Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Mikael Kotkavuori

Henkilöliikenteen mittaridatasta muodostettava yhteinen tietomalli

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tutkinto-ohjelma

Insinöörityö

14.5.2020

Tekijä Otsikko Sivumäärä Aika	Mikael Kotkavuori Henkilöliikenteen mittaridatasta muodostettava yhteinen tietomalli 32 sivua 14.5.2020
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikan tutkinto-ohjelma
Ammatillinen pääaine	Ohjelmistotuotanto
Ohjaajat	Lehtori Jorma Räty

Insinöörityön tarkoituksena oli kehittää henkilökuljetuksen mittaridatasta tietomalli ja yhdistelyjärjestelmä asiakasyrityksen palkanlaskentaa varten.

Insinöörityön päätavoitteiksi määriteltiin mittaridataa yhdistelevän tietomallin muodostaminen ja yhdistelystä vastaavan järjestelmän kehittäminen. Lisäksi tavoitteiksi määritettiin kansainvälisten standardien ja muiden tiedon tuontitapojen selvittäminen.

Työ oli osa suurempaa projektia, johon liittyi myös palkanlaskentajärjestelmän kehittäminen.

Työ toteutettiin käyttäen Microsoftin .Net-ympäristöä ja Azure-pilvipalveluita.

Yhdistelyjärjestelmä ja tietomalli saatiin valmiiksi ja kaikkiin päätavoitteisiin päästiin. Tiedonhankinnassa ilmenneiden vaikeuksien vuoksi kansainvälisten standardien selvittäminen oli suppeaa. Erilaisten tiedon tuontitapojen selvittäminen siirrettiin insinöörityön alkuvaiheessa osaksi käyttöjärjestelmän kehitysprojektia.

Avainsanat	Tietomalli, .Net, Azure, C#

Author	Mikael Kotkavuori
Title	Unified Data Model from Passenger Traffic Meter Data
Number of Pages	32 pages
Date	14 May 2020
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Software Development
Instructor	Jorma Räty, Senior Lecturer

The purpose of the Thesis was to develop a unified data model and combination system for calculating the client's payroll from passenger transportation meter data.

The main goals of the thesis were defined as forming the combining data model and the system responsible for the combination of data. Additional goals were set to determine international meter data standards and other means of delivering information into the system.

The thesis was a part of a larger project, which involved the development of a payroll system.

The system and data model were implemented with Microsoft .Net environment and Azure cloud services.

The combination system and data model were finished, and all the main goals were met. Due to difficulties in procuring information, the goal of determining international standards was only partially met. The goal of discovering other means of information delivery was transferred to be a part of the payroll system development project at the early stages of the project.

Keywords	Data model, .Net, Azure, C#

Sisällys

Lyhenteet

CRUD	Create, read, update, delete. Tietokantaan tehtävien muutosten perustoiminnot.
DTO	Data Transfer Object. Tiedonsiirto-olio.
GPS	Global Positioning System. Maailmanlaajuinen paikallistamisjärjestelmä.
IIS	Internet Information Services. Microsoftin kehittämä palvelinohjelmistokokonaisuus, jota käytetään Windows-pohjaisissa palvelimissa.
JSON	JavaScript object notation. Avoimen standardin tiedostomuoto.
KELA	Kansaneläkelaitos. Eduskunnan valvonnassa oleva itsenäinen sosiaaliturvalaitos, jolla on oma hallinto ja talous.
MPK	Matkapalvelukeskus. Yleistermi vammaispalvelulain ja sosiaalihuoltolain mukaisten kuljetuspalveluiden välitystä järjestävästä alueellisesta toimijasta.
ORM	Object-relational mapping. Oliomallin mukaisen esityksen kuvaus relaatiomallin mukaiseksi esitykseksi.
XML	Extensible Markup Language. Merkintäkieli.

.

1 Johdanto

Javatrans Oy tilasi keväällä 2019 kuljettajien palkanlaskentasovelluksen osaksi Movit-palveluaan. Javatrans Oy on vuonna 2002 perustettu henkilökuljetuksia tarjoava perheyritys, jolla on 110 ajoneuvon kalusto.

Intoit Oy on vuonna 2011 perustettu ohjelmistokonsultointi- ja ohjelmistokehitysyritys. Yrityksen päätuotteena on henkilökuljetusten hallintaan kehitetty Movit-palvelu.

Movittia käytetään Javatrans Oy:ssä pääasiassa koulukuljetusten järjestämiseen. Javatrans ajaa koulukuljetusten lisäksi myös matkapalvelukeskuksen kuljetuksia sekä harjoittaa taksitoimintaa.

Palkanlaskentaa varten eri kuljetustyypeistä saatava tieto kerätään eri järjestelmiin useiden eri mittareiden kautta. Mittareista saatavan tiedon yhdisteleminen käsin palkanlaskentaa varten koettiin aikaavieväksi ja tehottomaksi prosessiksi.

Insinöörityö suoritettiin osana palkanlaskentajärjestelmän kehitysprojektia. Insinöörityön tavoitteena oli kehittää tietomalli ja järjestelmä, joka keräisi ja yhdistäisi taksimittareista ja muista lähteistä saatavan datan yhtenäiseen muotoon.

Päätavoitteiden lisäksi määriteltiin alitavoitteet suunnittelua varten: tietomallin suunnittelun tavoitteina oli selvittää Suomen lisäksi kansainvälisiä standardeja, selvittää Javatrans Oy:n käytössä olevista mittareista saatava tieto sekä huomioida mahdolliset muut tiedonlähteet. Yhdistelyjärjestelmän tavoitteina oli luoda helposti muokattava ja ylläpidettävä järjestelmä, johon voidaan tulevaisuudessa liittää muita mittarirajapintoja.

Kehitysprojektiin kuului myös käyttöliittymän suunnittelu ja toteutus, jotka mahdollistaisivat kerätyn tiedon käyttämisen palkanlaskennan perusteena. Vaikka käyttöliittymän toteutus ei ollut osa opinnäytetyötä, se ohjasi työn vaiheita huomattavasti.

2 Lähtökohdat

Mittaridatan yhdistelyssä oli toteutuksen kannalta kolme erilaista tiedonlähdettä, joissa tietoa on tallennettu eri muotoihin: Intoit Oy:n Movit, Trippi Oy:n Mitax -mittarirajapinta sekä Semel Oy:n Vuoronet -rajapinta.

2.1 Movit

Movit on Intoit Oy:n päätuote, jota on kehitetty vuodesta 2014. Movit kuuluu Intoit Oy:n henkilökuljetusten hallintaan kehitettyyn Movit-tuoteperheeseen. Tuotteisiin kuuluvat Movitin lisäksi tilausten tekoon ja hallintaan suunniteltu Timit-järjestelmä sekä kuljetettavien asiakkaiden käyttöön suunnattu Rideit -sovellus. [1; 2; 3.]

Movitin avulla kuljetusyrityksen ajojärjestelijät suunnittelevat ja välittävät reitit kuljettajille. Kuljettajilla on matkapuhelimessaan Movit-sovellus, jonka kautta he vastaanottavat ja kuittaavat saamansa kuljetukset. Movitiin tallentuu kuljetuksista tietoa: kuljetuksen suorittanut kuljettaja, kuljetukseen käytetty ajoneuvo, kuljetukseen liittyvät maksu ja aikatiedot sekä kuljetuksen GPS-tiedot, jotka välittyvät kuljettajan Movit-sovelluksesta.

Movitin backend on rakennettu Microsoftin Entity Framework -ohjelmistokehyksen avulla, ja käyttää Microsoftin Azure pilvipalveluita. Movitista ei haettu tietoa, mutta yhdisteltävä mittaritieto oli sovitettava Movittiin, joten olemassaolevat tietomallit oli huomioitava projektissa. Kehitetty tietomalli yhdistettiin Movitissa jo olemassa oleviin vuorotietohin.

2.2 Mitax

Mitax on Trippi Oy:n valmistama taksamittari. Trippi oy on vuonna 1983 perustettu mittauslaitteistojen suunnitteluun ja valmistukseen erikoistunut yritys. Mitax-mittareita käytetään Javatrans oy:llä pääsääntöisesti Kelan kuljetuksiin. [4.]

Kuljettajat merkitsevät mittaritiedot Mobirouter-välityspalvelun tietojen perusteella Mitaxiin.

Mitaxin tarjoamasta rajapinnasta on mahdollista saada kahdella eri tavalla järjesteltyä tietoa: kljetuksen maksutiedot sekä mittariraportti, joka tekee yhteenvedon mittarilla suoritetuista kuljetuksista.

Mitaxista saatava tieto on XML-muodossa. XML on lyhenne sanoista Extensible Markup Language. XML on merkintäkieli, joka on lukukelpoinen ihmiselle ja koneelle. XML on määritelty World Wide Web Consortiumin toimesta XML 1.0 -spesifikaatiossa vuonna 1998. XML:n suunnittelutavoitteet painottavat yksinkertaisuutta, yleisyyttä ja käytettävyyttä internetissä. XML on suunniteltu dokumentteja varten, mutta sitä käytetään yleisesti myös tietorakenteiden, kuten olioiden välittämiseen. [5.]

```
<Maksutiedot xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="maksu_schema_v10_ext.xsd">
  <Eratunnus>ET11562334KST</Eratunnus>
  <Aikaleima>2008-11-14T15:30:01.123456</Aikaleima>
  <Tekija>Matti Meikäläinen</Tekija>
  <Ymparisto>TESTI</Ymparisto>
  <Varatila>0</Varatila>
  <Matka>
    <TransactionId>123</ TransactionId>
    <VuoroNumero>123</VuoroNumero>
    <KuljettajaNumero>1234</KuljettajaNumero>
    <AutoNumero>123</AutoNumero>
    <Alkupvm>2009-11-28</Alkupvm>
    <Matkapvm>2009-11-28</Matkapvm>
    <OmavastuuPeritty>X</OmavastuuPeritty>
    <LaskutusperusteitaMuokattu>K</LaskutusperusteitaMuokattu>
    <Varatila>0</Varatila>
    <Taksimatka>
      <Tilaustunnus>KST00000000012345678</Tilaustunnus>
      <Alkuaika>1145</Alkuaika>
      <Loppuaika>1230</Loppuaika>
      <Perusmaksu>500</Perusmaksu>
      <Taksa>
        <Taksatyyppi>S</Taksatyyppi>
        <Taksaluokka>1</Taksaluokka>
        <Taksanimi>KELA1</Taksanimi>
        <Kilometrihinta>130</Kilometrihinta>
        <Tuntihinta>3700</Tuntihinta>
        <Metri>32000</Metri>
        <Aika>0015</Aika>
        <Hinta>41600</Hinta>
      </Taksa>
      <Taksa>
        <Taksatyyppi>D</Taksatyyppi>
        <Taksaluokka>2</Taksaluokka>
        <Kilometrihinta>130</Kilometrihinta>
        <Tuntihinta>3700</Tuntihinta>
        <Metri>32000</Metri>
        <Aika>0015</Aika>
        <Hinta>41600</Hinta>
      </Taksa>
      <Taksa>
        <Taksatyyppi>F</Taksatyyppi>
```

```
      <Taksaluokka>3</Taksaluokka>
      <Kilometrihinta>130</Kilometrihinta>
      <Tuntihinta>3700</Tuntihinta>
      <Metri>32000</Metri>
      <Aika>0015</Aika>
      <Hinta>41600</Hinta>
    </Taksa>
    <Lisa>
      <Laji>A</Laji>
      <Maksu>1130</Maksu>
    </Lisa>
    <Lisa>
      <Laji>L</Laji>
      <Maksu>1230</Maksu>
      <Nimi>Lentokenttälisä</Nimi>
    </Lisa>
    <Lisa>
      <Laji>P</Laji>
      <Maksu>1330</Maksu>
    </Lisa>
    <Odotus>
      <Odotuslaji>O</Odotuslaji>
      <Tuntihinta>3700</Tuntihinta>
      <Odotusaika>0008</Odotusaika>
      <Maksu>493</Maksu>
    </Odotus>
    <Odotus>
      <Odotuslaji>H</Odotuslaji>
      <Tuntihinta>3700</Tuntihinta>
      <Odotusaika>0008</Odotusaika>
      <Maksu>499</Maksu>
    </Odotus>
    <Kokonaismetri>42000</Kokonaismetri>
    <Maksutapa>JM</Maksutapa>
    <Osuus>10000</Osuus>
    <Kustannus>7783</Kustannus>
    <Omavastuu>925</Omavastuu>
    <Korvaus>6858</Korvaus>
    <ReittiHinta>0</ReittiHinta>
    <Muutos>0</Muutos>
    <Muutoshinta>0</Muutoshinta>
    <Tippi>0</Tippi>
    <Maksupaatetunnus>000125000321</Maksupaatetunnus>
    <PalvelutuottajaYTunnus>11262334</PalvelutuottajaYTunnus>
    <PalvelutuottajaNimi>Kainuun Taksi Oy</PalvelutuottajaNimi>
    <Kuittinumero>12344</Kuittinumero>
    <TaksamittarinSarjanumero>16001234</TaksamittarinSarjanumero>
    <KorttiNumero>1234123412341234</KorttiNumero>
    <KorttiVoimassa>0319</KorttiVoimassa>
    <ALV>1000</ALV>
    <ALVVero>624</ALVVero>
    <ALVNetto>6234</ALVNetto>
    <Varatila>0</Varatila>
  </Taksimatka>
 </Matka>
</Maksutiedot>
```

Esimerkkikoodi 1. Esimerkki Mitaxin Transaction XML-oliosta.

Mitaxin rajapinnasta saatava tieto on jaoteltu kahteen erilliseen olioon: Mittariraportteihin (Reports) ja maksutietoihin (Transactions).

Mitaxin rajapinnasta saatava mittariraportti pitää sisällään tiedot raportin tunnisteesta, raportin tyypistä, joka rajapintakyselyssä on aina vuoro, raporttinumerosta, taksamittarin numerosta, autonumerosta, kuljettajanumerosta, alku- ja loppuajasta, maksutapahtumista sekä raa-asta mittaridatasta, joka on otsikoimaton, pilkuilla erotettu merkkijono.

Maksutietojen sisältämät mahdolliset tietokentät on kuvattu esimerkkikoodi 1:ssä.

2.3 Semel

Vuonna 1971 perustettu Semel kehittää ja tuottaa kuljetusalalle tietojärjestelmiä ja laitteita. Pääpainoalue on taksiliiketoiminnan tilaus-, maksu- ja seurantajärjestelmissä. Semel on Pohjoismaiden suurin taksien tietojärjestelmien toimittaja. Semel on kuulunut KGK-konserniin 1990-luvulta lähtien. [6.]

Semel valmistaa mittarilaitteet sekä tarjoaa rajapinnan ja ohjelmiston mittareista saadun tiedon seuraamiseen. Semelin rajapinnasta tietoa voidaan välittää JSON- ja XML-muodossa. Projektissa tietomuodoksi valittiin JSON, sillä tekijän mielestä se on XML:ää helpolukuisempi tietomuoto.[7]

JSON on lyhenne sanoista JavaScript Object Notation. JSON on avoimen standardin tiedostomuoto, joka käyttää ihmiselle lukukelpoista tekstiä tiedon tallentamiseen ja välittämiseen oliomuodossa. Tieto koostuu avain-arvo- ja taulukkotietotyypeistä. JSON määritettiin 2000-luvun alussa Douglas Crockfordin toimesta. JSON standardisoitiin vuonna 2013 ECMA-404-standardissa. Vuonna 2017 JSON määritettiin nykyisen internetstandardin STD-90 avulla, ja se on edelleen yhteensopiva ECMA-404:n kanssa. JSON on erittäin yleinen tiedostomuoto, jota käytetään hyvin laaja-alaisesti erilaisissa toteutuksissa. [8; 9.]

```
[
    {
        "Company": "Demo",
        "ShiftId": "1685",
        "ServiceNumber": "Demo",
        "DriverId": "38985",
```

```
        "MDXSCarNumber": "Demo",
        "ReceiptCarNumber": "Demo",
        "CarKilometersOnStart": "160041",
        "CarKilometersOnEnd": "160326",
        "ManualWorkTime": "5538094",
        "StartTime": "2019-10-21T07:09:10+03:00",
        "EndTime": "2019-10-21T22:59:38+03:00",
        "TotalMetersAtStart": "458831956",
        "TotalMetersAtEnd": "459128990",
        "Records": [
            {
                "Type": "Trip",
                "StartTime": "2019-10-21T08:01:04+03:00",
                "EndTime": "2019-10-21T08:20:37+03:00",
                "MetersDriven": 2900,
                "Price": 2015,
                "ReceiptNumber": "2645401-2645401",
                "Events": [
                    {
                        "PaymentType": "Company invoicing",
                        "Type": "51E",
                        "PaymentTime": "2019-10-21T08:20:37",
                        "Price": 2015,
                        "CurrencySymbol": null,
                        "ReceiptNumber": "2645401",
                        "MetersDriven": 2900,
                        "CustomerCode": "7002",
                        "PassengerId": "7002",
                        "CostCenterId": "7002",
                        "ManualCard": "0",
                        "CardNumber": "",
                        "TransportId": "",
                        "ExternalBookingId": "",
                        "CustomerTripId": "",
                        "BookingId": "10232219",
                        "ManualCustomerTripId": "",
                        "LicencePlateNumber": "Demo",
                        "Extras": [],
                        "VATs": [
                            {
                                "VatPercent": 1000,
                                "VatNetto": 1832,
                                "VatAmount": 183,
                                "VatBrutto": 2015
                            }
                        ],
                        "Discount": 175,
                        "Waste": 0,
                        "Excess": 0,
                        "Tips": 0
                    }
                ]
            },
        ]
    }
]
```


Esimerkkikoodi 2. Esimerkki Vuoronet -rajapinnasta saatavasta vuorotieto JSON-tiedostosta.

Semelistä saatava tieto on järjestelty mittarilaitteen tunnistenumeron perusteella. Tieto on järjestely mittarilla ajettavan vuoron mukaan.

Vuoro-olion tiedot on kuvattu esimerkkikoodi 2:ssa, vuoro-olio sisältää listauksen vuoron aikana ajetuista matkoista, sekä matkoille kirjatuista tapahtumista, jotka yleistävästä nimestä poiketen ovat aina maksutapahtumia.

3 Määrittely

Työn määrittely aloitettiin asiakastapaamisella, jossa asiakas esitteli nykyistä palkanlaskentaprosessia ja tarvittavia tietoja.

Määrittelyssä oli huomioitava asiakkaan palkanlaskentaan vaikuttavat tiedot. Nämä asiakasvaatimukset muodostivat tietomallin vähimmäisvaatimukset. Palkanlaskentaa varten tarvittiin tietoa muun muassa kuljettajan työajoista, ajetuista kilometreistä, kuljetusten tyypeistä (koulukuljetus, MPK matka, taksimatka) sekä kuljetuksista maksettu määrä.

Yhdistelyjärjestelmä määriteltiin käytettävien tekniikoiden perusteella, ja tarkempi kuva järjestelmästä alkoi muodostua vasta toteutusvaiheessa, kun tietomallin suunnittelu oli saatu päätökseen ja rajapintojen toiminta oli selvitetty.

3.1 Tekniikat

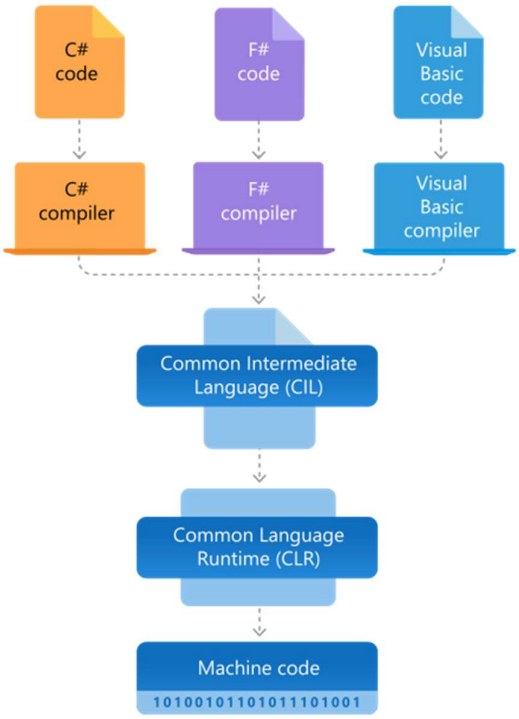
Käytettävät tekniikat määriteltiin Movitin mukaisiksi, jotta kehitys ja ylläpito olisi mahdollisimman yhdenmukaista ja tehokasta.

Tietokantana käytettäisiin jo olemassa olevaa, Azuren Azure SQL -tietokantaa. Azure SQL on yleiskäyttöinen relaatiotietokanta, joka toimitetaan hallinnoituna palveluna. Azure SQL -tietokantaan voi tallentaa relaatiomallin mukaisen datan lisäksi myös muita tietomalleja, kuten graafit, JSON:n, spatialin ja XML:n. [10]

Movitissa tieto on tallennettu relaatiomallin mukaisesti, joten tietokantamalliksi määritettiin relaatiomalli.

3.1.1 .Net Framework ja .Net Core

.Net Framework ja .Net Core ovat Microsoftin kehittämiä ohjelmistokehyksiä ja ohjelmistokomponenttikirjastoja. Nämä kehykset suorittavat suurimman osan ohjelmistojen vaatimista toiminnoista mahdollistaen ohjelmoijan keskittymisen olennaiseen, eli niin sanottuun business-logiikkaan. Kuvassa 1 esitellään .Net Frameworkin arkkitehtuuri.



Kuva 1. .Net Framework -arkkitehtuuri, lähde: https://dotnet.microsoft.com/static/images/illustrations/swimlane-architecture-framework.svg?v=ZuTW7j9pS1oiuMqx3E-Xvb9OEM_8ajDEcHbecyjRtLA

.Net Frameworkin suurimmat komponentit ovat Common Language Runtime (CLR) ja luokkakirjasto.

CLR on sovellusten käynnissä olosta vastaava toteutusmoottori. CLR tarjoaa palveluita, joita ovat säikeiden hallinnointi, roskan keräys, tyyppi turvallisuus ja poikkeusten hallinta.

Luokkakirjasto tarjoaa kokoelman rajapintoja ja tyyppejä yleisiä toiminnallisuuksia varten. Rajapintakokoelmissa on toteutuksia tiedostojen lukua- ja kirjoitusta varten, tietokantoihin yhdistämistä varten sekä esimerkiksi piirtämistä varten. Luokkakirjasto tarjoaa tyyppejä muun muassa merkkijonoja, päivämääriä ja numeroita varten.

.Net Framework on .Net -ympäristön alkuperäinen toteutus, joka toimi vain Windows -ympäristöissä, koska sen toiminnallisuus oli tiukasti sidottu Windowsin natiivikirjastoihin. Myöhemmät laajennukset, kuten ASP.NET, mahdollistivat IIS-pohjaisten websivujen kehityksen .NET -kielillä. Nykyään .Net Framework tukee muun muassa websivujen, palveluiden ja työpöytäsovellusten pyörittämistä Windows -ympäristöissä.

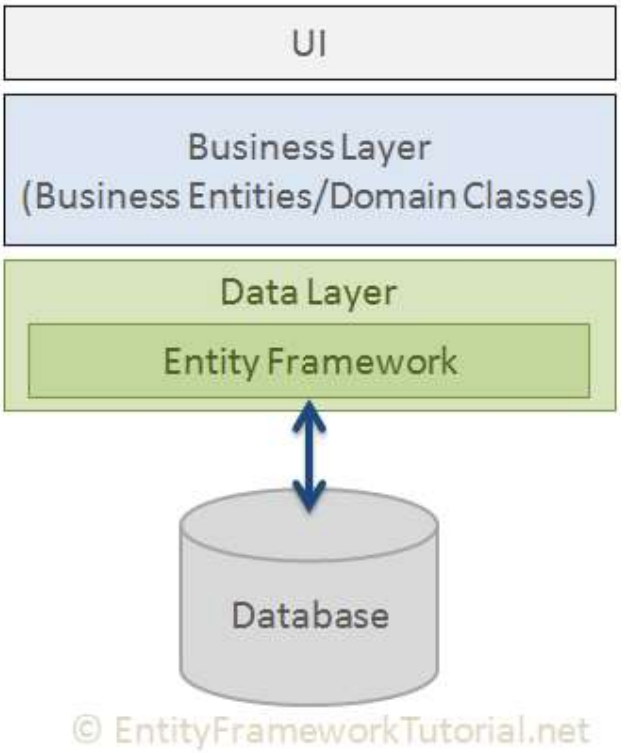
.Net Core on .Net Frameworkin seuraaja. .Net Core on alustariippumaton ja moderni avoimen lähdekoodin ohjelmistokehys, joka on edeltäjäänsä huomattavasti nopeampi ja pienikokoisempi modulaarisuutensa (package-based framework) vuoksi.

.Net Frameworkin tapaan .Net Core tarjoaa useita hyödyllisiä rajapintoja ohjelmistokehitykseen. Insinöörityössä osa kehityksestä tehtiin käyttäen .Net Frameworkkiä ja osa .Net Corella. [11; 12; 13; 14; 15.]

3.1.2 Entity Framework

Entity Framework on Microsoftin ylläpitämä ORM-viitekehys. Entity Frameworkin ensimmäinen versio on julkaistu vuonna 2008. Viimeisin Entity Frameworkin pääversio, versio 6, on julkaistu vuonna 2013. [16.]

ORM-viitekehyksen tavoitteena on mahdollistaa kahden yhteensopimattoman tietotyypin välinen kartoitus. Relaatiotietokannassa tieto on tallennettu relaatiomuodossa (taulukko). Ohjelmistoissa tietoa käytetään oliomuodossa. ORM-viitekehys luo niin kutsutun virtuaalisen olio tietokannan, jota voidaan käyttää ohjelmoinnissa.



Kuva 2. Entity Frameworkin arkkitehtuuri, lähde: https://www.entityframeworktutorial.net/what-is-entityframework.aspx

Entity Framework tarjoaa normaalien CRUD-toimintojen lisäksi muun muassa muutosten seurannan, joka seuraa olioihin tapahtuneita muutoksia myöhempää tallennusta varten.

Tietokantaan tehtävät rakenteelliset muutokset (taulujen ja sarakkeiden muutokset) ovat mahdollisia migraatioiden avulla. Migraatiot luovat alkuperäisen tietokannan, joka toimii Entity Framework -mallien avulla, tuottaa migraatiot, jotka pitävät lukua tietokantaan tehdyistä muutoksista, sekä pitävät tietokannan ajan tasalla tehtyjen muutosten suhteen.

Projektissa tietokantaan lisättiin kaksi täysin uutta taulua, reitit ja maksutapahtumat. Lisäksi jo olemassaoleva Movitin omaa vuorotietoa sisältävä taulu nimettiin uudelleen ja siihen lisättiin mittaridatan yhdistelemiseen tarvittavat sarakkeet migraatioiden avulla. Migraatiot mahdollistivat muutosten tekemisen turvallisesti ja tehokkaasti ilman, että tuotantotietokantoja olisi tarvinnut päivittää suorilla SQL-kyselyillä.

3.1.3 RESTful rajapinta

Representational State Transfer, REST voidaan nähdä arkkitehtuurisena suunnittelumallina, tai ohjelmistoarkkitehtuurin tyylinä, joka määrittelee kokoelman rajoitteita, jotka on huomioitava web-pohjaisen palvelun luonnissa.

Roy Fielding määritteli REST-rajapinnan väitöskirjassaan “Architectural Styles and the Design of Network-based Software Architectures” vuonna 2000.

RESTful rajapinnan rajoitteita on kuusi: Asiakas-palvelinarkkitehtuuri eriyttää käyttöliittymässä näytettävän tiedon tietokantaan tallennetusta tiedosta. Tilattomuus, eli palvelin ei säilytä asiakkaan tilaa kyselyiden välillä. Välimuistiin tallennettavuus eli asiakas ja välittäjät voivat säilyttää palvelimen vastaukset välimuistissaan. Kerrostettu järjestelmä, jossa asiakas ei välttämättä tiedä, onko yhteys suoraan loppupalvelimeen vai johonkin välittäjään. Koodi pyydettäessä on vapaaehtoinen rajoite, jossa palvelimet voivat tilapäisesti laajentaa tai muokata asiakkaan toiminnallisuutta palauttamalla toteutettavaa koodia. Yhtenäisen rajapinnan välittävän tiedon muoto on yhdenmukaista ja erillään tietokannan tietomuodosta. [17.]

Projektissa käytettävistä rajapinnoista kaikki toteuttavat RESTful rajapintaa.

3.1.4 Azure

Azure on Microsoftin tarjoama jatkuvasti kehittyvä pilvipalvelu, joka on julkaistu vuonna 2010. Azure tarjoaa monipuolisia palveluita. Sitä voidaan käyttää esimerkiksi virtuaalipalvelinten alustana sekä kehitysalustana. Projektissa käytettiin useaa Azuren tarjoamaa palvelua. [18.]

Azure App Service mahdollistaa websovellusten ja RESTful-rajapintojen kehityksen ja isännöinnin. App Servicen etuina on, ettei infrastruktuuria tarvitse pystyttää tai hallinnoida itse, App Service tarjoaa myös automaattista skaalausta ja erittäin korkeaa saatavuutta. Insinöörityön osalta App Servicessä isännöidään Movit-palvelua ja mittaritietoa keräämiä WebJobeja. App Servicessä isännöityä palvelua voidaan päivittää ja muuttaa suoraan versionhallinnasta. [19.]

Azure DevOps on kokoelma palveluita, jotka pyrkivät helpottamaan ja mahdollistamaan kehitystyön suunnittelua, versionhallintaa ja ohjelmistojen käyttöönottoa. Movitin kehitys ja versionhallinta hoidetaan Azure DevOpsin kautta.

3.2 Tietomalli

Tietomallin tuli koota ja kerätä Semel- ja Mitax-mittareista sekä Movitista saatavista tiedoista yhtenäinen, palkanlaskentaan soveltuva malli. Vaikka mittaridataa olisi saatavilla muistakin lähteistä, nämä olivat tärkeimmät järjestelmät asiakkaan palkanlaskentaa varten.

Tietomallin tuli olla laajennettavissa tiedon suhteen, ja tiedon keruussa käytettyjen menetelmien tuli olla räätälöitävissä mittarikohtaisesti, jotta uusien mittarityyppien lisääminen olisi mahdollisimman helppoa.

Vähimmäisvaatimukset tiedolle on esitetty taulukossa 1.

Taulukko 1. Mittareista saatavan tiedon vähimmäisvaatimukset

Kentän nimi	**Selitys**
Kuljettajan ID	Kuljettajan tunniste, kuljettajanumero
Pvm	Vuoron päivämäärä
Auto	Vuorossa käytetyn ajoneuvon tunniste
Vuoro	Vuoron tunniste
Kela	Kela kuljetusten yhteenlaskettu summa
Alv 0%	Yhteenlaskettu summa ilman arvonlisäveroa
Käteisajot	Vuoron aikana tapahtuneiden käteismaksujen yhteenlaskettu summa
Yhteensä	Vuoron aikaisten maksutapahtumien yhteenlaskettu summa
Reittien määrä	Vuorossa olleiden reittien määrä
Suunniteltu aloitus	Työsuunnittelun määrittämä vuoronaloitus
Suunniteltu lopetus	Työsuunnittelun määrittämä vuoronlopetus
Ensimmäinen kuittaus	Vuoron ensimmäinen mittarikuittaus

Viimeinen kuittaus	Vuoron viimeinen mittarikuittaus

4 Suunnittelu

Projekti toteutettiin ketterän ohjelmistokehityksen mukaisesti. Työtä ohjaa Kanban-taulu. Kanban-menetelmä on kehitetty 1900-luvun puolivälissä Toyotan autojen valmistukseen kehittämä järjestelmä, jonka on tarkoitus vähentää hävikkiä tuottamalla juuri riittävästi juuri oikeaan aikaan. Myöhemmin menetelmä on sovitettu ohjelmistotuotantoon David Andersonin toimesta vuonna 2005 hänen työskennellessään Microsoftilla. Projektin suunnitteluvaiheessa muodostettiin Lean periaatteiden mukaisesti kanban kortteja, jotka jaettiin useampaan tasoon (Feature, Work Item ja Task).

Feature on tässä tapauksessa työn ylin taso, jossa dokumentoitiin työn vaatimusmäärittely. Vaatimuksista muodostetiin työtehtäviä, jotka kirjattiin Work Itemeiksi.

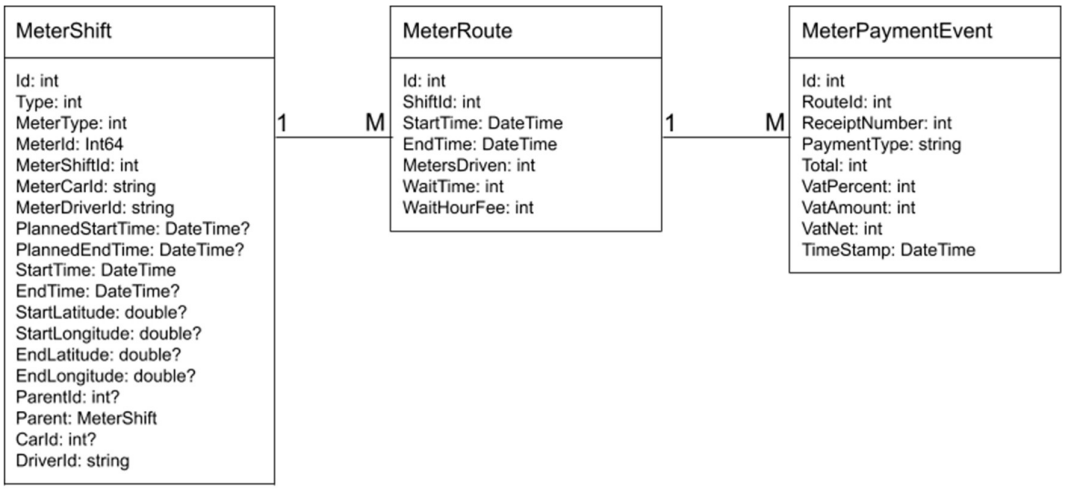
Work itemit ovat työnyksikkö, jonka suorittamisessa tulisi kulua noin kaksi päivää. Work itemit itsessään sisälsivät tehtäviä, Taskeja, jotka työn suorittaja itse määrittää. Tämä mahdollisti työn laajuuden ja ajoituksen arvioimisen.

Tietomallin muodostamiseksi saatavilla oleva tieto jaettiin erilaisiin ryhmittymiin. Näitä vertailtiin asiakasvaatimuksiin, joista muodostettiin ensimmäinen ehdotus mahdollisimman nopeasti. Tämän jälkeen ehdotus esiteltiin yrityksen sisäisesti. Tästä tapaamisesta saadun palautteen perusteella tehtiin muutoksia ehdotukseen. Tätä sykliä toistettiin suunnittelun aikana useaan kertaan. Nopealla suunnittelu-, esittely-, palautesyklillä pyrittiin suunnitteluvaiheessa mahdollisimman pieneen turhan tekemiseen sekä mahdollisimman tarkkoihin suunnitelmiin ennen toteutusta.

4.1 Tietomallin taulut

Movitissa käytetty tieto on tallennettu relaatiotietokantaan, joten uusi tietomalli toteutettiin relaatiomallina.

Mittareista saatava data on aina käyttäjäkohtaista, eli sidottavissa yhteen kuljettajaan.



Kuva 3. Suunnitellut mittaritaulut Movittiin

4.1.1 Vuoro

Vuoro on tietomallin suurin yksikkö. Vuoron muodostuksessa yhdistettiin Movitissa vuoroista jo kerätty tieto uusiin kenttiin. Kuljettajat voivat käyttää useampaa mittaria työpäivän aikana, eli yhdelle kuljettajalla voi muodostua useampi vuoro samalle päivälle. Joissain tapauksissa käyttäjä tai mittarivirheen vuoksi kuljettaja -tietoa ei tallennu mittariin eikä vuoroon. Vuoro voidaan kirjata ilman tietoa kuljettajasta, jotta tieto ei häviäisi, vaan voidaan tarvittaessa korjata käsin myöhemmin.

Vuoron alku- ja loppuaikojen lisäksi vuoroon voidaan lisätä tieto vuoron suunnitellusta aloitus- ja loppuajankohdasta. Näitä tietoja voidaan joissain tapauksissa käyttää palkanlaskennassa.

Vuoro voi sisältää useita kuljetuksia eli reittejä.

4.1.2 Reitti

Reitti on yksittäinen kuljetustapahtuma, joka sisältyy aina vuoroon. Reitti sisältää tiedon vuorosta, johon se liittyy, sekä yksittäisen kuljetuksen tiedot. Reittiin lisättiin projektin

edellyttämät kentät, ajetut metrit, mahdollinen odotusaika ja odotusajalle määritetty kustannus. Jo suunnitteluvaiheessa tiedostettiin, että reitille tulee tulevaisuudessa lisäkenttiä, myöhemmässä, palkanlaskenta käyttöliittymän toteuttavassa projektissa.

4.1.3 Maksutapahtuma

Maksutapahtuma on tietomallin pienin yksikkö, joka sisältää tiedon yksittäisestä maksutapahtumasta. Yhdessä kuljetuksessa, eli reitissä, voi olla useampi maksutapahtuma. Esimerkiksi MPK-kuljetuksessa on MPK:n maksama osuus sekä asiakkaan omavastuuosuus, joista kummastakin muodostuu oma maksutapahtuma.

Maksutapahtuma kuvaa maksutapahtumasta ajankohdan, kuittinumeron, maksutyypin, ALV-prosentin ja osuuden sekä maksetun kokonaissumman.

4.2 Tietomallia käyttävät palvelut

Suunnittelussa otettiin huomioon tietomallin vaikutukset ja tarkoitus hahmottelemalla palvelut, jotka käyttävät tietomallia tulevaisuudessa.

4.2.1 Palkanlaskenta

Tietomalli suunniteltiin, ja mittaridatan yhdisteleminen toteutettiin palkanlaskentajärjestelmän luomiseksi. Käyttöjärjestelmän tulee tarjota taulukkopohjainen verkkopalvelu asiakasyrityksille. Tavoitteena oli, että tietoa voidaan saada ja muokata suoraan palvelussa. Lisäksi tiedosta saataisiin ladattua CSV-tiedosto. Tiedoista voitaisiin myös muodostaa palkkakuitti.

Palkanlaskenta järjestelmän avulla palkkatietoja voi seurata huomattavasti lyhyemmältä aikaväliltä. Näin tietoa voidaan käyttää ohjaamaan uusien kuljettajien kirjaamiskäytäntöjä, toivottuna vaikutuksena kirjaamisvirheiden ja myöhempien selvittelyjen väheneminen. Järjestelmän jatko kehityksenä mietittiin myös mahdollisuutta luoda kuljettajille oma näkymä, josta he voisivat seurata palkkansa muodostumista, ja tarvittaessa ilmoittaa palkanlaskijoille virheistä jo etukäteen, mikä helpottaa myös palkanlaskijoiden työtä.

4.2.2 Tilastointi

Mittareista saatua tietoa voidaan yhdistellä myös liikkeenjohdon tarkoituksiin. Saatua dataa, esimerkiksi ajoneuvojen käyttöaste, kuljetusten keskimääräinen tuotto jne. voidaan esittää visuaalisesti niin sanotun Business Intelligence Dashboardin avulla. Tiedon avulla yrityksen johto saisi kohdennettua tietoa, joka voisi ohjata ja helpottaa tulevien päätösten tekoa.

4.3 Muut rajapinnat

Suunnittelussa huomioitiin myös mahdolliset muut rajapinnat, joita saatetaan hyödyntää tulevaisuudessa.

4.3.1 SUTI

Structures Unified Transportation Information, eli SUTI on ruotsalainen, vuonna 2002 perustettu voittoa tavoittelematon organisaatio. SUTI on jäseniensä omistama. SUTIlla on 42 jäsentä. Jäsenet ovat pohjoismaisia kuljetusalan yrityksiä, asiakkaita ja ohjelmistoyhtiöitä kuten esimerkiksi SLL, Cabonline Group sekä Frogne. [20.]

SUTIn tavoitteena on kehittää ja ylläpitää viestintä standardia asiakkaan ja palveluntarjoajan välillä. SUTI pyrkii kehittymään pohjoismaiseksi standardiksi kuljetusten viestinnässä. Vuosittain yli 30 miljoonaa kuljetusta välitetään SUTIn määrittelemien rajapintojen avulla. [20.]

SUTI määrittelee viestirajapintansa XML-skeeman avulla. Vaikka SUTI, kuten sen jäsenenä toimiva Frogne, ei suoranaisesti liittynyt projektiin, viestit kuitenkin huomioitiin suunnittelussa. [21.]

SUTIn viestit pyrkivät kuvaamaan mittaritapahtumia, viestikuvauksista löytyy tapahtumat vuoron aloitukselle ja lopetukselle, kuljetukset, eli reitit välitetään tilauksina, joihin sisältyy hinnoittelu. Välitysjärjestelmä voi muodostaa vuorot ja kuljetukset, ja näiden tietoja voidaan päivittää SUTIn määrityksen mukaisten viestien avulla. Kuvassa 4 on esitelty SUTIn määrittämä vuoron aloitusviesti. [21.]

1.8 MSG 1020: Resource Login

Message	MSG 1020: Resource Login
Description	MSG 1020 is a login message for an available resource (e.g. a vehicle). At the start of a shift a vehicle can login into the clients system with driverid, vehicle number and optionally a password. The message can also contain the vehicles' configuration and attributes.
Sender	Provider
Receiver	Client
Response required	YES
Response MSG	MSG 1021, MSG 1022
Client action	• check if the offered resource meets the demands • optionally - check if the supplied password is correct
Provider action	

Kuva 4. SUTI:n määrittelemä vuoron aloitusviesti.

4.3.2 Matkapalvelukeskus

Matkapalvelukeskus on yleistermi vammaispalvelulain ja sosiaalihuoltolain mukaisten kuljetuspalveluiden välitystä järjestävästä alueellisesta toimijasta. Helsingin Matkapalvelu on välittänyt edellä mainittuja kuljetuksia vuodesta 2009. Kuljetuspalvelut ovat joukkoliikennettä korvaava palvelu. Kuljetuspalveluita tarjotaan vaikeasti vammaisille, jotka eivät voi käyttää julkista liikennettä sekä liikkumisvaikeuksista kärsiville vanhuksille, jotka eivät kuitenkaan voi saada vammaispalvelulain mukaisia kuljetuspalveluita. Matkoja pyritään yhdistelemään, ja välittämään sosiaali- ja terveystoimialan kilpailuttamille liikennöitsijöille. [22; 23; 24.]

Palkanlaskentaa varten Matkapalvelukeskuksesta haetaan käsin CSV-tiedosto, joka lisätään palkanlaskennan tietoihin. Insinöörityön toteutushetkellä tätä tietoa ei ollut mahdollista hakea rajapinnasta, joten projektin osalta huomioitiin tilanne, jossa CSV:stä saatava tieto täytyisi muuttaa muun mittaritiedon mukaiseksi. Tämä tiedon yhdistäminen jätettiin osaksi palkkatietojen käyttöjärjestelmän kehittävää osuutta, jossa kehitettäisiin tapa ladata tiedoston sisältö suoraan palkkatietojärjestelmään.

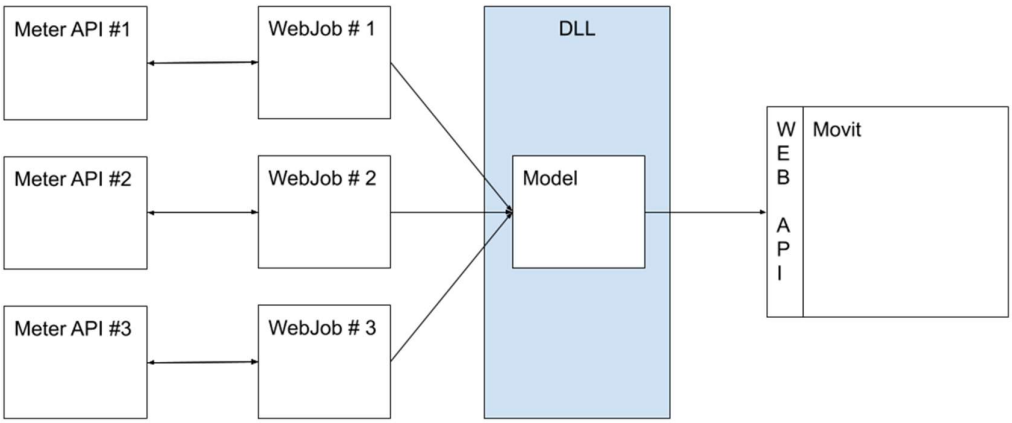
4.4 Yhdistelyjärjestelmä

Yhdistelyjärjestelmä suunniteltiin viimeisenä. Suunnittelun aikana harkittiin useampaa vaihtoehtoa. Yhdistelyjärjestelmän suurimpina tavoitteina oli muokattavuus ja itsenäi-

syys. Harkittuja metodeja vertailtiin keskenään. Mietittyjä vaihtoehtoja olivat muun muassa Linuxin cron, Hangfire ohjelmistokehys ja Azuren Webjobit. Hangfirestä oli jo kokemusta yrityksessä, eikä sen katsottu soveltuvan tarkoitukseen. Linuxin cron todettiin muun ympäristön kannalta hankalaksi. Movitin Azure App Servicet toimivat Windows -ympäristössä, joten tiedon keräämistä varten olisi pitänyt pystyttää täysin oma ympäristönsä. Tiedon hakeminen päätettiin toteuttaa Azure Webjobien avulla.

Tiedon välityksen haluttiin olevan mahdollisimman tehokasta. Mittaridatan suora välittäminen Movitiin vaatisi erilaisen rajapinnan jokaiselle mittarille. Mikäli mittarista tietoa hakeva Webjob kääntäisi mittarista saatavan tiedon halutun tietomallin mukaiseksi, tulisi jokaisen webjobin sisältää erilliset näkymämallikuvaukset, jotka tekisivät webjobien ja mittarimallin muutokset erittäin työläiksi ja vaikeiksi ylläpitää. Tämä ongelma ratkaistiin DLL:n avulla.

Kuvassa 5 esitellään yhdistelyjärjestelmän suunniteltu muoto.



Kuva 5. Yhdistelyjärjestelmän suunniteltu rakenne

4.4.1 Azure Webjob

Azure Webjob on Azuren App Servicessä oleva ominaisuus, jonka avulla voidaan ajaa ohjelma tai skripti samassa kontekstissa kuin Web-sovellus. Webjobit voivat olla laukaistuja, ajoitettuja tai jatkuvia. [25.]

Laukaistu Webjob suoritetaan osana tapahtumaketjua, esimerkiksi muutos ohjelmassa voi laukaista sähköpostien lähetyksen palvelun käyttäjille. Koska Webjob on erillinen prosessi osana palvelua, sen kaatuminen, pysähtyminen, päivittäminen tai uudelleen käynnisty, eikä näin ollen kuormita muuta palvelua, kun sähköpostit lähetetään. [26.]

Ajoitettu Webjob laukeaa määritettynä aikana. Esimerkkinä voisi olla esimerkiksi kerran viikossa tai kuukaudessa tapahtuva siivoustoiminto.

Jatkuva Webjob on käynnissä jatkuvasti, ilman erillistä aika herätettä. Ajoitetun ja jatkuvan Webjobin ero on ajoituksen määritteleminen.

Ajoitettu Webjob käynnistyy aikaherätteestä ja pysähtyy tiettyyn aikaan. Projektissa käytettiin jatkuvaa Webjobia, sillä projektin osalta oli tärkeää, että mikäli palvelu tai WebJob kaatuu, tiedonkeruu jatkuu automaattisesti Webjobin uudelleen käynnistyksen jälkeen, ja tiedonkeruun aikaväliä voidaan säätää rajapintakohtaisesti määrittämällä viive kyselysyklin jälkeen. [25.]

4.4.2 DLL

DLL eli Dynamic-link library, on Microsoftin toteutus jaetusta kirjastosta tai jaetusta oliosta. Jaetun kirjaston tarkoituksena on jakaa kirjaston resurssit suoritettavien (tässä tapauksessa projektien) tiedostojen välillä niin, etteivät projektit jaa omia resurssejaan, vaan toimivat toisistaan riippumattomasti. [27; 28.]

DLL:ien käyttö mahdollistaa koodin modulaarisuuden, koodin uudelleenkäytön, tehokkaan muistin käytön ja vähentää ohjelman kokoa. Tämän seurauksena ohjelmat latautuvat nopeammin, käyvät nopeammin ja vievät vähemmän tilaa. [29.]

Projektissa käytettiin DLL:ää pääasiassa tiedon välitysmuotona Movitin ja WebJobien välillä. Lisäksi DLL sisältää rajapinta-asiakkaan (API Client) Movitin rajapintaan. DLL:n ansiosta WebJobit voivat kartoittaa oman datansa yhtenäiseen muotoon, joka siirretään kartoituksen jälkeen jaetun asiakkaan kautta Movittiin. Aiemmin listattujen hyötyjen lisäksi tämä helpoittaa vian selvitystä ja tuo varmuutta tiedonsiirtoon.

5 Toteutus

Työn toteutus tapahtui osissa. Osa suunnittelusta, esimerkiksi SUTI-viestinvälitysstandardiin tutustuminen viivästyi huomattavasti, koska rajapinnan määrittely dokumentteja ei ollut saatavissa yhdistyksen sivuilta suunnittelun aikana. Kuvaukset saatiin kuitenkin myöhemmässä vaiheessa, ja ne sopivat jo suunniteltuun malliin. Myös rajapintojen testauksessa oli vaikeuksia, sillä rajapintojen tunnuksien saantiin kului huomattavasti odotettua kauemmin.

Toteutuksen vaiheet on kuvattu omissa otsikoissaan. Aluksi rajapinnasta saatavaa tietoa tarkasteltiin suorilla testirajapintakyselyillä Postman-ohjelman avulla. Postman on rajapinta-asiakasohjelma, jonka avulla voidaan muun muassa suorittaa REST-rajapintakyselyitä helposti. Testidata mahdollisti WebJobien, DLL:n ja Movitin rajapinnan muodostamisen ennen kuin oikeat rajapinta-avaimet olivat saatavilla.

Myös toteutettujen WebJobien siirtäminen Azure-palveluihin osoittautui ongelmalliseksi. WebJobien ensimmäinen toteutus rakennettiin vasta julkaistun .Net Core -version 3.0 avulla, joka osoittautui epävakaaksi, eikä WebJobeja saatu julkaistua Azureen kehittymättömän version takia. WebJobien päivittäminen .Net Coren versioon 3.1, joka oli ensimmäinen pitkän tuen saanut .Net Core 3.x -versio ratkaisi julkaisuongelmat.

5.1 Azure WebJob

Projektin osalta Webjob ajaa .Net Core -versio 3.1 konsolisovellusta, joka kerää tiedot mittarista, kartoittaa ne ViewModeliin sopivaksi, ja välittää ViewModelit Movitin rajapintaan.

Webjob on mittari- ja asiakaskohtainen: Jokaiselle mittarille on määritetty oma Webjob, jossa on määritetty mittaria varten tarvittavat kartoitustoiminnot sekä kysely- ja ajoituslogiikka. Tämä mahdollistaa sen, ettei WebJob ole riippuvainen tietystä välitysmuodosta, vaan jokainen on määritettävissä erikseen mittarin mukaisesti.

5.1.1 Ajoitus

Mitaxin ja Vuoronetin rajapinnoista saatava tieto on eri muodossa. Vaikka kummastakin rajapinnasta saadaan kyselyillä vain jo toteutunutta ja rajapintaan siirrettyä tietoa, kyselyiden ajoituksessa oli huomioitava rajapintojen palauttaman tiedon erot sekä kyselyiden aiheuttama kuormitus kaikille rajapinnoille. Kyselyiden väliin lisättiin viiveitä käyttämällä .Net Coren tarjoamaa Delay-toimintoa, joka pysäyttää kyselyiden suorituksen määrätyksi ajaksi. Kyselyiden väliset odotusajat määritettiin empiirisesti jokaisen rajapinnan kohdalla.

5.1.2 Asetukset

Mittarirajapinnat edellyttävät asiakkaan käyttäjätunnuksia, joten WebJob on aina asiakaskohtainen. Asiakaskohtaiset tunnukset määritetään Azure web -sovelluksen asetuksissa. Ilman näitä asetuksia WebJobit sammuttavat itsensä välittömästi, vaikka WebJob olisikin asennettu web-sovellukseen. WebJobin sammuttaminen estää sovelluksen sekä mittarirajapintojen turhan kuormittamisen.

5.1.3 Hallinnointi

Azuren web-sovellusasetusten lisäksi WebJobia voidaan suoraan hallinnoida Azure-portaalista käsin. Sen lisäksi, että WebJobin voi käynnistää ja pysäyttää, WebJobin toimintaa voi myös seurata reaaliajassa lokien avulla. WebJobit toimivat itsenäisesti, ilman käyttäjän syötettä, joten toiminnan seuraaminen ja lokien pitäminen on tärkeää välitetyn tiedon luotettavuuden varmistamiseksi. [25.]

5.2 Tiedon keräys

Tietoa haetaan aina takautuvasti, sillä mittaritapahtumista saatavat vuorot muodostuvat kumpaankin rajapintaan vasta vuoron päätyttyä, ja kun mittari välittää tiedon rajapintaan. Mahdollisten yhteysongelmien vuoksi jotkin vuorot saattavat kirjautua viiveellä.

Tiedon keräyksen hoitaa jokaisessa WebJobissa erikseen määritetty rajapinta-asiakas, joka on määritetty tarjotun rajapinnan mukaiseksi.

5.2.1 Mitax

Mitaxin tarjoama rajapinta palauttaa maksutietoja tai raportteja. Kysely palauttaa ensimmäiset käsittelemättömät maksutieto- tai raporttioliot, enimmillään 80 kerrallaan. Oliot voidaan merkitä käsitellyiksi lähettämällä lista saatujen olioiden tunnisteista, jonka jälkeen voidaan kysellä uusia käsittelemättömiä olioita.

Siirtotapa on erittäin luotettava, sillä mikäli tiedon muokkaaminen ja siirtäminen Movitiin epäonnistuu, saatua tietoa ei merkitä käsitellyksi. Näin voidaan varmistua siitä, ettei tietoa ole jäänyt siirtymättä.

Mikäli tiedonsiirrossa on tapahtunut virheitä, tietoa ei kuitenkaan voi hakea automaattisesti uudestaan, vaan on oltava yhteydessä Mitaxin rajapinnan toimittajaan, että he merkitsevät halutun tiedon jälleen käsittelemättömäksi.

5.2.2 Semel

Semelin rajapinnasta mittaridataa kysellään palvelu- eli mittarinumeron ja aikavälin avulla. Saatavissa olevat mittarinumerot voidaan hakea erillisellä kyselyllä.

Kyselyn aikavälit tulee huomioida hyvin tarkkaan, muussa tapauksessa vuoro, joka jää annetun aikavälin ulkopuolelle joko alku- tai loppuajankohtansa perusteella jää välittymättä. Välittymättä jäänyt vuoro vääristää palkanlaskennan tuloksia. Vuoro voidaan hakea jälkikäteen, mutta jatkuvasti puuttuvat vuorot rapauttaisivat tiedon luotettavuutta asiakkaan näkökulmasta.

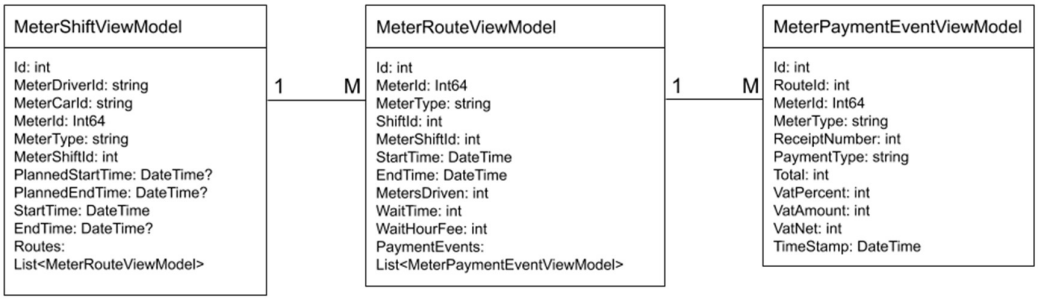
Vuoronetin osalta jokaisen palvelumittarin tietoja kysellään oletuksena kahden päivän takaa nykyisen päivän loppuun. Mikäli WebJob jostain syystä keskeytyy, tiedot haetaan viimeiseltä seitsemältä vuorokaudelta. WebJob sisältää myös tiedon mittarinumeron viimeisestä toteutuneesta vuorosta. Mikäli viimeinen toteutunut vuoro on tiedossa, aikaväli kyselylle on viimeisen toteutuneen vuoron loppuajankohdasta nykyisen päivän loppuun.

Vuoronet ei salli kyselyitä useammalta viikolta, joten mikäli viimeisestä vuorosta on kulunut yli viikko, käytetään oletus aikaväliä.

5.3 Tiedon välitys

Tiedon välitys mittarirajapinnan ja Movitin välillä toteutettiin käyttäen tiedonsiirto-olioita (DTO). Tiedonsiirto-olion tarkoituksena on helpottaa tietojen välitystä sovelluslogiikkakerroksen ja tietokantamalli kerroksen välillä. Movitissa käytetään Malli-Näkymä-NäkymäMalli (model-view-viewmodel, MVVM) suunnittelumallia, joten DTOt on nimetty NäkymäMalleiksei (ViewModel). Jokainen WebJob määrittää omissa malleissaan kartoituksen jaettuun NäkymäMalliin. Jaettua NäkymäMallia voidaan käyttää myös palkkatieto järjestelmässä tiedon esittämiseen ja muokkaamiseen. [30, s.441; 31; 32; 33.]

Kuvassa 6 näytetään muodostettujen näkymämallien väliset suhteet, ja samankaltaisuus kuvassa 3 esiteltyyn tietomalliin.



Kuva 6. DTO-kuvaus

Kuvassa 5 kuvatut tiedonsiirto-oliot (ViewModel) määritettiin DLL:ssä. Keskitetyn määrittelyn ansiosta ylläpito on selkeää. Mikäli mittarista halutaan kerätä lisätietoa, uusia kenttiä voidaan tarvittaessa lisätä null-arvoisina. Tämän ansiosta yksittäistä Webjobia voidaan muuttaa uusien mittarien ja asiakkaiden tarpeiden mukaan ilman, että jokaista mittaria tai rajapintaa pitäisi muuttaa erikseen.

Rajapinta asiakas on määritetty niin, että jokainen samassa Azuren instanssissa pyörivä WebJob jakaa tunnukset Movitin rajapintaan, joka mahdollistaa tunnuksen vaihtamisen

erittäin nopeasti, mikäli siihen havaitaan tarve. Dependency Injectionin avulla lokitustoiminnot voidaan välittää WebJobista, joka mahdollistaa virheen paikallistamisen tiettyyn Webjobiin. Tämän avulla voidaan helposti päätellä, johtuuko virhe Webjobin tavasta välittää tietoa, rajapinta asiakkaasta vaiko Movitista. Esimerkkikoodissa 3 esitellään rajapinta asiakkaan määrittely Dependency Injectionin avulla.

```
private static readonly HttpClient client = new HttpClient();
      private ILogger logger;
      private string apiKey;

      public MovitApiClient(
            string urlBase,
            string apiKey,
            ILogger<MovitApiClient> logger
      )
      {
            client.BaseAddress = new Uri(urlBase);
            this.logger = logger;
            this.apiKey = apiKey;
      }
```

Esimerkkikoodi 3. MovitApiClientin määritys Dependency Injectionin avulla

5.4 Movit

Tietojen tallennus Movitiin hoidetaan Mediator-suunnittelumallin mukaisesti. Ohjelmistoon on määritetty käsittelijä, joka ottaa vastaan vuoro- tai reittiolion NäkymäMallin. Controller välittää olion Mediator-mallille, joka hoitaa tiedon kirjoittamisen tietokantaan. Mediator-suunnittelumalli mahdollistaa tiedon siirron modulaarisuuden poistamalla olioiden välisen tiukan kytkennän, olioiden välisiä suhteita voidaan teoriassa määrittää irrallaan toisistaan. Esimerkkikoodissa 3 esitellään rajapinnan toteutus käyttäen Mediator-suunnittelumallia.

```
[WebJobAuthorize]
[RoutePrefix("api/webjob/meter")]
public class WebJobMeterController : ApiController
{
      private readonly IMediator _mediator;

      public WebJobMeterController(IMediator mediator)
      {
            _mediator = mediator;
      }

      [HttpPost]
      [Route("shift")]
      public async Task<IHttpActionResult> PostMeterShift(
```

```
            [FromBody]PostMeterShift shift,
            CancellationToken cancellationToken
        )
        {
            var result = await _mediator.Send(shift, cancellationToken);
            return Ok(result);
        }

        [HttpPost]
        [Route("route")]
        public async Task<IHttpActionResult> PostMeterRoute(
            [FromBody]PostMeterRoute route,
            CancellationToken cancellationToken
        )
        {
            var result = await _mediator.Send(route, cancellationToken);
            return Ok(result);
        }
}
```

Esimerkkikoodi 4. Rajapinnan toteutus käyttäen Mediator-suunnittelumallia

Tiedon oikeellisuuden varmistamiseksi käytetään kolmea tunnistetta, jolla tarkistetaan, ettei tietokantaan ole jo luoto vastaavaa vuoroa: tiedonlähteen tyyppi, taksamittarin numero, ja taksamittarin vuoron numero. Tietokantakyselyn nopeuttamiseksi nämä kolme vuorotaulun saraketta indeksoitiin, joka mahdollistaa haun valmistumisen epälineaarisessa ajassa. [34.]

Insinöörityön osalta Movitin rajapintaan määritettiin ainoastaan tavat lähettää yhdistelytietoa Movitin tietokantaan.

Movitin tietokantaan lisättiin suunnittelussa kuvatut taulut migraatioiden avulla, joista MeterShift-taulu muodostettiin yhdistämällä aiempi vuoro taulu ulkopuolisista mittareista saatuihin lisäkenttiin:

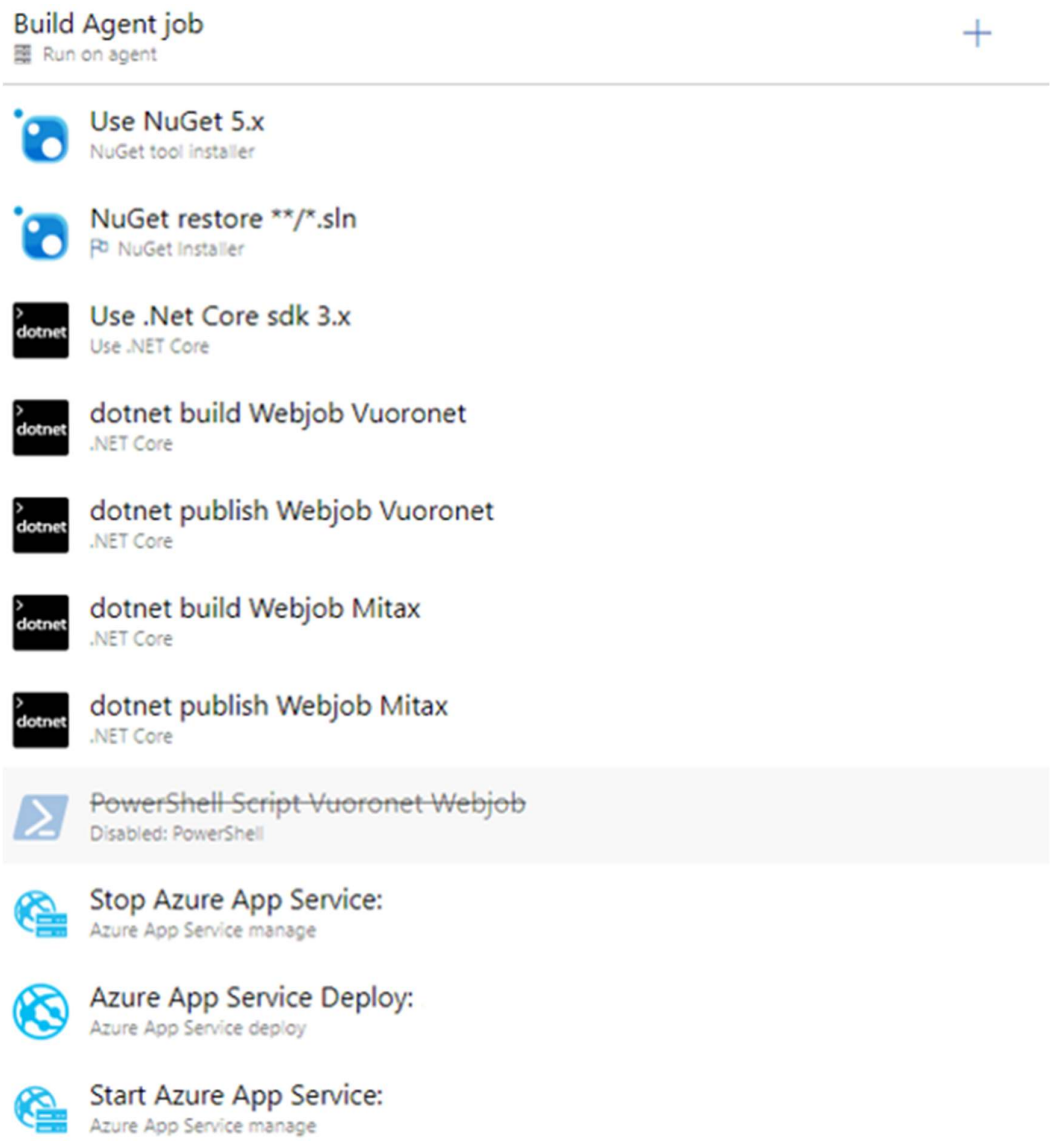
- MeterType
- MeterId
- MeterShiftId
- MeterCarId
- MeterDriverId
- PlannedStartTime
- PlannedEndTime.

5.5 Julkaisu

Toteutetun yhdistelyjärjestelmän julkaisu suunniteltiin ja toteutettiin, kun projektin muut osat olivat valmiita testausta varten. Webjob toimii osana Movit instanssin Azure App Serviceä. Serviceen julkaisu hoidettiin Azuren palveluiden avulla.

WebJobien rakennus (build) ja käyttöönotto (deploy) hoidetaan Azure DevOpsin Azure Pipelinesin kautta. Azure Pipelines mahdollistaa jatkuvan integraation (continuous integration, CI) ja jatkuvan toimituksen (continuous delivery, CD). Pipelines mahdollistaa ohjelmiston käyttöönoton lisäksi myös testauksen, jolloin hyvillä testeillä voidaan varmistaa, ettei tuotantoon pääse toimimatonta koodia.

WebJobien rakennus ja käyttöönottoa varten määritettiin oma Pipeline, jossa määritetään kuvan 7 mukaiset sovelluksen rakennusta varten tarvittavat toimenpiteet.



Kuva 7. Sovelluksen rakennukseen käytetyt toimenpiteet Azure Pipeline Build Agent jobissa

WebJobien rakennustoimenpiteet aloitetaan hakemalla NuGet-paketinhallintaohjelma, jonka jälkeen haetaan WebJob-projekteissa määritetyt NuGet-paketit. Pakettien jälkeen asennetaan .Net Coren ohjelmistokehityspaketti, jonka avulla Webjobit voidaan rakentaa ja julkaista. Tämän jälkeen kohteena oleva Azure App Service sammutetaan, uudet Webjobit otetaan käyttöön palvelussa, ja palvelu käynnistetään uudestaan.

Rakennuksen ja julkaisun määrittämisen yhteydessä tuli useita ongelmia. Webjobien ja Movitin eri ohjelmistokehykset aiheuttivat ongelmia rakennus- ja julkaisuvaiheessa, joten Webjobit eriytettiin omaan pipelineen. Vaikka päivitysprosessi saatiin osaksi yleistä julkaisuprosessia, käynnistämällä Webjobien pipeline osana Movitin julkaisu-pipelineä, täytyy jokainen Webjobia käyttävä Movit-instanssi lisätä erikseen.

Lisäksi Webjobit oli kehitetty käyttämään .Net Core -ohjelmistokehyksen versiota 3.0, joka oli uuden .Net Core 3 -version esikatseluversio. Versio oli niin epävakaa rakennuksen ja julkaisun osalta, että useamman viikon selvittelyn jälkeen tehtiin päätös päivittää WebJob-sovellukset käyttämään vastajulkaistua .Net Core 3.1 -versiota. Päivityksen jälkeen rakennus- ja julkaisuprosessi saatiin toimimaan.

6 Arviointi

Projektin päätteksi oli luotu järjestelmä, joka yhdistelee takseista saatavaa mittaridataa palkanlaskentaa varten. Kerättyä tietoa voidaan myös käyttää tulevaisuudessa esimerkiksi laivaston tehokkuuden arviointiin.

Webjobit ovat toimiva ratkaisu tiedon yhdistämiseen ja uusien tiedon lähteiden lisäämisen. Webjobien seurannassa on kuitenkin huomattavasti kehitettävää. Tällä hetkellä Webjobin ajautumisesta virhetilaan ei tule automaattista ilmoitusta, vaan tilaa pitää seurata manuaalisesti. Tämä voi pahimmillaan aiheuttaa pitkän tauon tiedon keräämisessä. Lokien avulla tieto voidaan hakea takautuvasti, mutta prosessi tulisi suunnitella tehokkaammaksi, virhetila ilmoitusten ja automaattisen uudelleenkäynnistyksen avulla.

Webjobien rakennus ja julkaisu toimii yhden asiakkaan tilanteessa riittävän hyvin, mutta mikäli useampi asiakas ottaa käyttöön mittaridatan yhdistelyn, julkaisuprosessi pitää suunnitella ja toteuttaa uudestaan.

Mittaridatan luotettavuudessa on myös parannettavaa. Keskusteluissa palkanlaskijoiden kanssa on havaittu virheitä toteutuneiden ajojen ja mittareista saatavan datan välillä. Virheet voivat aiheutua mittariin tulleesta virheestä tai mittarin virheellisestä käytöstä.

Näitä virheitä ei voida korjata tiedonkeruun avulla. Tulevan palkanlaskentajärjestelmän tulee siis huomioida syöttövirheiden mahdollisuus, ja mahdollistaa virheellisen tiedon korjaaminen käsin. Nykyistä tilannetta parantaa automaattisen tiedon keräyksen huomattavasti nopeampi palaute sykli.

Aikaisemman palkanlaskentaprosessin mukaisesti tieto virheestä oli havaittavissa vasta palkanlaskennan aikaan, eli pahimmillaan useamman viikon päästä virheen sattumisesta. Nopeampi palaute mahdollistaa virheen aikaisemman havaitsemisen. Mikäli virhe johtui esimerkiksi väärin ymmärretystä prosessista, virhe voidaan korjata huomattavasti tehokkaammin ja nopeammin.

7 Yhteenveto

Projektissa saavutettiin tärkeimmät tavoitteet: asiakkaan mittaridataa voidaan kerätä ja yhdistellä palkanlaskentaa varten mielekkäällä tavalla.

Lisäksi joitain tavoitteita saavutettiin osittain. Konkreettisen tiedon hankinnan vaikeudet, esimerkiksi SUTI-standardin kuvausten osalta, rajasi jonkin verran kansainvälisten standardien ja mittareista saatavan tiedon selvittämistä joihinkin kotimaisiin toimijoihin.

Myös kotimaisien tiedonhankintalähteiden selvittäminen oli haastavaa. Matkapalvelukeskukset ovat alueellisia toimijoita, joiden rajapintakuvauksia, tai tietoja ei löydy julkisesti, eikä niitä määrittele mikään standardi. [35, s. 54.]

Kehitettyjä rajapintoja ja menetelmiä voidaan ja tullaan jatkossa kehittämään paremmiksi. Webjobien avulla uusien mittarien lisääminen ja yhdisteleminen on tehokasta.

Projektissa ei käsitelty muita tapoja tuoda tietoa, koska suunnitteluvaiheessa tehtiin päätös siitä, että näiden tapojen lisääminen on luontevampi osa käyttöliittymän ja järjestelmän suunnittelua.

Lähteet

1 Intoit Oy. Verkkoaineisto. <https://www.intoit.fi/>, luettu 13.5.2020.

2 Intoit Oy. Verkkoaineisto. <https://www.movit.fi/>, luettu 13.5.2020.

3 Intoit Oy. Verkkoaineisto. <https://kouluun.fi/>, luettu 13.5.2020.

4 Luotettavaa matkan mittausta. Verkkoaineisto. <http://trippi.fi/>, luettu 7.5.2020.

5 Wikipedia. XML. Verkkoaineisto. <https://en.wikipedia.org/wiki/XML>, luettu 7.5.2020.

6 Semel Oy. Verkkoaineisto. <https://www.semel.fi/semel/yritys.html>, luettu 7.5.2020.

7 Semel Oy. ASP.NET Web API Help Page. Verkkoaineisto. <https://vuoro.net/VuoronetRestService/Help>, luettu 7.5.2020.

8 Wikipedia. JSON. Verkkoaineisto. <https://en.wikipedia.org/wiki/JSON>, luettu 7.5.2020.

9 T. Bray, Ed. The JavaScript Object Notation (JSON) Data Interchange Format. Verkkoaineisto. <https://tools.ietf.org/html/std90>. 1.12.2017, luettu 7.5.2020.

10 Microsoft. What is the Azure SQL Database service?. Verkkoaineisto. <https://docs.microsoft.com/en-us/azure/sql-database/sql-database-technical-overview>. 8.4.2019, luettu 7.5.2020.

11 Microsoft. What is .NET Framework?. Verkkoaineisto. <https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet-framework>, luettu 7.5.2020.

12 Wikipedia. .NET Framework. Verkkoaineisto. https://en.wikipedia.org/wiki/.NET_Framework, luettu 7.5.2020.

13 Microsoft. Get started with .NET Framework. Verkkoaineisto. <https://docs.microsoft.com/en-us/dotnet/framework/get-started/#Introducing>. 2.4.2020, luettu 7.5.2020.

14 Microsoft. .NET Core overview. Verkkoaineisto. <https://docs.microsoft.com/fi-fi/dotnet/core/about>. 26.3.2020, luettu 7.5.2020.

15 Microsoft. Packages, metapackages, and frameworks. Vekkoaineisto. <https://docs.microsoft.com/en-us/dotnet/core/packages>. 29.4.2020, luettu 7.5.2020.

16 Wikipedia. Entity Framework. Verkkoaineisto. <https://en.wikipedia.org/wiki/Entity_Framework>, luettu 7.5.2020.

17 Wikipedia. Representational state transfer. Verkkoaineisto. <https://en.wikipedia.org/wiki/Representational_state_transfer>, luettu 7.5.2020.

18 Wikipedia. Microsoft Azure. Verkkoaineisto. <https://en.wikipedia.org/wiki/Microsoft_Azure>, luettu 7.5.2020.

19 Microsoft. App Service overview. Verkkoaineisto. <https://docs.microsoft.com/en-us/azure/app-service/overview>. 30.4.2020, luettu 7.5.2020.

20 SUTI. About us. Verkkoaineisto. http://suti.se/facts/, luettu 7.5.2020.

21 SUTI. Public documents. Verkkoaineisto. <http://suti.se/2019-2/>, luettu 7.5.2020.

22 Helsingin Matkapalvelu. Tietoa Matkapalvelusta. Verkkoaineisto. <https://www.hel.fi/palvelukeskus/matkapalvelu-fi/matkustaminen/Tietoa+Matkapalvelusta/>. 29.3.2019, luettu 7.5.2020.

23 Helsingin Matkapalvelu. Tietoa Helsingin Matkapalvelusta. Verkkoaineisto. <https://www.hel.fi/palvelukeskus/matkapalvelu-fi/uutiset/tietoa>. 16.4.2014, luettu 7.5.2020.

24 Helsingin Matkapalvelu. Matkojen yhdistely. Verkkoaineisto. <https://www.hel.fi/palvelukeskus/matkapalvelu-fi/matkustaminen/yhdistely/>. 29.3.2019, luettu 7.5.2020.

25 Microsoft. Run background tasks with WebJobs in Azure App Service. Verkkoaineisto. <https://docs.microsoft.com/en-us/azure/app-service/webjobs-create#overview>. 16.10.2018, luettu 7.5.2020.

26 Scott Hanselman. Introducing Windows Azure WebJobs. Verkkoaineisto. <https://www.hanselman.com/blog/IntroducingWindowsAzureWebJobs.aspx>. 23.1.2014, luettu 7.5.2020.

27 Wikipedia. Dynamic-link library. Verkkoaineisto. <https://en.wikipedia.org/wiki/Dynamic-link_library>, luettu 7.5.2020.

28 Wikipedia. Library (computing). Verkkoaineisto. <https://en.wikipedia.org/wiki/Library_(computing)#Shared_libraries>, luettu 7.5.2020.

29 Microsoft. What is a DLL?. Verkkoaineisto. <https://support.microsoft.com/en-us/help/815065/what-is-a-dll>. 18.12.2019, luettu 7.5.2020.

30 Simo Silander, Vesa Ollikainen, Juha Peltomäki. 2010. Java

31 Wikipedia. Model–view–viewmodel. Verkkoaineisto. <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel>, luettu 7.5.2020.

32 Wikipedia. Data transfer object. Verkkoaineisto. <https://en.wikipedia.org/wiki/Data_transfer_object>, luettu 7.5.2020.

33 Microsoft. Views in ASP.NET Core MVC. Verkkoaineisto. <https://docs.microsoft.com/en-us/aspnet/core/mvc/views/overview?view=aspnetcore-3.1> 5.12.2019, luettu 7.5.2020.

34 Wikipedia. Database index. Verkkoaineisto. <https://en.wikipedia.org/wiki/Database_index>, luettu 7.5.2020.

35 Jouni Mutanen, Mika Piipponen. 2008. Matkapalvelukeskuksen hankinta. Verkkoaineisto. <https://julkaisut.valtioneuvosto.fi/bitstream/handle/10024/78322/LVM_40-2008.pdf?sequence=6&isAllowed=y/pdf>, luettu 7.5.2020.