**Harjoitustyö**

Big Data

Tapani Alastalo

M1475

**Sisältö**

[1 Pipelineketju 2](#_Toc56628844)

[2 Datalähde 2](#_Toc56628845)

[3 Highcharts komponentti 4](#_Toc56628846)

[4 Pohdinta 6](#_Toc56628847)

# Pipelineketju

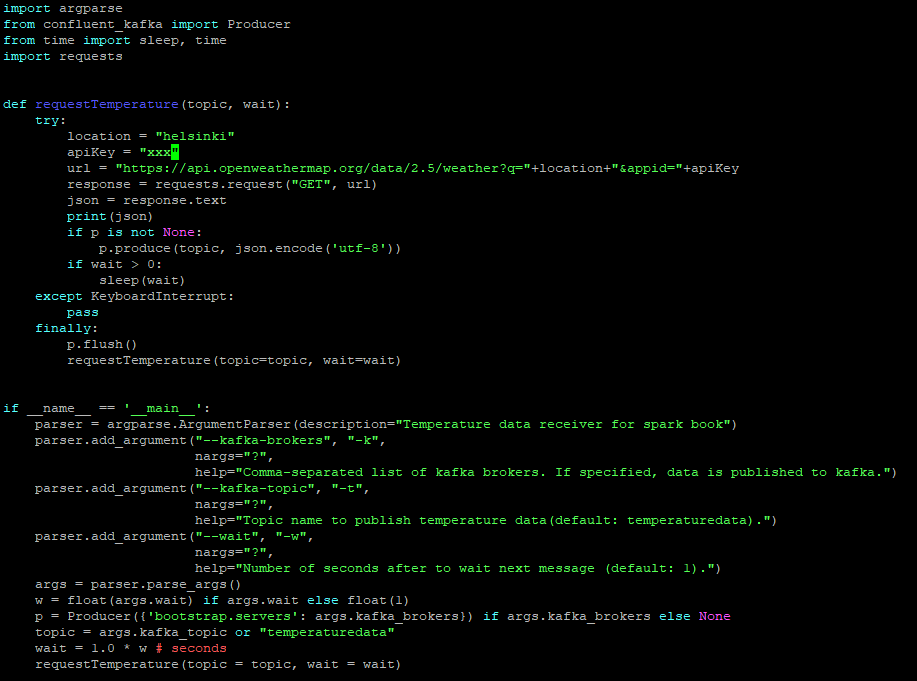
Pipelineketju koostui OpenWeatherin tarjoamasta API-rajapinnasta, mitä kautta data haettiin Apache Kafkalle, mistä se välitettii Apache Sparkille, minkä kautta se tallennettiin MongoDB:hen. Datan visualisointi tapahtui Highcharts kirjastolla, mikä haki datan MongoDB:ltä Node.js serverin kautta.

Pipelineketjun pohjana käytin harjoitustehtävissä rakennettu ketjua. Ketjun käynnistys alkoi laittamalla Brokerit ja Zookeeper päälle. Nämä piti käynnistää kerran uudestaan, jotta kaikki 3 Brokeria nousi pystyyn. Tämän varmistin netstat -an | grep LISTEN -komennolla.

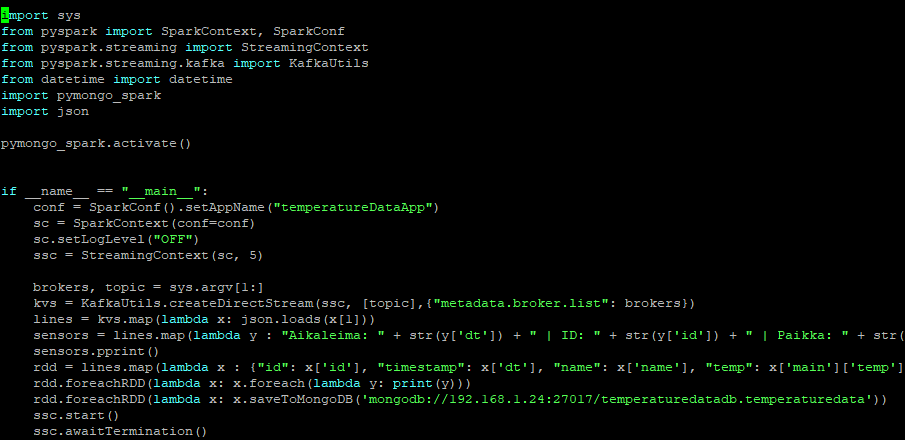
# Datalähde

Säätiedot OpenWeather palvelun ilmaisesta api rajapinnasta. Käytettävänä säätietona käytettiin Helsingin sääaseman havaintoja. Ei ole tiedossa, minkä tahon kautta OpenWeather saa säätietonsa.

Kyseinen palvelu tarjoaa ilmaisella versiolla 60 pyyntöä minuutissa. Palvelusta voi kutsua lähituntien ennusteita, viimeisimpiä havaintoja ja historiatietoja. Tässä palvelussa säätietoja pystyi hakemaan kaupungin nimellä ja sen takia valitsin tämän palvelun.

Kuva 1. Datan haku OpenWeatherin API:sta.

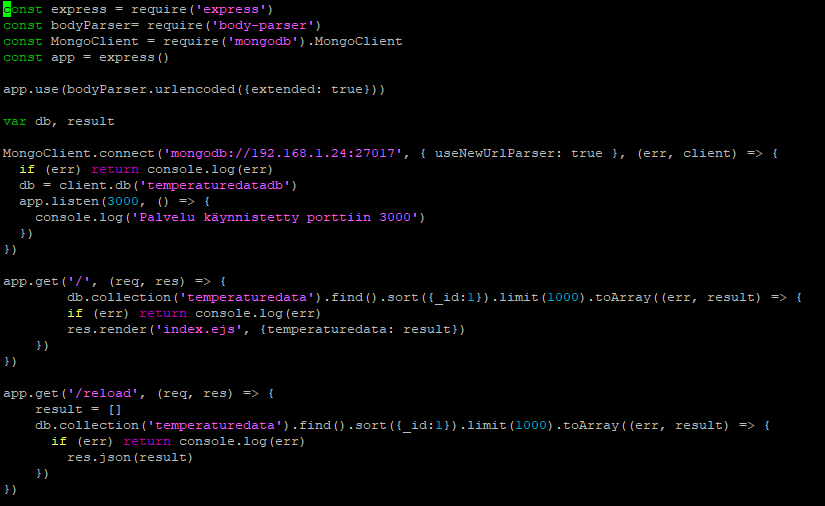
Datalähteen API rajapinta kyselyt tein Pythonin Request kirjaston avulla. Kyseinen palvelu tarjoaa vastaukset json muotoisena, mistä otin talteen haluamani lämpötilatiedot. Mukana tuli myös muunmuassa paikka-, kosteus- ja tuulitietoa. Nämä jätin tallentamatta.

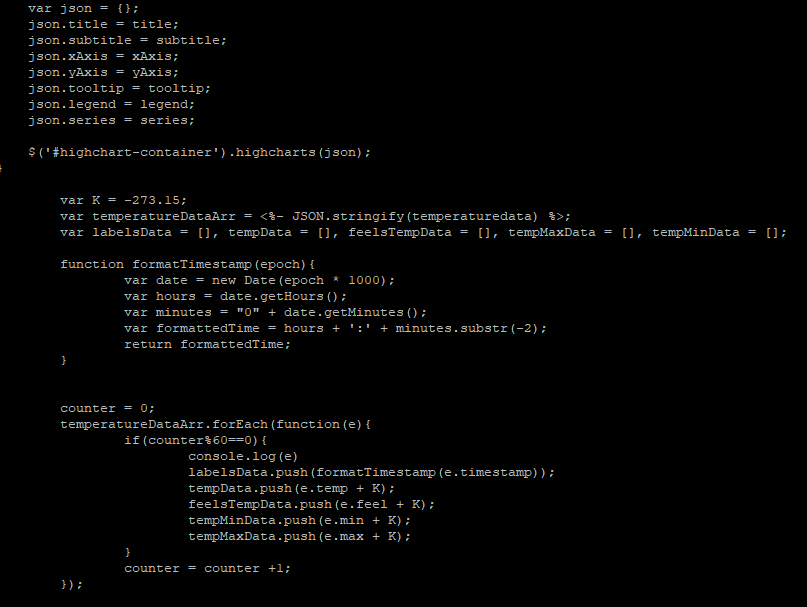
Kuva 2. Datan välitys Sparkilta MongoDB:hen.

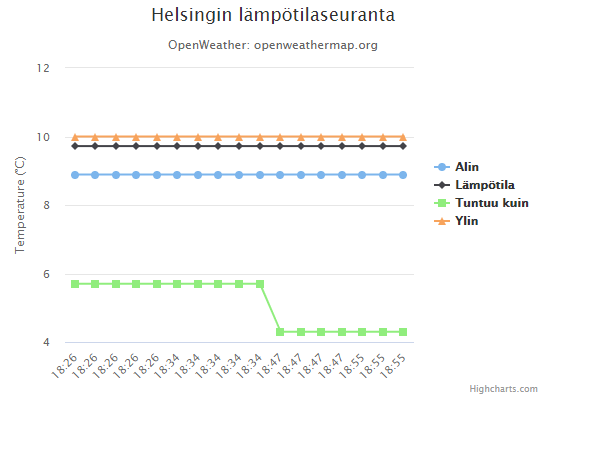
Kun data lähti liikkeelle oli se nähtävissä MongoDB koneen tietokannoissa. Minkä jälkeen oli aika siirtyä visualisointipuolelle.

# Highcharts komponentti

Highcharts tarjoaa visuaalisesti näyttäviä kuvaajia moneen lähtöön. Palvelussa on runsaasti esimerkkejä ja hyviä linkkejä, joiden avulla pääsee testaamaan palvelun kaavioita käytännössä.

Kuva 3. Server.js tiedoston lähdekoodi.

Kuva 4. Oleellinen kohta index.ejs tiedoston lähdekoodista.

Highcharts käyttöönotto tapahtui frontend puolella muokkaamalla index.js tiedostoa Views kansiossa. Server.js tiedostoon tuli tehdä muutoksia, jotta uuden datalähteen kyselyt osuivat kohdalleen. Halusin tuoda viimeisimmät havainnot Helsingin sääasemalta, joten aggregate kyselyn sijaan käytin Server.js puolella find() kyselyä mongodb:stä sorttaamalla viimeisimmät ensin ja rajoittamalla limit kutsulla datan määrää.Kuva 5. Esimerkki näkymästä.

Kuvasta 5 nähdään, että OpenWeatherin kuvaajat ovat selkeitä. Aikataulullisista syistä en alkanut värimaailmaa muokkaaman. Enkä pidä palvelua paria tuntia pitempää pystyssä. Lopuksi suljettu pipeline ja klusterit.

# Pohdinta

Tämän aiheen työstäminen on ollut virkistävää ikkunajumppaa, missä ei yksikään näyttö ole ylimääräinen. Ensimmäisenä haasteena törmäsin siihen, ettei cPouta ympäristön avainta saanutkaan kopioitua esimerkin mukaisesti. Eikä selvinnyt miten avaimen saisi kopioitua, joten jäi työ seisomaan hetkeksi työkiireiden takia, koska tehtävä ei tarjonnut paikkariippumattomuutta työn jatkamiseen. Uutta avainta ei haluttu ottaa käyttöön, koska se olisi ollut lopun alkua.

Seuraava haaste tuli vastaan, kun noin 1 kuukauden tauon jälkeen laitettiin pipelineketjua pystyyn. Muutaman kerran sai tarkastella käynnistysjärjestystä, että pystyi toteamaan ketjun toimivaksi.

Suurimmaksi haasteeksi nousi sopivan datalähteen valinta. Tällaisen pipeline ketjun hyödyntäminen pienellä datalla tuntuu liioittelulta. Lisäksi ulkoisten palveluiden API-rajapinnat pitävät huolen datan saatavuudesta, joten tuntui raskaalta työvaiheelta päästä ajatuksen yli, että koko pipelineketju on tarpeeton, sillä ainoata tarvittavaa komponettia voitaisiin kutsuoa suoraan frontendista. Tuskastumisen päätteeksi valitsin yksinkertaisen datalähteen säätietojen seurantaan.

Itse pipelineketjuun liittämisessä suurimmaksi haasteeksi nousi datan välittäminen Apache Kafkalta Sparkin kautta MongoDB:hen. Toistuvasti Spark pudotti skriptin pois päältä, koska RDD oli tyhjä. Kävi lopulta ilmi, että tätä kautta datan tulee päivittyä alle 10 sekunnin välein, jotta palvelu pysyy pystyssä. Päädyin lopulta päivittämään dataa 2 sekunnin välein halutun 10 minuutin sijaan.

Kurssilla vastaan tulleet Apachen Kafka, Zookeeper, Spark ja Kassandra olivat uusia tuttavuuksia ja sitä kautta kurssi on antanut paljon uutta päänvaivaa. Jatkohyödyntäminen riippuu pitkälti miten todennäköisesti näihin asioihin tulee törmättyä vastaisuudessa.