OULUN AMMATTIKORKEAKOULU

Valtteri Sandström

SUORATOISTODATAN LÄHETTÄMINEN JA VISUALISOINTI AZURE-PILVIPALVELUSSA

SUORATOISTODATAN LÄHETTÄMINEN JA VISUALISOINTI AZURE-PILVIPALVELUSSA

Valtteri Sandström
Opinnäytetyö
Kevät 2023
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, laite- ja tuotesuunnittelun suuntautumisvaihtoehto

---

Tekijä: Valtteri Sandström
Opinnäytetyön nimi: Suoratoistodatan lähettäminen ja visualisointi Azure-pilvipalvelussa
Työn ohjaaja: Teemu Korpela
Työn valmistumislukukausi ja -vuosi: Kevät 2023
Sivumäärä: 26

Opinnäytetyön tavoitteena oli tehdä yritykselle suoratoistodatan visualisointi Azure-pilvipalveluun. Opinnäytetyössä tutustutaan myös digitaalisiin kaksosiin. Opinnäytetyössä tehdään JSON-tiedostoon datamalli. Datamallissa on määritetty asiat, mitä yritys haluaa myöhemmin mitata. Opinnäytetyössä käytettiin Azuren omia työkaluja datan lähettämiseen, käsittelyyn ja visualisointiin.

Opinnäytetyö alkoi tutustumalla Azure Event Hub -työkaluun. Azure EventHub -työkalua käytettiin lähettämään dataa Azure-pilvipalveluun. Opinnäytetyössä data luotiin tekemällä Pythonilla datageneraattori, joka luo sattumanvaraista dataa. Datageneraattori tehtiin Linux-pohjaiselle virtuaaliselle koneelle.

Sattumanvarainen data käsiteltiin ja visualisoitiin Azure Data Explorer -työkalun avulla. Opinnäytetyössä tehtiin esiselvitys, miten digitaalinen kaksonen toimii ja miten sitä voisi hyödyntää opinnäytetyöprojektissa myöhemmin.

Lopputuloksena saatiin tehtyä virtuaalisessa koneessa datageneraattori, joka lähettää dataa Azure Event Hub -työkalun kautta Azure Data Explorer -työkaluun. Azure Data Explorer -työkalussa saatiin luotua datalle kuvaajia, joissa näkyy mitattu data suhteessa aikaan. Yritys pystyy jatkamaan opinnäytetyössä tehtyä projektia tästä eteenpäin haluamallaan tavalla. Esiselvitys on tehty tämän opinnäytetyön avulla. Opinnäytetyön projektin ohella saatiin tehtyä esiselvitys, miten digitaaliset kaksoset toimivat Azure-pilvipalvelun kanssa.

Asiasanat: Azure Event Hub, Azure Data Explorer, Digitaalinen kaksonen

ABSTRACT

Oulu University of Applied Sciences
Degree Programme of Information Technology, Option of Device and Product Design

---

Author: Valtteri Sandström
Title of thesis: Sending and visualizing streaming data in in the Azure Cloud Services
Supervisor: Teemu Korpela
Term and year when the thesis was submitted: spring 2023
Number of pages: 26

The goal of this thesis was to visualize streaming data in the Azure Cloud Services for the company. The thesis also introduces digital twins. In this thesis, a data model is created for the JSON file. The data model defines the things that the company wants to measure later. This thesis used Azure's own tools for sending, processing and visualizing data.

This thesis started by getting to know the Azure Event Hub tool. The Azure EventHub tool was used to send data to the Azure cloud service. In this thesis, the data was created by making a data generator with Python, which creates random data. The data generator was made on a Linux-based virtual machine. Random data was processed and visualized using the Azure Data Explorer tool. In the project, a preliminary investigation was carried out on how the digital twin works and how it could be used in the project later

The result was a data generator in a virtual machine that sends data via the Azure Event Hub tool to the Azure Data Explorer tool. Graphs were created for the data in the Azure Data Explorer tool, which shows the measured data in relation to time. The company will be able to continue the project from now on as it wishes. The preliminary investigation has been done with the help of this thesis. Along with this thesis, a preliminary investigation was done for the digital twins, how they work with the Azure Cloud Services.

Keywords: Azure Event Hub, Azure Data Explorer, Digital Twin

SISÄLLYS

1 JOHDANTO

Opinnäytetyön tavoitteena oli tehdä yritykselle suoratoistodatan visualisointi Azure-pilvipalveluun. Opinnäytetyössä tutustutaan myös digitaalisiin kaksosiin. Opinnäytetyössä tehdään JSON-tiedostoon datamalli. Datamallissa on määritetty asiat, mitä yritys haluaa myöhemmin mitata.

Opinnäytetyössä ei mitata dataa, mutta tehdään sattumanvaraista dataa. Tätä varten täytyy tehdä datageneraattori, joka luo JSON-tiedostoon dataa datamallin mukaisesti. Datageneraattori tehdään käyttämällä Python-ohjelmointikieltä. Data täytyy myös lähettää Azure-pilvipalveluun, käsitellä ja visualisoida.

Datan lähettämistä varten yritys halusi, että käytetään Azure Event Hub -työkalua. Azure Event Hub -työkaluun voi yhdistää Python-koodin avulla. Datan lähettämisen jälkeen se täytyy vielä käsitellä ja visualisoida.

Datan käsittelyyn ja visualisointiin käytetään Azure Data Explorer -työkalua. Datan voi lähettää siihen Azure Event Hub -työkalun kautta. Azure Data Explorer -työkalulla voi käsitellä sekä visualisoida datan käyttämällä sen omaa query-kieltä. Azure Data Explorer -työkaluun voi myös tehdä koontinäytön, johon kerätään visualisoinnit, joita yritys haluaa seurata.

Opinnäytetyössä myös tutustutaan digitaalisiin kaksosiin ja selvitetään, miten niitä voisi tässä työssä käyttää. Tämä tulee toimimaan esiselvityksenä tulevaisuutta varten projektissa.

Datageneraattoria varten täytyy myös ottaa käyttöön virtuaalinen kone, jotta data saadaan toimimaan suoratoistona. Tämä tehdään siksi, että datageneraattorin koodi ei olisi riippuvainen siitä, onko tietokone, jolla se on tehty, päällä vai kiinni. Tässä myös otetaan käyttöön Docker. Docker on alusta, joka käyttää virtualisointia konttien toimittamiseen. Nämä kontit ovat ohjelmistopaketteja. (1.)

2 EVENTHUBIN KÄYTTÖÖNOTTO JA TESTAUS

Opinnäytetyön aiheena on tehdä esiselvitys, sekä demo bts-datan monitoroinnista. Bts on lyhenne sanoista base transceiver station. Tämä on kiinteä radiolähetin-vastaanotin mobiiliverkoissa, joka liittää mobiililaitteet verkkoon. Bts lähettää ja vastaanottaa radiosignaaleja mobiililaitteisiin ja muuntaa ne digitaalisiksi signaaleiksi. Nämä signaalit reitittävät verkon muihin päätelaitteisiin, joita se välittää verkossa reitittääkseen verkon muihin päätelaitteisiin ja internettiin. (2.) Työssä käytetään dummydataa eli esimerkkidataa, jonka rakenne kuitenkin lähentelee oikeaa bts-dataa, jonka kyseinen yritys käyttää tai mittaa.

2.1 Työkalut

Työssä käytettiin Microsoft Azure -pilvipalvelupohjaa. Microsoft Azure on pilvipalvelupohja, jolla on yli 200 tuotetta, joista yksi on Azure Event Hub. Se lasketaan opinnäytetyön tekohetkellä kolmeen isoimpaan pilvipalveluista. Opinnäytetyön alkaessa yritys antoi prioriteetiksi käyttää Azure-pilvipalvelua ja Azure Event Hubia. Azure Event Hub on julkaisu-tilauspalvelu, jolla voi vastaanottaa ja lähettää miljoonia tapahtumia sekunnissa. Nämä tapahtumat voi myös suoratoistaa useille kuluttajille. Tämä mahdollistaa suurten datamäärien prosessoimisen ja analysoinnin. Azure Event Hubs -kirjasto mahdollistaa Event Hub -tapahtumien julkaisun. Azure Event Hubien kanssa voi käyttää apuna Capture-työkalua tallentamaan dataa esimerkiksi blob-kontteihin.

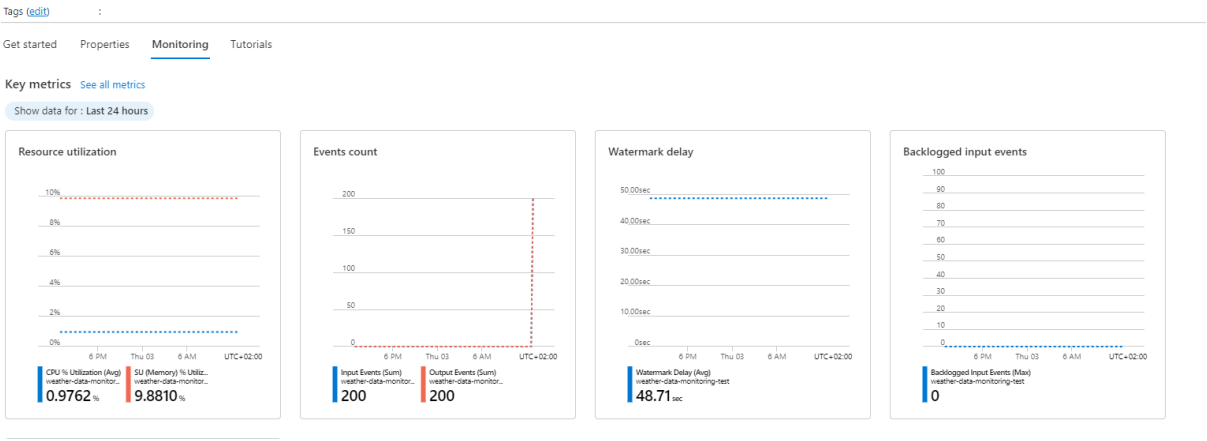
Azure Event Hub -työkalua käyttöön ottaessa täytyi valita hintapaketti. Vaihtoehtoja oli neljä: Basic, Standard, Premium ja Dedicated. Hinnat määräytyvät sen mukaan, paljonko muistia tarvittiin. Yritys antoi opinnäytetyöhön käyttöön Standard-paketin, jonka hinta oli 0.027€ / miljoona tapahtumaa.

Capturella ladataan suoratoistodataa Azure-pilvipalveluun. Tämä työkalu päästää käyttäjän aloittamaan nopeasti datan prosessoinnin, eikä tällöin tarvitse keskittyä datan tallentamiseen. Opinnäytetyössä täytyy datan tallennuksen lisäksi analysoida dataa. Tähän käytetään Pandas-kirjastoa. Pandas on Python-kirjasto, jota käytetään esimerkiksi data-analytiikassa ja tietojenkäsittelyssä. (3.)

2.2 Datan mittaus

Ennen datan mittausta opiskeltiin Azuren perusteita ja testattiin, kuinka Azure Event Hub toimii datan suoratoistossa. Ohjelmointiympäristönä käytettiin pääsääntöisesti Visual Studio Code- ja Azure Data Studio -ohjelmistoympäristöjä. Tämä suoratoiston testaus tehtiin käyttämällä omavalintaista datageneraattoria tai vapaavalintaista API:a, jotta saataisiin mitattua suoratoistodataa. Tähän kyseiseksi API:ksi valittiin OpenWheatherMap, jolla saadaan käyttöön suoratoistosäädataa. Hyvä vaihtoehto on myös JSONPlaceholder, jolla voitaisiin suorittaa kaikki testaukset paikallisena, jottei tarvitsisi tehdä erikseen API-pyyntöä.

Kumpaakin vaihtoehtoa testattiin, jotta nähtiin, miten Event Hub toimii erilaisten lähteiden kanssa. Kummassakin vaihtoehdossa pyritään lähettämään dataa Azure Event Hubiin. Kun data saatiin kerättyä API:n kautta ja lähetettyä Event Hubiin, täytyi tehdä datalle tallennusvaihe, jonka pystyy tekemään Azuren Capture-työkalulla. (Kuva 1.)



KUVA 1. Tallennettua testisäädataa

Data tulee konttiin Parquet-muodossa, jolloin data ei ole ihmissilmälle luettavassa muodossa. Parquet -tiedostomuotoa käytettiin, koska se operoi hyvin kompleksisen datan kanssa ja parquet pakkaa datan hyvin, jolloin se ei vie paljoa muistia. Mitatun datan voi myös muuntaa JSON-muotoon, jotta se on luettavissa. Se onnistuu käyttämällä Pandas-kirjastoa muuntamaan data *tietorakenteeksi* eli dataframeksi. Tämän jälkeen Pandas pystyy muuntamaan datan JSON-muotoon. (Kuva 2.)

```python
import uuid
import os
import json
import requests
import pyarrow
from dotenv import load_dotenv
from pyowm import OWM
from azure.eventhub import EventHubProducerClient, EventData

load_dotenv()

# api_url_json_placeholder = "https://jsonplaceholder.typicode.com/users/1/todos"
api_key = os.environ.get("API_KEY_OWM")
api_key_2 = OWM(api_key)
connection_string= os.environ.get("CONNECTION_STRING_EVEN_HUB")
lat = "65.01"
lon = "25.47"
part = "minutely"
api_url_openwheather = f"https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={api_key}"

wheather_manager = api_key_2.weather_manager()
wheather = wheather_manager.weather_at_place('Oulu,FIN')
w = wheather.weather

w.detailed_status
w.wind()
w.humidity
w.temperature('celsius')
w.rain
w.heat_index
w.clouds
producer = EventHubProducerClient.from_connection_string(conn_str= connection_string, eventhub_name="bts-monitoring-hub")
while True:
    devices = []
    for x in range(0, 10):
        devices.append(str(uuid.uuid4()))
    # for y in range(0,20):    # For each device, produce 20 events.
    event_data_batch = producer.create_batch() # Create a batch. You will add events to the batch later.
    for dev in devices:
        print('Getting data...')
        # Create a dummy reading.
        dummy_data = requests.get(api_url_openwheather)
        dummy_list = dummy_data.json()
        s = json.dumps(dummy_list) # Convert the reading into a JSON string.
        event_data_batch.add(EventData(s)) # Add event data to the batch.
    producer.send_batch(event_data_batch) # Send the batch of events to the event hub.
    print("Sending data...")
    # Close the producer.
    producer.close()
```

KUVA 2. API-data Event Hubiin

Koodin lopussa käytetään Pandas-kirjastoa ja Pyarrowia moottorina saamaan Python lukemaan dataa Parquet-tiedostosta (kuva 3).

```python
        if dummy_list == 0:
            break

    def read_parquet():
        parquet_file = '/c/bts-monitoring/.vscode/-1714053307_2499855d99674f0e8a85cc6ceaf66a10_1.parquet'
        pd.read_parquet(parquet_file, engine='pyarrow')
```

KUVA 3. Parquet-tiedoston luku

Pyarrow on Apache Arrowin Pythonilla toimiva kirjasto. Sitä käytetään big datan prosessointiin ja liikuttamiseen. (4.) Pyarrowia käytettiin opinnäytetyössä tiedostotyyppien muuntamisessa toisiksi tiedostotyypeiksi yhdessä Pandas-kirjaston avulla. Parquet-tiedosto ladattiin Azuressa olevasta kontista .vscode-kansioon Visual Studio Codeen, jotta testisäädataa voitaisiin testata. Lataus tapahtui valitsemalla kansio Azuren blob-kontista ja painamalla lataa-nappia vasemmasta yläkulmasta. Kun data oli ladattu tietokoneelle, se siirrettiin projektikansioon ja sen sisällä .vscode-nimiseen kansioon. Kansion nimi on .vscode, koska sinne tulevat kaikki testausvaiheessa käytettävät tiedostot.

2.3 Datan visualisointi ja uudelleen järjestäminen

Opinnäytetyössä luettiin Parquet-tiedostot käyttämällä Pythonin Pandas-kirjastoa ja tehtiin visualisointi kyseiselle datalle offlinessa. Koodi juoksutettiin läpi. Näin saatiin tiedoston sisältö luettavaan muotoon.

Aluksi datan mittauksessa asetetut koordinaatit olivat väärin (kuva 4). Tämän takia jouduttiin vaihtamaan koordinaatit datan mittauksessa käytettyyn koodiin, jotta saatiin käyttöön mahdollisimman hyvää dataa.

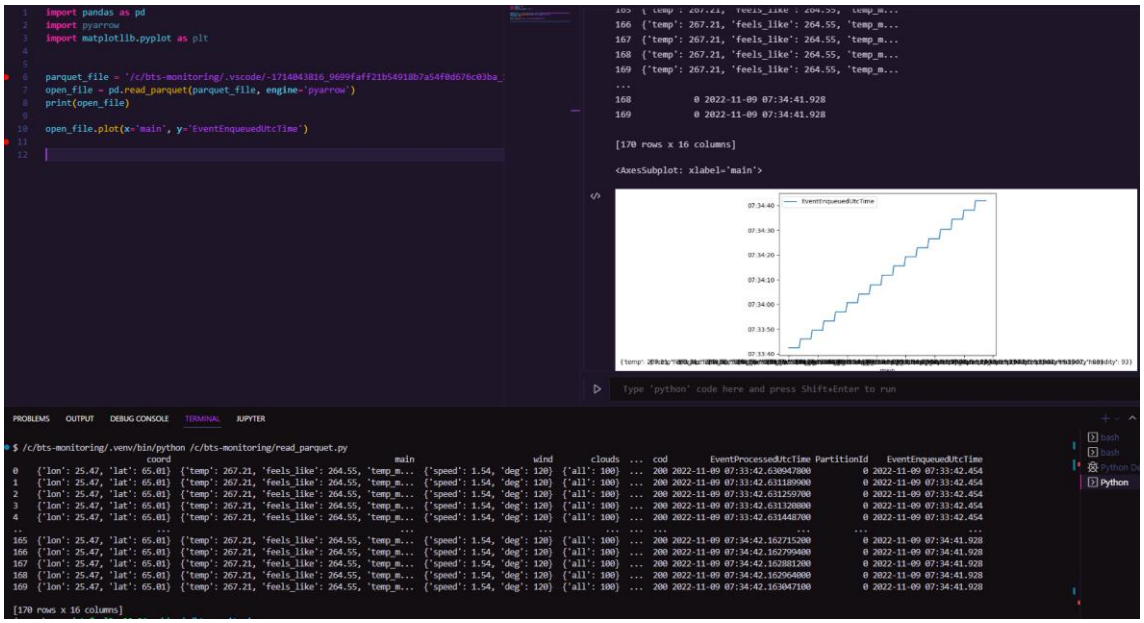
```
$ /c/bts-monitoring/.venv/bin/python /c/bts-monitoring/read_parquet.py
   coord  main  wind clouds   sys weather  cod          message         EventProcessedUtcTime  PartitionId    EventEnqueuedUtcTime  base  visibility  dt  timezone  id  name
0   None  None  None   None  None    None  400  wrong longitude 2022-11-03 11:50:57.887610100           0 2022-11-03 11:50:57.742  None         NaN NaN       NaN NaN  None
1   None  None  None   None  None    None  400  wrong longitude 2022-11-03 11:50:57.887690600           0 2022-11-03 11:50:57.742  None         NaN NaN       NaN NaN  None
2   None  None  None   None  None    None  400  wrong longitude 2022-11-03 11:50:57.887707300           0 2022-11-03 11:50:57.742  None         NaN NaN       NaN NaN  None
3   None  None  None   None  None    None  400  wrong longitude 2022-11-03 11:50:57.887719600           0 2022-11-03 11:50:57.742  None         NaN NaN       NaN NaN  None
4   None  None  None   None  None    None  400  wrong longitude 2022-11-03 11:50:57.887731600           0 2022-11-03 11:50:57.742  None         NaN NaN       NaN NaN  None
..   ...   ...   ...    ...   ...     ...  ...              ...                            ...         ...                   ...   ...         ...  ..       ...  ..   ...
85  None  None  None   None  None    None  400  wrong longitude 2022-11-03 11:51:38.107076000           0 2022-11-03 11:51:38.014  None         NaN NaN       NaN NaN  None
86  None  None  None   None  None    None  400  wrong longitude 2022-11-03 11:51:38.107088400           0 2022-11-03 11:51:38.014  None         NaN NaN       NaN NaN  None
87  None  None  None   None  None    None  400  wrong longitude 2022-11-03 11:51:38.107100300           0 2022-11-03 11:51:38.014  None         NaN NaN       NaN NaN  None
88  None  None  None   None  None    None  400  wrong longitude 2022-11-03 11:51:38.107112000           0 2022-11-03 11:51:38.014  None         NaN NaN       NaN NaN  None
89  None  None  None   None  None    None  400  wrong longitude 2022-11-03 11:51:38.107124000           0 2022-11-03 11:51:38.014  None         NaN NaN       NaN NaN  None

[90 rows x 17 columns]
```

KUVA 4. Epäonnistunut Parquet-tiedosto.

Datan keräämisen jälkeen perehdyttiin datan visualisointiin käyttämällä apuna Python-kirjastoja Pandas ja Matplotlib. Matplotlib on visualisointiin tarkoitettu Python-kirjasto. Sitä voi käyttää esimerkiksi animoitujen ja interaktiivisten visualisointien tekemiseen. (5.)

Datan visualisoinnissa käytettiin Matplotlib-kirjastoa visualisoimaan data Parquet-tiedostosta. Sitä varten Parquet-tiedosto täytyi muuttaa tietorakenteeksi, jotta Matplotlib pystyi visualisoimaan mitatun datan. Datan visualisoinnin jälkeen data täytyi järjestää uudelleen (Kuva 5).



KUVA 5. Datan visualisointi Matplotlib-kirjastolla.

Datan uudelleen järjestämisessä on kaksi tunnetumpaa metodia: Upsampling ja Downsampling. Nämä kumpikin on mahdollista tehdä Pythonilla käyttäen apuna Pandas-kirjastoa. Upsampling purkaa datan korkeammalta asteelta pienemmälle asteelle aikasarjan mukaan. Esimerkiksi Upsampling voi purkaa aikasarjadatassa vuoden kuukausiksi, kuukauden päiviksi ja niin edelleen. (6.)

Downsampling on käytännössä sama asia kuin Upsampling, mutta purkaa aikasarjadatan pienemmältä asteelta suuremmalle asteelle. Esimerkiksi Downsampling voi purkaa aikasarjadatassa päivät viikoiksi, viikot kuukausiksi ja niin edelleen. (7.) Downsamplingia ja Upsamplingia käyttämällä data saatiin järjestettyä uudelleen. Opinnäytetyössä testattiin kumpaakin menetelmää.

3 DIGITAALINEN KAKSONEN ELI DIGITAL TWIN

Opinnäytetyön aikana tarkasteltiin, voiko Digital Twin eli digitaalinen kaksonen toimia Event Hubin sijasta. Ajatuksena oli, että jos se toimii yrityksen haluamalla tavalla, se voidaan ottaa käyttöön myöhemmin opinnäytetyön projektissa. Tässä opinnäytetyössä testataan, voidaanko digitaalinen kaksonen ottaa käyttöön. Opinnäytetyössä vertaillaan myös digitaalista kaksosta ja Azure Event Hub -työkalua.

3.1 Yleistä digitaalisista kaksosista

Digitaalinen kaksonen on virtuaalinen malli, joka on suunniteltu mahdollisimman tarkasti vastaamaan fyysistä objektia. Tässä opinnäytetyössä oli ajatuksena mallintaa 5G-radio käyttäen digitaalista kaksosta. Dataa saatuaan digitaalista kaksosta voi käyttää simuloimiseen, opettelemaan suorituksen aikana tapahtuvia ongelmia ja tekemään mahdollisia parannuksia. (8.)

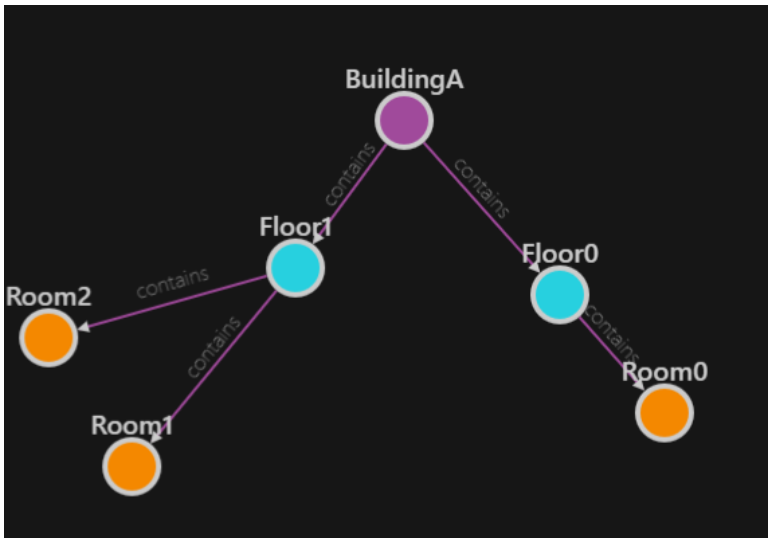
Vaikka simuloiminen ja digitaalinen kaksonen kuulostavatkin hyvin samanlaisilta, on niissä kuitenkin eroja. Simulointi käy läpi vain yhtä prosessia, kun taas digitaalinen kaksonen voi käydä läpi useampia simulaatioita yhtä aikaa. Digitaalinen kaksonen on myös suunniteltu käyttämään apunaan suoratoistodataa. Tämä mahdollistaa useampien ongelmien läpikäymisen. (8.)

3.2 Digital Twin Azuressa

Azuren oma digitaalinen kaksonen on nimeltään Azure Digital Twins, jota voi käyttää Azure-portaalin kautta. Azure Digital Twins Explorerin voi ottaa käyttöön, kun Azureen on luotu oma digitaalinen kaksonen -työkalu. Azure Digital Twins Explorerilla tehdään mallinnus fyysiselle objektille. (9.) Azure Digital Twin Explorer käyttää omaa JSON-tyyppistä kieltä mallien määrittämiseen. DTDL eli Digital Twin Definition Language perustuu JSON-LD-kieleen. Kun tehdään DTDL-tiedostoja, on tärkeä muistaa, että nämä täytyy tallentaa käyttäen .json-päätettä, koska DTDL seuraa JSON-syntaksia. Visual Studio Codella on oma lisäosa DTDLlle, jota voi käyttää luomaan valmiin DTDL-pohjan itse valitsemallaan nimellä. (10.)

```
{
  "@context": "dtmi:dtdl:context;2",
  "@id": "dtmi:com:example:bts_monitoring;1",
  "@type": "Interface",
  "displayName": "bts_monitoring",
  "contents": [
    {
      "@type": "Telemetry",
      "name": "temperature",
      "schema": "double"
    },
    {
      "@type": "Property",
      "name": "deviceStatus",
      "schema": "string"
    },
    {
      "@type": "Command",
      "name": "reboot",
      "request": {
        "name": "delay",
        "schema": "integer"
      }
    }
  ]
}
```


KUVA 6. DTDL-tiedosto tehty Visual Studio Code -lisäosalla.



KUVA 7. Digitaalinen kaksonen -malli Digital Twin Explorerissa.

Digital Twin Explorer tarvitsee myös DTDL:n lisäksi käyttöönsä .xlsx-tiedoston, jossa määritellään datan sisältö. Tämä .xlsx-tiedosto täytyy liittää Digital Twin Exploreriin, jolloin se muodostaa osan digitaalinen kaksonen -mallista (kuva 7). (11.) Loppuosa mallista muodostuu, kun käyttäjä itse liittää kaksosia Digital Twin Exploreriin liitettyjen DTDL-tiedostojen avulla. (Kuva 6.)

Digital Twin Explorer käyttää SQL-tyyppistä Azure Digital Twins Query Language -kieltä (kuva 8.). Sitä voi käyttää hakemaan digitaalinen kaksonen -mallista yhtä tiettyä kaksosta. Tämä on hyvä silloin, kun halutaan etsiä isosta mallista yksi tietty kaksonen esimerkiksi id:n perusteella.

```
SELECT * FROM digitaltwins
```

KUVA 8. Esimerkki Azure Digital Twins Query Language -kielestä.

3.3 Digital Twin opinnäytetyössä

Lisäselvityksen aikana huomattiin, että digitaalisen kaksosen voi liittää myös suoraan Azure Event Hubiin. Pystytään siis lähettämään dataa digitaalisella kaksosella tehdystä mallista Azure Event Hubiin ja sitä kautta blob-konttiin. Opinnäytetyön jälkeisessä kehitystyössä digitaalinen kaksonen voi helpottaa simulointivaihetta, koska digitaalinen kaksonen pystyy käsittelemään suoratoisto-dataa ja tekemään useampia simulointeja yhtä aikaa. Digitaalista kaksosta käyttämällä jatkoprojektissa voidaan mallintaa esimerkiksi useampi 5G-radio ja mitata niistä tulevaa dataa.

Opinnäytetyön loppuvaiheessa sovittiin, että digitaalinen kaksonen voisi tulla myöhemmin käyttöön, mikäli projekti jatkuu yrityksessä vielä myöhemmin. Ajatus oli, että projekti tulee jatkumaan vielä pitkään ja tämän voi ottaa käyttöön myöhemminkin. Osa tehtävänantoa oli tutustua digitaalinen kaksonen -työkaluun, joten tämä osio toimii esiselvityksenä, miten digitaalisia kaksosia voisi käyttää tulevaisuudessa.

4 AZURE COSMOSDB JA AIKASARJADATA

4.1 Aikasarjadata

Aikasarjadata on ajan mukaan järjestetty arvojoukko. Aikasarjadatan keskeinen ominaisuus on järjestää tapahtumat samassa järjestyksessä, jossa nämä tapahtuvat ja saapuvat käsittelyyn. Aikasarjadataa voi käyttää tarkkailemaan vanhaa dataa, jotta voi tarkastella muutoksia datassa. Sillä on myös mahdollista katsoa dataa eteenpäin, mikä mahdollistaa tulevaisuuteen ennustamisen datan avulla. (12.)

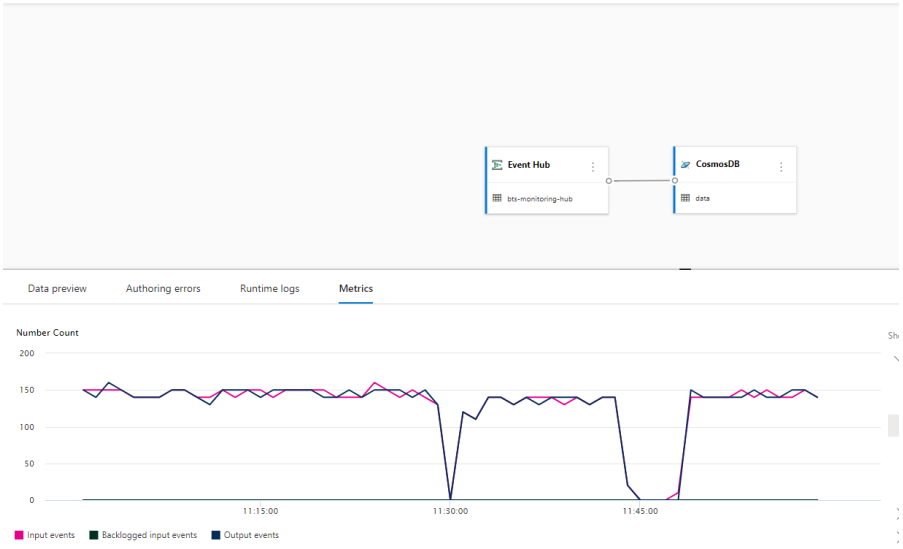
4.2 Azure Cosmos DB

Tässä luvussa selvitetään, miten aikasarjadata toimii Azure Cosmos DB -tietokantapalvelun kanssa. Opinnäytetyössä käytetään tietokantapalveluna Azure Cosmos DB:tä. Azure Cosmos DB on NoSQL-tyyppinen tietokantapalvelu. Toisin kuin moni muu tietokantapalvelu, Azure Cosmos DB ei käytä SQL-kieltä. (13.) Opinnäytetyössä käytetään Azure Cosmos DB:tä MongoDB:n kanssa. Sen käytöstä sovittiin, sillä yritys, jolle opinnäytetyötä tehtiin, käyttää muissakin projekteissaan MongoDB -tietokantoja. Tämä mahdollistaa tarvittaessa hyvät yhdistysmahdollisuudet muihin projekteihin. Azure Cosmos DB liitetään MongoDB:hen käyttäen Azure Cosmos DB for MongoDB -työkalua, jotta saataisiin käytettyä MongoDB:n työkaluja ja liitettyä opinnäytetyönä tehty projekti helposti Azureen. Hyvä työkalu, jota voi käyttää opinnäytetyön aikana on esimerkiksi ”Real time analytics (HTAP) at any scaly”. MongoDB:n kyseinen työkalu pystyy tekemään analytiikan datalle suoratoistona ilman, että tämä vaikuttaa tietokantaan.

Azuren kanssa Azure Cosmos DB:tä voi käyttää esimerkiksi Pythonilla, Javalla, Node.js:llä ja muilla vaihtoehtoisilla ohjelmointityökaluilla. Tässä opinnäytetyössä käytetään Pythonia, koska se on tutuin kieli. Aluksi tehtiin Pythonilla koodi, joka lähettää tietokantaan kovakoodattua dataa. (13.) Tämä koodi oli tehty alustavaa testausta varten, jotta nähdään, miten tietokantaan lähettäminen toimii. Tämän jälkeen tavoitteena oli saada Event Hub yhdistettyä Azure Cosmos DB -tietokantaan.

4.3 Azure Cosmos DB for NoSQL

Myöhemmin opinnäytetyössä tuli ongelmia liittää MongoDB Event Hubiin. Ongelmana oli, että data ei mennyt läpi Event Hubin kautta tietokantaan. Tämän vuoksi päätettiin testata toista Azure Cosmos DB -palvelua. Opinnäytetyössä otettiin käyttöön Azure Cosmos DB for NoSQL -palvelu. Kun tietokanta oli rakennettu Azure-portaalissa, se oli mahdollista liittää ulostulona Event Hubiin.



KUVA 9. Event Hub ja CosmosDB yhdistettynä.

Event Hubiin lähetettiin samaa säädataa testinä (kuva 9.). Data tuli tietokantaan JSON-muodossa, jossa näkyivät lämpötila, ilmankosteus, koordinaatit, näkyvyys, pilvien määrä ja millainen sää on. OpenWheatherMap-API, jonka kautta dataa kerättiin, muodosti JSON-tiedoston. (Kuva 10.)

```
{
    "coord": {
        "lon": 25.4699,
        "lat": 65.0116
    },
    "weather": [
        {
            "id": 600,
            "main": "Snow",
            "description": "light snow",
            "icon": "13d"
        }
    ],
    "base": "stations",
    "main": {
        "temp": 268.83,
        "feels_like": 262.12,
        "temp_min": 268.83,
        "temp_max": 269.21,
        "pressure": 1007,
        "humidity": 100
    },
    "visibility": 10000,
    "wind": {
        "speed": 6.17,
        "deg": 90
    },
    "snow": {
        "1h": 0.15
    },
    "clouds": {
        "all": 20
    },
    "dt": 1670493040,
    "sys": {
        "type": 2,
        "id": 2003583,
        "country": "FI",
```


KUVA 10. Oulussa kerättyä säädataa 8.12.2022.

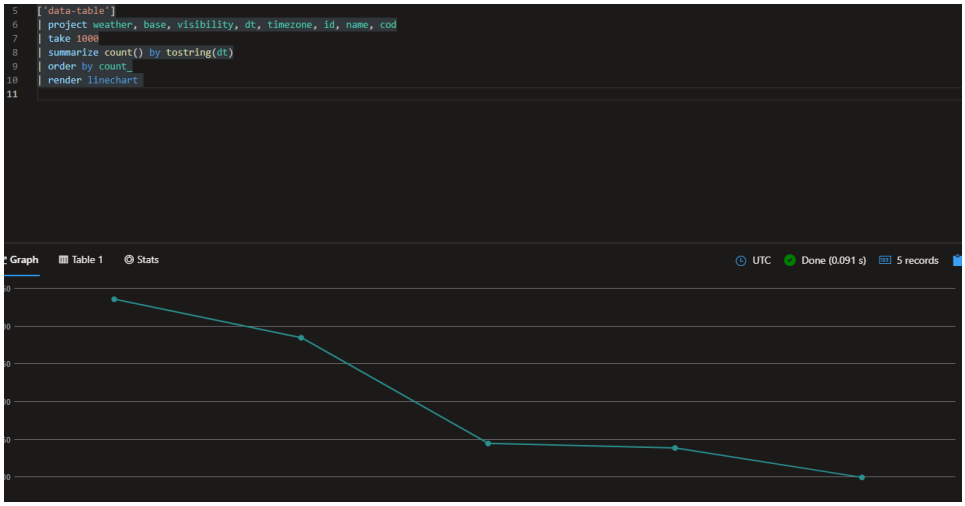
5 AZURE DATA EXPLORER

5.1 Azure Data Explorer

Azure Data Explorer on big datan käsittelyyn ja visualisointiin tehty ohjelma. Sitä on käytetty monissa tunnetuissa Microsoftin järjestelmissä. Näitä järjestelmiä ovat esimerkiksi Windows, Skype, Xbox, LinkedIn, Visual Studio, SQL Server, Bing, Office ohjelmat, Azure ja Power BI. Azure Data Explorer käyttää omaa ohjelmointikieltä: Kusto Query Language eli KQL. KQL on avoimen lähdekoodin kieli, jota täytyy käyttää Azure Data Explorerin kanssa. KQL on samantyylinen kieli kuin SQL. (14.)

5.2 Azure Data Explorer opinnäytetyössä

Opinnäytetyössä Azure Data Exploreria käytetään monitoroimaan data, joka tuodaan Azure Data Exploreriin Event Hubin kautta. Aluksi tehtiin Azure Data Explorerin sisällä oma tietokanta ja tietokannan sisälle pöytä. Tämän jälkeen tehtiin liitäntä Event Hubiin ja lähetettiin dataa Event Hubin kautta tietokantaan. Datan lähettäminen Event Hubiin toimi samalla tavalla kuin edellisissä vaiheissa. Datan käsittely ja visualisointi voitiin tehdä Kusto Query Languagella, kun data oli saatu siirrettyä Azure Data Exploreriin. API:n kautta haettua säädataa käytettiin testaamaan Kusto Query Languagea ja Azure Data Exploreria. (14.)



KUVA 11. Säädatan visualisointi KQL:n avulla.

6 DATAGENERAATTORI JA VIRTUAALINEN KONE

Opinnäytetyössä täytyi korvata API, jota käytettiin alkuvaiheessa esittämään suoratoistodataa. Sen tilalle tehtiin oma datageneraattori Pythonilla. API korvattiin opinnäytetyössä, sillä opinnäytetyössä haluttiin käyttää datamallin mukaan tehtyä dataa. API oli testauksessa mukana, koska Azure Event Hub haluttiin saada toimimaan suoratoistodatan kanssa.
Tämä datamalli luo sattumanvaraista dataa JSON-tiedostoon tehdylle datamallille, joka kuvantaa radiodataa, jota halutaan tällä opinnäytetyön projektilla myöhemmin tarkkailla.

Opinnäytetyössä täytyy myös siirtää koodi Virtual Machineen eli virtuaaliseen koneeseen, jotta koodi saadaan juoksemaan koko ajan Azuren sisällä. Tämä mahdollistaa sen, että ohjelman toisto ei katkea kesken kaiken, mikäli tietokone sammuu tai menee lepotilaan. Tämä mahdollistaa pitkien aikojen datan mittauksen.

6.1 Datageneraattorin teko

Datageneraattori tehtiin muodostamaan sattumanvaraista dataa JSON-datamalliin. Datageneraattori simuloi 5G-radiosta kerättyä bts-dataa. Datamalliin on kirjattu id, testlineId, resultId, timestamp, moduleName, serialNumber, featured, btsConfigBlockId, platformName, bracketSlot, moduleType, memory, cpu, temperature ja power. ModuleTypen määrittämiseen tehtiin lista, jonka sisällä on neljä moduuli tyyppiä: RU, BB, SM ja SM+BB. Opinnäytetyössä luodaan sattumanvaraista dataa, joten käytimme Pythonin Random-kirjastoa apuna valitsemaan aina jonkun näistä moduuleista. Muiden kohtien arvot määriteltiin käyttämällä Faker- ja Random-kirjastoa.

Datan muodostamiseen käytettiin for-silmukkaa (loop) while-silmukan sisällä. Tämä mahdollisti datan muuttumisen JSON-tiedostossa ja suoratoistodatan mittaamisen. Näillä keinoilla koodi saatiin muodostamaan sattumanvaraista dataa ja se voitiin lähettää Event Hubiin. Datan lähettämiseen käytettiin apuna aiemmin käytettyä koodia (kuva 2). Ainoastaan datan lähde vaihdettiin koodiin. Event Hubista data siirrettiin Azure Data Explorer -työkaluun käyttäen Stream Analytics Job -työkalua (kuva 9). Tässä tapauksessa valitsimme toiseksi ulostuloksi Azure Data Explorerin, jotta voimme käyttää sitä datan käsittelyyn ja visualisointiin. Visualisointi Data Explorer -työkalussa tehtiin käyttämällä datan käsittelyyn KQL-kieltä. Data Explorer -työkalulla tehtiin neljä kuvaajaa: power

suhteessa aikaan, cpu suhteessa aikaan, temperature suhteessa aikaan ja memory suhteessa aikaan. KQL-kielen avulla datasta valittiin aina oma kolumni, josta haluttiin tehdä kuvaaja ja aikaleima. Data järjestettiin aikaleiman mukaan ja tämän jälkeen piirrettiin kuvaaja jokaisesta neljästä arvosta, jotka oli vapaasti valittu mitatusta datasta.

6.2 Generoidun datan käyttö Data Explorerissa ja virtuaalisessa koneessa

Datageneraattori siirrettiin virtuaaliseen koneeseen, kun datageneraattori oli saatu toimimaan paikallisesti tietokoneella. Tämä virtuaalinen kone tehtiin käyttämällä Azuren omaa Virtual Machine työkalua. Tämä mahdollisti sen, että Azure itse pitää ylläpitää virtuaalista konetta. Opinnäytetyössä tehtiin virtuaalinen kone, datanmittauskoodia varten. Tämä virtuaalinen kone tehtiin Azure-pilvipalvelupohjalla. Virtuaalinen kone tehtiin Linux-pohjalle, joten yhdistäminen täytyi tehdä SSH-yhteydellä. SSH-yhteyden muodostamisen jälkeen tehtiin opinnäytetyön projektille oma kansio virtuaaliseen koneeseen. Tähän käytettiin terminaalissa Nano-ohjelmistoympäristöä. Koodi siirrettiin Nano-ohjelmistoympäristöön suoraan kopioimalla ja liittämällä. Nanolla täytyi tehdä vielä toinen, .env-niminen, tiedosto, johon oli laitettu API-avaimet ja Conection stringit, joita tarvittiin Azuren yhdistämiseen. Koodi juoksutettiin läpi käyttämällä terminaalissa komentoa ”python3 send.py”. Tämä komento aloitti datan mittauksen käyttäen apunaan virtuaalista konetta.

Docker otettiin mukaan opinnäytetyöhön, kun koodi oli saatu toimimaan Nanolla virtuaalisessa koneessa. Docker on alusta, joka käyttää virtualisointia konttien toimittamiseen. Nämä kontit ovat ohjelmistopaketteja. (1.) Dockerin käyttöönotto vaatii Docker-tiedoston, jossa määriteltiin kaikki, mitä koodin toimimista varten täytyi asentaa. Docker image pystyttiin rakentamaan terminaalissa yhdellä komennolla, kun Docker-tiedosto oli luotu. Tehtäessä muutoksia koodiin täytyy rakentaa Docker image ”sudo docker build -t <tiedoston nimi>.” komennolla uudestaan. Kun Docker image oli rakennettu, pystyi Dockerin käynnistämään komennolla ” sudo docker run <tiedoston nimi>”. Docker mahdollisti koodin toimimisen virtuaalisessa koneessa ja datan lähettämisen Data Exploreriin koko ajan.

Azure Data Explorerissa visualisoitiin data käyttämällä apuna KQL-kieltä (kuva 11.). Azure Data Exploreriin pystyi tekemään koontinäytön, jossa voitiin näyttää kerralla kaikki visualisoinnit, mitä haluttiin nähdä. Koontinäytöllä oli opinnäytetyön lopussa neljä kuvaajaa: cpu, memory, power ja

temperature. Näissä näkyi datageneraattorilla luotu sattumanvarainen data suhteessa aikaan vapaasti valituista kolumneista.

7 DATAN LUOMINEN JA VISUALISOINTI ILMAN PILVIPALVELUA

Tämä opinnäytetyö olisi myös voitu tehdä ilman pilvipalvelua. Tässä luvussa kootaan vaihtoehtoinen ratkaisu opinnäytetyön toteutukselle ilman pilvipalvelua.

Mikäli opinnäytetyö olisi haluttu tehdä ilman pilvipalvelua, olisi tämän opinnäytetyön voinut tehdä esimerkiksi juoksuttamalla koodin täysin paikallisesti tietokoneella. Silloin Event Hubin käytön olisi voinut jättää pois, koska dataa ei tarvitsisi lähettää Azure-pilvipalveluun. Silloin datan olisi voinut suoraan visualisoida samalla, kun sitä luodaan datageneraattorin avulla.

Visualisointiin voi käyttää sitä varten luotuja Python-kirjastoja, kuten Matplotlib. Tähän tutustuttiin jo opinnäytetyön alkuvaiheessa. Datan tallentaminen olisi voinut tapahtua myös itse tehtyyn SQL-tietokantaan tai johonkin valmiiseen vaihtoehtoiseen tietokantaan, kuten MongoDB.

Datan luominen voitaisiin tehdä samalla tavalla kuin opinnäytetyössä, jossa käytettiin Random- ja Faker-kirjastoja luomaan sattumanvaraista dataa JSON-datamalliin. Tämä olisi vaihtoehtoinen tapa tehdä opinnäytetyö ilman Azure-pilvipalvelupohjaa.

8 POHDINTA

Tämän työn tavoitteena oli tehdä eräälle yritykselle suoratoistodatan visualisointi Azure-pilvipalveluun. Tavoitteena oli myös tehdä esiselvitys digitaalisille kaksosille.

Opinnäytetyö suoritettiin onnistuneesti siihen vaiheeseen saakka, johon yritys sen halusi. Opinnäytetyössä saatiin tehtyä datageneraattori, joka loi sattumanvaraista dataa valmiiksi määriteltyyn datamalliin. Datageneraattorilla luotu data lähetettiin Azure-pilvipalvelupohjan Azure Event Hub -työkaluun ja sitä kautta Azure Data Exploreriin. Azure Data Explorer -työkalussa datageneraattorilla luotu data käsiteltiin ja visualisoitiin. Opinnäytetyössä käytettiin myös Dockeria ja virtuaalista konetta sekä tutustuttiin digitaalisiin kaksosiin.

Opinnäytetyöhön kuului datan keräämistä, luomista, käsittelyä ja visualisointia. Opinnäytetyössä käytettiin Python-ohjelmointikieltä tekemään datageneraattori ja lähettämään dataa Azure Event Hub -työkaluun. Opinnäytetyössä käytettiin myös kahta uutta kieltä, Digital Twin Definition Language (DTDL) ja Kusto Query Lanquage (KQL). Näitä kieliä käyttivät digitaalinen kaksonen ja Azure Data Explorer. Opinnäytetyössä käytettiin myös JSON:lla tehtyä datamallia pohjana datageneraattorille.

Lopputuloksena opinnäytetyössä luotiin 5G-datamallin mukaan tehty datageneraattori, joka lähetti dataa Azure Event Hub -työkaluun. Azure Event Hub -työkalusta data lähetettiin Azure Data Explorer -työkaluun, jossa data käsiteltiin ja visualisoitiin koontinäytölle. Koontinäytöllä data näkyy useampana kuvaajana.

Opinnäytetyön aikana opittiin paljon datan käsittelystä ja visualisoinnista. Opinnäytetyössä tutustuttiin Azure-pilvipalvelun eri toimintoihin ja työkaluihin. Tällaisia työkaluja ovat Azure Event Hub, Azure CosmosDb, Azure Digital Twin ja Azure Data Explorer.

Tässä opinnäytetyössä selvitettiin myös, miten digitaalinen kaksonen voisi toimia opinnäytetyössä. Opinnäytetyössä ei vielä käytetty digitaalista kaksosta, mutta opinnäytetyön projektin jatkuessa sille on jo tehty esiselvitys, mikä helpottaa jatkoa.

LÄHTEET:

1. Docker docs 2023. Overview. Hakupäivä 3.1.2023. https://docs.docker.com/get-started/

2. Gartner 2023. Base Transceiver Station (BTS). Hakupäivä 18.1.2023. shttps://www.gartner.com/en/information-technology/glossary/bts-base-transceiver-station

3. Pandas. Package overview. Hakupäivä 22.11.2022. https://pandas.pydata.org/docs/getting_started/overview.html#package-overview

4. Apache Arrow 2022. Apache Arrow, Hakupäivä 30.12.2022 https://arrow.apache.org/docs/index.html

5. Matplotlib 2023. Matplotlib Visualization with Python. Hakupäivä 2.1.2023. https://matplotlib.org/

6. Microsoft. What is Azure? Hakupäivä 16.11.2022. https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure/

7. Geeks for geeks. How to Resample Time Series Data in Python? Jssuriyakumar 19.12.2021. Hakupäivä 17.11.2022. https://www.geeksforgeeks.org/how-to-resample-time-series-data-in-python/?ref=gcse

8. IBM. What is a digital twin? Aashish Mehra, Markets and Markets. Hakupäivä 23.11.2022. https://www.ibm.com/topics/what-is-a-digital-twin

9. Microsoft 2022. Azure Digital Twins Explorer (preview). Hakupäivä 24.11.2022. https://learn.microsoft.com/en-us/azure/digital-twins/concepts-azure-digital-twins-explorer

10. Microsoft 2022. Learn about twin models and how to define them in Azure Digital Twins. Hakupäivä 24.11.2022. https://learn.microsoft.com/en-us/azure/digital-twins/concepts-models#digital-twin-definition-language-dtdl-for-models

11. Microsoft 2022. Quickstart - Get started with a sample scenario in Azure Digital Twins Explorer. Hakupäivä 25.11.2022. https://learn.microsoft.com/en-us/azure/digital-twins/quickstart-azure-digital-twins-explorer

12. Microsoft. Time series data. Hakupäivä 30.11.2022.https://learn.microsoft.com/en-us/azure/architecture/data-guide/scenarios/time-series

13. Microsoft 2022. Welcome to Azure Cosmos DB. Hakupäivä 30.11.2022. https://learn.microsoft.com/en-us/azure/cosmos-db/introduction

14. Microsoft 2022. What is Data Explorer? Hakupäivä 16.12.2022. https://learn.microsoft.com/en-us/azure/data-explorer/data-explorer-overview

15. Docker docs 2023. Overview. Hakupäivä 3.1.2023. https://docs.docker.com/get-started/

16. Microsoft 2022. Azure Event Hubs client library for Python - version 5.10.1. Hakupäivä 16.11.2022.https://learn.microsoft.com/en-us/python/api/overview/azure/eventhub-readme?source=recommendations&view=azure-python

17. Microsoft. Capture events through Azure Event Hubs in Azure Blob Storage or Azure Data Lake StorageHakupäivä 17.11.2022. https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-capture-overview?WT.mc_id=Portal-Microsoft_Azure_EventHub

18. Geeks for geeks. How to Resample Time Series Data in Python? Jssuriyakumar 19.12.2021. Hakupäivä 17.11.2022. https://www.geeksforgeeks.org/how-to-resample-time-series-data-in-python/?ref=gcse

19. Microsoft 2022. What is Axure Digital Twins? Hakupäivä 23.11.2022. https://learn.microsoft.com/en-us/azure/digital-twins/overview

20. Microsoft. Azure Digital Twins. Hakupäivä 24.11.2022. https://azure.microsoft.com/en-us/products/digital-twins/#overview

21. Microsoft Azure 2022. What is a virtual machine (VM)? Hakupäivä 13.12.2022. https://azure.microsoft.com/en-in/resources/cloud-computing-dictionary/what-is-a-virtual-machine/#layout-container-uid0c75

22. Microsoft 2022. Kusto Query Language (KQL) overview. Hakupäivä 16.12.2022 https://learn.microsoft.com/en-us/azure/data-explorer/kusto/query/

23. Tableau 2023. What Is Data Visualization? Definition, Examples, And Learning Resources. Hakupäivä 2.1.2023. https://www.tableau.com/learn/articles/data-visualization