Creating an API with Node.js and Express. The first thing that we need to do is make sure that Node and Express are installed. The next thing we need to do is define the Node packages that we need. Express will be used to start and connect to our server and it will declare our routes.

When it comes to defining our Node packages we will need to create a .json file, for example package.json. These packages help Express which is Node’s framework. I believe that we need a ORM (Object relational mapping) to connect to our sql database. An ORM is a technique or a design pattern used to access a relational database from and object-oriented language. The best ones that I have come up with are: Mapper, Persistence.js, sequelize, and Node ORM2. From what I have read Mapper and sequelize both seem to need the least configuration and set-up and it are the quickest ones to jump right into using.

How to use sequelize with Express:

Installing express and creating an Express project:

$ mkdir example-app

$ cd example-app

$ npm install express express-generator

$ node\_modules/.bin/express . -f

$ npm install

$ ./bin/www

Now that there is an Express application we can now add sequelize.

$ npm install --save sequelize@2.0.0-rc1 sequelize-cli sqlite3

Or

$ npm install sequelize –save

Then you can install connectors for the database by typing

npm install pg --save or npm install mysql --save

Next we have to install the Node packages that we will be using. This is fairly straight forward. All you need to do is go to the command line and type “$ npm install”. Node package manager will then pull all of the packages defined in a node\_modules folder in out project. So, basically it wil pull all of the packages that we defined in the package.json file.

For installing Postgres-

Command prompt: $ npm install pg

var pg = require('pg');

var conString = "postgres://username:password@localhost/database";

var conString = "postgres://bruce:marist@localhost/database";

pg.connect(conString, function(err, client, done) {

if (err) {

return console.error('error fetching client from pool', err);

}

client.query('SELECT $1::int AS number', ['1'], function(err, result) {

done();

if (err) {

return console.error('error running query', err);

}

console.log(result.rows[0].number);

});

});

Sources:

<https://scotch.io/tutorials/build-a-restful-api-using-node-and-express-4>

<https://medium.com/@jeffandersen/building-a-node-js-rest-api-with-express-46b0901f29b6#.yezdtecwo>

<https://milinaudara.wordpress.com/2014/05/24/an-introduction-to-sequelize-js/>

<http://sequelize.readthedocs.org/en/1.7.0/articles/express/>

<https://www.quora.com/What-is-the-best-MySQL-ORM-for-Node-js>

<http://blog.modulus.io/absolute-beginners-guide-to-nodejs>

<http://docs.sequelizejs.com/en/latest/>

<http://docs.sequelizejs.com/en/latest/docs/getting-started/>