Janne Ingalsuo

KULUNVALVONNAN REST-RAJAPINTA

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Tieto- ja viestintätekniikka
Toukokuu 2021

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Toukokuu 2021	**Tekijä/tekijät** Janne Ingalsuo
Koulutus Insinööri (AMK), tieto- ja viestintätekniikka		☒ AMK ☐ YAMK
Työn nimi KULUNVALVONNAN REST-RAJAPINTA		
Työn ohjaaja Kyösti Marjakangas		**Sivumäärä** 29
Työelämäohjaaja -		

Kokkotyö-säätiön konkurssin myötä tätä työtä ei tehdä yritykselle. Sen takia aiheeksi valikoitui kulunvalvonnan REST-rajapinta.

Tavoitteena oli luoda kulunvalvonnan REST-rajapinta sekä käydä läpi REST:n vaatimuksia. Rajapinnan on vastaanotettava ja lähetettävä sekä käsiteltävä tietoa. Tieto on tallennettava palvelimella olevaan tietokantaan.

Touteuksessa käydään läpi laitteistoa ja luodaan tietokanta, johon tieto tallennetaan. REST toteutettiin PHP:llä sekä testattiin toimintaa Postman:lla.

Lukija saa käsityksen mitä vaatimuksia REST-rajapinnan pitää täyttää ja millaisella laitteistolla se voitaisiin toteuttaa edullisesti.

Rajapinnasta saatiin toimiva, mutta resurssit pitäisi nimetä uudelleen ja korjata yhden resurssin toiminta. Asiakaspuolen käyttöliittymä jäi toteutumatta aikataulun takia.

Asiasanat
REST-rajapinta, Postman

ABSTRACT

Centria University of Applied Sciences	**Date** May 2020	**Author** Janne Ingalsuo
Degree programme Bachelor of Science, Information and Communication Technology		
Name of thesis REST interface for access control		
Centria supervisor Kyösti Marjakangas		**Pages** 29
Instructor representing commissioning institution or company -		

With the bankruptcy of the Kokkotyö Foundation, this work not be done for the company. Therefore, the REST interface for access control was chosen as the topic.

The aim was to create an REST interface for access control, as well as to go through the requirements of REST. The interface must receive, send and process information. The information must be stored in a database on the server.

In the implementation, the hardware is reviewed, a database is created in which the information is stored. REST was implemented with PHP and tested with Postman.

The reader gets an idea of what requirements the REST interface must meet and with what devices it could be implemented advantageously.

The interface became functional, but resources should be renamed and the operation of one resource corrected. The client side interface did not materialize due to the schedule.

Key words
Rest interface, Postman

KÄSITTEIDEN MÄÄRITTELY

API

Application Programming Interface on ohjelmistovälittäjä, jonka avulla kaksi sovellusta voi puhua keskenään.

CRUD

(Create Read Update Delete) on neljä toimintoa, joilla lisätään, luetaan, päivitetään ja tuhotaan tietokannasta.

GDPR

General Data Protection Regulation on yleinen tietosuoja-asteus.

HATEOAS

Hypermedia as the Engine of Application State eli hypermedia sovellustilan moottorina. Yksinkertaisesti liikutaan sivulta toiselle linkkiä napsauttamalla.

JSON

JavaScript Onbject Notation on avoimen standardin tiedostomuota tiedonvälitykseen.

MariaDB

Relaatiotietokantajärjestelmä

MySQL

Tietokantapalvelu pilvipohjaisten sovellusten käyttöönottoon.

PHP

PHP: Hypertext Preprocessor komentosarjakieli verkkosivujen kehittämiseen.

phpMyAdmin

Ilmainen ohjelmistotyökalu, joka on luotu PHP:llä. Tukee suurta osaa MySQL- ja MariaDB-toimintoja.

POSTMAN

Yhteistyöalusta API-kehitykselle

Raspberry pi 3

Yhden piirilevyn ns. korttitietokone

RDBMS

Relational Database Management System on ohjelmistojärjestelmä, joka käyttää tavanomaista menetelmää tietojen luettelointiin, hakemiseen ja suorittamiseen.

REST

REpresentional State Transfer on serverillä oleva resurssi josta asiakas voi pyytää esityksen.

RFID

Radio-Frequency Identification radiotaajuustunnistus

URI

Uniform Resource Identifier on merkkijono, jota käytetään nimen tai resurssin tunnistamiseen.

XAMPP

Täysin ilmainen Apache-jakelu, joka sisältää MariaDB:n, PHP:n ja perlin.

1 JOHDANTO

Työssä tehtiin pieni kulunvalvonta REST:iä apuna käyttäen. REST valittiin, koska se on tällä hetkellä käytetyimpiä ohjelmistorajapintoja internetin maailmassa. Kulunvalvonta piti tehdä alun perin Kokkotyö-säätiölle, mutta säätiö meni konkursssiin. Työssä kiinnittiin siksi enemmän huomiota REST-rajapintaan ja sen testaamiseen..

Työssä tutkittiin, mitä kulunvalvontajärjestelmän toteuttaminen vaatisi. Työssä luodaan REST-rajapinta kulunvalvontaan, joka soveltuisi pienelle yritykselle. Pienen yrityksen ongelma on yleensä raha, joten luodaan halpa vaihtoehto yritykselle, jolla ei ole liikaa käyttää rahaa kalliisiin järjestelmiin. Kulunvalvonnassa käytetty laitteistokin pidetään mahdollisuuksien mukaan halpana, ja täten valikoitui Raspberry pi -korttitietokoneen ympärille rakentuva järjestelmä.

REST-rajapinta tehtiinn PHP:llä ja kulunvalvonnasta tulevat tiedot tallentuvat palvelimella olevaan tietokantaan. REST:in testaukseen kätytetään Postman API-kehitysalustaa.

2 KULUNVALVONTA

Kulunvalvonta on järjestelmä, jolla seurataan, kuka kulkee ja missä kulkee ja mahdollisesti mitä laitteita käyttää. Järjestelmät voivat koostua sähköisestä lukoista, ja sisälle pääsee näyttämällä korttia tai antamalla koodin, jonka jälkeen aika tallentuu tietokantaan. Tähän kuuluu myös ovien hallinta, jolla nähdään, mitkä ovet ovat auki tai kiinni ja lisäksi voidaan avata tietty ovi vaikka 10 minuutiksi tavarantoimittajalle.

Mikäli tietoja kerätään ja tallennetaan tietokantaan, sillointyöntekijän on annettava suostumus, että organisaatio voi kerätä tietoja.

GDPR (General Data Protection Regulation) eli yleistä tietosuoja-asetusta alettiin soveltamaan kaikissa EU-maissa keväällä 2018. Asetus antaa paremman suojan henkilötiedoilla ja keinoja hallita niiden käsittelyä. Organisaation on nimitettävä tietosuojavastaava mikäli käsitellään arkaluonteisia tietoja tai seurataan laajamittaisesti, säännöllisesti ja järjestelmällisesti ihmisiä. (Tietosuojavaltuutetun toimisto).

Organisaation on laadittava seloste henkilötietojen käsittelytoimista. Selosteesta on käytävä ilmi, mitä tietoja kerätään, miten ja mihin tarkoitukseen henkilötietoja käsitellään. (Tietosuojavaltuutetun toimisto).

Tähän työhön valikoitui seurannan osalta tulo- ja poistumisajan tallennus tietokantaan ja työntekijän tietojen tallennus tietokantaan sekä tietojen hakeminen tietokannasta ja päivittäminen. Laitteistoksi osalta valikoitui Raspberry Pi 3 ympärille rakentuva halpa laitteisto.

3 REST

REST (REpresentional State Transfer) tarkoittaa , että serverillä on resurssi ja asiakas voi pyytää esityksen resurssin tiedosta. Esimerkiksi resurssi on tietokantaan tallennettu blogi, jonka esitysmuoto voi olla yksinkertainen lista tietokannan tauluun tallennettuja arvoja. Tämä tarkoittaa, ettei asiakas välitä palvelimella olevan resurssin toteutuksesta. Asiakas välittää vain palvelimelta saamastaan esityksestä. (Lange 2016, 3.)

JSON (JavaScript Object Notation) on yleisesti käytetty resurssin esitysmuoto REST:ssä. Esimerkiksi blogiesitys voisi näyttää tältä JSON:lla :

```
{"title": "7 Things You Didn't Know about Star Wars",
"content": "George Lucas accidentally revealed that..."}
```

(Lange 2016, 4.)

JSON:n avain-arvopareja käytetään edustamaan objektin ominaisuutta. Tässä tapauksessa ominaisuuksia ovat First Name, Last Name, Age ja Profession. Jokaisella Ominaisuudella on siihen liittyvä arvo. Esimerkiksi First Name -arvo on Virender ja Age -arvo on 34. (Singh 2017.)

Avain-arvoparit erotetaan kaksoispisteellä, avain aina esitetään heittomerkeillä ja arvo voi olla mitä tahansa riippuen tietotyypistä. Tietotyypit ovat Boolean (True, False), numero, objekti (avain-arvoparien assosiatiivinen taulukko) ja taulukko (arvojen assosiatiivinen taulukko). (Singh 2017.)

```
{
"FirstName": "Virender",
"LastName": "Singh"
"Age": 34,
"Profession": "Engineer"
}
```

JSON-objekti esitetään Avain-arvoparien kokoelmana. Tämä kokoelma avain-arvopareja on ryhmitelty aaltosulkeilla. Avain-arvoparit erotetaan pilkulla, objekti alkaa aaltosulkeella { ja päättyy aaltosulkeella }. (Singh 2017.)

JSON-taulukko on kokoelma arvoja eroteltuna pilkulla. Taulukko alkaa hakasulkeella [ja päättyy hakasulkeella] ja arvot erotetaan pilkulla. (Singh 2017.)

```
{
"FirstName": "Virender",
"LastName": "Singh"
"Age": 34,
"Profession": "Engineer",
“Hobbies”: [“Games”, “Computers”, “Music”]
}
```

(Singh 2017).

3.1 REST metodit

CRUD (Create, Read, Update, Delete) on neljä päätoimintoa, joita käytetään vuorovaikutuksessa tietokantasovellusten kanssa. Monilla ohjelmointikielillä ja -protokollilla on oma vastaavuus. SQL on suosittu kieli, joka on vuorovaikutuksessa tietokantojen kanssa. SQL:n neljä toimintoa ovat Insert, Select, Update ja Delete. (Bush 2020.)

Seuraavia HTTP metodeita on yleisesti käytetty REST-pohjaisessa arkkitehtuurissa. Resurssi luetaan metodilla GET. Luodaan uusi resurssi metodilla POST. Poistetaan resurssi metodilla DELETE. Päivitetään resurssi metodilla PUT. (Tutorialspoint.)

3.2 Asiakas ja palvelin

Asiakas ja palvelin on ensimmäinen rajoite kuudesta. Järjestelmän on koostuttava asiakkaasta ja palvelimesta. Palvelimella on resurssi, jota asiakas haluaa käyttää. Esimerkiksi palvelimella on lista hinnoista jotka asiakas haluaa näyttää taulukossa. Palvelin hoitaa viestien vastaanottamisen, tallentamisen ja tietokannan ylläpitämisen. Asiakas hoitaaa viestin lähettämisen ja käyttöliittymät. (Lange, 2016 5.)

3.3 Tilattomuus

Asiakkaan ja palvelimen viestinnän on oltava tilatonta. Asiakas pitää istuntonsa tiedon itsellään ja palvelin ei tiedä mitään eli siellä ei pitäisi olla evästeitä, istuntomuuttujia tai muita tilallisia ominaisuuksia. (Lange 2016, 6.)

3.4 Välimuisti

Palvelimelta saadut vastaukset merkitään tallenettavaksi tai ei tallennettavaksi välimuistiin. Välimuistiin tallentaminen vähentää asiakkaan ja palvelimen välistä vuorovaikutusta, mikä parantaa järjestelmän suorituskykyä. (Lange 2016, 7.)

3.5 Yhtenäinen käyttöliittymä

Selaimella voidaan lukea uutisia ja hoitaa pankkiasiat samalla selaimella vaikka ne ovatkin erinlaisia sovelluksia. Tämä voidaan tehdä, koska yhtenäinen käyttöliittymä irroittaa käyttöliittymän toteutuksesta, mikä tekee vuorovaikutuksesta niin yksinkertaista , että jokainen , joka tuntee tyylin voi nopeasti ymmärtää käyttöliittymän. (Lange 2016, 8.)

3.5.1 Resurssien tunnistaminen

REST-tyyli keskittyy resursseihin. Resurssi on pohjimmiltaan kaikki, mikä voidaan nimetä. Tämä on melko laaja määritelmä, johon voi sisältyä staattisia kuvia ja reaaliaikaisia syötteitä. (Lange 2016, 8.)

Jokainen RESTful-mallin resurssi on oltava yksilöitävissä URI:n (Uniform Resource Identifier) kautta ja tunnisteen on oltava vakaa vaikka taustalla olevaa resurssia päivitettäisiin. Tämä tarkoittaa, että jokaisella resurssilla on oltava oma URI. (Lange 2016, 8.)

Yleinen malli on käyttää alla olevaa URI-tyyliä, jolla pääsee kokoelma-aineistoon autoista.

```
https://api.autokauppa.com/cars
```

Resurssin nimi on yleensä monikkomuoto, koska se on resurssikokoelma. Kyselyparametria käytetään, jos halutaan etsiä vain tietyt kohteet kokoelmasta. Alla oleva esimerkki hakee kaikki, joiden nimi on Maserati.

```
https://api.autokauppa.com/cars?name=Maserati
```

3.5.2 REST-resurssien käsittely esityksen kautta

Asiakas ei käsittele suoraan palvelimen resurssia. Asiakkaan ei siis sallita suorittaa SQL-käskyjä tietokantatauluihin. Sen sijaan palvelin paljastaa resurssin tilan, mikä tarkoittaa resurssin tiedon näyttämistä neutraalissa muodossa. Yleisin REST-resurssin esitysmuoto on JSON.

Kun asiakas haluaa päivittää resurssin, se saa resurssin esityksen palvelimelta. Asiakas päivittää esityksen uusilla tiedoilla ja lähettää päivitetyn esityksen palvelimelle, sekä pyytää palvelinta päivittämmän resurssin, että se vastaa lähetettyä esitystä. (Lange, 2016, 11 -12).

3.5.3 Itsekuvaavat viestit

Pyynnössä ja vastausviestissä on oltava tarpeeksi tietoa, jotta vastaanottaja ymmärtää sen erikseen. Jokaisella viestillä on oltava mediatyyppi (application/json), joka kertoo vastaanottajalle miten viesti jäsennetään. HTTP:tä ei virallisesti vaadita RESTful-verkkopalveluille. HTTP:tä käyttäessä metodit ovat GET, POST, PUT ja DELETE.

Asiakas voi käyttää esimerkiksi taulukossa 1 esitettyjä menetelmiä. Selkeästi määritettyjen HTTP-menetelmien etu on, jos API-kayttäjä tietää HTTP:n muttei järjestelmäämme voi arvata palvelun toiminnan katsomalla vain HTTP-menetelmää ja URI-polkua. (Lange 2016, 12–14.)

TAULUKKO 1. HTTP-menetelmät ja URI-polut. (Lange, 2016, 14.)

Tehtävä	Menetelmä	Polku
Lisää asiakas	POST	/customers
Poista asiakas	DELETE	/customers/{id}
Tulosta asiakkaan tiedot	GET	/customers/{id}
Etsi asiakkaat	GET	/customers
Päivitä asiakkaan tiedot	PUT	/customers/{id}

3.5.4 Hypermedia sovellustilan moottorina

HATEOAS (Hypermedia as the Engine of Application State) eli hypermedia sovellustilan moottorina tarkoittaa sitä, että verkkosivu on sovellustilan ilmentymä ja hypermedia on tekstiä hyperlinkkien kanssa. Hypermedia toimii sovellustilan moottorina. Toisin sanoen se tarkoittaa vain sitä, että siirrymme linkkiä napsauttamalla uusille sivuille. (Lange, 2016, 15).

Surfatessa netissä käytetään HATEOAS:ta. Rajoitus sanoo, että pitäisi käyttää linkkejä navikoidessamme sovelluksessa. (Lange 2016, 15.)

3.6 Kerroksinen järjestelmä

Asiakkaan tulisi tietää vain se välitön kerros, jonka kanssa hän on yhteydessä, eikä hänen tarvitse tietää muista kerroksista, jotka ovat tai eivät ole palvelimen takana tämä tarkoittaa sitä, että asiakkaan ei tarvitse tietää puuhuuko hän väli tai varsinaisen palvelimen kanssa. (Lange 2016, 17.)

Välityspalvelin sijoitettaisiin asiakkaan ja palvelimen väliin, se ei vaikuta heidän viestintään eika tarvitse päivittää asiakas- tai palvelinkoodia. Voimme lisätä suojauksen kerroksena verkkopalveluihin ja erottaa siten liiketoimintalogiikan selkeästi suojauslogiikasta. Tarkoittaa myös, että palvelin voi soittaa useille muille palvelimille vastauksen tuottamiseksi asiakkaalle. (Lange 2016, 17.)

3.7 Ladattava koodi (Code-On-Demand)

Ladattava koodi tarkoittaa, että palvelin voi laajentaa asiakkaan toimintoja ajon aikana lähettämällä koodia , joka sen tulisi suorittaa esimerkiksi Java Applets tai JavaScript. (Lange 2016, 18–19.)

Nettisivuilla koodin lataaminen on hyvin yleistä. Selain lataa JavaScriptiä riipuen siitä minkä sivun avaa ja siten laajentaa selaimen toiminnallisuutta. (Lange 2016, 19.)

4 TIETOKANTA

Tietokanta on jäsennelty tietetokokoelma, joka on järjestetty helppoa käyttöä ja hakua varten. Tietokanta jaetaan useisiin erillisiin tallennusalueisiin, joita kutsutaan tauluiksi. (Kinsta Inc., 2021).

Tietokannan taulussa on aina yksi pääavain, jota kutsutaan nimellä primary key, joka ei salli samoja arvoja. Tauluja voidaan linkittää toisiinsa vierasavaimella (foreign key), ja kun taulut on linkitetty yhteen voidaan samalla kyselyllä hakea molemnpien taulujen tiedot ja yhdistää ne.

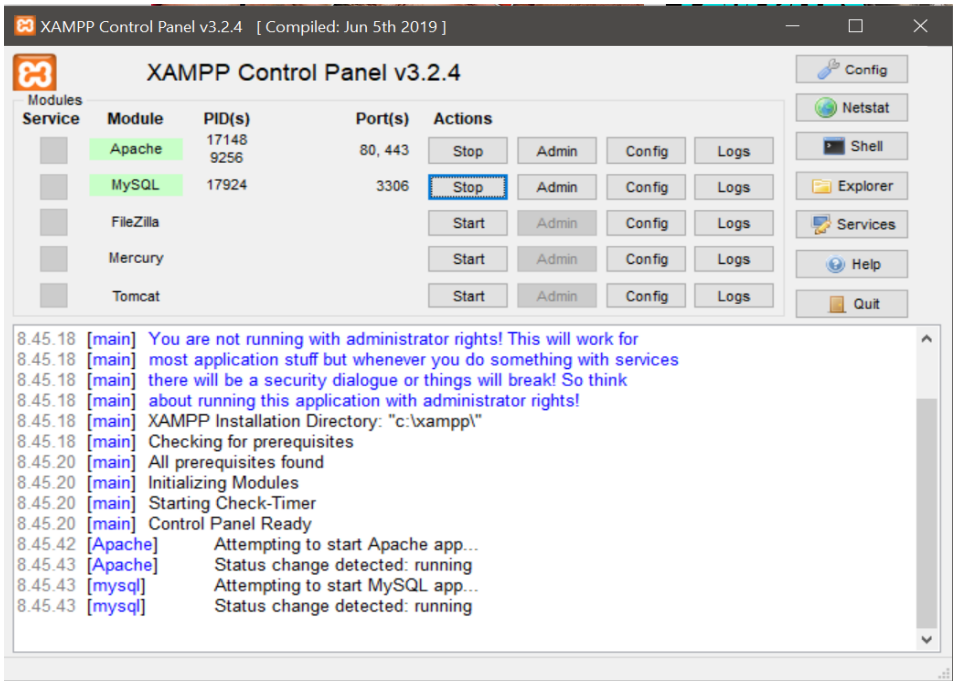
4.1 MySQL

MySQL on avoimenlähdekoodin relaatiotietokantojen hallintajärjestelmä. MySQL on järjestelmä, joka voi tallentaa ja hakea tietoja. MySQL käynnistettiin vuonna 1995. Se on käynnyt läpi muutaman omistajuuden, ennen kuin päätyi Oracle Corporationiin vuonna 2010. Nimi tulee yhdistelmästä “My”, joko on perustajan tyttären nimi ja SQL, joka taas tarkoittaa Structured Query Language, joka on ohjelmointikieli, joka auttaa hallitsemaan relaatiotietokannan tietoja. (Kinsta Inc. 2021.)

4.2 XAMPP

Pienellä harjoittelulla kehitysympäristön peruskäyttö on suhteellisen helppoa. Luodaan nettisivu htdocs-kansioon ja käynnistetään Apache XAMPP-hallintapaneelista (KUVA 1.). Kirjoitetaan selaimen kenttään http:\\localhost\”nettisivu.php”.

XAMPP on suosituin PHP-kehitysympäristö (PHP: Hypertext Preprocessor) kehitysympäristö. Ilmainen Apache-jakelu sisältää MariaDB, PHP ja Perlin. (Apache friends.)



KUVA 1. XAMPP-hallintapaneeli

Perl on korkean tason yleiskäyttöinen, tulkittu dynaaminen ohjelmointikieli. Perl on termi, joka tarkoittaa Practical Extraction and Reporting Language vaikka perl:lle ei ole varsinaista lyhennettä. Sen esitteli Larry Wall 1987. (GURU99.)

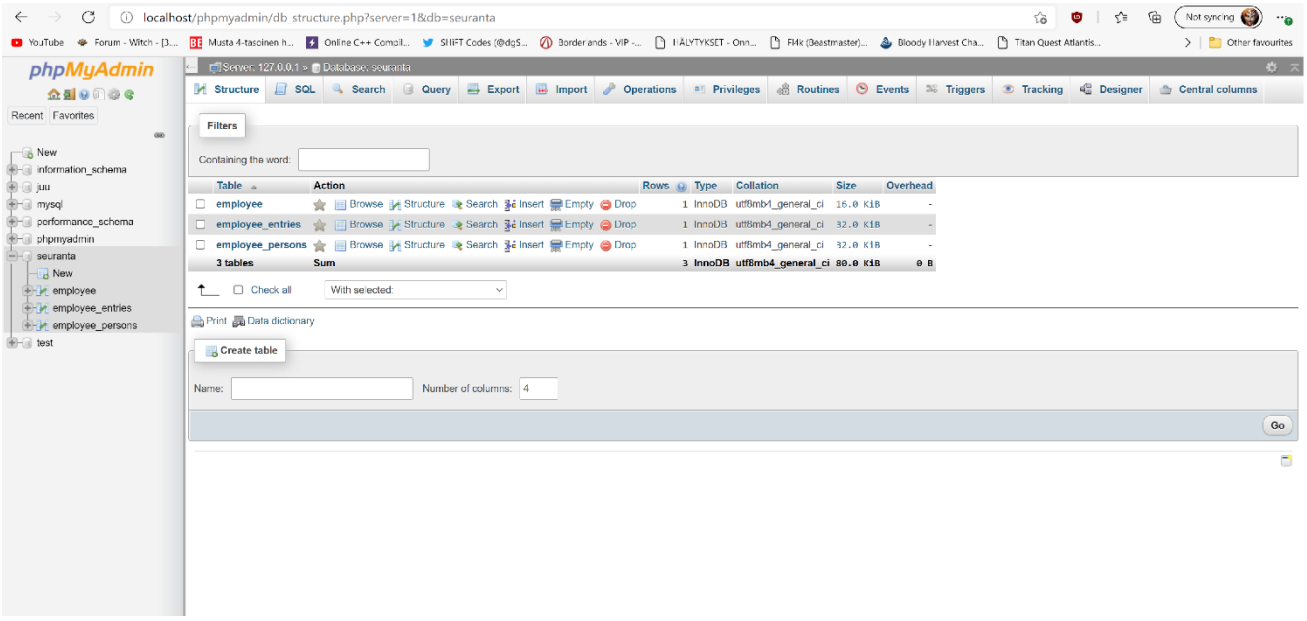
4.3 MariaDB

MariaDB Server on relaatiotietokantojen hallintajärjestelmä, joka on paranneltu versio MySQL:stä. MySQL:n voi korvata helposti MariaDB:llä, koska MariaDB tarjoaa samat ominaisuudet kuin MySQL. (Kinsta Inc. 2021.)

MariaDB Server on yksi suosituimmista avoimen lähdekoodin relaatiotietokantajärjestelmä. Sen ovat tehneet MySQL:n kehittäjät ja MariaDB pysyy avoimena lähdekoodina. (MariaDB Foundation).

4.4 phpMyAdmin

Ilmainen ohjelmisto työkalu (KUVA 2.), joka tarjoaa käyttöliittymän tietokantojen ja niiden taulujen hallintaan. PhpMyAdmin tukee suurinta osaa MySQL ominaisuuksista. Yleisimpiä toimintoja ovat tietokannan hallinta, taulujen hallinta, taulujen suhteet, indeksointi ja käyttäjien hallinta. (phpMyAdmin.)



KUVA 2. phpMyAdmin-työkalu.

4.5 PHP

PHP on yksi mailman suosituimmista ohjelmointikielista verkkosivujen kehittämiseen. Vuonna 2009 PHP:tä käytettiin yli 20 miljoonassa nettisivussa ja yli kolmanneksessa maailman verkkopalvelimissa. (Vaswani 2009, 4.)

5 TOTEUTUS

Työn REST toteutetaan suurimmaksi osin PHP:llä. Olen itse käyttänyt PHP:tä netisivujen koodauksen yhteydessä, joten se on luonnollinen valinta pääkieleksi. Tietokantojen yhteydet ja hallinta onnistuu PHP:llä, kun käyttää vähän aikaa opettelemiseen.

Raspberry Pi 3 liitetään RFID-lukija, jolle näytetään RFID-tagia ja kuuluu piippaus. Tagia on näytetty lukijaan ja aika vilahtaa tekstikentässä, ja näytöllä on ilmoitus onnistuneesta tai epäonnistuneesta leimauksesta. Tekstikentässä olevaa numerosarjaa verrataan tietokannassa olevaan numerosarjaan ja mikäli numerosarjat ovat samat aika tallentuu tietokantaan.

5.1 Laitteisto

Järjestelmän päälaitteet ovat Raspberry Pi 3, näyttö tai Raspberry Pi:lle asennettu kosketusnäyttö, palvelin, RFID-tagi tai avaimenperä ja RFID-lukija. Palvelin ei ole pakollinen ja sen tilalla voi olla vaikka työasema, jossa XAMPP ja tietokanta sijaitsevat. Näyttökään ei periaatteessa ole pakollinen, mutta mahdollisten virhetilanteiden takia sitä olisi suositeltavaa käyttää.

5.1.1 Raspberry pi 3

Raspberry Pi on yhden piirilevyn tietokone. Raespberry Pi 3:n suorituskyky riittää hyvin, koska avattuna on vain selain. Selaimessa on nettisivu, jossa on yksinkertaisesti vain tekstikenttä, johon luettu tagin numerosarja vilahtaa. Tässä työssä käytetään Raspberry Pi 3 (KUVA 3 ja TAULUKKO 2.).

Taulukko 2. Raspberry Pi 3 Model B+ ominaisuuksia. (Raspberry Pi Foundation.)

RaspBerry Pi 3 Model B+	
Prosessori	1.4 GHz 64-bit neliytiminen
Verkko	Kaksikaistainen langaton lähiverkko, ethernet
Bluetooth	Bluetooth 4.2
Muisti	1Gt RAM
Portit	HDMI, 4 x usb

KUVA 3. Raspberry Pi 3 Model B+.

5.1.2 RFID-tagi

Koostuu antennista ja RFID-piiristä. Antenni lähettää ja vastaanottaa signaaleja. RFID-siru tai integroitu piiri varastoi ID-numeron ja muita tietoja. Tagi lähettää tiedon radioaaltojen kautta. Tagi vastaanottaa energiaa. Tagia näytetään lukijaan, ja kun tagi saa lähetyksen lukijasta niin energia menee sisäisen antennin kautta ja aktivoi piirin. (atlasRFIDstore, 2021.)

RFID-tagina (KUVA 4.) käytetään passiivista tagia, jonka lukuetäisyys on pieni. Tagin on oltava ns. lähikontaktissa lukijaan. Tagissa olevassa piirissä on neljä muistipankkia, jotka sisältävät tietoa tagista. (atlasRFIDstore, 2021.)

KUVA 4. RFID-tagi / avaimenperä.

5.1.3 RFID-lukija

RFID-lukija (KUVA 5.) liitetään USB-porttiin, ja kun tagia näytetään lukijaan niin tagi lähettää numerosarjan lukijan kautta näytölle.

Taulukko 3. RFID-lukijan tietoja. (Ihmevekotin.fi 2011.)

RFID-lukija	
Taajuus	125 kHz
Sirut	Tukee mm siruja : EM4001, EM4100, EM4200, TK4100
Tiedonsiirtonopeus	106 kbps
lukuetäisyys	5 – 8 cm
Muuta	Lukee ensimmäiset 10 -merkkiä, ei tarvitse ajureita

KUVA 5. RFID-lukija.

5.2 Tietokanta

Tietokanta luodaan phpMyAdmin-työklalulla, joka tarjoaa graaffisen käyttöliittymän. Tietokannan luominen tapahtuu CREATE DATABASE-komennolla käyttöliittymän SQL-välilehdessä olevaan tekstikenttään.

```
CREATE DATABASE seuranta ;
```

Tietokannan taulujen luonti CREATE TABLE-komennolla SQL-välilehdessä olevaan tekstikenttään.

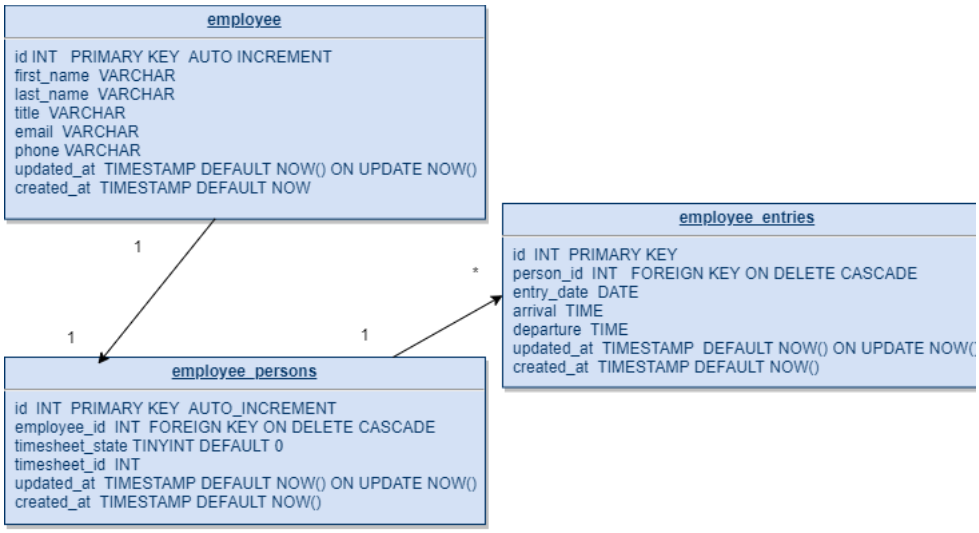
```
CREATE TABLE employee (
    id INT(10) PRIMARY KEY AUTO_INCREMENT NOT NULL,
    first_name VARCHAR(10) NOT NULL,
    last_name VARCHAR(20) NOT NULL,
    title VARCHAR(20) NOT NULL,
    email VARCHAR(20),
    phone VARCHAR(10),
    updated_at TIMESTAMP NOT NULL DEFAULT NOW() ON UPDATE NOW(),
    created_at TIMESTAMP NOT NULL DEFAULT NOW()
    );

CREATE TABLE employee_persons (
     id INT(10) PRIMARY KEY AUTO_INCREMENT NOT NULL,
     employee_id INT, FOREIGN KEY(employee_id) REFERENCES employee(id) ON DELETE CASCADE,
     timesheet_state TINYINT(1) DEFAULT 0,
     timesheet_id INT(10),
     updated_at TIMESTAMP NOT NULL DEFAULT NOW() ON UPDATE NOW(),
     created_at TIMESTAMP NOT NULL DEFAULT NOW()
     );

CREATE TABLE employee_entries (
     id INT(10) PRIMARY KEY AUTO_INCREMENT NOT NULL,
     person_id INT, FOREIGN KEY(person_id)REFERENCES employee_persons(employee_id) ON DELETE CASCADE,
     entry_date date,
     arrival time,
     departure time,
     updated_at TIMESTAMP NOT NULL DEFAULT NOW() ON UPDATE NOW(),
     created_at TIMESTAMP NOT NULL DEFAULT NOW()
     );
```

Ensimmäisessä taulussa on työntekijän tiedot. Toisessa taulussa on tagin id timesheet_id ja leimatessa timesheet_state muuttuu nollasta ykköseksi. Kolmanteen tauluu tallennetaan saapumis- ja poistum-isaika.

Kuvassa 6 tietokannan rakenne, josta käy ilmi miksi poistettaessa employee-taulusta työntekijä poistuu myös muista taulusta työntekijän tiedot. Kaikissa tauluissa pääavaimena (PRIMARY KEY) on id. Employee-taulu on linkitetty employee_persons-taulun vierasavaimeen (FOREIGN KEY) employee_id, sekä employee_persons-taulu on linkitetty employee_entries-taulun vierasavaimeen person_id. Kaikissa vierasavaimissa on ON DELETE CASCADE-määritys, jonka takia muidenkin taulujen tiedot poistuvat, jos työntekijä poistetaan employee-taulusta.



KUVA 6. Tietokannan rakenne ja tietotyypit

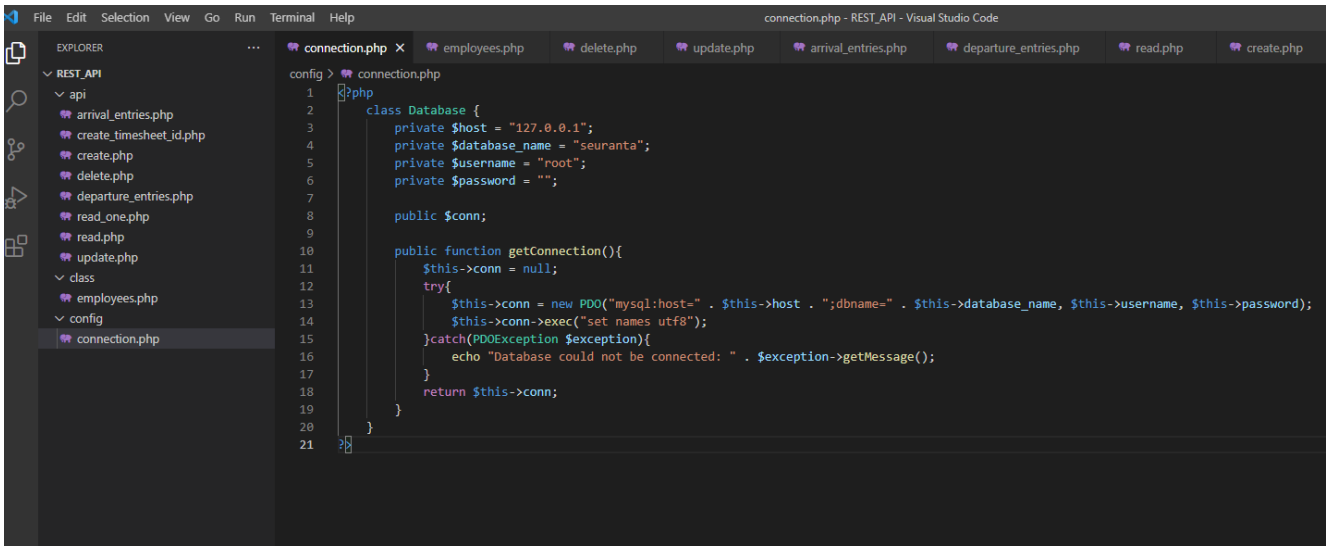
5.3 REST

Luodaan ensiksi config-kansioonconnection.php, josta löytyy tiedot tietokantayhteyden muodostamiseen. Seuraavaksi luotiin class-kansioon employees.php, josta löytyy kaikki funktiot joita tarvitaan tiedon tulostamiseen, päivittämiseen, tuhoamiseen tai lukemiseen tietokannasta. Api-kansiossa on URI-tiedostot, joilla lisätään, tuhotaan, päivitetään, luetaan tai tulostetaan tietoa tietokannasta. Taulukossa 4 on kaikki työn URI-polut.

TAULUKKO 4. Menetelmät ja URI-polut

Tehtävä	**Menetelmä**	**Polku**
Lisää työntekijä	POST	localhost/REST_API/api/create
Poista työntekijä	DELETE	localhost/REST_API/api /delete
Tulosta työntekijän tiedot	GET	localhost/REST_API/api /read_one/{id}
Tulosta kaikki työntekijöiden tiedot	GET	localhost/REST_API/api /read
Päivitä työntekijän tiedot	UPDATE	localhost/REST_API/api /update
Lisätään tagin numerosarja	POST	localhost/REST_API/api /create_time-sheet_id
Lisätään saapumisaika tietokantaan	POST	localhost/REST_API/api /arrival_entries
Lisätään poistumisaika tietokantaan	UPDATE	localhost/REST_API/api /departure_entries

Luotiin ensiksi config-kansioon connection.php, jotta saadaan yhteys tietokanataan (KUVA 7.).



KUVA 7. Database class, jossa tarvittavat tiedot tietokantayhteyden muodostamiseen.

Luotiin PHP-luokka class-kansioon, jossa on kaikki CRUD-toiminnot. Kuvassa 8 on funktio, jonka avulla lähetetään työntekijän tiedot tietokantaan. Kuvassa 9 oleva create.php, jonka avulla voidaan syöttää tiedot JSON:lla.

```php
// CREATE
public function createEmployee(){
    $sqlQuery = "INSERT INTO ". $this->db_table ." SET first_name = :first_name, last_name = :last_name, title = :title, email = :email, phone = :phone";

    $stmt = $this->conn->prepare($sqlQuery);

    // sanitize
    $this->first_name=htmlspecialchars(strip_tags($this->first_name));
    $this->last_name=htmlspecialchars(strip_tags($this->last_name));
    $this->title=htmlspecialchars(strip_tags($this->title));
    $this->email=htmlspecialchars(strip_tags($this->email));
    $this->phone=htmlspecialchars(strip_tags($this->phone));

    // bind data
    $stmt->bindParam(":first_name", $this->first_name);
    $stmt->bindParam(":last_name", $this->last_name);
    $stmt->bindParam(":title", $this->title);
    $stmt->bindParam(":email", $this->email);
    $stmt->bindParam(":phone", $this->phone);

    if($stmt->execute() ){
        return true;
    }
    return false;
}
```

KUVA 8. Työntekijän perustiedot funktio createEmployee().

create.php

```php
<?php
    header("Access-Control-Allow-Origin: *");
    header("Content-Type: application/json; charset=UTF-8");
    header("Access-Control-Allow-Methods: POST");
    header("Access-Control-Max-Age: 3600");
    header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");

    include_once '../config/connection.php';
    include_once '../class/employees.php';

    $database = new Database();
    $db = $database->getConnection();

    $item = new Employee($db);

    $data = json_decode(file_get_contents("php://input"));

    $item->first_name = $data->first_name;
    $item->last_name = $data->last_name;
    $item->title = $data->title;
    $item->email = $data->email;
    $item->phone = $data->phone;

    if($item->createEmployee()){
        echo 'Employee created successfully.';
    } else{
        echo 'Employee could not be created.';
    }
?>
```

KUVA 9. Tallennetaan annetut tiedot tietokantaan.

Poistetaan työntekijä ja kaikki tiedot tietokannasta (KUVA 10.). Kaikki tiedot poistuvat tietokannasta, koska tietokannassa viitataan toisiin tauluihin.

```php
// DELETE
function deleteEmployee(){
    $sqlQuery = "DELETE FROM " . $this->db_table . " WHERE id = ?";
    $stmt = $this->conn->prepare($sqlQuery);

    $this->id=htmlspecialchars(strip_tags($this->id));

    $stmt->bindParam(1, $this->id);

    if($stmt->execute()){
        return true;
    }
    return false;
}
```

KUVA 10. deleteEmployee() poistaa tiedot tietokannasta

Kuvassa 11 poistetaan työntekijä JSON:lla annetulla id:llä. Poistetun työntekijän kaikki tiedot poistuvat tietokannassa olevien viittauksien takia.

```php
<?php
    header("Access-Control-Allow-Origin: *");
    header("Content-Type: application/json; charset=UTF-8");
    header("Access-Control-Allow-Methods: POST");
    header("Access-Control-Max-Age: 3600");
    header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");

    include_once '../config/connection.php';
    include_once '../class/employees.php';

    $database = new Database();
    $db = $database->getConnection();

    $item = new Employee($db);

    $data = json_decode(file_get_contents("php://input"));

    $item->id = $data->id;

    if($item->deleteEmployee()){
        echo json_encode("Employee deleted.");
    } else{
        echo json_encode("Data could not be deleted");
    }
?>
```

KUVA 11. Poistetaan työntekijä ja tiedot annetulla id:llä

Kuvassa 12 on funktio, joka päivittää työntekijä perustiedot tietokannan employee-tauluun.

```php
// UPDATE
public function updateEmployee(){
    $sqlQuery = "UPDATE ". $this->db_table ." SET first_name = :first_name, last_name = :last_name, title = :title, email = :email, phone = :phone WHERE  id = :id";

    $stmt = $this->conn->prepare($sqlQuery);

    // sanitize
    $this->id=htmlspecialchars(strip_tags($this->id));
    $this->first_name=htmlspecialchars(strip_tags($this->first_name));
    $this->last_name=htmlspecialchars(strip_tags($this->last_name));
    $this->title=htmlspecialchars(strip_tags($this->title));
    $this->email=htmlspecialchars(strip_tags($this->email));
    $this->phone=htmlspecialchars(strip_tags($this->phone));

    // bind data
    $stmt->bindParam(":id", $this->id);
    $stmt->bindParam(":first_name", $this->first_name);
    $stmt->bindParam(":last_name", $this->last_name);
    $stmt->bindParam(":title", $this->title);
    $stmt->bindParam(":email", $this->email);
    $stmt->bindParam(":phone", $this->phone);

    if($stmt->execute()){
       return true;
    }
    return false;
}
```

KUVA 12. updateEmployee() päivittää työntekijän tiedot.

Kuvassa 13 annetaan ne tiedot , jotka halutaan päivittää JSON:lla.

```php
<?php
    header("Access-Control-Allow-Origin: *");
    header("Content-Type: application/json; charset=UTF-8");
    header("Access-Control-Allow-Methods: POST");
    header("Access-Control-Max-Age: 3600");
    header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");

    include_once '../config/connection.php';
    include_once '../class/employees.php';

    $database = new Database();
    $db = $database->getConnection();

    $item = new Employee($db);

    $data = json_decode(file_get_contents("php://input"));

    $item->id = $data->id;

    // employee values
    $item->first_name = $data->first_name;
    $item->last_name = $data->last_name;
    $item->email = $data->email;
    $item->title = $data->title;
    $item->phone = $data->phone;

    if($item->updateEmployee()){
        echo json_encode("Employee data updated.");
    } else{
        echo json_encode("Data could not be updated");
    }
?>
```

KUVA 13. Päivitetään tiedot.

Haetaan SQL-lauseessa määritetyt tiedot tietokannasta (KUVA 14.).

```php
// GET ALL
public function readEmployees(){
    $sqlQuery = "SELECT employee.id, first_name, last_name, email, phone, title, timesheet_id, employee.created_at FROM " . $this->db_table .
    " JOIN " . $this->db_table2 . " ON employee.id = employee_persons.employee_id";
    $stmt = $this->conn->prepare($sqlQuery);
    $stmt->execute();
    return $stmt;
}
```

KUVA 14. readEmployees() tulostaa kaikkien työntekijöiden tiedot.

Tulostetaan kaikki työntekijät ja tiedot JSON-muotoon (KUVA 15.).

```php
<?php
    header("Access-Control-Allow-Origin: *");
    header("Content-Type: application/json; charset=UTF-8");

    include_once '../config/connection.php';
    include_once '../class/employees.php';

    $database = new Database();
    $db = $database->getConnection();

    $items = new Employee($db);

    $stmt = $items->readEmployees();
    $itemCount = $stmt->rowCount();

    echo json_encode($itemCount);

    if($itemCount > 0){

        $employeeArr = array();
        $employeeArr["body"] = array();
        $employeeArr["itemCount"] = $itemCount;

        while ($row = $stmt->fetch(PDO::FETCH_ASSOC)){
            extract($row);
            $e = array(
                "id" => $id,
                "first_name" => $first_name,
                "last_name" => $last_name,
                "title" => $title,
                "email" => $email,
                "phone" => $phone,
                "timesheet_id" => $timesheet_id,
                "cretaed_at" => $created_at
            );

            array_push($employeeArr["body"], $e);
        }
        echo json_encode($employeeArr);
    }

    else{
        http_response_code(404);
        echo json_encode(
            array("message" => "No record found.")
        );
    }
?>
```

KUVA 15. Tulostetaan kaikki tiedot.

6 TESTAUS

Testauksessa testataan REST API:n toimintaa. Aikataulun tiukkuudesta johtuen käyttöliittymä jäi toteutumatta ja sitä ei voida testata. Raspberry Pi 3 toimii lukijan kanssa päätelaitteena.

Teoreetttisesti leimaustapahtuman kulku olisi : käyttäjä saapuu päätelaitteelle ja asettaa RFID-tagin lukijan päälle tai läheisyyteen. RFID-tagin numerosarja vilahtaa tekstikentässä ja näytössä on ilmoitus onnistuneesta tai epäonnistuneesta leimauksesta

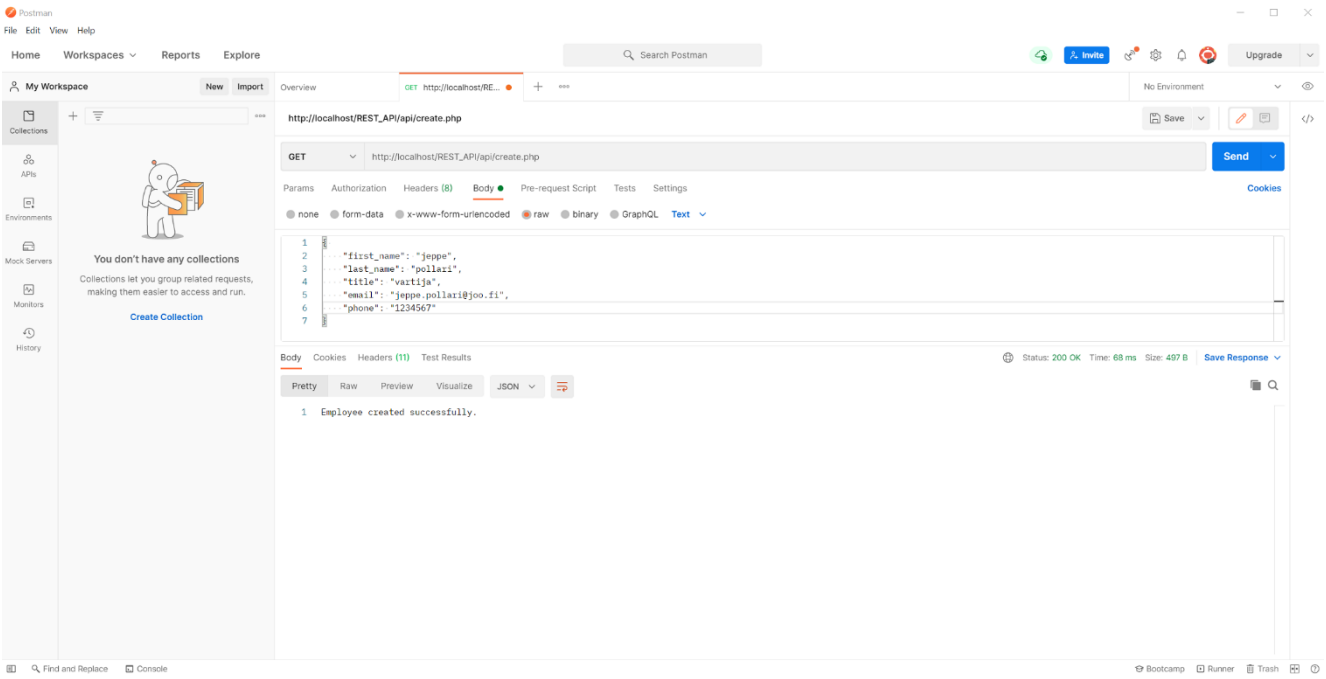
6.1 Postman

Postman on yhteistyöalusta API-kehitykselle. Postmanin ominaisuudet yksinkertaistavat sovellusliittymän rakentamisen vaiheita ja tehostavat yhteistyötä, jotta voidaan luoda parempia sovellusliittymiä nopeammin. (Postman, Inc, 2021).

Testaamisessa käytetään Postmanin työpöytäsovellsuta. Sovelluksella voidaan testata URI-polkujen toimintaa antamalla body-kenttään JSON:lla työntekijän tiedot jotka haluttaisiin esimerkiksi päivittää update.php URI:lla.

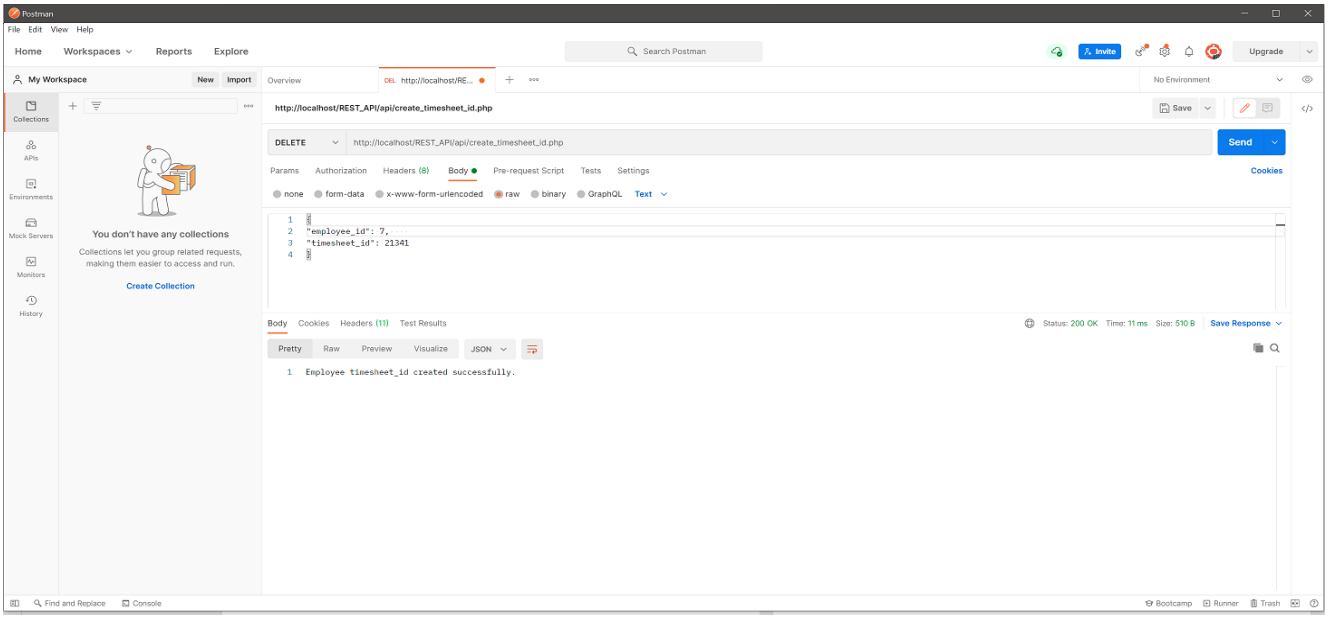
6.2 REST

Kuvassa 16 testataan työntekijän luominen tietokantaan kirjoittamalla http://localhost/REST_API/api/create.php ja antamalla tiedot bodyn tekstikenttään. Vastauksena saatiin employee created successfully sekä status 200 OK. Työntekijä luotiin onnistuneesti tietokantaan.



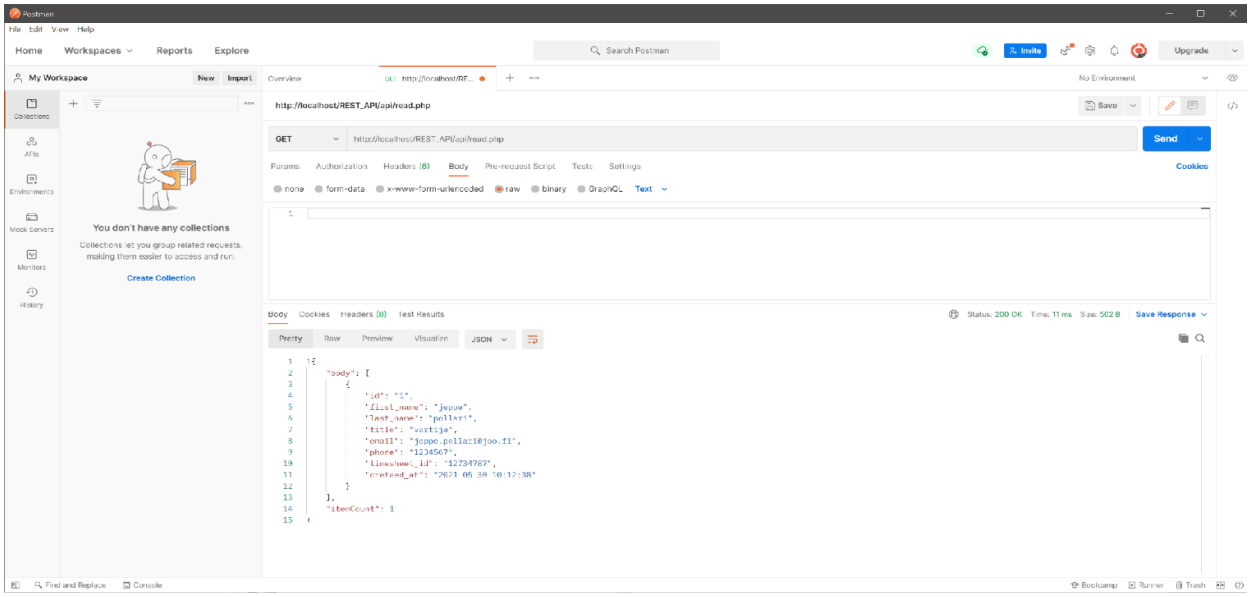
KUVA 16. Työntekijän luominen tietokantaan.

Kuvassa 17 testataan timesheet_id:n antaminen kirjoittamalla http://localhost/REST_API/api/create_timesheet_id.php ja antamalla tieto body:n tekstikenttään. Id luotiin onnistuneesti.



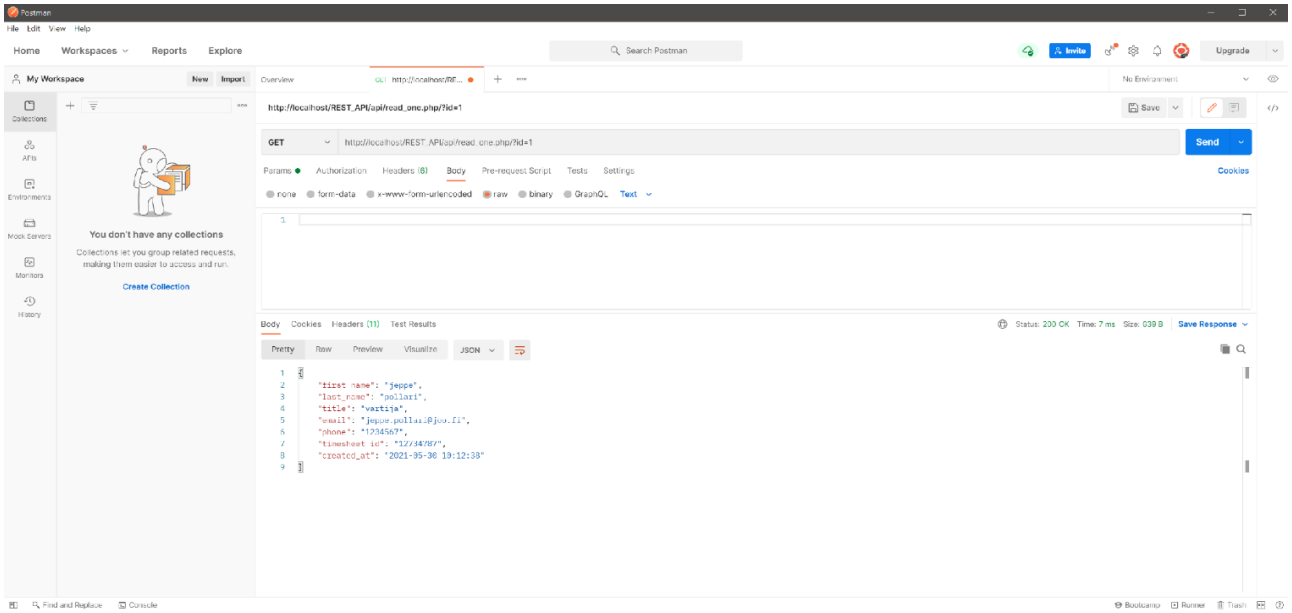
KUVA 17. RFID-tagin numerosarjan tallennus tietokantaan.

Kuvassa 18 testataan työntekijöiden tulostus ja samalla nähdään menivätkö tiedot tietokantaan. Kirjoitetaan http://localhost/REST_API/api/read.php ja vastauksena saatiin aikaisemmin luodun työntekijän tiedot.



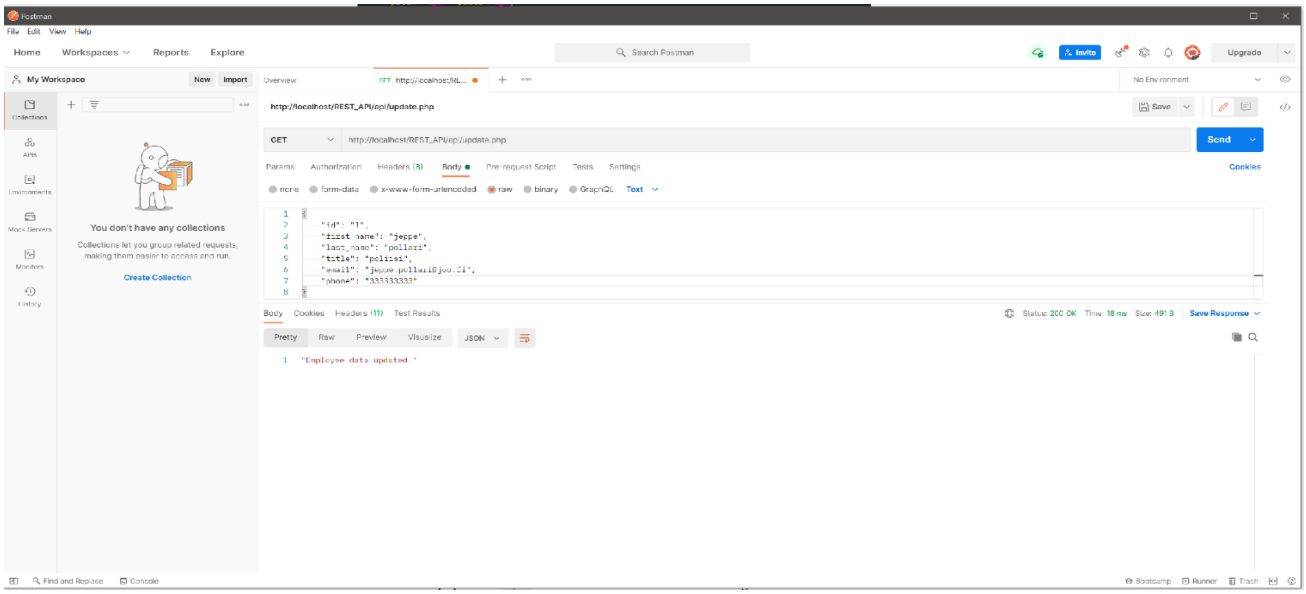
KUVA 18. Tulostetaan työntekijät ja tiedot.

Kuvassa 19 tulostetaan työntekijä id:n mukaan kirjoittamalla http://localhost/REST_API/api/read_one.php/?id=1. Saatiin tiedot tietokannasta missä id on 1.



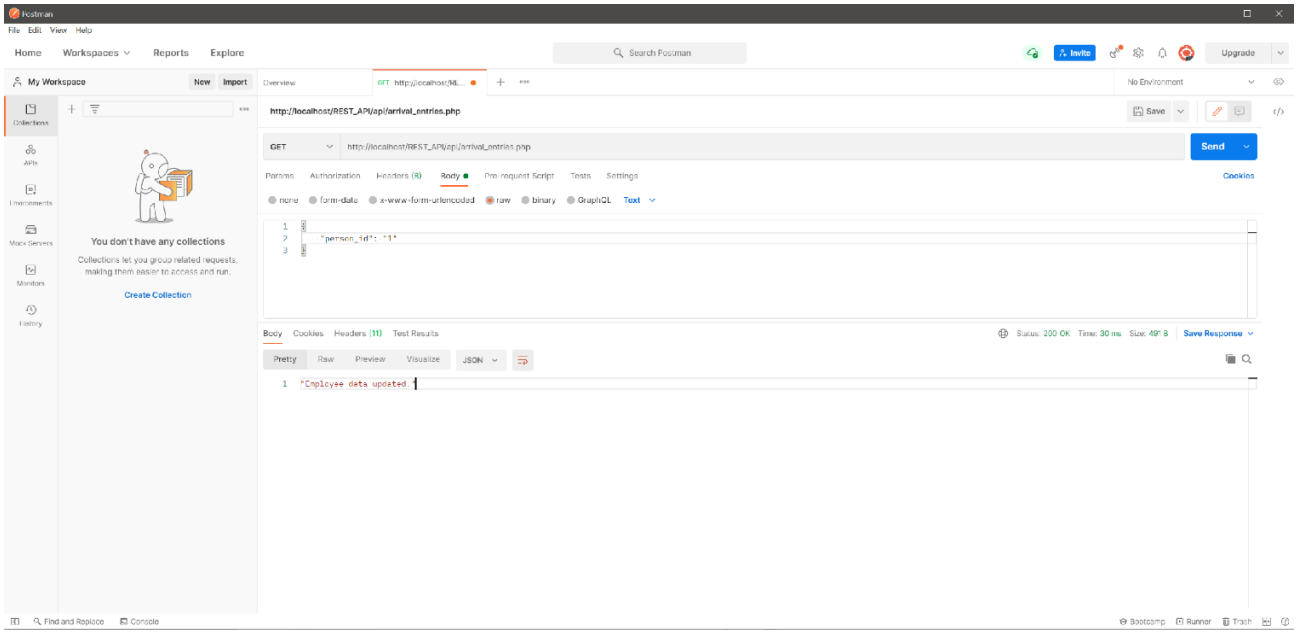
KUVA 19. Työntekijän tulostus id mukaan.

Kuvassa 20 testataan työntekijän tietojen päivittäminen kirjoittamalla http://localhost/REST_API/api/update.php ja annetaan tiedot bodyn tekstikenttään. Annetaan tiedot kaikki työntekiän tiedot ja tiedot jotka ovat muuttuneet päivittyvät tietokantaan. Id annetaan, jotta SQL UPDATE tietää kenen tiedot muutetaan.



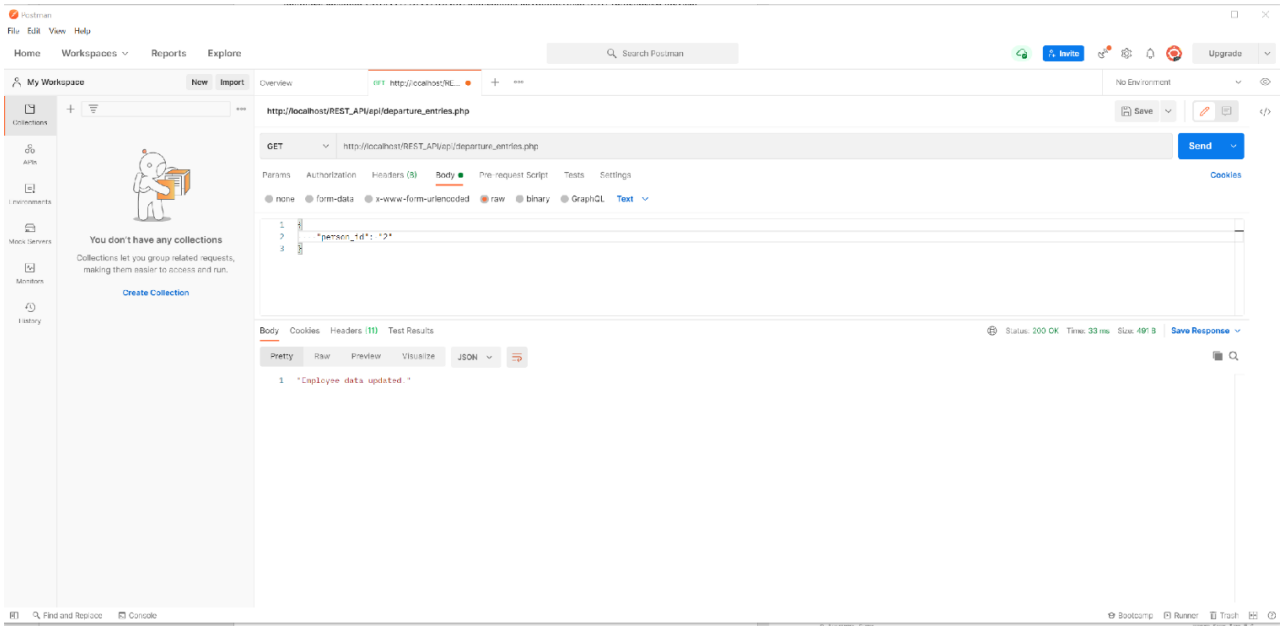
KUVA 20. Työntekijän tietojen päivitys.

Kuvassa 21 lisätään saapumisaika tietokantaan kirjoittamalla http://localhost/REST_API/api/arrival_entries.php ja antamalla person_id, joka on sama kuin employee_id.



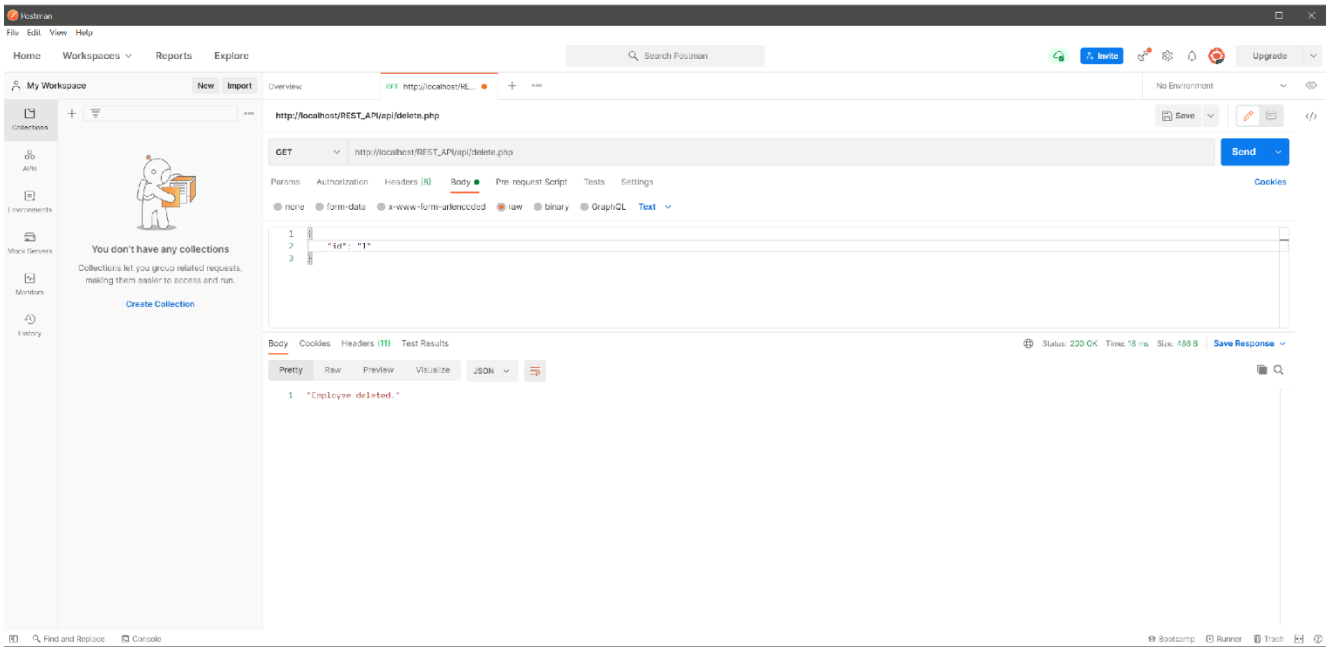
KUVA 21. Tallennetaan saapumisaika tietokantaan.

Kuvassa 22 lisätään poistumisaika tietokantaan kirjoittamalla http://localhost/REST_API/api/departure_entries.php ja annetaan person_id.



KUVA 22. Päivitetään poistumisaika tietokantaan.

Kuvassa 23 poistetaan työntekijä tietokannasta ja kaikki tiedot kirjoittamalla http://localhost/REST_API/api/delete.php ja antamalla työntekijän id.



KUVA 23. Poistetaan työntekijä tietokannasta

7 YHTEENVETO

Tarkoituksena oli luoda pieni kulunvalvontajärjestelmä, joka soveltuisi pienelle yritykselle käyttöön, ja tutkia mitä järjestelmän tekeminen vaatisi. Pienen yrityksen ongelmana on monesti rahaeikä voida välttämättä investoida järjestelmään Laiteisto valittiin halvaksi, mutta kuitenkin sen verran automaattiseksi ettei käyttäjän joutuisi syöttämään tietoa itse kirjautuessaan.

Järjestelmän olisi pitänyt toimia niin, että työntekijä tulee työpaikalle ja näyttää taigia lukijalle. Lukija piippaa ja tallentaa saapumis- tai poistumisajan tietokantaan. Toimisto tai joku muu taho olisi lisännyt työntekijän tietokantaan, tulostanu tietoja, päivittänyt tai poistanut työntekijän tietokannasta.

Työ onnistui osaltaan vaikkei REST API toiminut ihan halutulla tavalla. Tulo- ja poistumisajan tallennus tietokantaan tapahtuu person_id avulla, kun se olisi pitänyt tapahtua RFID.tagin id:n mukaan. En vain saanut sitä jostain syystä toimimaan.

Käyttöliittymä oli suunniteltu tehtäväksi. Ikävä kyllä aikataulun tiukkuudesta johtuen suunniteltu käyttöliittymä jäi tekemättä.

Mikäli käyttöliittymä saataisiin tehtyä Api:ssa pitäisi ainakin käyttäjien lisäämisen, poistamisen, päivittämisen ja tulostamisen toimia oikein.

Työ onnistui ainakin laitteiston kannalta se pidettiin halpana ja API toimi vaikkei kaikilta osin ihan oikein. Täysin tyytyväinen ei voi siis olla, kun kaikki ei mennyt niin kuin oli suunnitellut, mutta tuli opittua uutta REST:n, JSON:n osalta ja mitä kulunvalvonnan tekeminen vaatisi.

LÄHTEET

Apache friends. XAMPP Apache + MariaDB + PHP + Perl. Saatavissa: https://www.apachefriends.org/index.html. Viitattu: 09.06.2021.

atlasRFIDstore. 2021. What is RFID | The Beginner's Guide to RFID Systems. Saatavissa: https://www.atlasrfidstore.com/rfid-beginners-guide/. Viitattu: 09.06.2021.

Bush. T. 2020. CRUD vs REST: What's the Difference?, Saatavissa: https://nordicapis.com/crud-vs-rest-whats-the-difference/. Viitattu: 09.06.2021.

GURU99. Perl Tutorial: Variable, Array, Hashes with Programming Example. Saatavissa: https://www.guru99.com/perl-tutorials.html. Viitattu: 09.06.2021.

Ihmevekotin.fi. 2021. Saatavissa: https://ihmevekotin.fi/rfid-ja-nfc/624-rfid-lukija-usb-porttiin-125-khz.html. Viitattu: 09.06.2021.

Lange, K. 2016. The little book on REST services, Copenhagen. Saatavissa: https://www.kenneth-lange.com/books/The-Little-Book-on-REST-Services.pdf. Viitattu: 09.06.2021.

Kinsta Inc. 2021. What is MySQL ? A beginner-friendly explanation. Saatavissa: https://kinsta.com/knowledgebase/what-is-mysql/. Viitattu: 09.06.2021.

MariaDB Foundation. MariaDB Server: The open source relational database. Saatavissa: https://mariadb.org/. Viitattu: 09.06.2021.

phpMyAdmin. Bringing MySQL to the web. Saatavissa: https://www.phpmyadmin.net/. Viitattu: 09.06.2021.

Postman inc. 2021. The Collaboration Platform for API Development. Saatavissa: https://www.postman.com/. Viitattu: 09.06.2021.

Raspberry Pi Foundation. Saatavissa: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. Viitattu: 09.06.2021.

Singh, V. 2017. What is JSON. Saatavissa: https://www.toolsqa.com/rest-assured/what-is-json/. Viitattu: 09.06.2021.

Tietosuojavaltuutetun toimisto. Saatavissa: https://tietosuoja.fi/gdpr. Viitattu: 09.06.2021.

Tutorialspoint. RESTful Web Services – Introduction. Saatavissa: https://www.tutorialspoint.com/restful/restful_introduction.htm. Viitattu: 09.06.2021.

Vaswani, V. 2009. PHP – A BEGINNER'S GUIDE. Saatavissa: http://englishonline-club.com/pdf/PHP%20-%20A%20Beginner%E2%80%99s%20Guide%20[EnglishOnline-Club.com].pdf. Viitattu: 09.06.2021.