1. Asenna WebStorm, Node.js, Git ja MySQL Workbench (tai vastaava) erillisen ohjeen perustella. Laita myös ESLint toimimaan.

2. Lue seuraava tutoriaali: https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm Asenna Express-sovelluskehys NPM:ää käyttäen ja kopioi tutoriaalin ensimmäinen server.js-versio PHPStormiin ja aja se. Totea sriptin toimivuus.

3. Keksi tietosisältö niin, että joudut määrittelemään muutaman taulun relaatiokannan, jossa käytät pääavain - viiteavain pareja. Jos et keksi muuta, niin Oman työtilassa on CreateTestDatabase.sql, jonka voi ajaa tyyliin: mysql > source CreateTestDatabase.sql. Jos olet käyttänyt Mongodb:tä, voit tehdä oman tietokannan myös sillä.

   Lisää tietokantaan tietoa käsin tyyliin `insert into location values ('1', 'Tavastia', 'Urho Kekkosen katu', 'Helsinki', '00100', 'Finland');` koska käyttöliittymä puuttuu vielä. Myös event ja eventDate tauluihin täytyy laittaa jotakin.

4. Liitä tietokantayhteys palvelinskriptiin ja kokeile kyselyä. Yksinkertainen esimerkki löytyy täältä: https://www.w3schools.com/nodejs/nodejs_mysql_select.asp Monimutkaisempi (ja parametroitu)

```
var q = url.parse(req.url, true).query;
var startDate = q.start;
var endDate = q.end;
var sql = "SELECT event_date.Date, event.Name, event.Type, event.Time,
Location.Location_place_name"
    + " FROM event_date, event, location"
    + " WHERE event_date.Event_Event_id = event.Event_id and
event.Location_Location_id = Location.Location_id"
    + " and event_date.Date >= ? and event_date.Date <= ?"
    + " GROUP BY Name"
    + " ORDER BY event_date.Date";
```

   kysely voisi olla oheisen näköinen. Kokeile esimerkkiä.

5. Jos lopputulos näkyy vain konsolissa, mutta ei selaimessa, katso oheinen keskustelu https://github.com/mysqljs/mysql/issues/1361 ja muuta toteutustasi niin, että vaste näkyy selaimessa saakka. Vaste on suoraan JSON-muotoinen (ainakin uudemmilla MySQL-toteutuksilla). Jos haluat muuntaa JSON-formaattia, voit käsitellä sitä merkkijonona ja tehdä tarpeellisia muunnoksia.

6. Lue artikkeli Node.js ja SQL-kyselyjen asynkronisesta toteutuksesta:https://codeburst.io/node-js-mysql-and-async-await-6fb25b01b628. Callback-operaatioiden käyttö johtaa niiden toistuvaan tarkistamiseen tilanteessa, jossa tietokantakyselyn toteutus perustuu useampaan toisiaan kutsuvaan operaatioon.

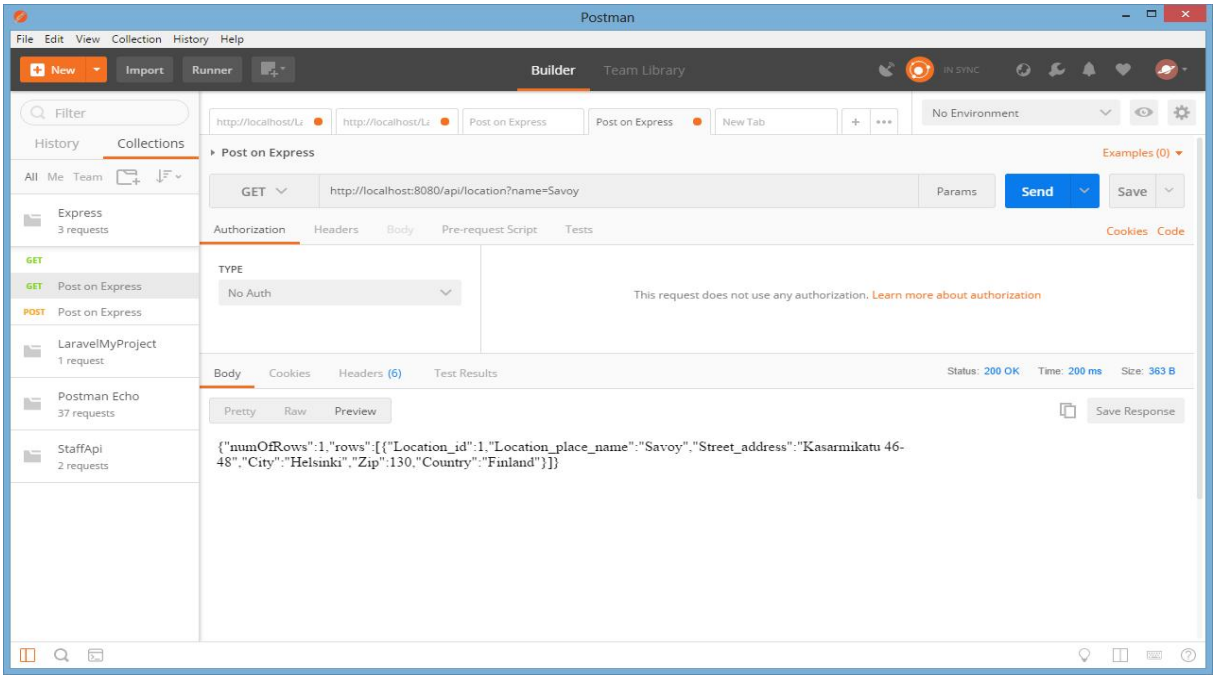
Kutsut voidaan toteuttaa async/await -tyylisesti, jolloin callback-operaatiota ei tarvitse määritellä. Tämä edellyttää, että Javascript-kieli on ainakin ECMAScript 6 tasolla ja Eslint-tarkistimen asetuksista löytyy seuraavan kaltaisia rivejä:

```
"env": {
    "node": true,
    "es6": true
},
"parserOptions": {
    "ecmaVersion": 2018
}
```

Toteuta palvelinskriptisi kysely/kyselyt avainsanoilla async/await. Oheinen linkki voi olla hyödyllinen: https://stackoverflow.com/questions/44004418/node-js-async-await-using-with-mysql

7. Kirjoita html-lomake, joka näyttää tietokannan tiedot siistinä listana, kun lähtötietona annetaan päivämääräväli, jolta tapahtumia halutaan nähdä. Työtilassa Dokumentit/Aputiedostoja-kansiossa on esimerkki, jota voi halutessaan käyttää apuna. Huomaa, että lomake täytyy ohjelmallisesti lähettää selaimelle, koska lomake on palvelimen resurssi. Tee tämä.

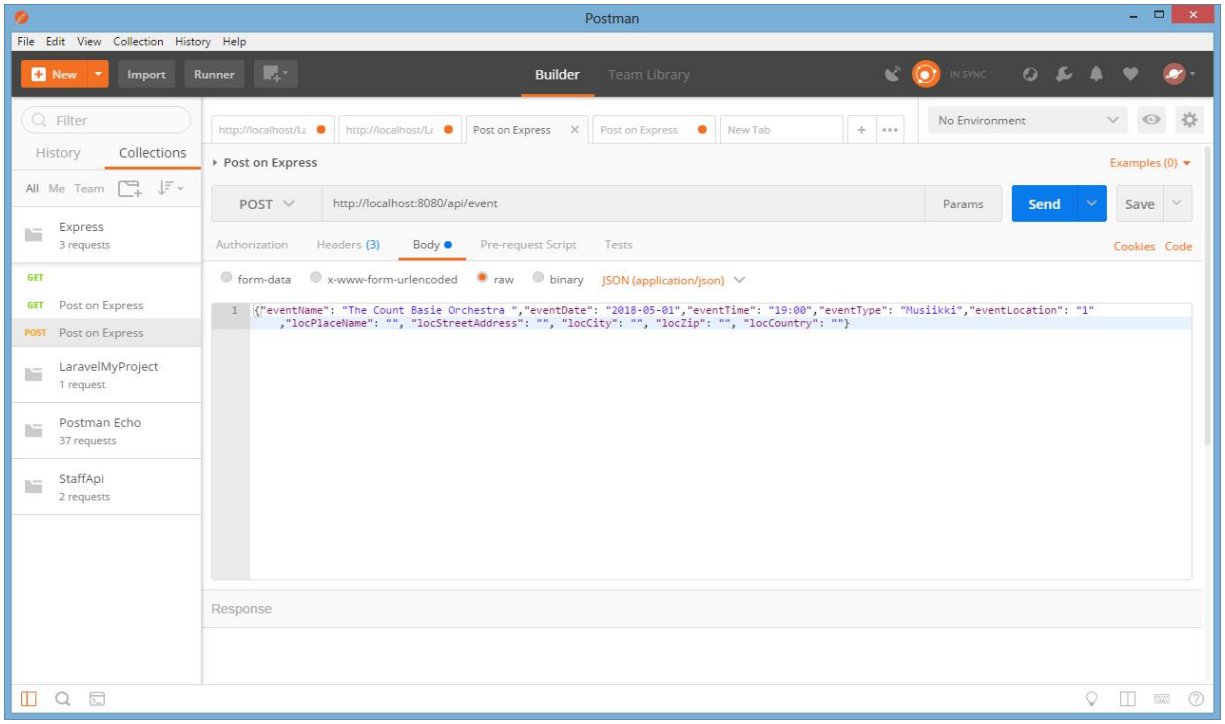
8. Toteuta rajapintaan toiminto, joka palauttaa JSON-muodossa huvittelupaikan osoitteen, kun sen nimi syötetään lähtötietona (kuva).




9. Toteuta toiminto, jonka avulla voit syöttää JSON-muotoisen tiedon http:n post-viestillä sisään järjestelmään. Oheiset apurakenteet helpottanevat http-sanomaosan prosessointia:

```
app.use(bodyParser.urlencoded({
extended: false }));
app.use(bodyParser.json()); // for
reading JSON
```

Testausta kannattaa tehdä aluksi Postmanilla:




Konsolissa voi tutkia, onko http:n sanoma-osassa mitään. Sanoman Content-Type kannattaa määritella application/json-tyyppiseksi. Päivitys tietokantaan tapahtuu joko päivittämällä kahta taulua (tai kolmea, jos tapahtumapaikka (location) on ennestään tuntematon). Päivityksessä kannattaa huomata, että mikäli sen tekee erillisillä con.query-kutsuilla (kuten varmaan täytyy), ne ovat asynkronisia.

```
console.log("body: %j", req.body);
```

Linkkejä, jotka voivat auttaa: https://stackoverflow.com/questions/32327858/how-to-send-a-post-request-from-node-js-express
https://stackoverflow.com/questions/38306569/what-does-body-parser-do-with-express
https://stackoverflow.com/questions/34665221/retrieve-data-from-just-inserted-row-mysql-and-node-js

10. Palvelin-skripti on tällä hetkellä yhtä pakettia. Mieti, miten se kannattaisi jakaa loogisiin osakokonaisuuksiin ja suorita paloittelu. Tee tarvittaessa tiedonhaku asiasta.

11. Klikkaa projektisi jotakin käyttöliittymätiedostoa (esim. listofplaces.html) hiiren oikealla napilla ja valitse run. WebStormin mukana asentunut mini-web-palvelin on portissa 63342 ja käynnistää html-lomakkeen. Lomake on nyt eri www-domain-alueessa kuin palvelin. Syötä lomakkeeseen päivämääräväli. Mitä huomaat?

    Käytettäessä Ajax-kutsuja kommunikointi on perinteisesti rajoittunut siihen www-domain-alueeseen, missä Ajax-kutsu tehdään. Tähän viitataan termillä ”same-origin-policy” https://en.wikipedia.org/wiki/Same-origin_policy Usein Ajax-kutsuja halutaan kuitenkin tehdä www-domainista toiseen.

    Yksi tapa toteuttaa tämä on ”Cross-Origin Resource Sharing” (CORS) menettely, joka edellyttää asiakkaan käyttävän Origin Request headeria sekä palvelimen määrittävän ”Access-Control-Allow-Origin”-http otsikon palvelinpuolelta lähteviin sanomiin. Kokeile, saatko mekanismin toimimaan omassa ympäristössäsi.

Katso esim. oheiset URL:t: https://en.wikipedia.org/wiki/Cross-origin_resource_sharing
http://www.html5rocks.com/en/tutorials/cors/#toc-withcredentials
https://developer.chrome.com/extensions/xhr liittyen CORS-tekniikkaan.