Modules used: Express.js, node.js, pg-promise, pg-node

\*DB = database.

**Purpose of test:**

To independently test all of the CRUD functions and core feature APIs before integrating them into a complete unit. CRUD functions include: ability to create and insert data into the DB, ability to read and extract data from DB, ability to update data and ability to remove and delete data from the DB.

**Test cases:**

1) Read and pull data from DB from backend (Successful)

2) Create and insert data into DB from backend (Successful)

3) Update data in DB from backend (Successful)

4) Delete data from DB from backend (Successful)

**Test data set:**

There are 2 tables created in the DB called tester1 and tester2. Each table as 2 columns: ID and Name. Column ID is set to be an integer and cannot be a null value. Column Name is set as a varchar (20) that cannot be null as well.

\*Left is tester1 and right is tester2.

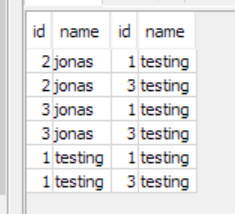
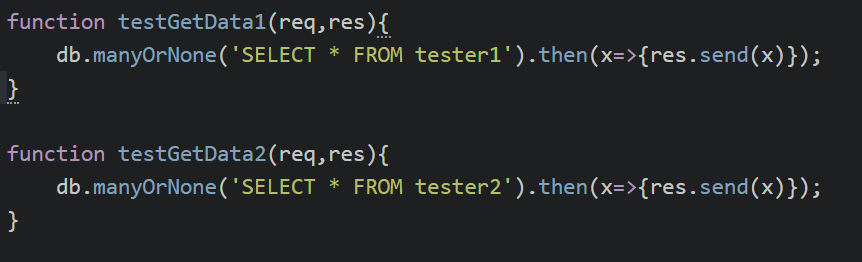


Fig 1.1

**Test case 1:**



**Purpose of test:** To extract data from the backend and send as a GET request to the backend.

**Expected results:**

**testGetData1:** An array that contains all distinct rows as no duplicates are shown.

**testGetData2:** An array that contains all distinct rows as no duplicates are shown.

**Actual Results**:

**testGetData1:**

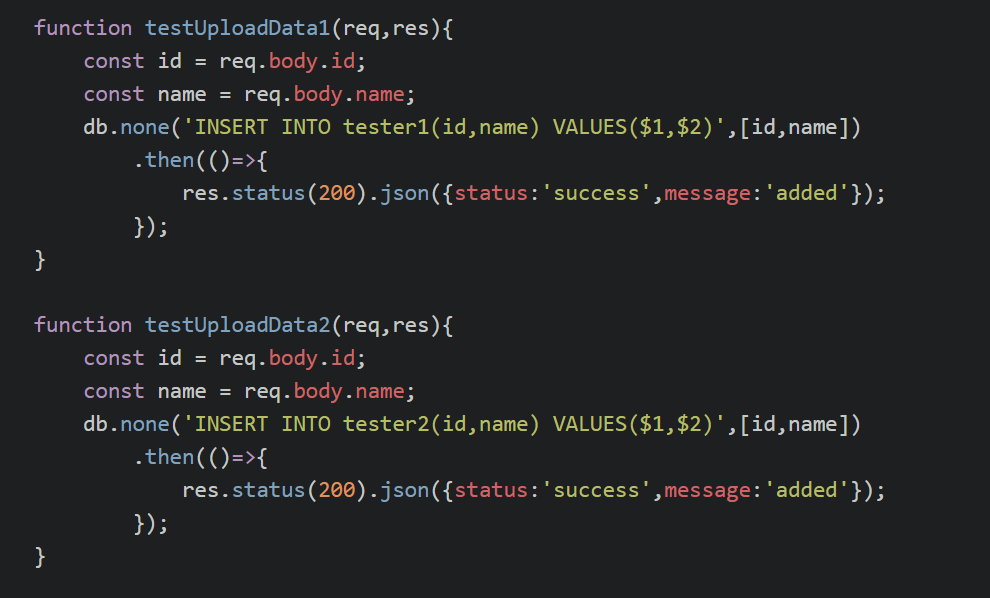


**testGetData2:**



**Conclusion:** Both tests are successful in getting data out from the DB.

**Test case 2:**



**Purpose of test:** To see if we are able to send data and insert into the DB from the backend.

**Test case 2.1:** insert data into tester2 before tester1.

**Data to add:**

**Tester2:** ID(5), Name(unit testing2)

**Test case 2.2:** insert data into tester1, then tester2

**Tester1:** ID(5), Name(unit testing1)

**Tester2:** ID(5), Name(unit testing2)

**Expected results:**

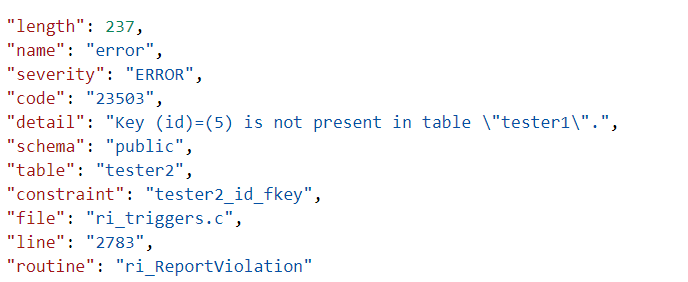
Able to add data into both tables. Since tester2 is related to tester1 with a 1 – 1 relation, we must add data into tester1 first, before we can add into tester2. This is an additional test to check for conditional data.

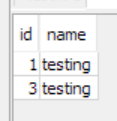
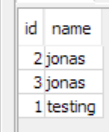
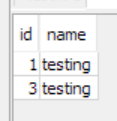
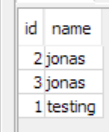
**Test case 2.1:** expect error, unable to insert data into DB.

**Test case 2.2:** successful insertion.

**Actual results:**

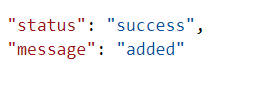
**Test case 2.1:**

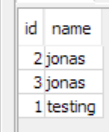
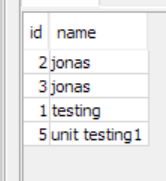


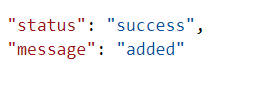
Before After

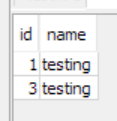
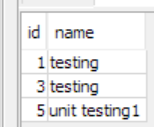
**Test case 2.2:**



Before(tester1) After(tester1)



Before(tester2) After(tester2)

**Conclusion:** Successful testing of POST requests and insertion into DB from backend.

**Test case 3:**

**Purpose of the test:** To update and change existing data in the DB as per requested.

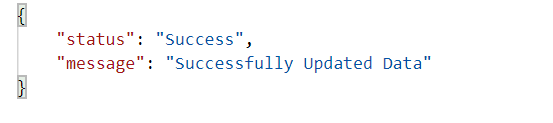
**Data to modify:**

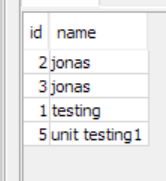
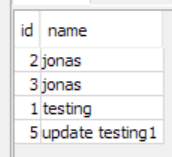
**Tester1:** From **id=5,name=unit testing1** to **id=5,name=update testing1**

**Expected results:**

Data in table tester1 should have no rows with id=5,name=unit testing1 and have a row with id=5,name=update testing1.

**Actual results:**

****

 ****

Before(tester1)After(tester1)

**Conclusion:** Successfully update the data in the DB from the backend.

**Test case 4:**

**Purpose of the test:** To remove and delete data from the DB from backend.

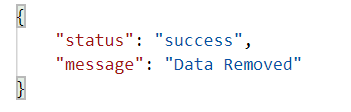
**Data to delete:**

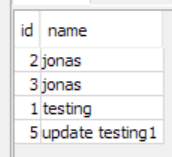
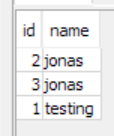
**Tester1:** delete the row where id=5,name=update testing1

**Expected results:**

There should be only 3 rows of data of id 1,2,3.

**Actual results:**

****

** **

Before(tester1) After(tester1)

**Conclusion:** Successful removal of data from DB through the backend.