FIRST STEP : installation

Create the folder and the fileindex.js, and after this I have init npm, then I have install in the folder Knex postgres and express with the next command:

*npm install express knex postgres*

and finally: (

*npm install -g knex*

then I do my first test with the step 4&5 (website) on port 3000 and write my first route

<https://medium.com/@tobie.tsuzuki/getting-started-with-node-js-express-and-knex-5640f595df98>

following this I initialize Knex and install on my laptop postgres (postgres default port is 5432) and I changer sqlite to do pg.

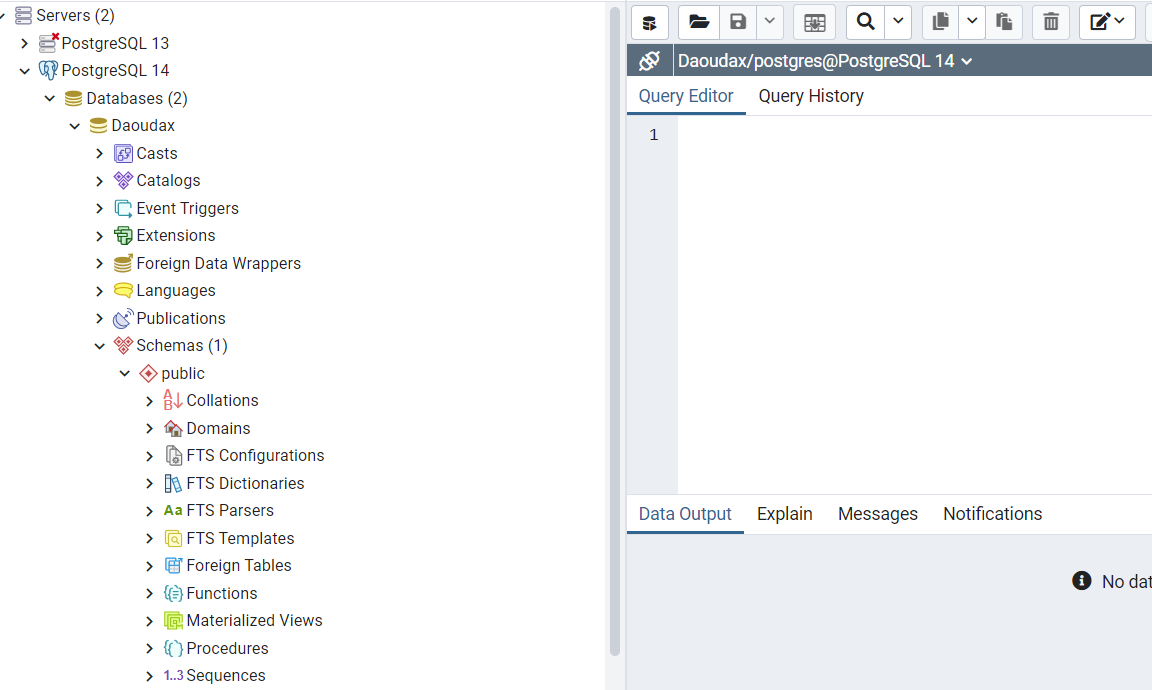
*createdb <database name>*

*knex init*

to create local database with postgres create knex.js for environment configuration

SECOND STEP: insert data

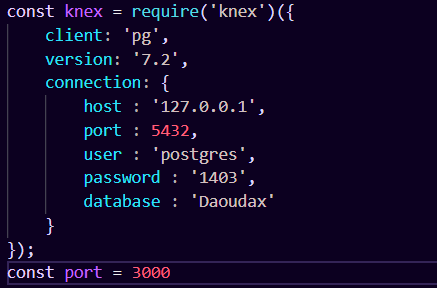
Now I installe pgAdmin 4



I have copy/paste the sql file on query tool, on the database I created. But I must update the sql file like by example quote for the name of table. And then I add the content of table.

THIRD STEP: Code

To connect Knex on pg admin I need to create the connection:



Here we have the database software who is pg, host: localhost (127.0.0.1), default pg port, username take by me, password of my pg and finally the name of the database.

After this I can begin to answer to the consign and I begin by READ because it’s the simplest and it’s just display information. For this: (method GET)

I create the function getCity to get information. With Knex method, I request on the table city the name and them I return this.

    knex('city').where('name', *name*)

    .then(*knexres* => {

*res*.send(*knexres*)

In the Url of google I need to write localhost:port/city/theNameSearch to have information like that

app.get('/city/:city', (*req*, *res*) => {

    getCity(*res*, *req*.params.city);

})



Now I write le method DELETE who take name of city in param, the functioning is sql sql request with condition where.

app.delete('/city/:city', (*req*, *res*) => {

    knex('city').where('name', *req*.params.city).del()

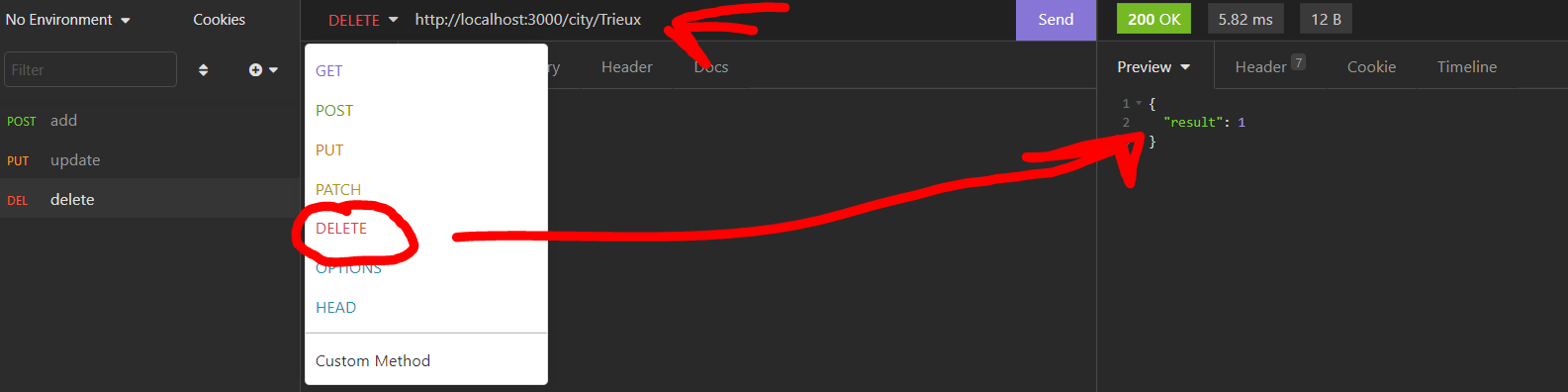
    .then(*knexres* => {

*res*.send({result:*knexres*});

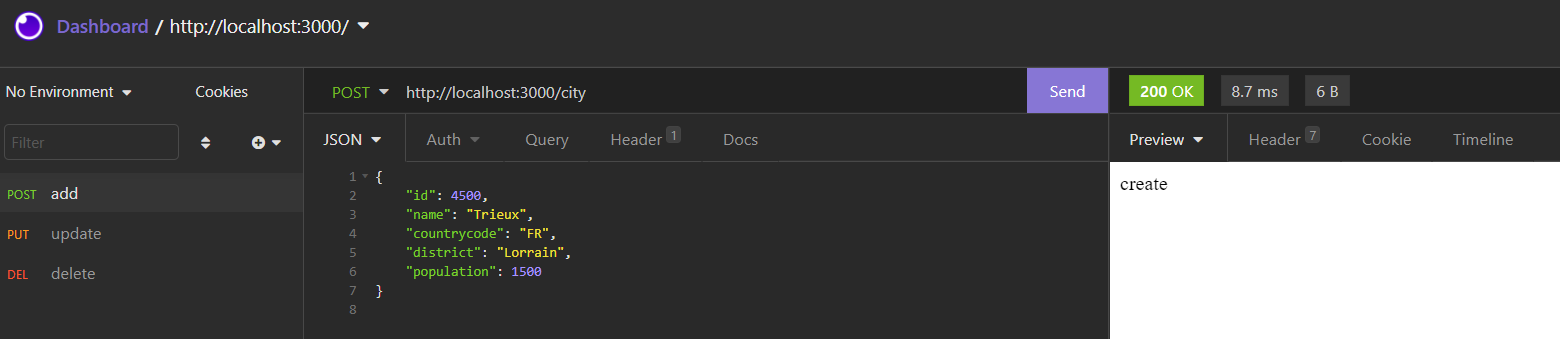
    });

})

To do this method we need to introduce some information because this methode is POST, I use Insomia, I create new request and I select the “delete”. The city will be delete with the name in URL.



Now the method POST, I use insomnia too, do pass some information on json request



This method take all json data in a request in the insert methode for insert new data in the table

app.post('/city', (*req*, *res*) => {

    knex('city').insert(*req*.body).then(*result* => {

        console.log(*result*);

    });

*res*.send('create')

})

finaly the method PUT, this is POST too but modified, knex here we can see knex is very easy to use, like sql request we just need condition and the action

app.put('/city/:city', (*req*, *res*) => {

    knex('city')

    .update({ population: *req*.body.population })

    .where('name', '=', *req*.params.city)

    .then(*knexres* => {

        if (!*knexres*) *res*.send('Error')

        getCity(*res*, *req*.params.city);

    });

})

