## Sequelize Application

### Sequelize Setup - Part 4

For part 4 of the lab work I first initialised a new project by executing the command

* ***npm init -y***

I then installed the following packages for the project

* ***npm install --save express nodemon sequelize sequelize-cli pg pg-hstore***

Once I had all the packages installed to use sequelize and the sequeliz-cli I then configure

a **.sequelizerc** file which would tell sequelize-cli where to store the migration, model, seed and config files.

Finally, I ran the following command

* ***node\_modules/.bin/sequelize init***

Which created a server folder, that contain migrations, models and seeders folders in it as well as a config.js file.

config.js contained the configurations for the database I was going to work with.

After I finished configuring the connection to the database It was time to generate the table models for the database through sequeliz-cli

by running this command:

* ***node\_modules/.bin/sequelize model:create Name\_of\_Model --attributes attribute\_name:data\_type, attribute\_name:data\_type"***

The models that were created had an identical structure to the tables in the pgguide database.

* **Note**: Corresponding migration files were also generated for each model

Finally, I configured the associations between the models, so they would mimic the relationship between the tables that were already in the pgguide database

Using the command `node\_modules/.bin/sequelize db:migrate` I ran the migration files which created 4 new tables in the database:

* seqProducts
* seqUsers
* seqPurchases
* seqPurchses\_Items

Which had the exact same structure and relationships as the tables

* + products
  + users
  + purchases
  + purchase\_items

That were initially in the pgguide database

Sequelize seeding - populating the database - part 5

After the models were created it was time to populate the database with test some data.

For this I used seeders which are files that are generated using the following command:

* ***node\_modules/.bin/sequelize seed:create --name seed\_file\_name***

I created a seed file for each of the sequelize models I made and added data in them that I wanted to add to the corresponding models

Finally, I populated the sequelize models by running the seed files with the following command:

* ***node\_modules/.bin/sequelize seed:all***

### Routes for API - Part 6

For part 6 of the worksheet I created a new app.js file where I configured a REST API with the following routes:

1. GET ***/products[?title=product\_title]***

* Returns a list of all the products available in the seqProducts table
* You can also look up a specific product by title using the query string like ***/products?title=product\_title***

2. GET */products/:id*

* + Returns a product with the id being passed in through the URL
  + Example ***/products/4***

3. POST ***/products***

* Adds a new record for the product
* The body of the POST request can contain the following:
  + - product
    - title
    - price
    - quantity
    - tags

4. PUT ***/products/:id***

* This PUT request updates a specific product by passing the product id of the product to be updated through the URL
* The values that can be updated are the price and quantity of the product and they are passed through the body

5. DELETE ***/product/:id***

* By making a http DELETE request to the API and passing a product ID through the URL, the record with that product ID gets deleted

For testing purposes, I also left in some additional routes:

- GET ***/addUser[?email=""&password=""&details={}&deleted\_at=]***

* This route adds a user by passing the following parameters through the query string in the URL:
* email :string
* password : string
* details : JSON
* deleted\_at : timestamp

- GET ***/addProducts[?title=""&price=&quantity=&tags=[]]***

* This route adds a product by passing the following parameters through the query string in the URL:
  + title : string
  + price : float
  + quantity : integer
  + tags : [strings]

- GET ***/purchases[?name="string"&address="string"&state="string"&zipcode=integer&created\_at=timestamp&userId=integer]***

* This route returns the contents of the purchases table, but you can also add a record to the seqPurchases through the query string of the URL like:

***/pruchases?name="string"&address="string"&state="string"&zipcode=integer&created\_at=timestamp&userId=integer***

-GET ***/purchase\_items[?price=&quantity&state=""&purchaseId=&productId=]***

* This returns the contents of the seqPurchses\_Items table but you can also add a record to it through the query string of the URL like :

***/purchase\_items?price=&quantity&state=""&purchaseId=&productId=***