VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Petri Rinta-Halkola

# PRIMA POWER AVOIMEN DATAN ALUSTA

Tekniikka
2020

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

# TIIVISTELMÄ

| | |
|---|---|
| Tekijä | Petri Rinta-Halkola |
| Opinnäytetyön nimi | Prima Power avoimen datan alusta |
| Vuosi | 2020 |
| Kieli | suomi |
| Sivumäärä | 37 |
| Ohjaaja | Jani Ahvonen |

Tämän opinnäytetyön tavoitteena oli kehittää sekä dokumentoida web-pohjainen alusta, jonka avulla käyttäjät pystyisivät noutamaan Prima Powerin hallussa olevaa levytyökonedataa. Tätä dataa voitaisiin käyttää opetuksessa sekä tutkimustoiminnassa. Alusta on tarkoitettu muun muassa ammattikorkeakoulujen käyttöön.

Alusta on toteutettu niin sanottua SERN-teknologiapinoa käyttäen, johon kuuluvat SQL, Express, React sekä Node.js.

Alusta koostuu kolmesta osasta, nettisivustosta, REST-rajapinnasta sekä tietokannasta. Sivusto sekä REST-rajapinta on toteutettu JavaScript-ohjelmointikielellä käyttäen React-kirjastoa sekä Node.js runtime -ympäristöä.

SQL-tietokannassa varastoidaan levytyökonedatan lisäksi käyttäjätietoja sekä API-avaimia.

Opinnäytetyö oli laajuudeltaan oletettua suurempi, mutta lopputuloksena saatiin kuitenkin toimiva lopputuote, jonka kautta sen käyttäjät pystyvät noutamaan levytyökonedataa erilaisiin käyttötarkoituksiin.

Avainsanat REST, JavaScript, Node.js, React, SQL

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikka

## ABSTRACT

| | |
|---|---|
| Author | Petri Rinta-Halkola |
| Title | Prima Power Open Data Platform |
| Year | 2020 |
| Language | Finnish |
| Pages | 37 |
| Name of Supervisor | Jani Ahvonen |

The aim of this thesis was to create and to document a platform which would provide laser and sheet metal machinery data to its end users. This data could be then used in teaching as well as in research and development operations. The platform is intended to be used by universities of applied sciences.

The platform was made with a so called SERN stack, which includes SQL, Express, React and Node.js.

The platform consists of three parts, Web page, REST API and SQL-database. Both the web page and the REST API were implemented in the JavaScript-language using React-library and Node.js runtime environment.

In addition to the laser and sheet metal machinery data the SQL database stores user data and API keys.

The scope of this thesis was larger than first expected, but the result ended up being fully functional platform which offers laser and sheet metal machinery data to its end users for various purposes.

Keywords REST, JavaScript, Node.js, React, SQL

# SISÄLLYS

# KUVA- JA TAULUKKOLUETTELO

# 1 JOHDANTO

Julkishallinto, organisaatiot, yritykset tai jopa yksityishenkilöt voivat jakaa heille kertynyttä dataa muiden hyödynnettäväksi. Datalla tarkoitetaan koneluettavassa muodossa olevaa tietoa, kuten tekstiä, kuvia, karttoja ja niin edelleen.

Datalla on käyttötarkoituksensa sitä keräävällä taholla, mutta tätä voidaan hyödyntää vieläkin laajemmin jakamalla sitä muiden hyödynnettäväksi. Tämä edistää muun muassa uusien palvelujen sekä innovaatioiden syntyä.

## 1.1 Työn tarkoitus

Tämän opinnäytetyön toimeksiantajana toimi Prima Power.

Opinnäytetyön tarkoituksena oli kehittää alusta, jonka kautta Prima Powerin hallussa olevaa koneluettavaa levytyökonedataa voitaisiin jakaa muiden hyödynnettäväksi osaksi opetusta sekä tutkimustoimintaa.

## 1.2 Prima Power

Prima Power on levytyökoneisiin ja -järjestelmiin erikoistunut yritys, jolla on tuotantoyksiköitä Suomessa, Italiassa, Yhdysvalloissa sekä Kiinassa. /1/

Se palvelee asiakkaitaan yli 70 maassa ja sen tuotevalikoimasta löytyy laserleikkaus, lävistys, kulmaleikkuu sekä taivutusautomaatteja. /1/

Suomen toimipiste sijaitsee Seinäjoella, jossa työskentelee tällä hetkellä yli 300 työntekijää.

# 2 VAATIMUSMÄÄRITTELY

Alustalle asetettiin tiettyjä vaatimuksia lähinnä toiminnallisuuksien osalta. Ulkoasun sekä käytettävien teknologioiden osalta oli suhteellisen vapaat kädet.

Alustan tuli koostua rajapinnasta, tietokannasta sekä web-pohjaisesta käyttöliittymästä. Rajapinnan käyttöä haluttiin rajoittaa niin, ettei kuka tahansa pystyisi käyttämään sieltä saatavaa dataa.

Koska alusta on tarkoitettu pääasiassa koulujen käyttöön, haluttiin siihen toiminnallisuus, jossa opettajat pystyisivät kontrolloimaan kuka saa käyttöoikeuden rajapintaan sekä sieltä saatavaan dataan. Tässä tapauksessa kyse on opiskelijoista, joiden pääsyä rajapintaan opettajat pystyvät kontrolloimaan.

Toiveena oli myös saada jonkinlainen monitorointi rajapintaan kohdistuvasta liikenteestä. Tämän jouduin kuitenkin hylkäämään työn loppuvaiheessa aikataulun vuoksi.

Tietokantana tulisi olemaan SQL ja se tulisi pyörimään Azuren-pilvipalvelussa, johon rajapinnan kautta saatava data tulisi siirtää.

Lopulta koko kokonaisuus tulisi pyörimään samaisessa Azuren-pilviympäristössä.

# 3 KÄYTETYT TEKNOLOGIAT

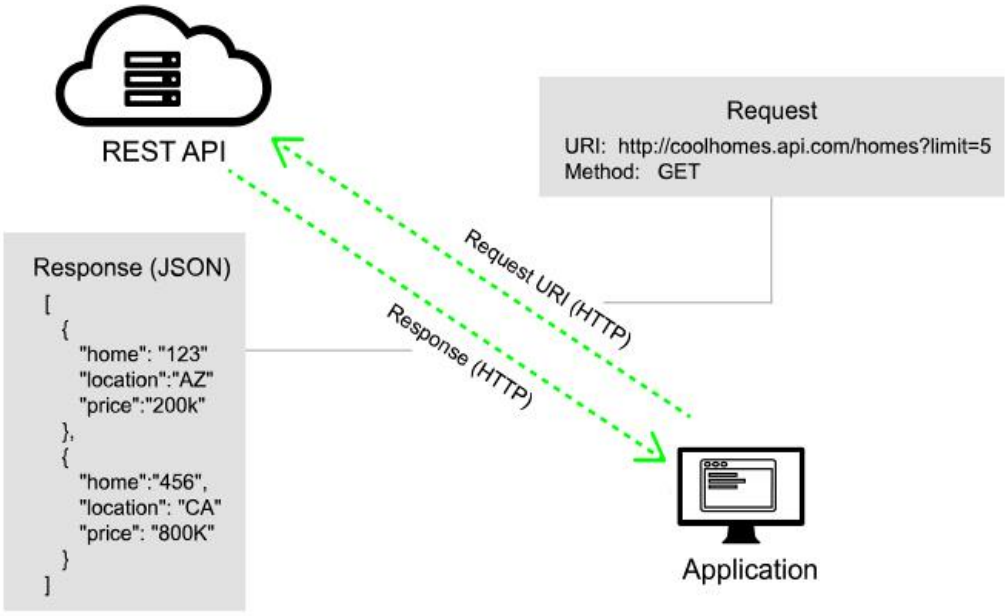
Tässä luvussa käydään läpi opinnäytetyössä käytettyjä teknologioita ja arkkitehtuurimalleja sekä niiden taustoja.

## 3.1 REST-arkkitehtuuri

REST on lyhenne sanoista Representational State Transfer ja se on HTTP-protokollaan perustuva arkkitehtuurimalli ohjelmointirajapintojen toteuttamiseen. /2/

REST-arkkitehtuuri käsittelee sen jokaista sisältöä resurssina. Resurssit voivat olla lähes mitä tahansa, kuten henkilöitä, kirjoja ja niin edelleen. /2/

Resursseja käsittelevien osien yhteys perustuu tyypillisesti asiakaspalvelinmalliin, jossa asiakas (*engl. client*) tekee pyynnön ja palvelin kuuntelee sekä käsittelee vastaanottamiaan pyyntöjä ja palauttaa niihin vastauksia. (**Kuva 1.**) /3/




**Kuva 1.** Asiakaspalvelinmalli.

### 3.2 JSON

JSON tulee sanoista JavaScript Object Notation ja se on standardoitu tekstipohjainen formaatti, joka perustuu JavaScript-oliosyntaksiin. Sitä käytetään perinteisestin tiedon välittämiseen web-sovelluksissa. /10/

JSON-muotoisen datan yhtenä etuna on sen helppo luettavuus. (**Kuva 2.**)

```
1  {
2    "firstName": "John",
3    "lastName": "Doe"
4  }
5
```

**Kuva 2.** Esimerkki JSON-muotoisesta datasta.

### 3.3 API- avain

API on lyhenne sanoista Application Programming Interface, eli ohjelmointirajapinta. Avaimella viitataan tässä tapauksessa yksilölliseen tunnisteeseen, jolla sen käyttäjä pystyy todentamaan pääsynsä rajapintaan. Avain tulee liittää osaksi jokaista rajapintaan tehtävää palvelupyyntöä.

Avain on tässä tapauksessa 31 merkkiä pitkä merkkijono, joka koostuu neljästä 7 merkin ryhmästä väliviivalla eroteltuna. (**Kuva 3.**)

API-avaimia käytetään yleensä pääsyn rajoittamisessa ohjelmointirajapintoihin tai käyttäjän pyyntöjen seurantaan.


**Kuva 3.** Esimerkki opinnäytetyössä käytetystä API-avaimesta.

### 3.4 React.js

React.js on avoimeen lähdekoodiin perustuva Facebookin vuonna 2013 kehittämä sekä ylläpitämä komponenttipohjainen JavaScript-kirjasto, joka on suunniteltu selainsovellusten käyttöliittymien toteuttamiseen. /11/

Komponenttien avulla käyttöliittymät voidaan pilkkoa pienempiin uudelleen käytettäviin osiin. Komponentit ovat siis pala koodia, jotka kuvaavat tiettyä osaa käyttöliittymästä.

Jokainen React-sovellus sisältää vähintään yhden komponentin, jota kutsutaan juurikomponentiksi.

Tämä komponentti sisältää muut sovelluksen komponentit puunmuotoisessa asetelmassa, jossa juurikomponentti on sen ylin osa. (**Kuva 4.**)

```
import React from "react";
import ChildComponent1 from './ChildComponent1';
import ChildComponent2 from './ChildComponent2';
import "./styles.css";

export default function App() {
  return (
    <div className="App">
      <ChildComponent1 />
      <ChildComponent2 />
    </div>
  );
}

```

**Kuva 4.** Esimerkki tyypillisestä React-komponenttipuusta.

### 3.5 Node.js

Node.js on avoimen lähdekoodin alustariippumaton JavaScript runtime-ympäristö, joka mahdollistaa JavaScript-koodin suorittamisen selaimen ulkopuolella. Se pohjautuu Googlen kehittämään JavaScript-moottoriin, jota kutsutaan nimellä V8. /4/ Kyseistä moottoria käytetään myös muun muassa Google Chrome- sekä Microsoft Edge-selaimissa.

Node.js sisältää joukon sisäänrakennettuja moduuleita käytettäväksi. Yksi näistä on HTTP-moduuli, jonka avulla voidaan luoda web-palvelimia. (**Kuva 5.**)

```
const http = require('http');
const PORT = process.env.PORT || 5000;

const app = http.createServer((req, res) => {
    res.writeHead(200, { 'Content-Type': 'text/plain' })
    res.end('Hello World')
})

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

```

**Kuva 5.** HTTP-moduulin avulla luotu yksinkertainen web-palvelin.

#### 3.5.1 Express

Web-palvelimen rakentaminen Node.js:n sisäänrakennetun HTTP-moduulin avulla on mahdollista, mutta varsin työlästä. Siksipä on mielekkäämpää käyttää siihen tarkoitukseen tehtyä verkkokehystä, kuten Expressiä. /3/

Express on yksi Node.js:lle kehitetyistä verkkokehyksistä, joka on suunniteltu helpottamaan verkkosivustojen, verkkosovellusten sekä rajapintojen kehittämistä. (**Kuva 6.**)

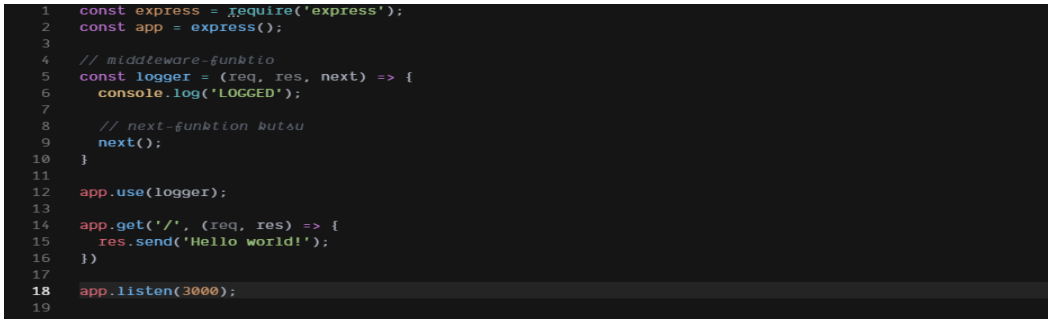
```
const express = require('express');
const app = express();
const PORT = process.env.PORT || 5000;

app.get('/', (req, res) => {
    res.send('Hello World');
})

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

```

**Kuva 6.** Express-verkkokehyksen avulla luotu yksinkertainen web-palvelin.

### 3.5.2 Middleware

Middleware on funktio, jolla on pääsy request -ja response-olioihin sekä next-funktioon**.** (**Kuva 7.**)

Middleware-funktioita voi olla useita, jolloin ne suoritetaan siinä järjestyksessä kuin ne on otettu koodissa käyttöön. Kutsumalla next-funktiota voidaan siirtyä seuraavaan middleware-funktioon. /5/



```
const express = require('express');
const app = express();

// middleware-funktio
const logger = (req, res, next) => {
  console.log('LOGGED');

  // next-funktion kutsu
  next();
}

app.use(logger);

app.get('/', (req, res) => {
  res.send('Hello world!');
})

app.listen(3000);

```

**Kuva 7.** Esimerkki middleware-funktiosta.

### 3.5.3 Node Package Manager

Node Package Manager tai npm on Node.js:lle kehitetty ilmainen paketinhallintajärjestelmä, jonka kautta käyttäjä voi asentaa ja hallinnoida paketteja sekä integroida näitä osaksi sovellustaan. /12/

Paketit ovat käytännössä vain joukko koodeja, jota joku muu on kirjoittanut ja jakanut muiden hyödynnettäväksi.

## 3.6 Structured Query Language

Structured Query Language tai SQL on rakenteinen kyselykieli, jonka historia alkoi jo 1970-luvun lopulla. Sen avulla pystytään kommunikoimaan relaatiotietokannan kanssa.

SQL-käskyjä käytetään tehtävien, kuten tiedon hakemiseen tai päivittämiseen tietokannasta. /13/

## 3.7 Microsoft Azure

Azure on microsoftin luoma pilvipalvelu sovellusten ja palveluiden rakentamiseen, testaamiseen, käyttöönottoon ja hallintaan Microsoftin hallinnoimien palvelinkeskusten kautta. /7/

# 4 KÄYTTÖLIITTYMÄ

Web-pohjainen käyttöliittymä on toteutettu kokonaisuudessaan käyttäen React-kirjastoa ja se on jaettu kahteen osaan. Toinen osa sisältää salasanalla suojatun hallintapaneelin ja toinen osa varsinaisen sivuston.

Ulkoasu on lisäksi täysin responsiivinen, jolloin se toimii moitteettomastin paitsi työpöydällä, myös mobiililaitteilla. Nykyisin suurin osa käyttäjistä tulee sivuille mobiililaitteilla, jolloin se on otettava toteutuksessa huomioon.

## 4.1 Hallintapaneeli

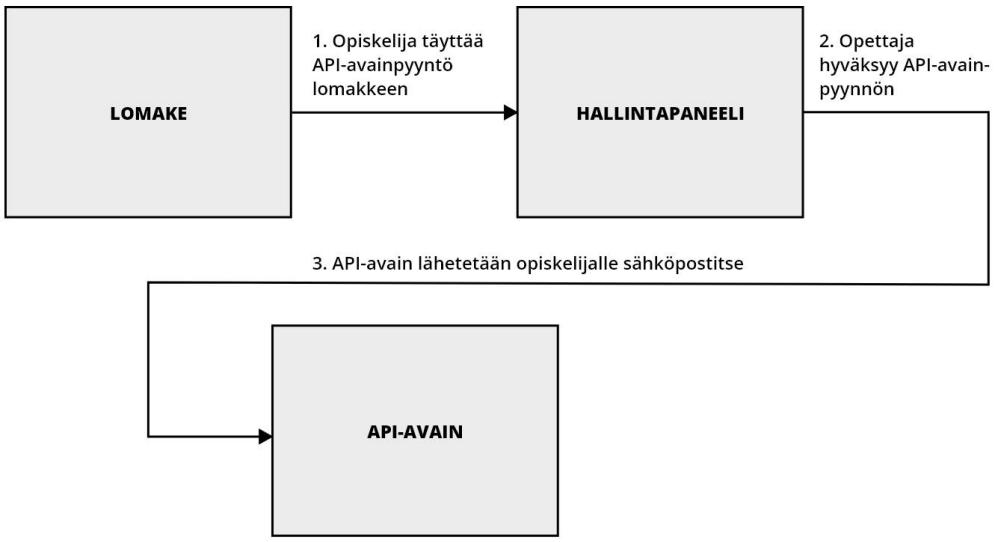
Rajapinnan käyttöä haluttiin kontrolloida, joten sen käyttö vaatii käyttöoikeuden. Tässä tapauksessa se tarkoittaa API-avaimen käyttöä. API-avain on merkkijono, joka liitetään osaksi jokaista rajapinnalle tehtävää palvelupyyntöä.

Hallintapaneeli mahdollistaa näiden avaimien kontrolloinnin. Avaimen luonti rajapintaan on prosessi, jota opettajat hallinnoivat. Opiskelijoilla on mahdollisuus pyytää avainta täyttämällä sivustolta löytyvä lomake. Tämä opiskelijan lähettämä API-avainpyyntö tulee näkyviin hallintapaneelissa, johon opettajilla on pääsy.

Hallintapaneelissa opiskelijoiden API-avainpyynnöt esitetään taulukossa, jossa jokainen rivi sisältää tietoa kyseisestä pyynnön tehneestä henkilöstä. Näitä tietoja ovat nimi, sähköpostiosoite, luontiajankohta sekä kommentti. Kommenttikenttä voi sisältää yksilöitävää tietoa, kuten esimerkiksi projektiryhmän.

Pyynnön hyväksymisen jälkeen kyseiselle opiskelijalle lähetetään sähköpostiviesti, joka sisältää henkilökohtaisen API-avaimen rajapintaan.

Tätä API-avain hyväksyntäketjua voidaan havainnollistaa sekvenssikaaviolla. (**Kuva 8.**)

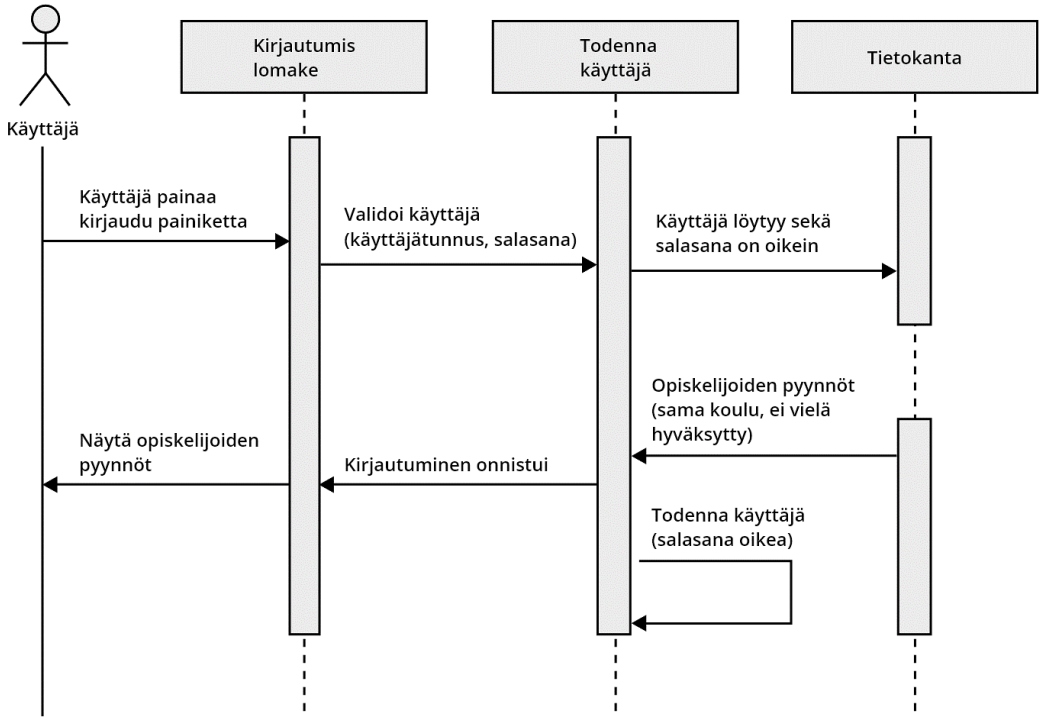



**Kuva 8.** API-avaimen hyväksyntäketju.

### 4.2 Kirjautuminen

Opettajan kirjautuessa sisään, tietokannasta palautetaan opiskelijoiden API-avainpyynnöt, jotka liittyvät kyseiseen opettajaan. Opiskelijoiden pyynnöt yhdistetään opettajiin, opiskelijan ilmoittaman koulun perusteella. (**Kuva 9.**)

Opiskelijoiden pyynnöt jotka ovat jo hyväksytty, eivät tule näkyviin hallintapaneelissa. Näitä hallitaan ylimääräisellä kentällä tietokannassa.




**Kuva 9.** Sisäänkirjautuminen sekvenssikaaviona.

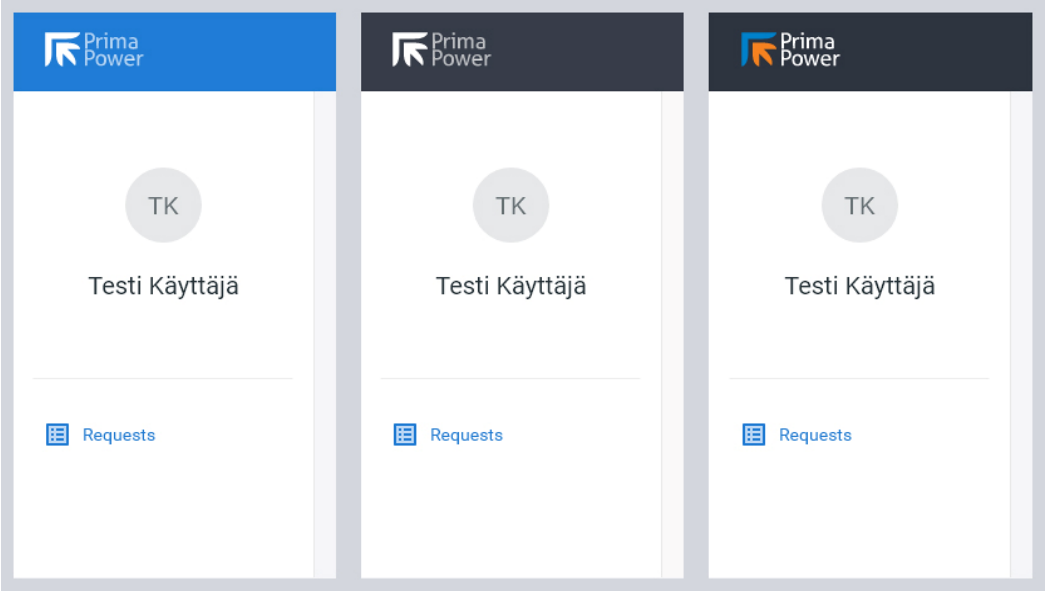
### 4.3 Ulkoasu

Hallintapaneelin pohjalla on käytetty valmista Reactilla luotua ilmaista julkaisua, josta löytyi joitain tarvitsemiani toiminnallisuuksia valmiiksi rakennettuna. Suurin syy tähän oli merkittävä ajansäästö.

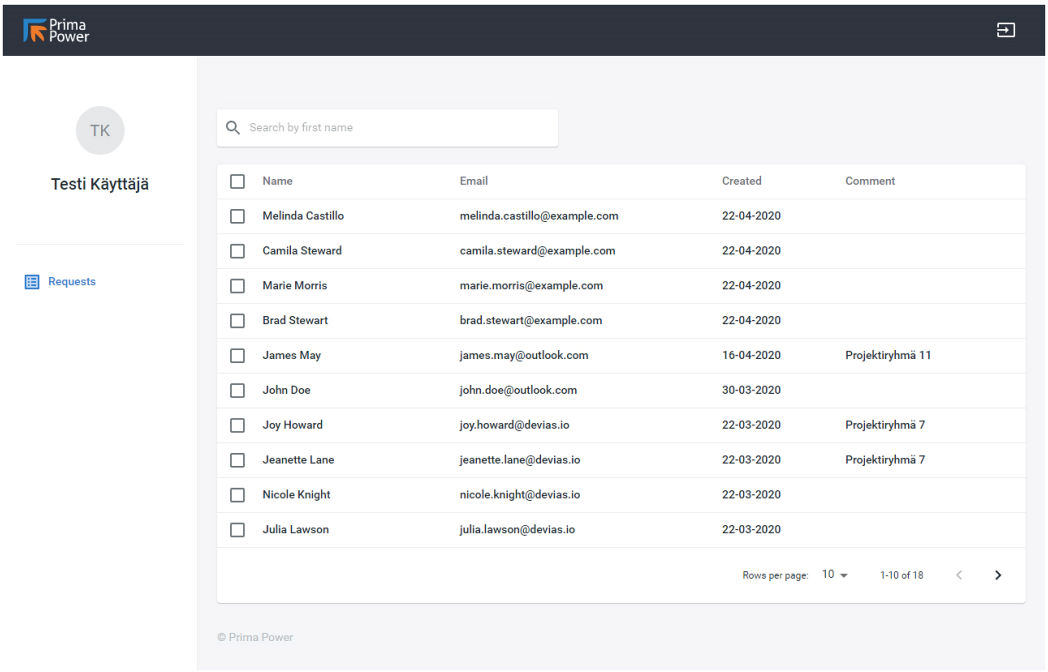
Ulkoasu on hyvin pitkälti pysynyt koskemattomana, ainoastaan värimaailmaan tehtiin joitain muutoksia, joka kävikin läpi muutaman luonnoksen (**Kuva 10.**) ennen kuin päädyttiin lopulliseen versioon. (**Kuva 11.**)

Hallintapaneeli käyttää Googlen luomia Material-UI -komponentteja hyväkseen, jotka nopeuttavat käyttöliittymäkomponenttien toteuttamista entisestään.

Google on lisäksi luonut näille todella kattavan dokumentaation esimerkkeineen.




**Kuva 10.** Ulkoasun värimaailman luonnoksia.

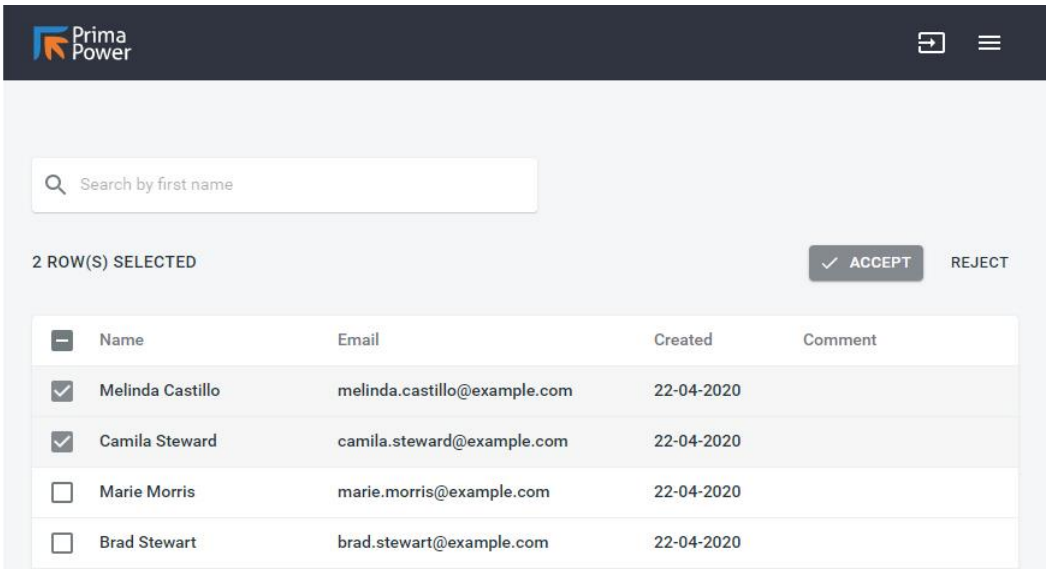



**Kuva 11.** Lopullinen hallintapaneelin ulkoasu.

### 4.4 API-avainpyyntöjen -hallinta

API-avainpyyntöjen -hallinta tapahtuu valitsemalla yksi tai useampi rivi, jolloin taulukon yläpuolelle ilmestyy kaksi painiketta. Näiden avulla pyyntöjä voidaan joko hyväksyä tai hylätä.

Tavoitteena oli rakentaa mahdollisimman käyttäjäystävällinen käyttöliittymä pyyntöjen hallintaan, jolloin päädyin ratkaisuun, jossa useampia pyyntöjä on mahdollista hallita samanaikaisesti. (**Kuva 12.**)




**Kuva 12.** API-avainpyyntöjen -hallinta hallintapaneelissa.

Käyttäjän suorittamasta toimenpiteestä tiedotetaan ilmoituksella, joka ilmestyy hallintapaneelin alanurkkaan. (**Kuva 13.**)

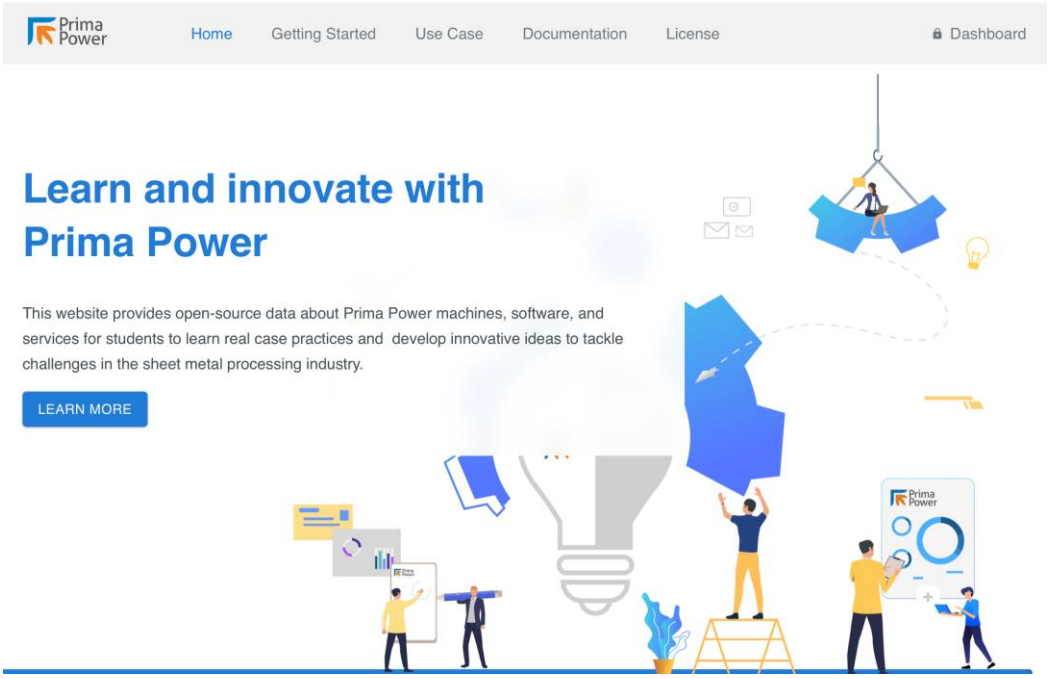
Tällä haluttiin viestiä käyttäjälle hänen suorittamansa toimenpiteen tila, sekä mahdolliset virhetilanteet sovelluksen käytön aikana.


**Kuva 13.** Ilmoitus käyttäjän suorittamasta toimenpiteestä.

### 4.5 Sivusto

Sivusto sisältää yleistä tietoa alustasta sekä sen käyttötarkoituksesta, dokumentaation rajapinnan käyttöön, datan käyttölisenssin, lomakkeen API-avaimen pyyntöön sekä linkin hallintapaneeliin.

Ulkoasu noudattaa toimeksiantajaltani saamaani PowerPoint-esityksen ulkoasua. (**Kuva 14.**)




**Kuva 14.** Kuvakaappaus sivuston etusivulta.

# 5 REST-RAJAPINTA

Rajapinta on yhteydessä SQL-tietokantaan, jossa levytyökonedataa varastoidaan. Rajapinta mahdollistaa muun muassa tiedon hakemisen tietokannasta sekä sen palauttamisen pyynnön tehneelle asiakkaalle.

### 5.1.1 REST-päätepiste

Rajapinta tarjoaa yhden julkisen päätepisteen (*engl. endpoint*), joka tarjoaa JSON-muotoista levytyökonedataa. Tämä päätepiste vaatii API-avaimen sisällyttämisen rajapintaan tehtävässä palvelupyynnössä.

Avaimen lisäksi URL-parametreina tulee ilmoittaa taulukossa pakollisena ilmoitetut parametrit. (**Taulukko 1.**) Taulukko on osa rajapinnan julkista dokumentaatiota.

**Taulukko 1.** REST-rajapinnan parametritaulukko.

| **Parametri** | **Pakollinen** | **Selite** | **Tyyppi** |
|---|---|---|---|
| start | Kyllä | Aloitus päivämäärä v-k-p muodossa. | Merkkijono |
| end | Kyllä | Lopetus päivämäärä v-k-p muodossa. Esimerkiksi 2020-01-24. | Merkkijono |
| machineID | Kyllä | Koneen id, jolta dataa haetaan. | Kokonaisluku |
| metric | Kyllä | Määrittelee palautettavan datan tyypin. Nämä on listattuna dokumentaatiossa. | Merkkijono |
| limit | Ei | Tietokannasta palautettavien rivien lukumäärä. Oletuksena palautetaan 150 riviä. | Kokonaisluku |
| key | Kyllä | Yksilöllinen avain, jota käytetään todentamiseen. Pakollinen jokaisessa palvelupyynnössä. | Merkkijono |

### 5.1.2 URL-parametri

URL-parametreja käytetään esimerkiksi määrittelemään mitä rajapinnan tulisi palauttaa sitä tehneelle asiakkaalle.

URL-parametrit liitetään osaksi URL-osoitetta avainarvopareina. (**Kuva 15.**) Avain sekä arvo erotetaan toisistaan yhtäsuuruus-merkillä (=) ja yhdistetään muihin mahdollisiin parametreihin et-merkillä (*&*).

URL-osoitteen ensimmäisen parametrin edessä tulee aina olla kysymysmerkki (*?*).

```
https://example.com/search?parametri1=1234&parametri2=google
```

**Kuva 15.** Esimerkki URL-parametreista.

URL-parametrit ovat luettavissa palvelimen päässä request-oliosta, jolloin voidaan suorittaa asiakkaan toivoma toimenpide.

Parametrit ovat rajapintakohtaisia ja täten täysin rajapinnan kehittäjän päätettävissä.

### 5.1.3 Request -ja response-oliot

Request-olio edustaa HTTP-pyyntöä ja sillä on pääsy muun muassa asiakkaan palvelimelle lähettämän palvelupyynnön parametreihin. (**Kuva 16.**)

Response-olion avulla voidaan vastaavastin vastata asiakkaan palvelinpyyntöön, jolloin voidaan esimerkiksi palauttaa tietokantakyselyn tulos.

```
// GET https://example.com/search?keyword=url-parameter

app.get('/search', (request, response) => {
  console.log(request.query.keyword); // "url-parameter"
});
```

**Kuva 16.** URL-parametrin lukeminen request-olion avulla.

### 5.1.4 Syötteen todentaminen

Jokainen käyttäjän suorittama toimenpide rajapintaan todennetaan eli sen oikeellisuus tarkastetaan. Oli kyseessä sitten käyttäjän lähettämä lomake tai rajapintaan tehty palvelupyyntö. Käyttäjän tekemät syötteet tuleekin aina tarkastaa vähintään palvelimen puolella, mahdollisuuksien mukaan molemmissa, jolloin voidaan tarjota tietoturvan ohella parempi käyttäjäkokemus.

Todentamisella estetään muun muassa vääränmuotoisen datan pääsy järjestelmään mikä voisi aiheuttaa mahdollisia ongelmia myöhäisemmässä vaiheessa. (**Kuva 17.**)

```
const queryValidationRules = () => {
  return [
    query('start', 'Valid date is required.').matches(validDateFormat),
    query('end', 'Valid date is required.')
      .matches(validDateFormat)
      .bail()
      .custom((value, { req }) => {
        if (formatDate(value) < formatDate(req.query.start)) {
          throw new Error('End date must be after or same as the start date.');
        }
        return true;
      }),
    query('machineID', 'Valid machineID is required.').isInt({ gt: 0 }),
    query('metric', 'Valid metric type is required.')
      .isString()
      .bail()
      .isLength({ min: 1 }),
    check('limit', 'Invalid value.')
      .optional({ checkFalsy: true, nullable: true })
      .isInt({ gt: 0 }),
    query('key').custom(async value => {
      const isValid = await checkKeyValidity(value);
      if (!isValid) {
        throw new Error('Valid API key is required.');
      }
      return true;
    })
  ];
};

```

**Kuva 17.** Palvelupyynnön todentaminen käyttäen express-validator -pakettia.

Virheellisestä tai puutteellisesta palvelupyynnöstä palautetaan JSON-muotoinen virheilmoitus sekä virhettä vastaava HTTP-tilakoodi. Virheilmoituksista pyrittiin tekemään mahdollisimman selkeitä sekä kuvaavia. (**Kuva 18.**)

```
{
    "code": 400,
    "message": [
        {
            "key": "Valid API key is required."
        }
    ]
}
```

**Kuva 18.** Rajapinnan palauttama JSON-muotoinen virheilmoitus.

HTTP-tilakoodi 400 viittaa niin sanottuun Bad Request-tilaan, joka tarkoittaa, ettei pyyntöä voida käsitellä asiakkaalta tulleen puutteellisen tai virheellisen pyynnön johdosta.

HTTP-tilakoodit on standardoitu, mutta se mitä palvelin missäkin tilanteessa palauttaa on täysin kehittäjän päätettävissä.

### 5.1.5 API-avaimen todentaminen

API-avaimen käyttökelpoisuuden tarkastamiseen luotiin oma funktio. Funktio tarkastaa avaimen kelpoisuuden ennen jokaista rajapinnan julkiseen päätepisteeseen tehtävää palvelupyyntöä. Funktio tekee tietokantakyselyn tarkastaakseen avaimen olemassaolon. (**Kuva 19.**)

```
const sql = require('mssql');
const dbConfig = require('../config/dbConfig');
const uuidAPIKey = require('uuid-apikey');

const checkKeyValidity = async key => {
  try {
    // tarkasta onko avain validi api-avain
    const isValid = uuidAPIKey.isAPIKey(key);

    if (!isValid) {
      return false;
    }

    const toUUID = uuidAPIKey.toUUID(key);
    const findUserUUID = `SELECT uuid FROM users WHERE uuid = '${toUUID}' AND verified = 'true'`;
    const findAdminUUID = `SELECT uuid FROM admins WHERE uuid = '${toUUID}'`;

    await sql.connect(dbConfig);
    const user = await sql.query(findUserUUID);
    const admin = await sql.query(findAdminUUID);

    const result = user.rowsAffected[0] > 0 ? user : admin;

    if (result.rowsAffected[0] < 1) {
      return false;
    }

    return true;
  } catch (err) {
    return false;
  }
};

module.exports = checkKeyValidity;
```

**Kuva 19.** API-avaimen käyttökelpoisuuden tarkastava funktio.

Avainta ei kuitenkaan liitetä millään tavalla itse käyttäjään, vaan kuka tahansa avaimen saatuaan pystyisi tekemään palvelupyyntöjä rajapintaan. Avain on kuitenkin tarvittaessa mahdollista mitätöidä, jolloin käyttäjälle lähetetään uusi avain sähköpostitse.

Ottaen huomioon käyttötarkoituksen, on kyseinen autentikointimenetelmä kuitenkin täysin riittävä tähän tarkoitukseen.

### 5.1.6 CORS

Cross Origin Resource Sharing on selainmekanismi, mikä mahdollistaa hallitun pääsyn tietyn toimialueen ulkopuolella sijaitseviin resursseihin. /14/

Nykyaikaiset selaimet noudattavat niin sanottua saman alkuperän käytäntöä (*engl. Same Origin Policy*), jolloin ne hylkäävät muihin lähteisiin tehdyt palvelupyynnöt oletuksena tietoturvan takia. /15/

Toisin sanoen CORS estää selaimesta JavaScriptilla tehdyn palvelupyynnön palvelimelle, joka sijaitsee jossain muussa lähteessä kuin mistä palvelupyyntö tehtiin. Tätä toiminnallisuutta voidaan kuitenkin tarvittaessa säädellä palvelimen päästä, jolloin tällainen toiminta voidaan jopa sallia.

Tämän rajapinnan tapauksessa haluttiin käyttäjän pystyvän tekemään palvelupyyntöjä myös suoraan selaimesta käsin rajapinnan julkiseen päätepisteeseen. Tähän tarkoitukseen käytettiin erillistä npm-paketinhallintajärjestelmän kautta asennettua middleware-pakettia, joka sallii tämän. Paketin CORS-asetuksia on mahdollista säädellä omien tarpeiden mukaan. Tässä tapauksessa oletusasetukset kuitenkin toimivat mainiosti.

## 5.2 Kirjautuminen

Tietoturvaan liittyvissä asioissa pyrittiin noudattamaan hyväksi havaittuja käytäntöjä ja tämä onkin yksittäisistä aiheista suhteellisestin eniten aikaa vievä.

### 5.2.1 Salasanan todentaminen

Käyttäjän kirjautuessa sisään, verrataan hänen syöttämäänsä salasanaa tietokannassa olevaan salasanan tiivisteeseen (*engl. hash*). Salasanasta on siis luotu tiiviste käyttäjätilin luonnin yhteydessä ja tämä on tallennettuna tietokantaan.

Näiden ollessa samat, asiakkaalle lähetetään eväste, jolla käyttäjä valtuutetaan (*engl. authorize*) rajapinnan suojatuille päätepisteille.

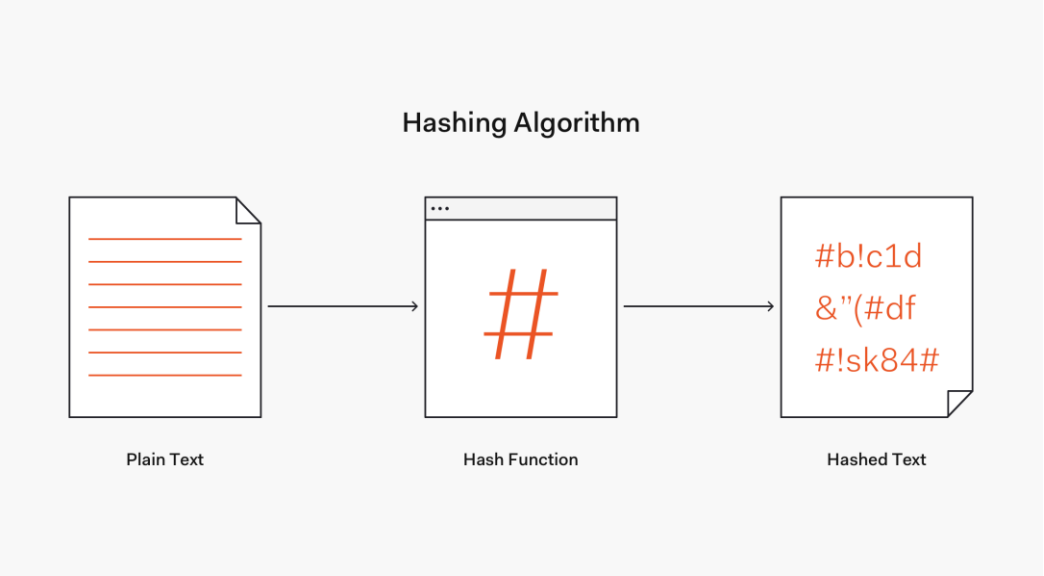
Evästeiden tietoturvaa voidaan parantaa asettamalla niille tiettyjä attribuutteja tai määritteitä niiden luonnin yhteydessä, joista ehkä tärkeimpänä HttpOnly-määrite.

### 5.2.2 Salasanan tiiviste

Salasanojen varastoiminen tietokannassa selkeälukuisena tekstinä ei ole missään nimessä suositeltavaa, sen sijaan salasana tulee ajaa matemaattisen algoritmin läpi, joka luo salasanasta tiivisteen. Tiiviste on salasanasta luotu tiivistetty merkkijono. (**Kuva 20.**)

Tiivisteen luontialgoritmi on yksisuuntainen, joten sen kääntäminen takaisin alkuperäiseksi salasanaksi on lähes mahdotonta. /20/

Tallentamalla selkeälukuisen salasanan sijasta siitä luotu tiiviste tietokantaan, vähennetään riskiä salasanojen päätymisestä vääriin käsiin mahdollisen tietomurron sattuessa.




**Kuva 20.** Tiivistealgoritmin tuottama tiiviste.

### 5.3 Evästeet

Eväste tai keksi on pieni datan palanen, jonka palvelin lähettää käyttäjän selaimeen. Selain voi tallentaa tämän evästeen ja lähettää sen myöhempien pyyntöjen mukana samalle palvelimelle. Tyypillisesti sillä yritetään selvittää tulivatko kaksi pyyntöä samalta selaimelta, esimerkiksi pitäen käyttäjän sisäänkirjautuneena.

Evästeitä käytetään tyypillisesti istunnon hallinnassa, käyttäjäkohtaisissa asetuksissa sekä käyttäjän seurannassa ja analysoinnissa. /8/

Evästeiden käytössä tulee kuitenkin huomata, että käyttäjän selaimeen tallennettuihin evästeisiin pääsee käsiksi suoraan JavaScriptilla ja, että niiden tietoja on mahdollista peukaloida.

Jos tämä ei ole toivottua, tulisikin käyttää erityistä HttpOnly-määritettä evästeen luonnin yhteydessä.

Seuraavassa osiossa on esitetty muutama tämän opinnäytetyön kannalta oleellinen evästeen-määrite.

### 5.3.1 Name

Name-määrite kertoo evästeen nimen ja se onkin ainut pakollinen määrite, joka evästeelle täytyy määrittää sen luonnin yhteydessä, muiden ollessa vaihtoehtoisia. /16/

### 5.3.2 HttpOnly

HttpOnly-määrite estää evästeen lukemisen JavaScriptin avulla selaimesta käsin ja siten myös sen mahdollisen peukaloinnin.

### 5.3.3 Secure

Secure-määrite varmistaa, että eväste lähetetään ainoastaan salatun HTTPS-protokollan kautta. /8/

### 5.3.4 Expires

Expires-määrite määrittelee evästeen eliniän, jonka jälkeen se tuhotaan. Jos tätä ei ole erikseen määritetty, eväste tuhotaan selainistunnon päätyttyä. Vanhentumisaika ilmoitetaan muodossa *Thu, 24 May 2020 08:55:17 GMT*. /16/

### 5.3.5 Domain

Domain-määrite määrittele missä verkkotunnuksessa (*engl. domain*) eväste on voimassa. Jos tätä ei määritetä, on eväste oletuksena voimassa vain siinä verkkotunnuksessa missä se on asetettu, muttei sen aliverkkotunnuksissa (*engl*. *subdomain*). /8/

# 6 TIETOKANTA

Rajapinnan kautta saatavilla oleva data varastoidaan Azuren pilvessä sijaitsevassa SQL-tietokannassa.

## 6.1 Datan siirto

Toimeksiantajalta saatua dataa siirrettiin Microsoft SQL server management studio-hallintasovelluksella SQL-tietokantaan. Taulut nimettiin tiedostojen nimien perusteella, sillä ne olivat kuvaavia sellaisenaan.

Kyseisestä sovelluksesta löytyy tarvittavat työkalut datan siirtämiseen.

## 6.2 Näkymät

Aiemmin luomistani tauluista luotiin jokaisesta uusi näkymä (*engl. view*), jolloin niistä voitiin piilottaa tarpeettomia sarakkeita muokkaamatta alkuperäistä dataa millään tavalla.

Näkymät ovat ikään kuin virtuaalisia tauluja, joita ei tallenneta pysyvään muistiin, vaan ne luodaan niihin kohdistuneen kyselyn seurauksena. Tämän vuoksi niissä oleva tieto on aina ajantasaista.

Käyttäjälle näkymät näkyvät kuten perinteiset taulutkin ja niihin voidaan kohdistaa kyselyjä samaan tapaan kuin perinteisiin tauluihinkin. /17/

Uusi näkymä voidaan luodaan käyttäen CREATE VIEW -komentoa. (**Kuva 21.**)

```
CREATE VIEW [uusi_nakyma] AS
SELECT CustomerName, ContactName
FROM Customers WHERE Country = 'Finland';
```

**Kuva 21.** Esimerkki uuden näkymän luonnista.

### 6.3 Taulut

Opiskelijoille sekä opettajille luotiin omat taulut, sillä ne eroavat toisistaan muun muassa opettajilla olevan salasanan takia.

Taulut pitävät sisällään käyttäjätietojen ohella myös henkilökohtaisen API-avaimen.

Uusi taulu luodaan käyttäen CREATE TABLE -komentoa. (**Kuva 22.**)

```
CREATE TABLE Persons (
    Personid int IDENTITY(1,1) PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Email varchar(255) NOT NULL
);
```

**Kuva 22.** Esimerkki uuden taulun luonnista.

Pääavain (*engl. primary key*) on uniikki tunniste, joka yksilöi jokaisen taulun tietueen. /18/ Esimerkissä esiintyvä IDENTITY-avainsana korottaa kokonaislukuarvoa automaattisestin aina yhdellä, kun uusi tietue lisätään, jolloin esimerkissä oleva Personid pysyy aina yksilöllisenä.

### 6.4 Indeksi

Indeksi on ikään kuin kirjan sisällysluettelo, jonka avulla tietokantahakuja voidaan nopeuttaa merkittävästi. /19/

Indeksi luodaan taulun sarakkeesta käyttäen CREATE INDEX -komentoa. (**Kuva 23**.)

```
CREATE INDEX idx_lastname
ON Persons (LastName);
```

**Kuva 23.** Esimerkki indeksin luonnista.

Indeksoinnilla saavutettiinkin huomattavia parannuksia tietokantahakujen nopeudessa. Tyypillinen haku tietokantaan saattoi viedä aiemmin sekunteja, kun indeksoinnin jälkeen se oli enää muutamia satoja millisekunteja.

On kuitenkin huomattava, että jokainen luotu indeksi vie levytilaa, joten riippuen tietokannan koosta, voi näistä kertyä merkittävä määrä levytilaa vievää dataa.

Indeksi kannattaakin luoda vain sellaisista sarakkeista, joihin kohdistuu kyselyjä usein.

# 7 TESTAUS

Testauksella pyritään varmistamaan sovelluksen tai palvelun oikeanlainen toiminta. Kaikkia mahdollisia skenaarioita ei ole kuitenkaan mahdollista testata, joten testauksessa tuleekin pyrkiä tunnistamaan palvelun kannalta tärkeimmät käyttötapaukset ja rakentamaan testit näiden pohjalta. /9/

## 7.1 Postman

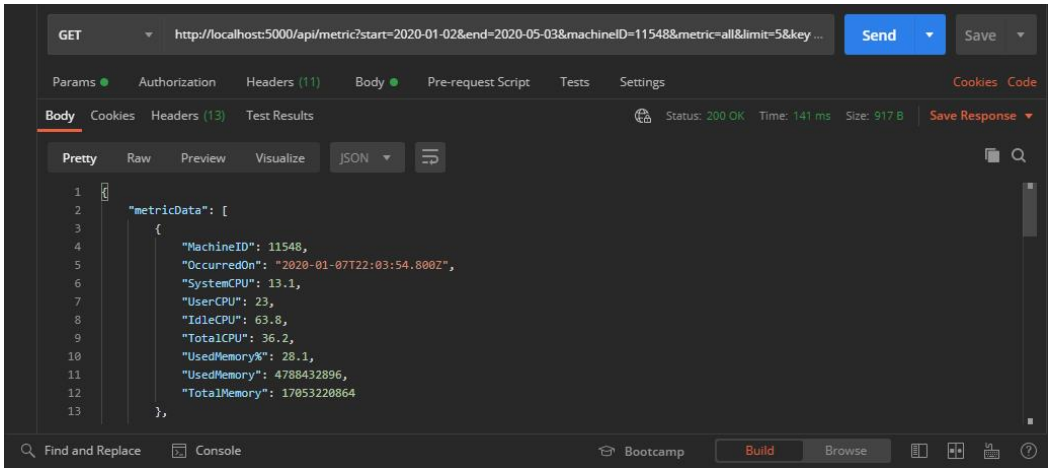
REST-rajapintojen testaukseen löytyy monenlaisia työkaluja. Päädyin kuitenkin käyttämään jo aiemmin kokeilemaani sovellusta nimeltään Postman.

Postman on sovellus rajapintojen testaukseen. Sovelluksesta on saatavilla maksullinen sekä ilmaisversio. Ilmaisversio sisältää kuitenkin varsin kattavat toiminnallisuudet testaamiseen. Postmanin käyttöön löytyy tarvittaessa kattava dokumentaatio heidän sivuiltaan.

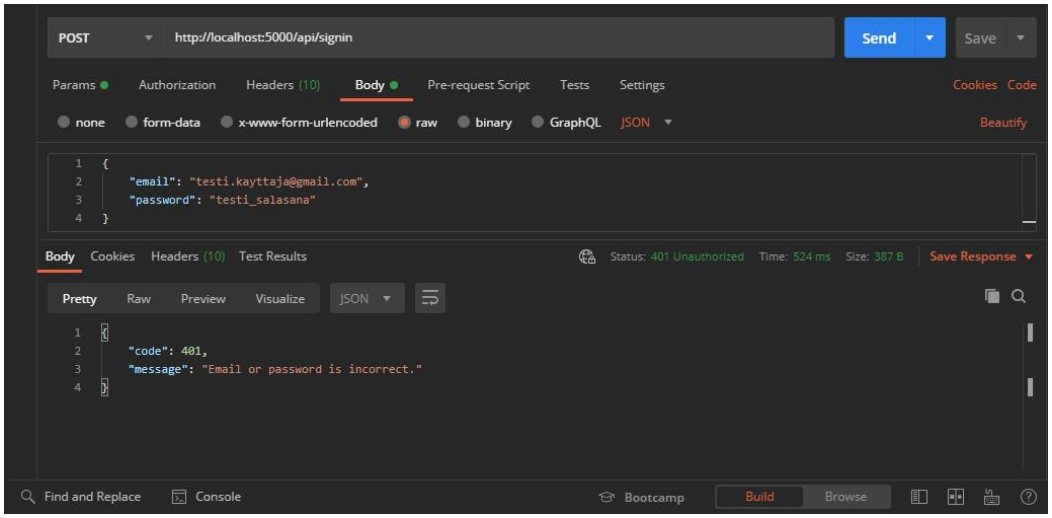
## 7.2 REST-rajapinnan testaus

Rajapinnan testausta suoritettiin kehittämisen ohella, jolloin pystyttiin havaitsemaan mahdolliset ohjelmointivirheet mahdollisimman aikaisessa vaiheessa.

Yksinkertaisimmillaan Postmanilla tehtiin palvelupyyntöjä rajapintaan ja katsottiin miten se reagoi erilaisiin tilanteisiin. (**Kuva 24 ja 25.**) Testien perusteella rajapintaa kehitettiin eteenpäin.




**Kuva 24.** Rajapinnan testausta Postman-sovelluksella.




**Kuva 25.** Rajapinnan testausta Postman-sovelluksella.

Postmanin avulla testattiin myös kirjautumista syöttämällä virheellistä kirjautumistietoa ja lähettämällä tämä POST-pyyntönä palvelimelle. Kirjautumistiedot sisällytetään pyynnön runkoon (*engl. body*) JSON-muodossa.

Rajapinta palauttaa tähän asianmukaisen virheilmoituksen. (**Kuva 25.**)

# 8 YHTEENVETO

Opinnäytetyö oli laajuudeltaan huomattavasti laajempi, kuin mitä alunperin ajattelin sen olevan, tarjoten kuitenkin sopivasti haastetta sekä uusien asioiden opiskelua.

Autentikointiprosessi olikin yksi näistä uusista asioista, joihin täytyi perehtyä hieman pintaa syvemmältä tämän opinnäytetyön edetessä. Päämääränä oli hyödyntää hyväksi havaittuja käytäntöjä ja toteuttaa näitä osana opinnäytetyötä. Koko autentikointi prosessi luotiin näiden käytäntöjen pohjalta.

Opinnäytetyön laajuuden vuoksi yksi vaatimusmäärittelyn toiminnallisuuksista täytyi pudottaa pois työn loppuvaiheessa aikataulun vuoksi. Kyseessä oli toiminnallisuus, jolla pystyttäisiin monitoroimaan rajapinnan liikennettä. Kyseinen toiminnallisuus olisi kuitenkin täysin mahdollista lisätä jälkikäteen.

Muilta osin lopputuote on vaatimusmäärittelyn mukainen.

## LÄHTEET

/1/ Prima Power. Viitattu 1.11.2020. https://www.primapower.com/fi/yritys/

/2/ RESTful Web Services-Resources. Viitattu 1.11.2020. https://www.tutorialspoint.com/restful/restful_resources.htm

/3/ Rajapinnat ja REST. Viitattu 1.11.2020. https://web-palvelinohjelmointi-19.mooc.fi/osa-6/4-rajapinnat-ja-rest

/4/ Node.js. Viitattu 1.11.2020. https://fi.wikipedia.org/wiki/Node.js

/5/ Node.js ja Express. Viitattu 1.11.2020. https://fullstackopen.com/osa3/node_js_ja_express

/6/ Writing middleware for use in Express apps. Viitattu 1.11.2020. https://expressjs.com/en/guide/writing-middleware.html

/7/ Microsoft Azure. Viitattu 1.11.2020. https://fi.wikipedia.org/wiki/Microsoft_Azure

/8/ Using HTTP cookies. Viitattu 1.11.2020. https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

/9/ Niemi, D, 2017. REST-rajapintojen testaus Postmanilla. Viitattu 1.11.2020. https://blog.nemit.fi/rest-rajapintojen-testaus-postmanilla-44c02cbb4661

/10/ Introducing JSON. Viitattu 8.11.2020. https://www.json.org/json-en.html

/11/ React (web framework). Viitattu 8.11.2020. https://en.wikipedia.org/wiki/React_(web_framework)

/12/ About npm. Viitattu 8.11.2020. https://docs.npmjs.com/about-npm

/13/ SQL. Viitattu 8.11.2020. https://fi.wikipedia.org/wiki/SQL

/14/ Cross-origin resource sharing. Viitattu 8.11.2020. https://en.wikipedia.org/wiki/Cross-origin_resource_sharing

/15/ Same-origin policy. Viitattu 8.11.2020. https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

/16/ Evästeet, sessiot ja tiedostot – luento 9. Viitattu 8.11.2020. http://appro.mit.jyu.fi/2007/kevat/sovellukset/luennot/luento9/

/17/ SQL Views. Viitattu 8.11.2020. https://www.w3schools.com/sql/sql_view.asp

/18/ SQL–Primary Key. Viitattu 8.11.2020. https://www.tutorialspoint.com/sql/sql-primary-key.htm

/19/ SQL-indexes. Viitattu 8.11.2020. https://www.tutorialspoint.com/sql/sql-indexes.htm

/20/ Hashing Passwords: One-Way Road to Security. Viitattu 8.11.2020. https://auth0.com/blog/hashing-passwords-one-way-road-to-security/