tieto kyseisen aseman sijainnista ja tiedot myöhästymisestä, jos juna kulkee myöhässä aikataulusta.

Ohjelma rakentaa aina käynnistettäessä tietokannan uudelleen käyttäen viimeisimpiä aikataulutietoja. Tietokannan alustamisen jälkeen jatketaan tietojen päivittämistä kahdessa syklissä, junien sijainti ja nopeustiedot päivitetään viiden sekunnin välein ja junien aikataulutiedot 30 sekunnin välein. Tietoja päivitetään eri rytmeissä, sillä aikataulutietojen päivittäminen on näistä raskaampi prosessi, jossa liikutellaan huomattavasti suurempia datamääriä, jotka tarvitsevat myös enemmän käsittelyä. Lopullisessa käyttötilanteessa 30 sekuntia vanhaa aikataulutietoa voidaan pitää edelleen erittäin ajantasaisena, kun taas sijainti ja nopeustietojen tulee olla lähes reaaliaikaisia, jotta niitä voitaisiin pitää ajantasaisina. Nopeus ja sijaintitiedot eivät rajapinnassakaan päivity reaaliajassa. Viiden sekunnin päivitystiheys on kompromissi, joka testikäytössä on todettu olevan sopivin päivitystiheys uusimman tiedon vastaanottamiseksi samalla palvelimen resursseja säästäen.

Palvelinta käynnistettäessä tietokannan rakentaminen vie aikaa vaihtelevasti noin 20-40 sekuntia, jolloin kaikkien päivän junien tiedot käsitellään ja ajetaan tietokantaan. Tietojen päivittäminen sen sijaan vie sijaintitietojen osalta vain sekunninsadasosia ja aikataulutietojen osalta vaihtelevasti puolesta sekunnista lähes kymmeneen sekuntiin.

4.2.4 Datan hakeminen tietokannasta

Datan hakemisen tietokannasta hoitaa yksinkertainen PHP-ohjelma, jonka ainoa pääasiallinen tehtävä on hakea tietoja tietokannasta ja luoda niistä JSON-muotoinen dokumentti Javascriptin käytettäväksi. PHP-ohjelmalle on luotu tietokantatunnus, jolla on ainoastaan lukuoikeus junatietoja käsitteleviin tauluihin. Rajoitetut käyttöoikeudet yhdistettynä tietokantaan esilähetettäviin kyselyihin, estää tehokkaasti SQL-injektiohyökkäyksen onnistumisen.

Käyttäjä kirjoittaa hakukenttään hakusanan, joka voi olla junan tunnus, junan numero tai jokin asema, jossa juna pysähtyy tai josta se lähtee. Käyttäjän käynnistäessä haun, sovellus pyytää palvelimelta tietoja get.php-ohjelmalta, välittäen sille parametrina kyselyn tyypin, hakukentän tapauksessa ’getTrainsByName’ ja käyttäjän antaman hakusanan.

PHP-ohjelma suorittaa SQL-kyselyn tietokantaan, jossa se yhdistää kummatkin junatietokannan sisältämät taulut LEFT JOIN-kyselyllä ja muodostaa junan tunnuksesta, junan numerosta ja aseman nimestä yhden merkkijonon vertailua varten. Merkkijonosta etsitään LIKE-vertailulla käyttäjän antamaa hakusanaa. Kyselyssä verrataan myös, pysähtyykö juna haetulla asemalla ja löytyykö sille tietokannasta sijaintitieto. Haettava juna ei myöskään saa olla vielä saapunut haetulle asemalle ja sen tiedot tulee olla päivitetty viimeisen viiden minuutin sisällä. PHP-ohjelma muodostaa tietokannan vastauksesta JSON-dokumentin, jonka se palauttaa pyynnön tehneelle Javascript-ohjelmalle.

Käyttäjä valitsee hakutuloksista oikean junan, jonka jälkeen hän siirtyy junan tiedot näyttävään näkymään. Junatiedoissa näkyville tulee reaaliaikaiset tiedot junasta, jotka päivitetään viiden sekunnin välein. Tietokantakyselyt tehdään asynkronisesti hyödyntäen jQuery-kirjastoa, joka myös hoitaa JSON-datan parsimisen Javascript-olioksi.

4.2.5 Datan käsittely

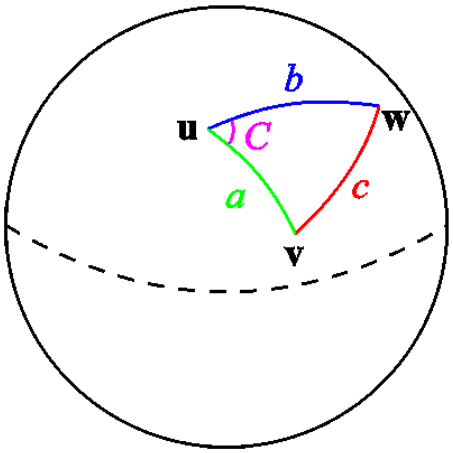
Junien tiedot ladataan palvelimelle ja lähetetään palvelimelta JSON-muodossa, mutta dataa käsitellään palvelimella, eikä se ole tietokannassakaan JSON-muodossa. Datasta on palvelimella parsittu pois kaikki ylimääräinen tieto, jota sovellus ei hyödynnä. Sovelluksen hyödyntämiä osia käsitellään ja järjestellään paremmin sovelluksen toimintaa tukevaan muotoon.

Selaimen vastaanottaessa palvelimelta datan JSON-muotoisena dokumenttina, siitä parsitaan Javascript-olio. Olion sisältöön pystytään viittaamaan ja sen sisältämää moniulotteista dataa pystytään käymään läpi rekursiivisesti.

Junan tietoihin kuuluu sen reaaliaikainen sijainti GPS-koordinaatteina. Tämä sama tieto on myös aikataulutiedoissa, jokaisesta asemasta erikseen. Koordinaateista pystytään laskemaan junan ja aseman etäisyys maan päällä käyttäen kaavaa,

$$d = 2r\ arcsin\left(\sqrt{sin^2\left(\frac{\varphi_2-\varphi_1}{2}\right) + cos(\varphi_1)\cdot cos(\varphi_2)\cdot sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right) \quad (1)$$

Jossa r on maapallon säde, φ_1 ja φ_2 ovat verrattavien koordinaattien leveysasteita ja λ_1, sekä λ_2 verrattavien koordinaattien pituusasteita. Koordinaateista on annetulla kaavalla mahdollista määrittää kahden pisteen välinen etäisyys d. Kuvassa 10 on havainnollistettu kahden koordinaattipisteen välisen etäisyyden määrittäminen geometrisen pallon päällä.



Kuva 10. Pisteiden välisen etäisyyden määrittäminen pallon pinnalla

4.2.6 Datan esittäminen käyttäjälle

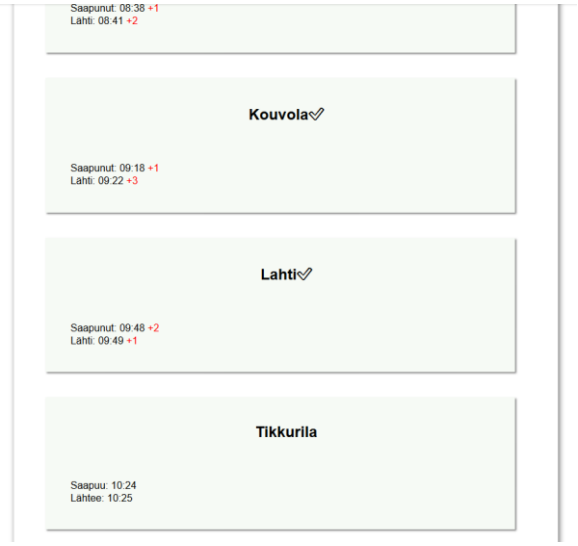
Junan tiedot esitetään käyttäjälle kuvassa 11 nähtävässä näkymässä. Näkymästä käyttäjä pystyy yhdellä silmäyksellä katsomaan junan ja asemien sijainnin kartalla, junan nopeuden, etäisyyden ja arvioidun saapumisajan seuraavalle asemalle ja määränpäähän. Näkymässä on myös esillä tieto, milloin näytettävät tiedot on viimeksi päivitetty tietokantaan, jolloin käyttäjä voi varmistua tietojen ajantasaisuudesta.

Kuva 11. Junatietojen näkymä sovelluksessa tietokoneella

Selaamalla näkymää alemmas, käyttäjä näkee kuvassa 12 näkyvässä näkymässä junan aikataulutiedot kronologisessa järjestyksessä junan lähtöase-

masta alkaen. Aikataulutiedoissa näkyy myös junan myöhästymiset minuutin tarkkuudella, sekä tulevien asemien arvioitu ja aikataulun mukainen saapumis- ja lähtöaika.

Asemien kohdalla, joissa juna on jo pysähtynyt, on aseman nimen perässä merkintä. Ohitetuista asemista näytetään todellinen saapumis- ja lähtöaika. Mahdollinen ero aikataulussa ilmoitettuun aikaan näytetään todellisen ajan perässä minuutteina.



Kuva 12. Aikataulutiedot sovelluksessa

4.2.7 Käyttäjäkohtaiset toiminnot

Sovelluksessa on käyttäjäkohtaisia toimintoja, joilla käyttäjä voi vaikuttaa näkemäänsä informaatioon muutenkin, kuin sillä minkä junan valitsee. Kuvassa 13 olevassa näkymässä on mahdollista aikataulutiedoista valita oma määränpääasemansa. Valittu asema tulee näkyviin junatietojen perusnäkymään, junan nopeustietojen vasemmalle puolelle.

Kuva 13. Määränpään valitseminen

Käyttäjän on myös mahdollista tallentaa junavuoro käyttämiinsä juniin, jolloin käyttäjän saapuessa sivustolle, tarkistaa sovellus automaattisesti onko jokin käyttäjän tallentamista junista juuri sillä hetkellä liikkeellä ja siirtää

sitten käyttäjän suoraan kyseisen junan sivulle. Jos tallennettuja junavuoroja on liikkeellä useampi, näytetään käyttäjälle lista junista, jolloin käyttäjä voi valita näistä oikean tai siirtyä etusivulle etsimään muuta junaa hakukonetta käyttäen.

Käyttäjän ei tarvitse rekisteröityä palveluun käyttääkseen henkilökohtaisia toimintoja, sillä nämä toteutetaan evästeiden avulla. Tämä on työlle asetetuiden tavoitteiden mukaista, sillä nopea pääsy omien junavuorojen tietoihin on tarpeellinen erityisesti mobiilikäytössä ja on todennäköistä, että käyttäjä käyttää palvelua toistuvasti samalla mobiililaitteella.

5 POHDINTA

Työn lopputulos on tavoitteiden mukainen palvelu, jolla junaliikenteen matkustajan on mahdollista saada oman junavuoronsa yksityiskohtaiset ja ajantasaiset kulkutiedot nopeasti katseltavaksi. Sovelluksesta tehtiin responsiivinen, niin mobiili, kuin työpöytäkäyttöönkin soveltuva kokonaisuus.

Web-käyttöliittymän toteutuksessa onnistuttiin luomaan kevyt ja suorituskykyinen sivusto, joka pyörii tehokkaasti kaikilla käyttöalustoilla. Sivusto sai Googlen Page Insights testissä nopeudesta pisteet 98/100 mobiilikäyttöliittymässä ja 100/100 työpöytäversiossa. (Google, n.d.)

Samainen käyttöliittymä skaalautuu responsiivisesti niin kapealle älypuhelimen näytölle kuin leveälle tietokoneen näytölle. Samat toiminnot ja tiedot ovat näkyvissä kaikilla laitteilla ja toteutus on täysin alustariippumaton. Sovellusta voidaan siis käyttää millä tahansa modernilla mobiililaitteella tai tietokoneella, selaimesta tai käyttöjärjestelmästä riippumatta.

Sovelluksessa on paljon mahdollisuuksia kehittämiselle tulevaisuudessa ja sovelluksen testikäytössä on noussut esille mahdollisia lisäyksiä, joita palveluun olisi hyvä tulevaisuudessa lisätä. Muutamia sovelluksen toimintaan liittyviä puutteita onkin jo korjattu ulkopuolisten testaajien palautteen perusteella.

Sovellus käyttää tällä hetkellä vain pientä osaa rajapinnasta saatavista junien tiedoista, joten saatavilla olevaa dataa hyödyntäen sovellukseen on mahdollista lisätä vielä paljon hyödyllisiä ominaisuuksia. Uusia ominaisuuksia lisättäessä on kuitenkin otettava huomioon, että käyttöliittymän tulee pysyä helppokäyttöisenä ja kaiken tärkeimmän datan tulee mahtua mobiililaitteen näytölle ilman tarvetta vierittämiselle.

Työn tavoitteet ja tarkoitus nousivat tekijän omista kokemuksista päivittäisenä junamatkustajana ja siitä muodostuneesta tarpeesta toteutetun kaltaiselle sovellukselle, joten on aiheellista pohtia, onko sovellus vastannut

näihin tarpeisiin. Kehitetty sovellus on tekijällä itsellään päivittäisessä käytössä ja sitä kokeilleet ihmiset ovat myös kokeneet sen hyödylliseksi.

Onnistuneet ohjelmistokehitysprojektit ovatkin usein lähtöisin tekijänsä henkilökohtaisesta tarpeesta ja ratkaisevat tekijää itseään koskettavan ongelman. Tämän työn tuloksena syntynyt sovellus on asennettu osoitteeseen https://junassa.petrimalja.com/ ja sen lähdekoodi on tuotu saataville osoitteessa https://github.com/olasirtep/junassa/.

LÄHDELUETTELO

Amazon. (n.d.). *AWS Free Tier*. Haettu 7.12.2018 osoitteesta https://aws.amazon.com/free/

Amazon Web Services. (n.d.). *Amazon EC2 Security Groups for Linux Instances*. Haettu 7.12.2018 osoitteesta https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html

Arsenault, C. (2017). *The Pros and Cons of 8 Popular Databases*. Haettu 16.11.2018 osoitteesta https://www.keycdn.com/blog/popular-databases

Avoindata.fi. (2018). *3.4. Tekniset valinnat*. Haettu 16.11.2018 osoitteesta https://www.avoindata.fi/fi/content/34-tekniset-valinnat

Bashah Mat Ali, A., Shakhatreh, A. Y., Abdullah, M. & Alostad, J. (2011). SQL-injection vulnerability scanning tool for automatic creation of. *Procedia Computer Science, 3*, 453-458. doi:https://doi.org/10.1016/j.procs.2010.12.076

Belshe, M., Peon, R. & Thomson, M. (2015). *Hypertext Transfer Protocol Version 2 (HTTP/2)*. Haettu 3.12.2018 osoitteesta HTTP Working Group: https://httpwg.org/specs/rfc7540.html

Brown, Z. (2018). *A Git Origin Story*. Haettu 7.12.2018 osoitteesta Linux Journal: https://www.linuxjournal.com/content/git-origin-story

Byrne, M. (2016). *The Rise and Fall of the Java Applet: Creative Coding's Awkward Little Square.* Haettu 15.11.2018 osoitteesta https://motherboard.vice.com/en_us/article/8q8n3k/a-brief-history-of-the-java-applet

Comodo CA. (n.d.). *What is HTTPS?* Haettu 3.12.2018 osoitteesta https://www.instantssl.com/ssl-certificate-products/https.html

dnsimple. (n.d.). *A Records*. Noudettu 19.11.2018 osoitteesta https://support.dnsimple.com/articles/a-record/

Ellingwood, J. (2014). *Understanding the SSH Encryption and Connection Process*. Haettu 7.12.2018 osoitteesta https://www.digitalocean.com/community/tutorials/understanding-the-ssh-encryption-and-connection-process

ENISA. (2016). *The Dangers of Trusting User Input*. Haettu 7.12.2018 osoitteesta https://www.enisa.europa.eu/publications/info-notes/the-dangers-of-trusting-user-input

Evans, B. (2017). *The Top 5 Cloud-Computing Vendors*. Haettu 7.12.2018 osoitteesta Forbes: https://www.forbes.com/sites/bobevans1/2017/11/07/the-top-5-cloud-computing-vendors-1-microsoft-2-amazon-3-ibm-4-salesforce-5-sap/#

Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P. & Berners-Lee, T. (1999). *RFC2616*. Haettu 3.12.2018 osoitteesta https://www.w3.org/Protocols/rfc2616/rfc2616.html

Garbade, M. (2017). *Could Python's Popularity Outperform JavaScript in the Next Five Years?* Haettu 7.12.2018 osoitteesta https://hackernoon.com/could-pythons-popularity-outperform-javascript-in-the-next-five-years-abed4e307224

Gebhart, G. & Schoen, S. (2018). *Let's Encrypt Hits 50 Million Active Certificates and Counting*. Haettu 7.12.2018 osoitteesta https://www.eff.org/deeplinks/2018/02/lets-encrypt-hits-50-million-active-certificates-and-counting

Geerling, J. (2014). *A brief history of SSH and remote access*. Haettu 7.12.2018 osoitteesta https://www.jeffgeerling.com/blog/brief-history-ssh-and-remote-access

Geerling, J. (2016). *Should I Disable PHP Warnings and Notices?* Haettu 7.12.2018 osoitteesta https://dev.acquia.com/blog/-should-i-disable-php-warnings-and-notices/30/06/2016/15786

Google. (n.d.). *PageSpeed Insights*. Noudettu 7.12.2018 osoitteesta https://developers.google.com/speed/pagespeed/insights/?url=junassa.petrimalja.com

Helsinki Region Infoshare. (2017). *Mitä on avoin data?* Haettu 7.12.2018 osoitteesta https://hri.fi/fi/ohjeet/mita-on-avoin-data/

Hovi, A., Huotari, J. & Lahdenmäki, T. (2003). *Tietokantojen suunnittelu & indeksointi.* Jyväskylä: Docendo Finland Oy.

Kaur, D. & Kaur, P. (2017). Cross-Site-Scripting Attacks and Their Prevention during Development. *International Journal of Engineering Development and Research, 5*(3), 153-159.

Liikennevirasto. (2018). *Digitraffic*. Haettu 7.12.2018 osoitteesta https://www.liikennevirasto.fi/avoindata/digitraffic#.XApYWSBoSUk

McMillan, R. (2012). *Lord of the Files: How GitHub Tamed Free Software (And More)*. Haettu 7.12.2018 osoitteesta https://www.wired.com/2012/02/github-2/

McQuaid, M. (2014). *Why Use Version Control?* Haettu 7.12.2018 osoitteesta https://mikemcquaid.com/2014/01/18/why-use-version-control/

Nguyen, T. (2018). *On-Demand vs Reserved vs Spot AWS EC2 Pricing Comparison*. Haettu 7.12.2018 osoitteesta https://blog.boltops.com/2018/07/13/on-demand-vs-reserved-vs-spot-aws-ec2-pricing-comparison

Nicholson, J. (2016). *Self-Signed SSL Certificate Warning*. Haettu 7.12.2018 osoitteesta https://www.inmotionhosting.com/support/website/ssl/self-signed-ssl-certificate-warning

Panda Security. (2017). *Default Settings, and Why the Initial Configuration is not the Most Secure*. Haettu 7.12.2018 osoitteesta https://www.pandasecurity.com/mediacenter/security/default-settings-initial-configuration-not-secure/

Pillai, P. (2004). *Introduction to Static and Dynamic Typing*. Haettu 8.12.2018 osoitteesta https://www.sitepoint.com/typing-versus-dynamic-typing/

Porup, J. (2018). *SQL injection explained: How these attacks work and how to prevent them*. Haettu 7.12.2018 osoitteesta https://www.csoonline.com/article/3257429/application-security/what-is-sql-injection-this-oldie-but-goodie-can-make-your-web-applications-hurt.html

Ranger, S. (2018). *What is cloud computing? Everything you need to know about the cloud, explained*. Haettu 7.12.2018 osoitteesta https://www.zdnet.com/article/what-is-cloud-computing-everything-you-need-to-know-from-public-and-private-cloud-to-software-as-a/

SSH Communications Security. (2017). *SSH Tunnel*. Haettu 7.12.2018 osoitteesta https://www.ssh.com/ssh/tunneling/

SSH Communications Security. (2018). *SSH Key*. Haettu 7.12.2018 osoitteesta https://www.ssh.com/ssh/key/

Strauss, V. (2017). *What is an SSL certificate and why do you need one?* Haettu 3.12.2018 osoitteesta https://www.voog.com/blog/what-is-an-ssl-certificate-and-why-do-you-need-one

The PHP Group. (n.d.). *History of PHP*. Haettu 7.12.2018 osoitteesta http://php.net/manual/en/history.php.php

W3Schools. (n.d.). *Javascript JSON*. Haettu 7.12.2018 osoitteesta https://www.w3schools.com/js/js_json.asp

W3Techs. (2018). *Usage of client-side programming languages for websites*. Haettu 15.11.2018 osoitteesta https://w3techs.com/technologies/overview/client_side_language/all

W3Techs. (2018). *Usage of JavaScript libraries for websites*. Haettu 19.11.2018 osoitteesta https://w3techs.com/technologies/overview/javascript_library/all

W3Techs. (2018). *Usage of server-side programming languages for websites*. Haettu 15.11.2018 osoitteesta https://w3techs.com/technologies/overview/programming_language/all

W3Techs. (2018). *Usage of SSL certificate authorities for websites*. Haettu 16.11.2018 osoitteesta https://w3techs.com/technologies/overview/ssl_certificate/all

W3Techs. (2018). *Usage of web servers for websites*. Haettu 13.11.2018 osoitteesta https://w3techs.com/technologies/overview/web_server/all

W3Techs. (2018). *Usage statistics and market share of Linux for websites*. Haettu 13.11.2018 osoitteesta https://w3techs.com/technologies/details/os-linux/all/all

Walsh, D. (2007). *6 Reasons To Use JavaScript Libraries & Frameworks*. Haettu 7.12.2018 osoitteesta https://davidwalsh.name/6-reasons-to-use-javascript-libraries-frameworks

Wootton, B. (2017). *Who's Using Amazon Web Services?* Haettu 7.12.2018 osoitteesta https://www.contino.io/insights/whos-using-aws