

Enterprise Application Development – Lab 1

C15478448



Enterprise Application Development

Course: DT228/4

Lab 1



Table of Contents

**Part 1 – HTTP API Endpoints 2**

**Part 2 – Products Extension + SQL Injection 6**

**Part 3 – Eliminate Security Holes 8**

**Part 4 – Sequelize ORM 10**

**Part 5 – Test Data (Seeders) 11**

**Part 6 – Restful API 15**

Folder Structure

Massive-Express – Contains Parts 1-3

Sequelize-ORM – Contains Parts 4-6

Document – This Document in .docx and .pdf

Screenshots – Screenshots for all Outputs

**Note:**

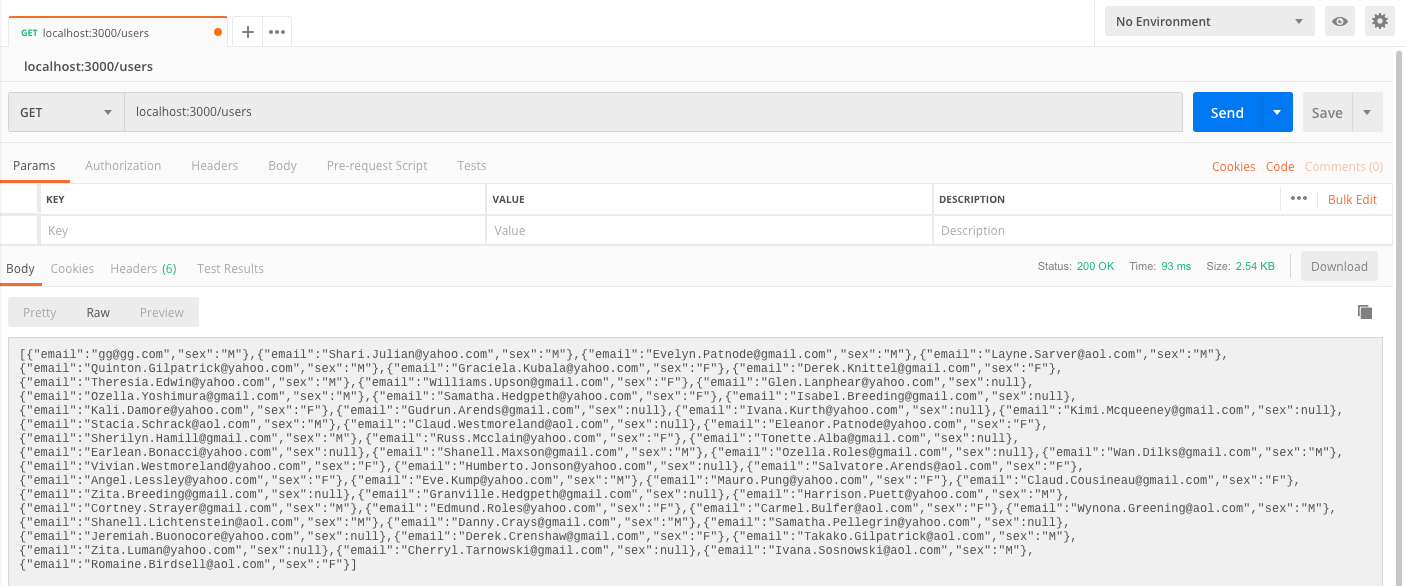
* To run this project you must enter in the following command in the **Massive-Express** and **Sequelize-ORM** folder: *npm install*
* Sequelize-ORM contains a ***README.txt*** which contains all of the commands on how to set up the models, migrations and seeders

**Problem Sets**

**Part 1 – HTTP API Endpoints**

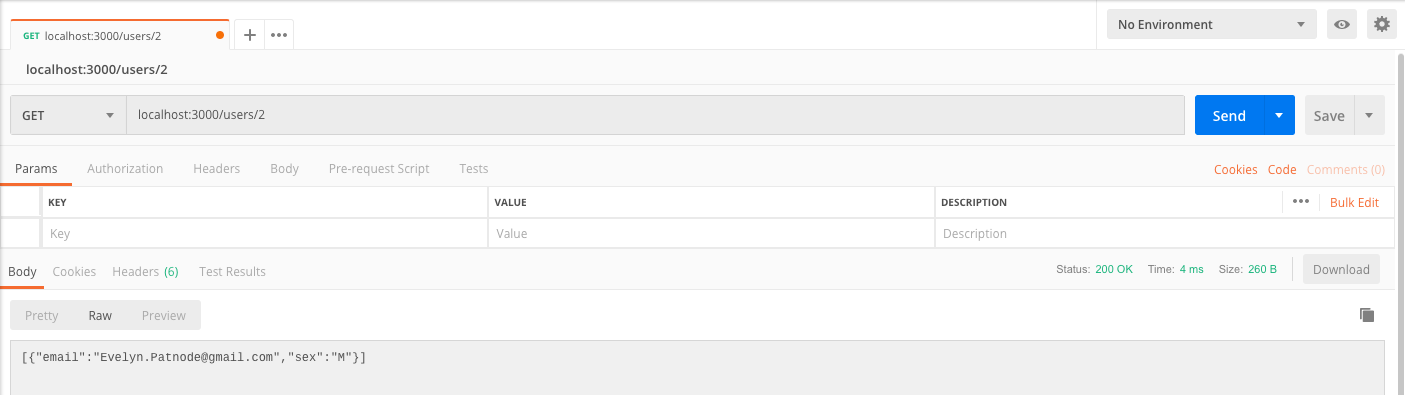
**GET /users**

Details: List all users email and sex in order of most recently created. Do not include password hash in your output.



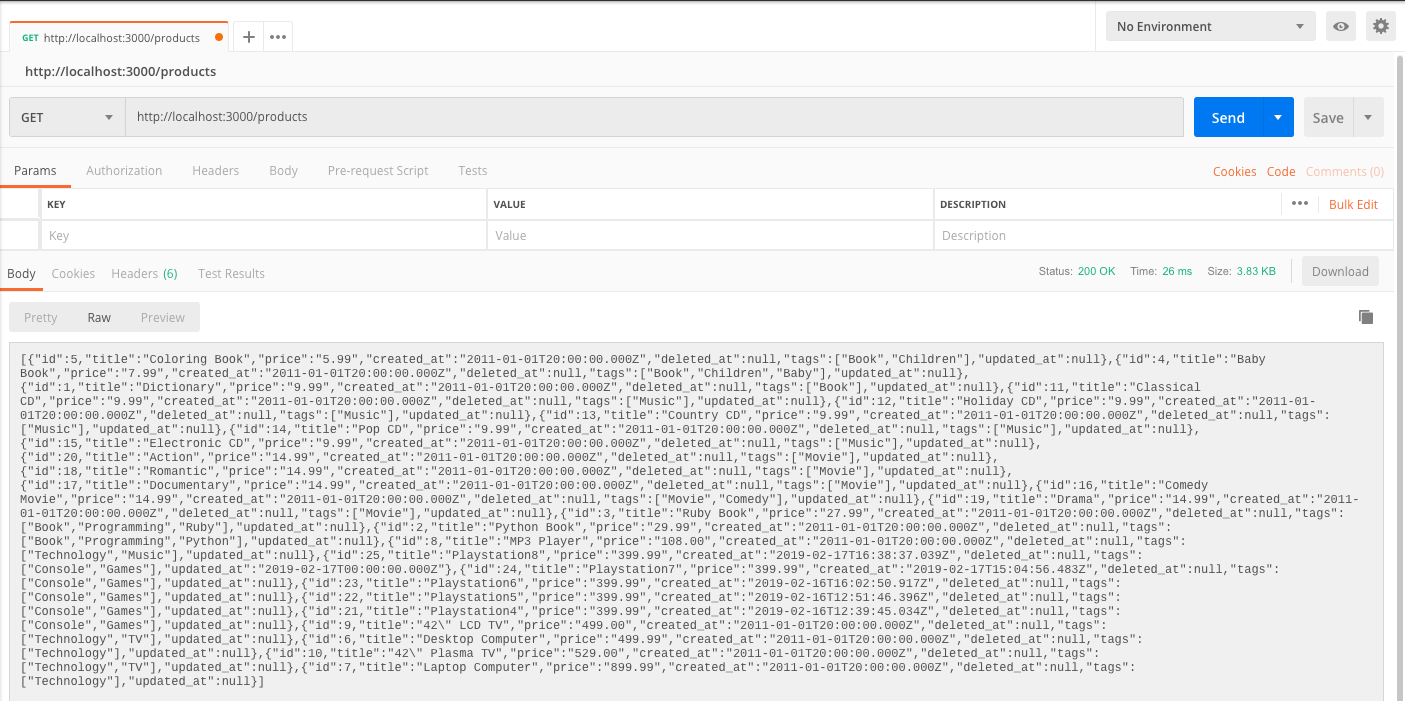
**GET /users/:id**

Show above details of the specified user.

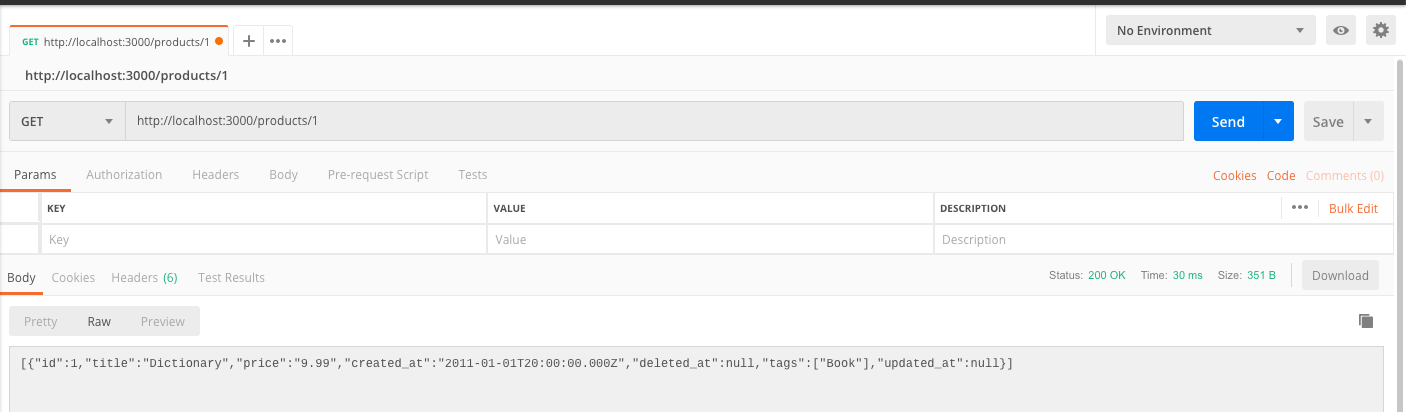
****

**GET /products**

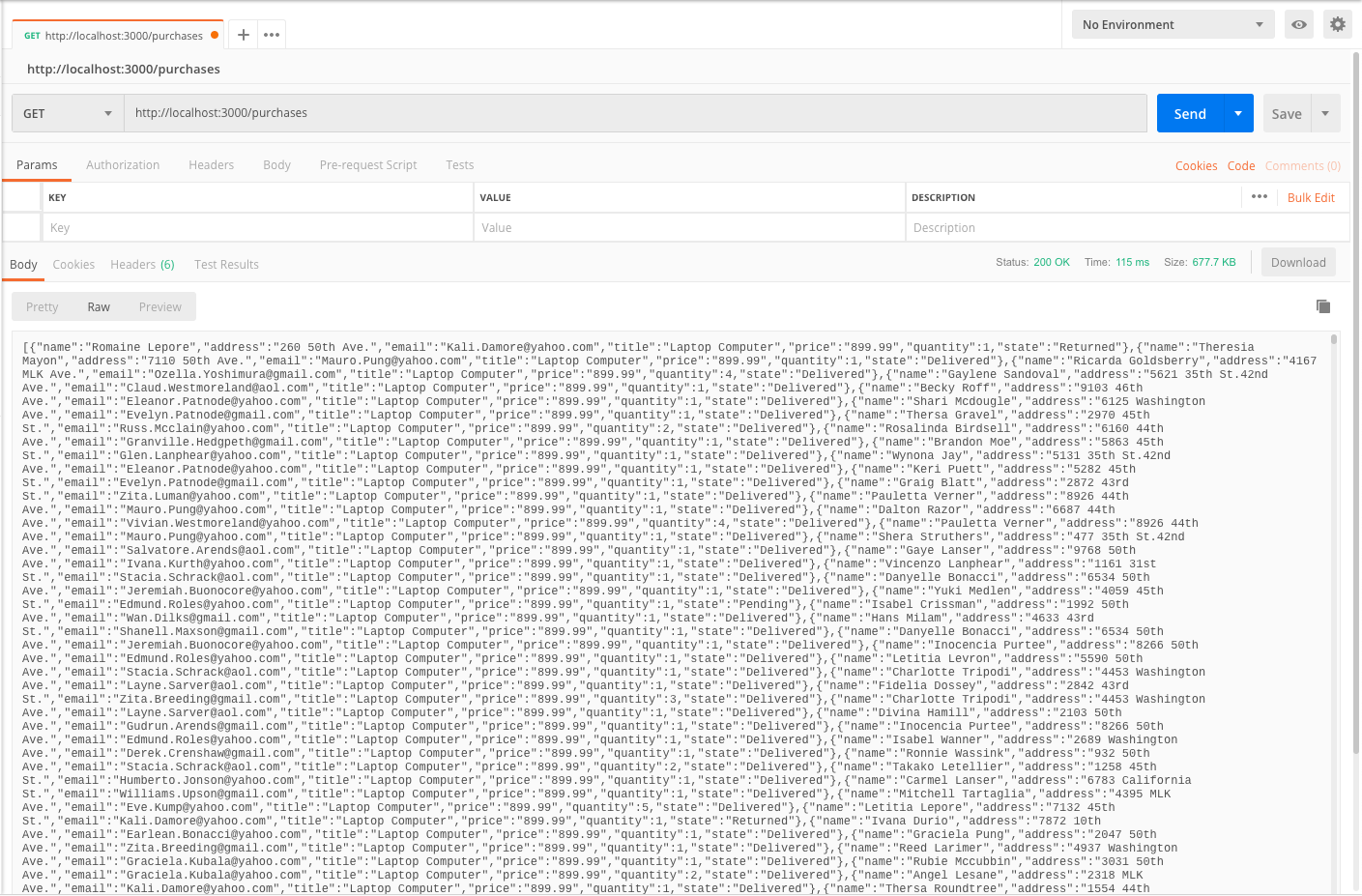
List all products in ascending order of price.

****

**GET /products/:id**

Show details of the specified products.

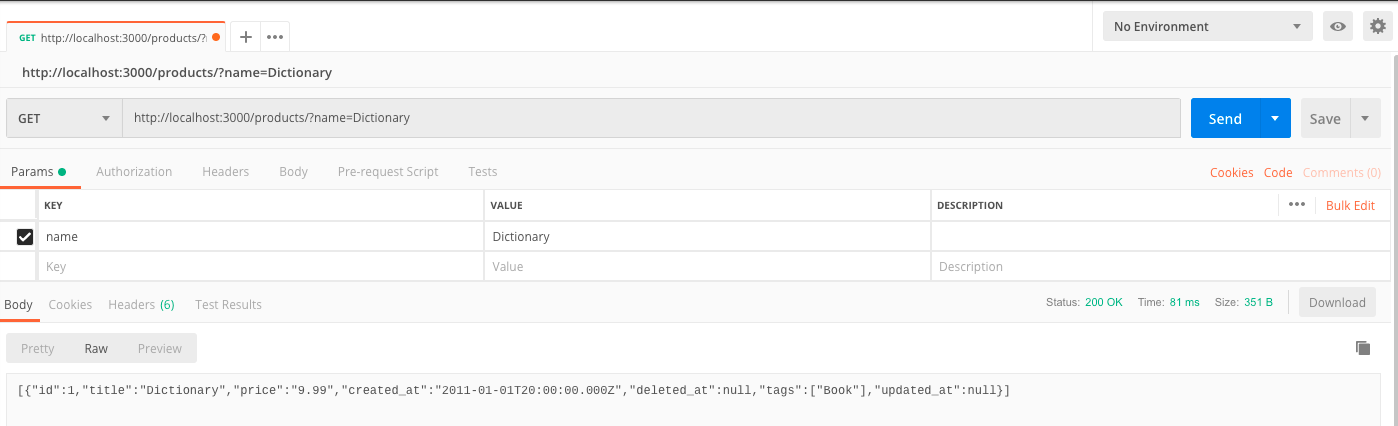
**GET /purchases**

List purchase items to include the receiver’s name and, address, the purchaser’s email address and the price, quantity and delivery status of the purchased item. Order by price in descending order.

All endpoints serve the expected data as checked with the database against it, explanation is in the code.

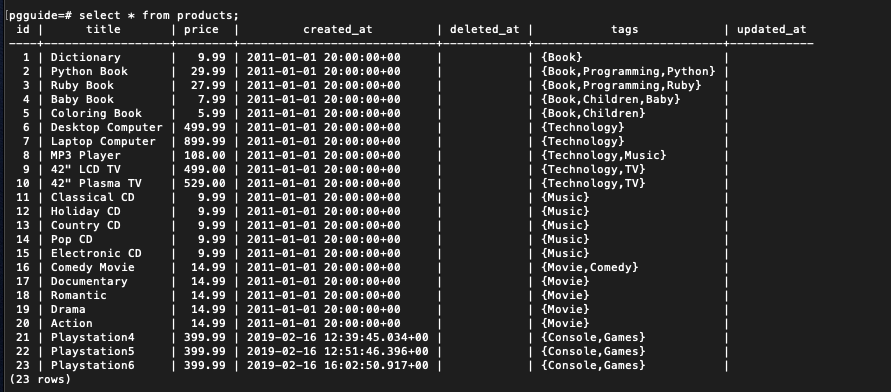
**Part 2 – Products Extension + SQL Injection**

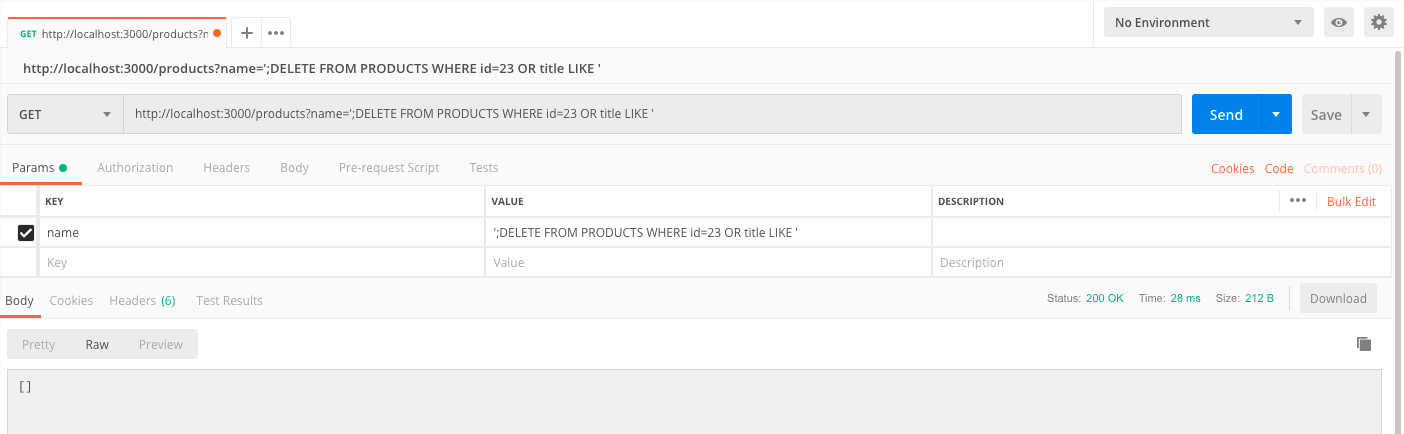
**GET products[?name=string]**

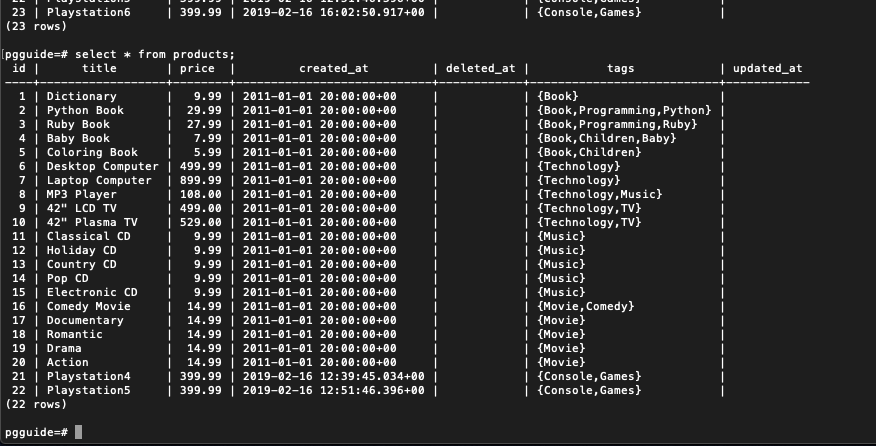


**SQL Injection**

Database before the query.

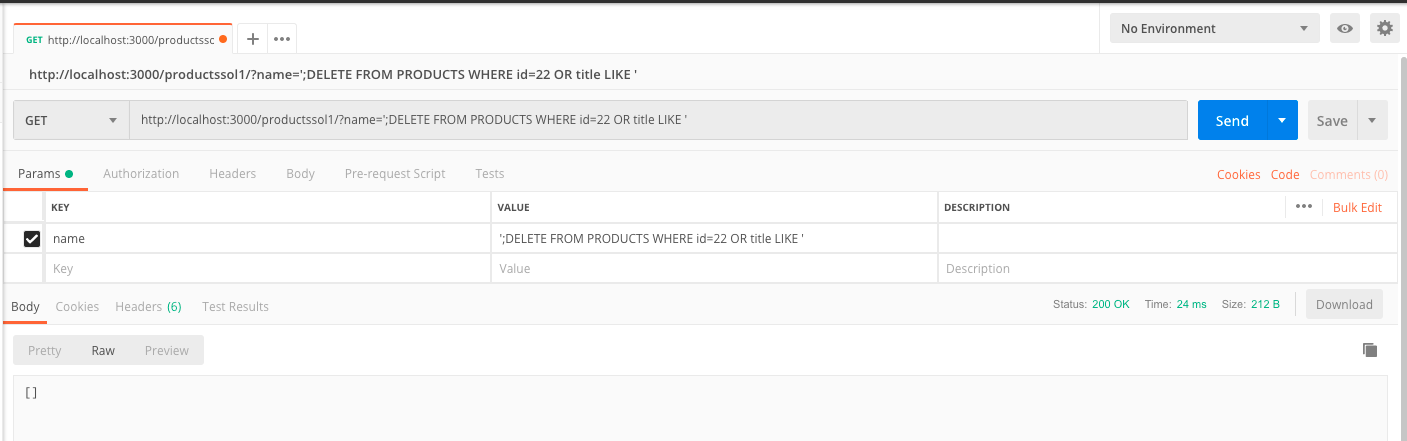


Entering in the query.

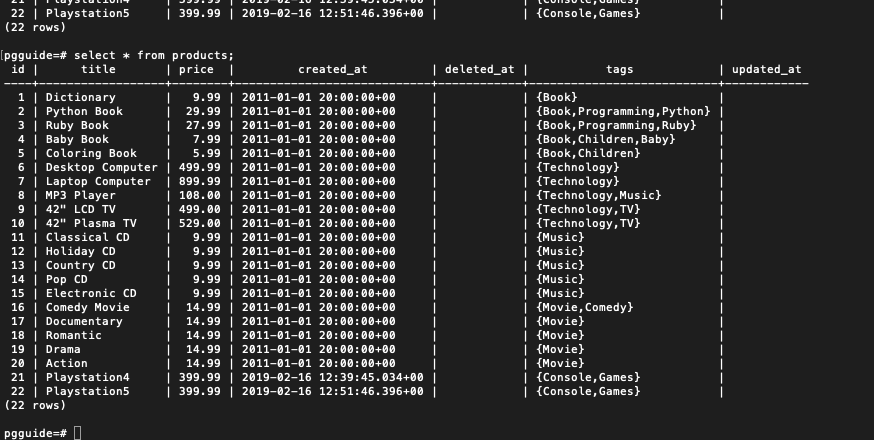
Database after the query.

**Part 3 – Two Solutions to Eliminate Security Holes**

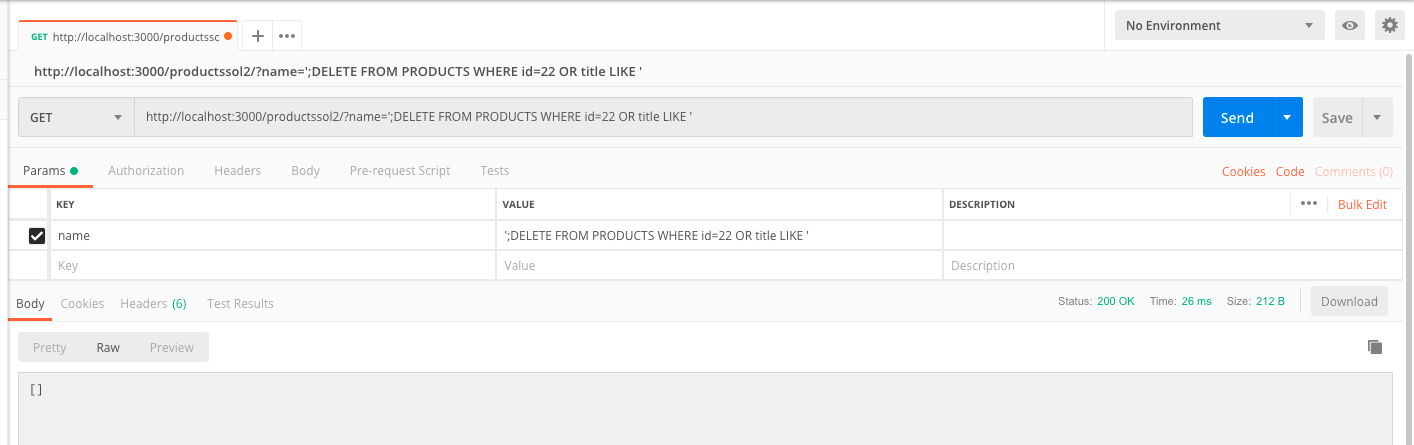
**Parameterised Query**



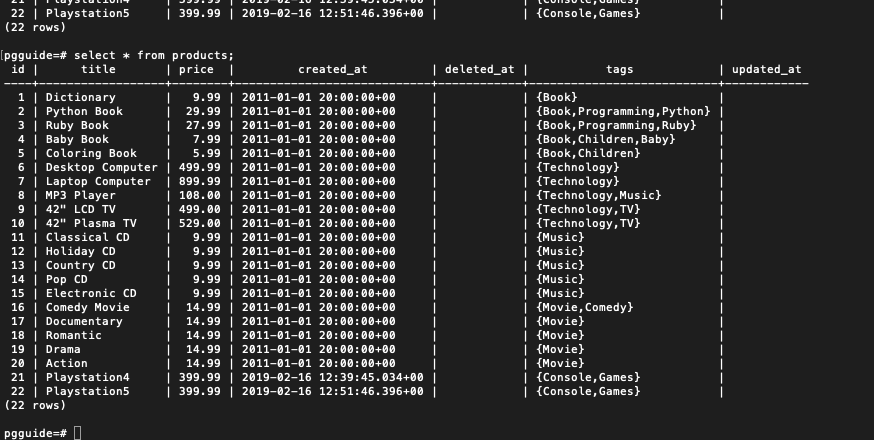
After entering in the query, it does not delete from the database as it is prone to SQL Injection.



The database is unaffected after entering in the SQL Injection query.

**Stored Procedure**

After entering in the query, it does not delete from the database as it is prone to SQL Injection.



The database is unaffected after entering in the SQL Injection query.

**Note:**

* /products/?name=… - Ability to perform SQL Injection
* /productssol1/?name=… - Parameterised Query (SQL Injection Prevention)
* /productssol2/?name=… - Stored Procedure (SQL Injection Prevention)

***All is explained in the code on how this is done.***

**Part 4 – Sequelize ORM**

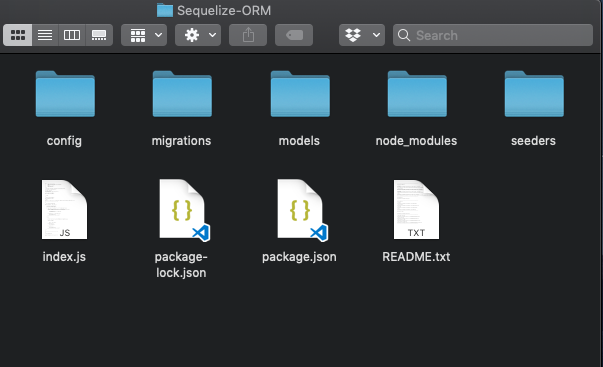
**Setup**

1. npm install --save sequelize-cli
2. node\_modules/.bin/sequelize init

The following folders are created:

* config
* models
* migrations
* seeders

**Folder Structure**



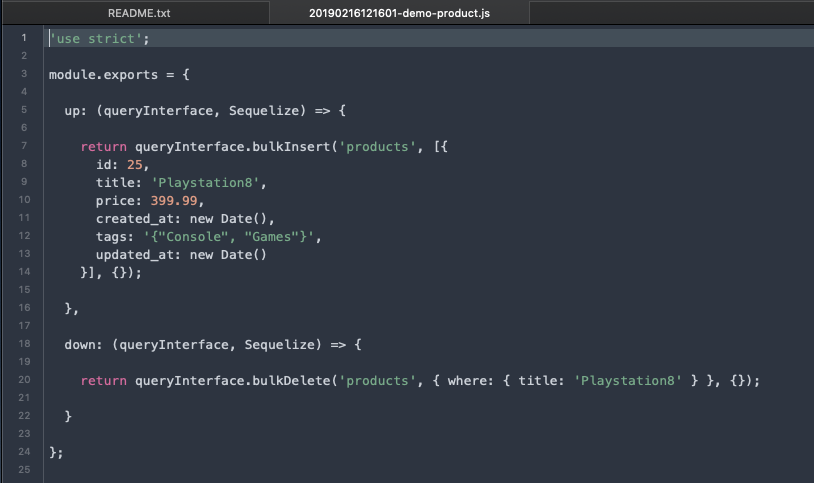
Inside the folder **Sequelize-ORM** the *README.txt* contains the setup commands in creating models and migrations.

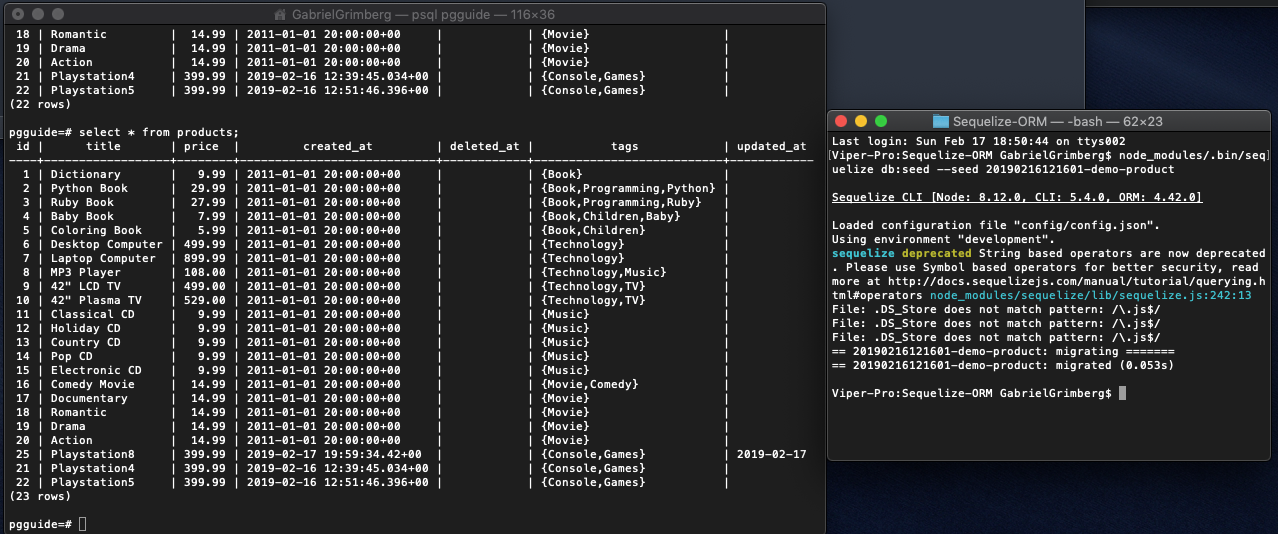
**Note**

* It’s important to add the “**updated\_at”** column to ensure that the appropriate associations are all set up and working with the models.

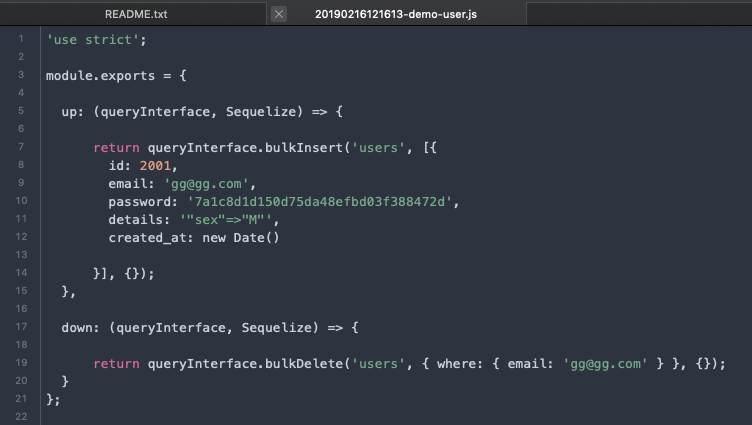
**Part 5 – Test Data (Seeders)**

**Product – Sample Data**

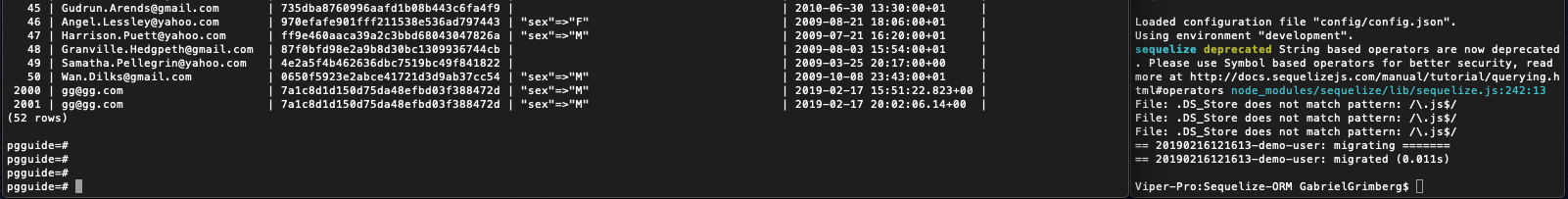
****

**Output**

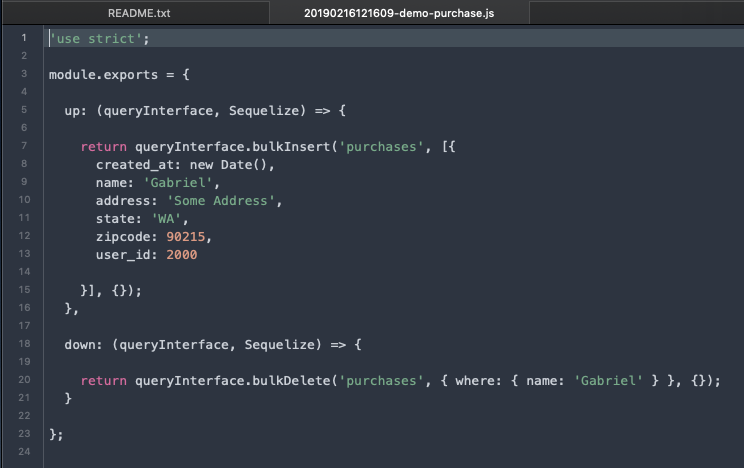
**User – Sample Data**

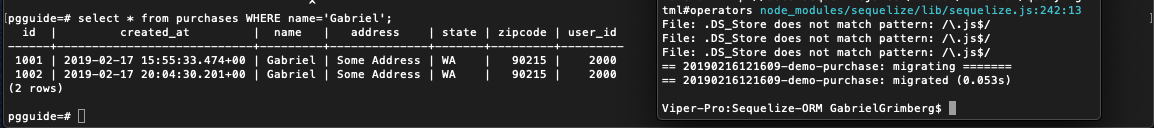
****

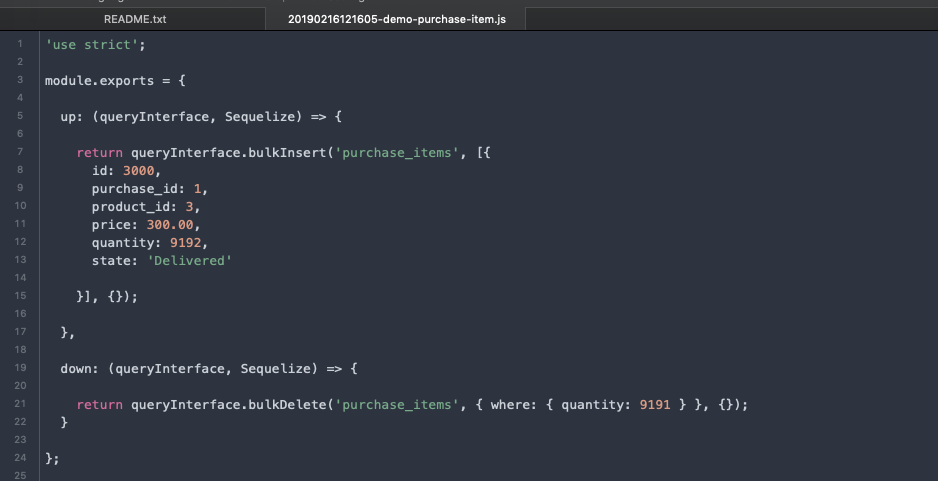
**Output**

******

**Purchase – Sample Data**



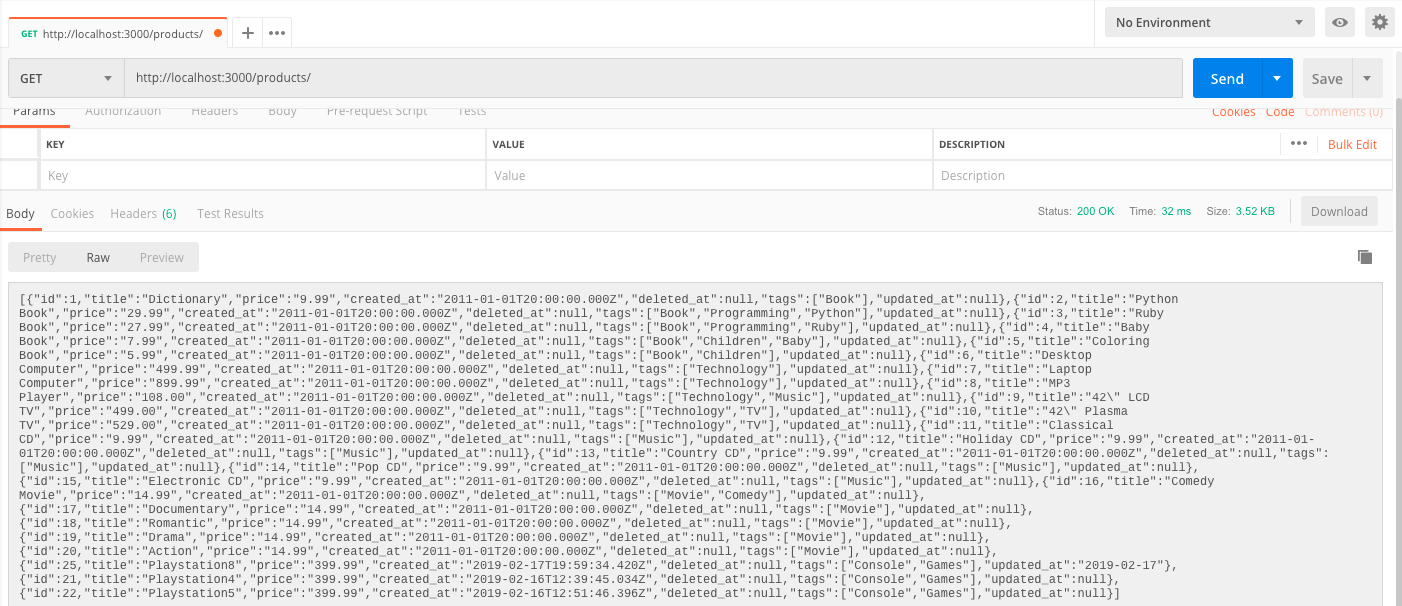
**Output**

**Purchase Item – Sample Data**

**Output**

**Part 6 – RESTful API**

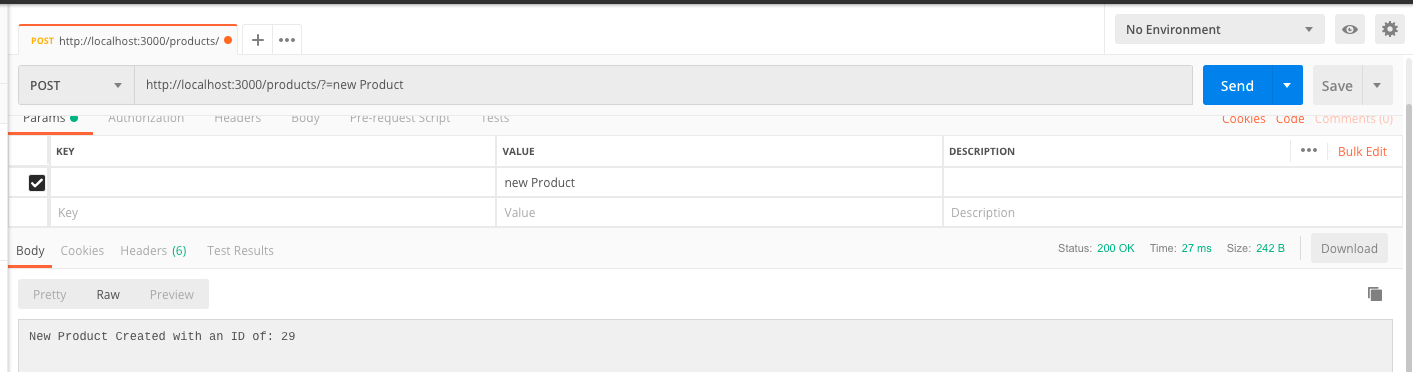
**GET /products[?name=string]**



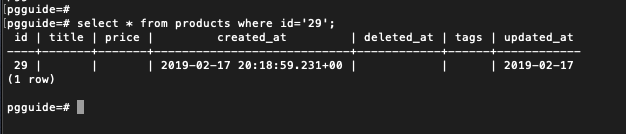
**GET /products/:id**



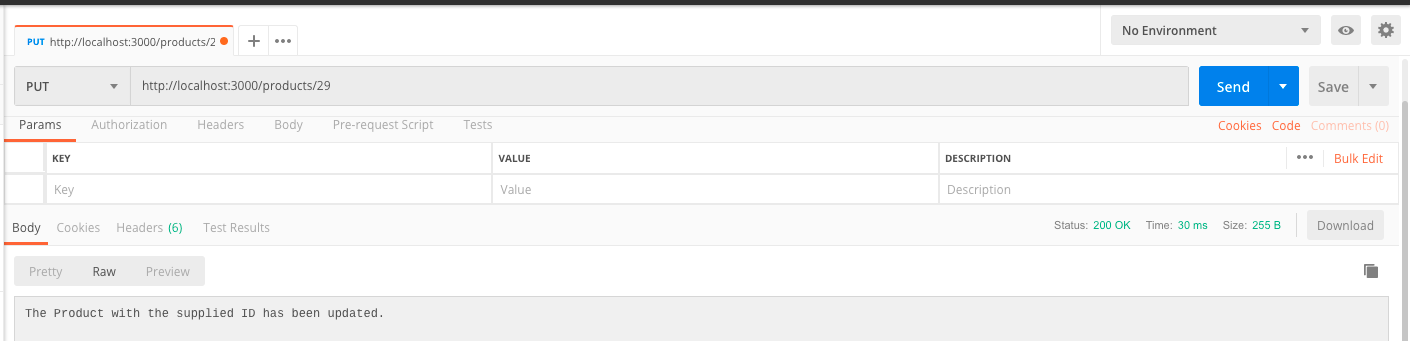
**POST /products**



**New Instance Confirmation**

****

**PUT /products/:id**

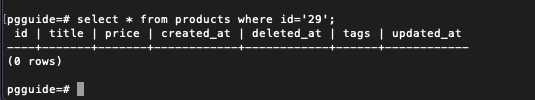


You supply the ID and the updated\_at field gets updated, that’s why this column is needed.

**DELETE /products/:id**



**Deletion Confirmation**

****