Aries Ogaco

C15477298

Dt228/4

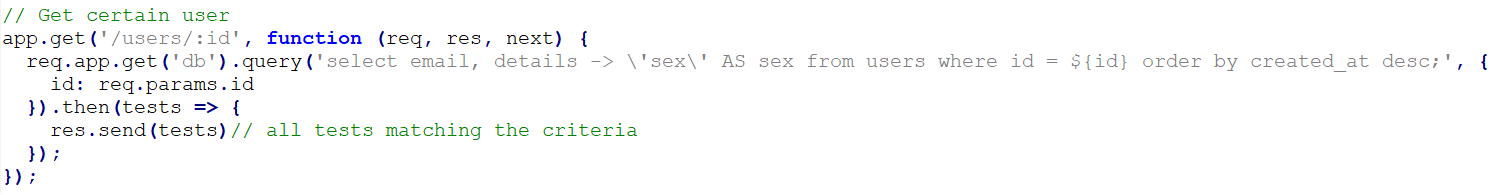
Lab One

Enterprise Application Development

1. Using Node, Express and Massive create the following HTTP API endpoints serving the following resources as JSON documents

|  |  |
| --- | --- |
| GET /users | List all users email and sex in order of most recently created. Do not include password hash in your output |

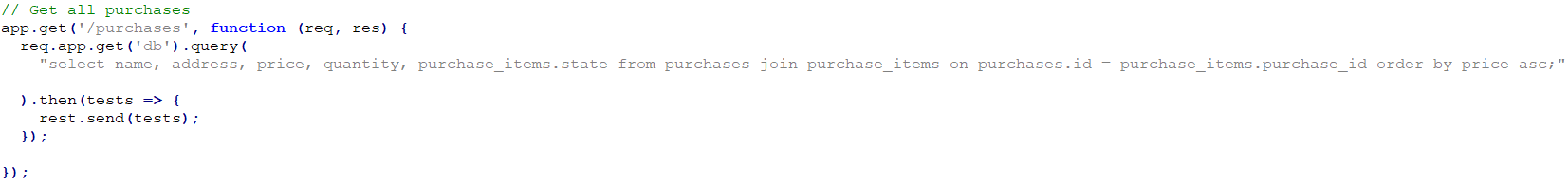
|  |  |
| --- | --- |
| GET /users/:id | Show above details of the specified user |



|  |  |
| --- | --- |
| 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 |

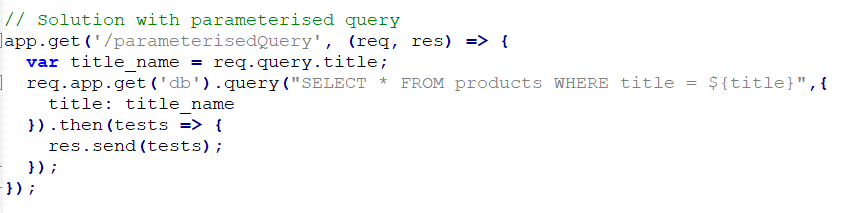


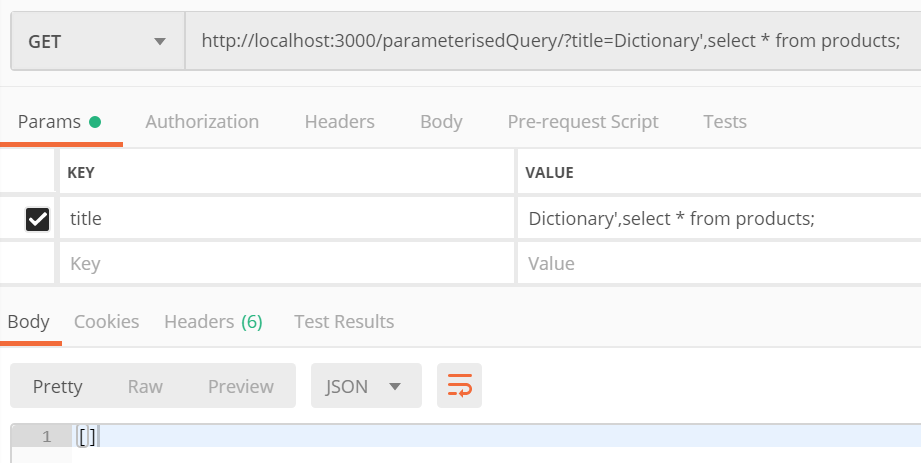
1. Building on your solution to part 1 for the API to the products resource from the pgguide database, extend the product indexing endpoint to allow the filtering of products by name as follows

|  |
| --- |
| GET /bad/products |

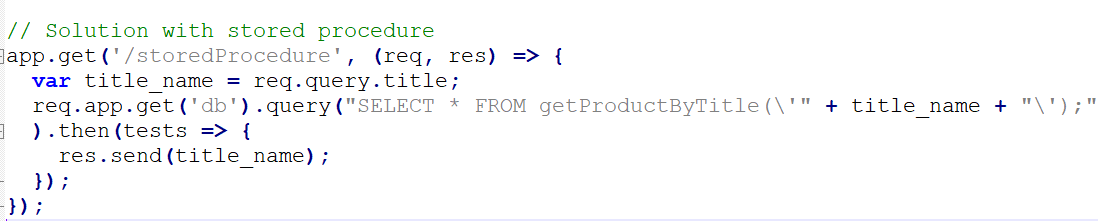
1. Provide two solutions to eliminate the security hole in your approach from the previous section as follows:

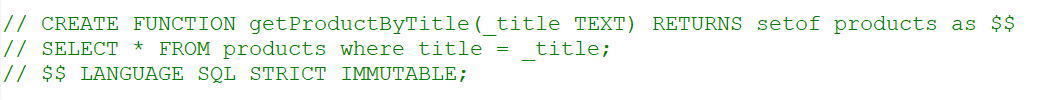
* **Using a parameterised query:** verifies that SQL injection doesn’t work.

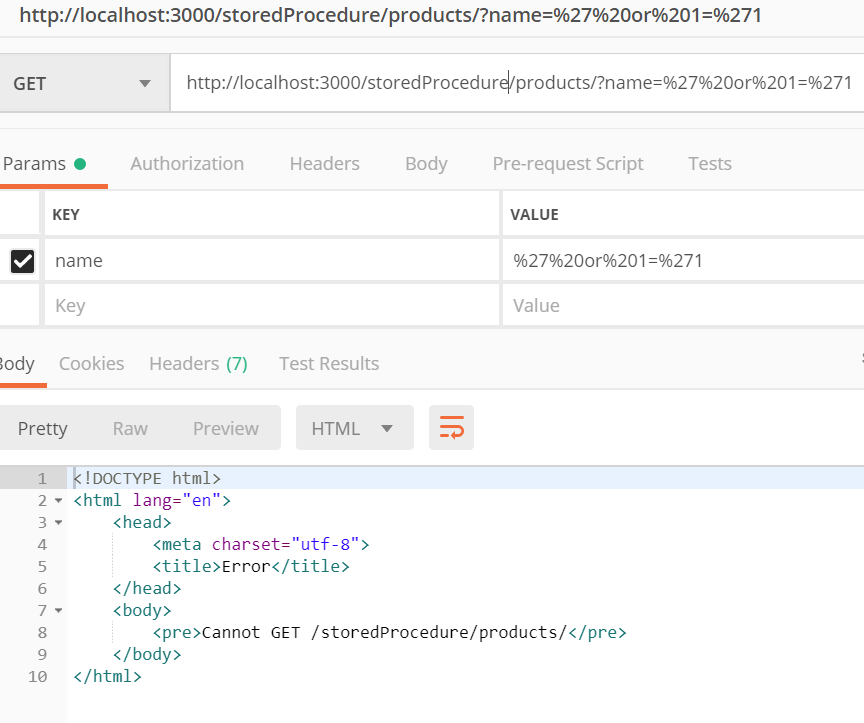




* **Using a stored procedure using SQL or PLPGSQL**



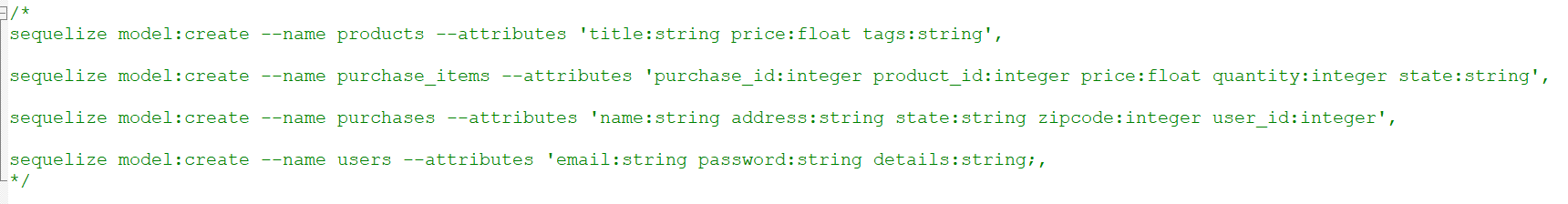


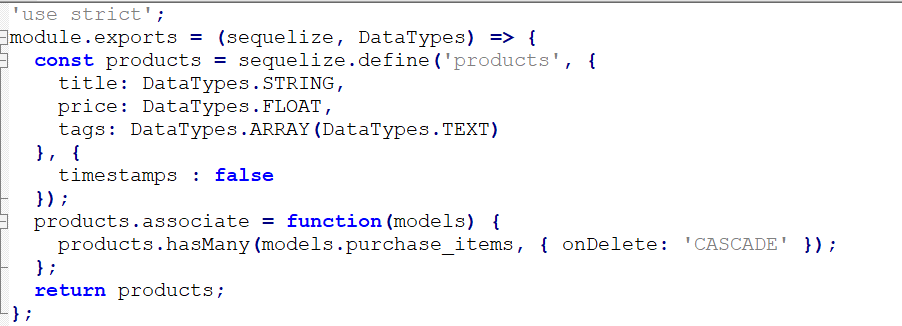


1. Create a brand-new Express project using the Sequelize ORM. Install and configure Sequelize and wire it up to the pgguide database. Verify that you have basic connectivity before proceeding.

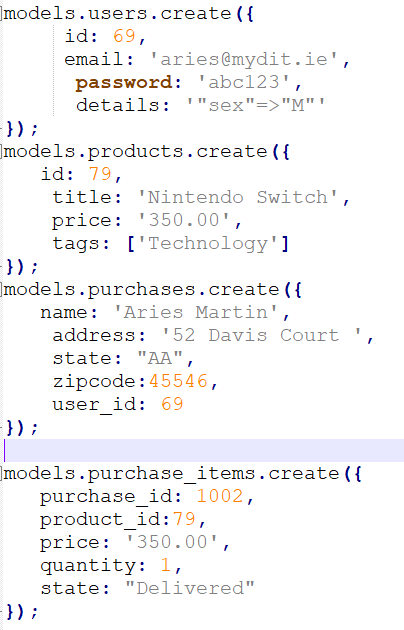
Create Sequelize migrations for the pgguide sample database

1. Use your models and Javascript code to populate the database with some additional test data for all the models above



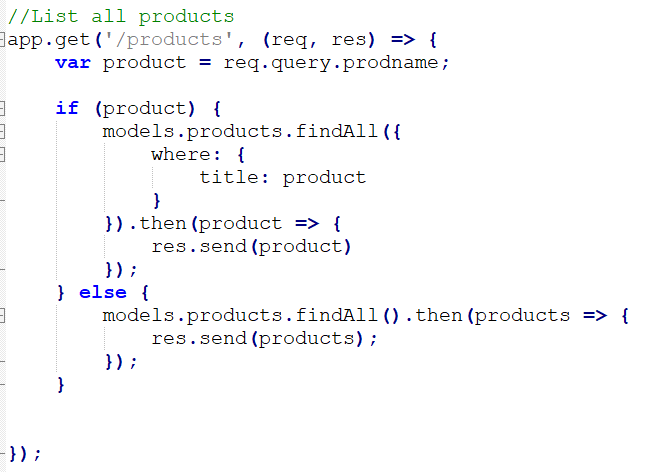


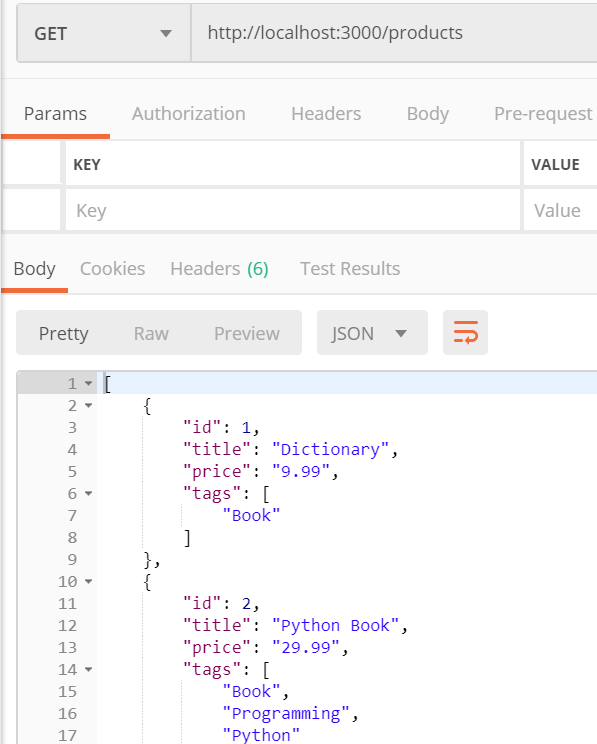
Now to populate the tables:



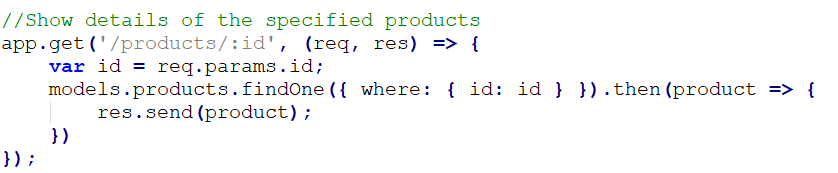
1. Reimplement the RESTful API using Sequelize and Express for your system. Your API should support the following CRUD operations as follows, returning JSON responses

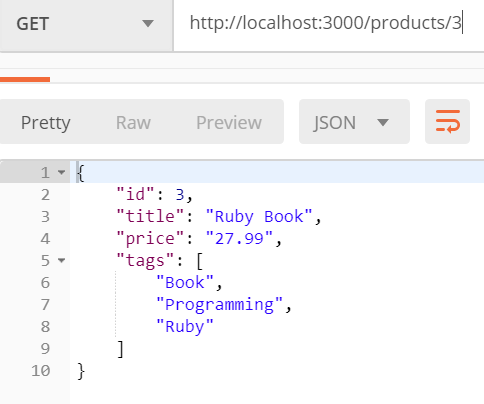
|  |  |
| --- | --- |
| GET /products[?name=string] | List all products |



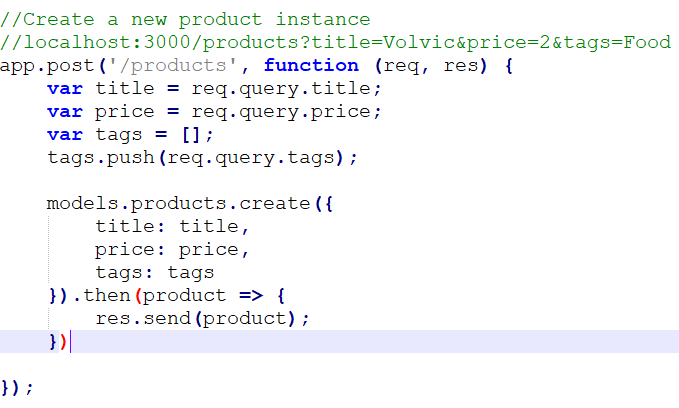


|  |  |
| --- | --- |
| GET /products/:id | Show details of the specified products |

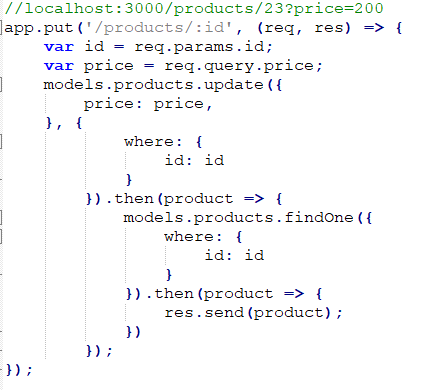




|  |  |
| --- | --- |
| POST /products | Create a new product instance |



|  |  |
| --- | --- |
| PUT /products/:id | Update an existing product |



|  |  |
| --- | --- |
| DELETE /products/:id | Remove an existing product |

