

# 18. Opintorekisteri- CRUD, REST API ja token-todennus

Kirjoittanut Joni Mäyrä

## Tehtävä

Loin tämän ohjelmistokokonaisuuden osana Tietokannat- ja rajapinnat kurssia Ammattikorkeakoulussa. Tehtävänä oli luoda CRUD (Create, Read, Update, Delete) -projekti, joka vaatii käyttäjätunnistautumisen tietojen hakemiseksi HTTP-pyyntöjen avulla. Olisin voinut käyttää myös basic authentication -menetelmää, joka olisi ollut helpompi suorittaa, mutta päätin käyttää tokeni pohjaista todennusmetodia, koska koen, että se on modernimpi ja turvallisempi. Tämä metodi myös helpottaa kirjautumisien hallitsemista ja ajoitusta.

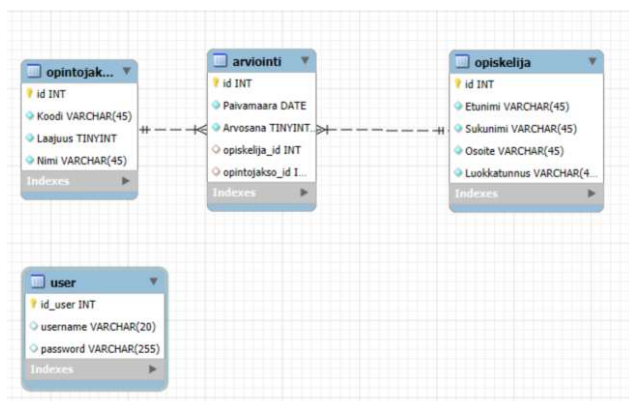
Käytin tehtävän toteutuksessa myös kryptausta kirjautumisen yhteydessä. Valitsin kryptaus algoritmiksi BcryptJS, joka on tunnettu metodi salasanojen salaukseen. Bcrypt algoritmi on ”yksisuuntainen” eli kun kryptaat salasanan se on vaikea palauttaa takaisin alkuperäiseen muotoon, mutta voimme kryptata käyttäjän antaman salasanan ja verrata sitä tietokannassa olevaan kryptattuun salasanaan ja katsoa täsmäävätkö ne.

Kun käyttäjä on antanut tunnukset, jotka vastaavat tietokannan käyttäjätunnuksia, palautamme käyttäjälle todennustunnuksen (Authentication token), jota käyttämällä käyttäjä voi tehdä tietokantaan CRUD – operaatioita käyttämällä Http- pyyntöjä.

## Tietokantarakenne

Loin tietokantaan neljä taulua joista, opintojakso ja opiskelija tauluilla on ”non - identifying” yhdestä moneen suhde arviointi tauluun. Tämä tarkoittaa, että opiskelijalla voi olla monta arviointia tietokannassa ja opintojaksolla voi myös olla myös monta arviointia tietokannassa.

Lisäsin tietokantaan User-tietokannan käyttäjä salasanojen ja käyttäjä tunnuksien tallentamiseen. Tätä tietokantaa käytetään rekisteröintiin kirjautumiseen ja käyttäjäntodentamiseen.



## Projektin rakenne

Aloin luomaan REST –API sovellusta ja päädyin käyttämään alla olevaa tiedostorakennetta.

**Bin kansio :** Sisältää ajettavia skriptejä. Tässä tapauksessa vain sovelluksen käynnistys skriptin.

**Models kansio:** Sisältää mallit eri taulujen SQL-kyselyihin.

**Node\_modules kansio:** Sisältää Node.js kirjastoja kuten BcryptJS ,Express ja JsonWebToken.

**Public kansio:** Julkisia tiedostoja kuten index.html, jota en käytä tässä projektissa.

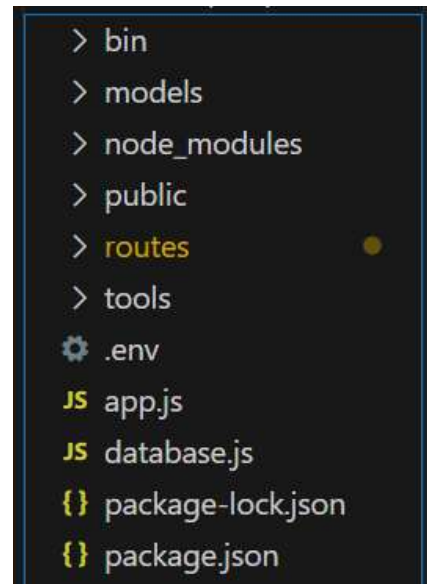
**Routes kansio:** Tähän määritetään reitit HTTP pyynnöille (GET, POST, PUT, DELETE jne.) ja ohjataan ne oikeisiin aliohjelmiin.

**Tools kansio:** Kansio erilaisille työkaluille, kuten todennustunnuksen varmistaminen ja niiden luonti.

**ENV. Tiedosto:** Arkaluontoiset muuttujat kuten JsonWebToken salaus avain ja tietokannan tunnukset.

**app.js:** Express REST sovelluksen luominen ja määrittäminen. HTTP polkujen lisääminen.

**database.js:** Skripti MySql-tietokanta yhteyden luomiseen.



## Toiminnallisuus

Rest-API projektin tarkoitus on vastaanottaa HTTP- pyyntöjä. Esitän alla miten sovellusta voi käyttää Postman - työkalulla datan hakemiseen tietokannasta.

## Kirjautuminen ja rekisteröityminen

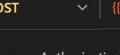
Huom {{baseUrl}} tarkoittaa http://localhost/3000

## Rekisteröityminen toimii lähettämällä

http://localhost/3000/register  
polkuun POST menetelmällä,  
json muodossa käyttäjänimen ja  
salasanan. Jos käyttäjänimi on  
varattu tai virheellinen. Käyttäjän  
luominen epäonnistuu.

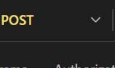
**Kirjautuminen** toimii lähettämällä `http://localhost/3000/` polkuun POST menetelmällä, json muodossa käyttäjänimen ja salasanan. Sovellus hashaa salasanan ja vertaa sitä tietokannassa olevaan salasanään. Jos hashatut salasanat ovat samat sovellus palauttaa todennustunnuksen, jolla voi tehdä CRUD operaatioita 30 min ajan.

```
{
  "message": "User registered successfully",
  "result": {
    "fieldCount": 0,
    "affectedRows": 1,
    "insertId": 9,
    "info": "",
    "serverStatus": 2,
    "warningStatus": 0,
    "changedRows": 0
  }
}
```



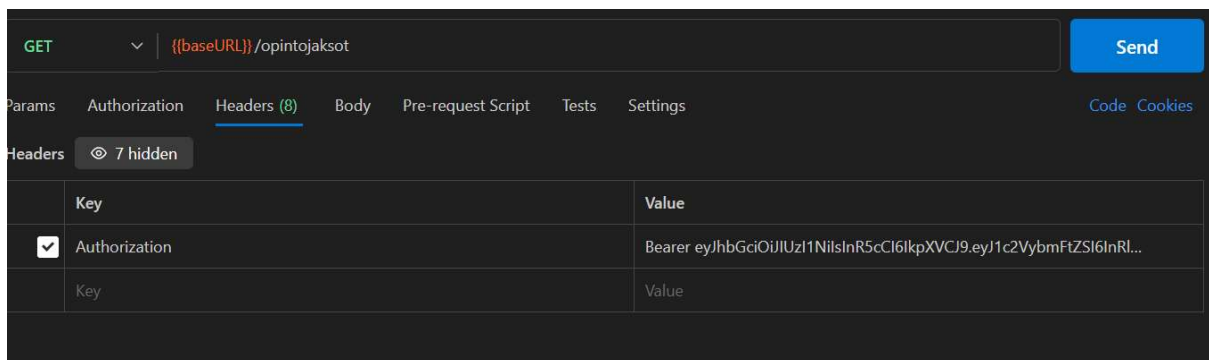
The screenshot shows a REST client interface. The method is **POST** and the URL is `{(baseUrl)}/register`. The **Headers** tab is selected, showing `Content-Type: application/json`. The **Body** tab is also visible. The request body is a JSON object: `{ "username": "test", "password": "test" }`.

```
eyJhbGciOiJIUzI1NiIsInR5cC  
eyJ1c2VybmFtZSI6InRlc3  
H0.NmG_g1y9YuG-piulHdK
```



The screenshot shows the REST Client interface. The method is **POST** and the URL is `{baseUrl}/`. The request body is a JSON object: `{ "username": "test", "password": "test" }`. The 'Headers' tab is selected, showing no headers. The 'Params' and 'Authorization' tabs are also visible.

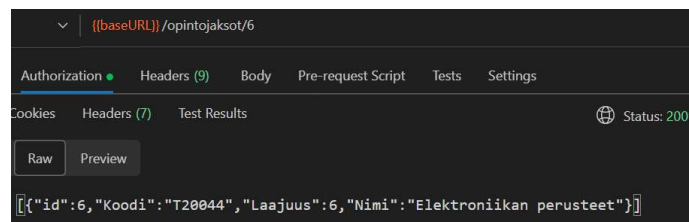
## Opintojaksot tietokannan CRUD- operaatiot HTTP pyynnöillä



Kirjautumisen jälkeen voimme tehdä CRUD operaation asettamalla todennustunnuksen Authorization headeriin. Muissa opintojaksot tietokannan operaatioissa tarvitsee myös todennustunnusta.

## READ OPERAATIO

Esimerkki Yhden opintojakson hakemisesta ID:en perusteella todennustunnuksen kanssa.

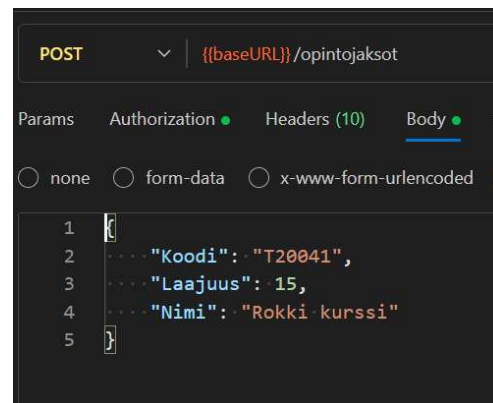


## CREATE OPERAATIO

Jos käyttäjällä on voimassa oleva todennustunnus voi hän luoda uusia kursseja tietokantaan.

Tieto lähetetään POST metodilla Json – muodossa. Ja tieto tallentuu MySql tietokantaan.

Alla kuva tietokantaan lisätyistä opintojaksoista, joista alin on juuri lisätty “Rokki kurssi”.

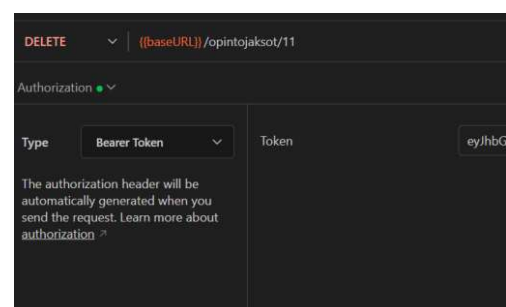


3	T20041	3	Tietokannat
4	T20042	3	Liike-elaman tapatietous
5	T20043	3	Fysiikka 3
6	T20044	6	Elektroniikan perusteet
7	T20045	5	Kellarihumppa
8	T20046	3	Matematiikka 2
9	T20047	3	Ruotsin kieli
10	T20048	3	VHDL-kieli
11	T20041	15	Rokki kurssi

## DELETE OPERAATIO

Todennettu käyttäjä pystyy myös poistamaan opintojaksoja

Kun lisäämme polkuun opintojakso ID:een ja kutsumme DELETE –metodia, voimme esimerkiksi poistaa äsken luodun “Rokki kurssin” ja siihen liittyvät arvioinnit.



Alla kuva tietokannasta josta on poistettu kyseinen kurssi ja kuva HTTP- pyynnön vastauksesta.

3	T20041	3	Tietokannat
4	T20042	3	Liike-elämän tapatietous
5	T20043	3	Fysiikka 3
6	T20044	6	Elektroniikan perusteet
7	T20045	5	Kellarihumppa
8	T20046	3	Matematiikka 2
9	T20047	3	Ruotsin kieli
10	T20048	3	VHDL-kieli

```

1 {
2   "fieldCount": 0,
3   "affectedRows": 1,
4   "insertId": 0,
5   "info": "",
6   "serverStatus": 2,
7   "warningStatus": 0,
8   "changedRows": 0
9 }

```

## UPDATE OPERAATIO

Todennettu käyttäjä pystyy myös muokkaamaan opintojaksoja

Kun lisäämme polkuun opintojakso ID:een ja kutsumme PUT-metodia, voimme esimerkiksi muokata tietokannat kurssia.

PUT `{{baseUrl}}/opintojaksot/3`

Params Authorization Headers (10) Body Pre-request Script

none form-data x-www-form-urlencoded raw binary

```

1 {
2   "Koodi": "T20041",
3   "Laajuus": 5,
4   "Nimi": "Tietokannat"
5 }

```

Alla kaksi kuvaa tietokannasta ennen ja jälkeen muokkauksen.

Ennen

id	Koodi	Laajuus	Nimi
3	T20041	3	Tietokannat

Jälkeen

id	Koodi	Laajuus	Nimi
3	T20041	5	Tietokannat

## Muut taulut

Kerroin aiemmin, että tietokannassa on myös kaksi muuta taulua, jotka ovat opiskelijat ja arvioinnit. Voisin tähän vielä alkaa selittämään niiden toimintaa, mutta ne ovat samanlaisia kuin opintojakson CRUD- operaatiot, joten en koe sitä tarpeelliseksi. Näihin tauluihin pätee samantyyppiset CREATE, READ, UPDATE ja DELETE -metodit ja käyttäjän todennus.

## Kehitys ideoita

REST-API sovellukseen voisi lisätä vielä esimerkiksi haku toimintoja, jos tietokanta kasvaisi isoksi, mutta koska tämä on vain kurssin yksi tehtävä en koe sitä tarpeelliseksi.

Sovellukseen voisi myös lisätä erilaisia käyttäjä tyyppejä ja erilaisia oikeuksia niille. Esimerkiksi voisi olla admin, opettaja ja opiskelija -käyttäjiä. Joista adminilla olisi

oikeus kaikkiin kursseihin, opettajalla omiin kursseihin ja arviointeihin ja oppilailla olisi vain luku oikeudet ei arkaluontoiseen tietoon.

Tietoturva voisi myös kehittää rajaamalla kirjautumisyritysten määrää. Niin voisi estää esimerkiksi Brute force- hyökkäyksiä.

Voisimme myös lisätä kirjautumiseen automaattisesti monitoroidun loki systeemin, joka varoittaa ylläpitäjiä epäilyttävästä toiminnasta.