Name: Jaime Valencia
Program: CLOUD DEVELOPER
Project: #4 **Serverless**

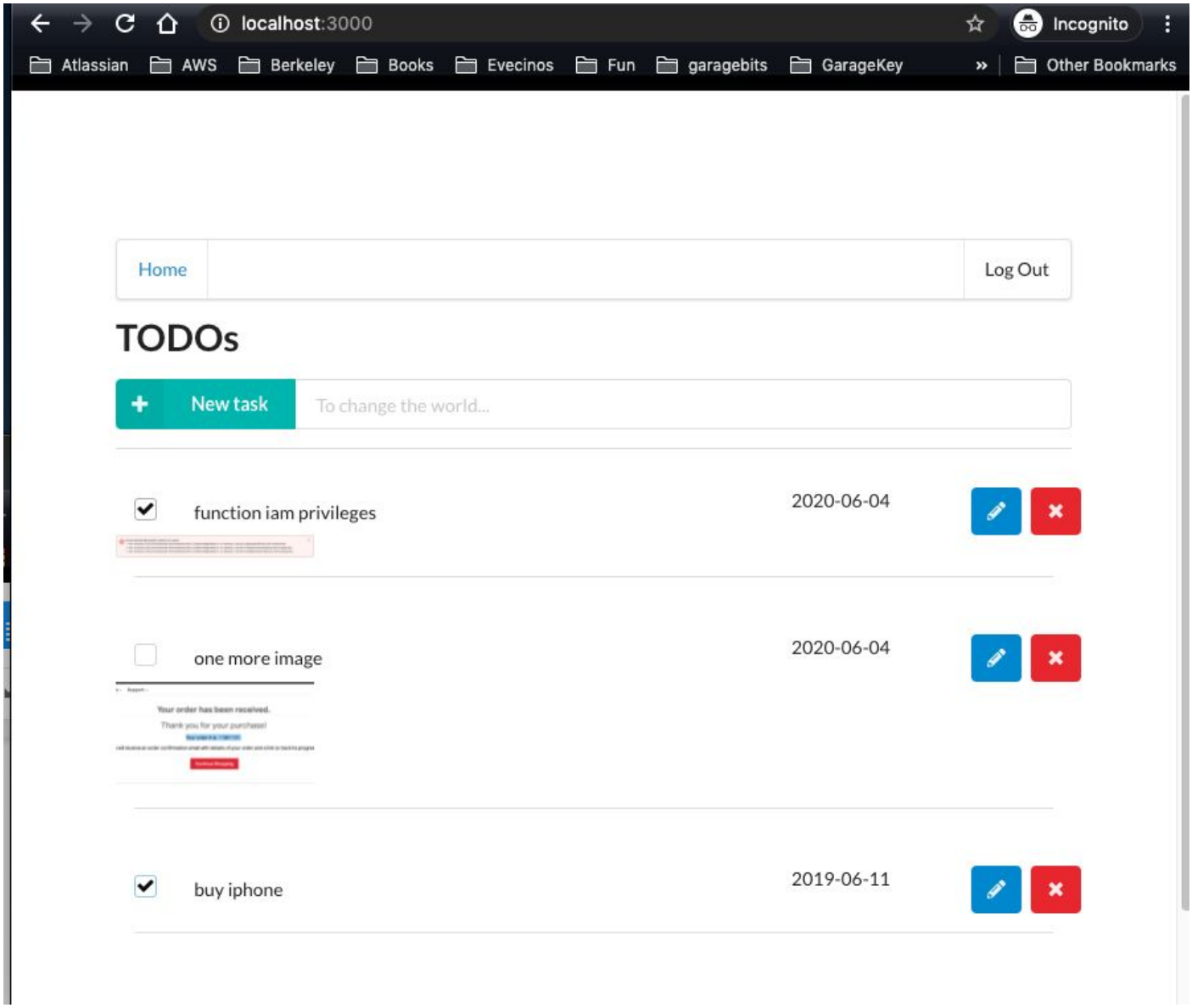
**User to test with the Front End**

User: jaimetest@vaduinc.com
Password: Jaimetest123

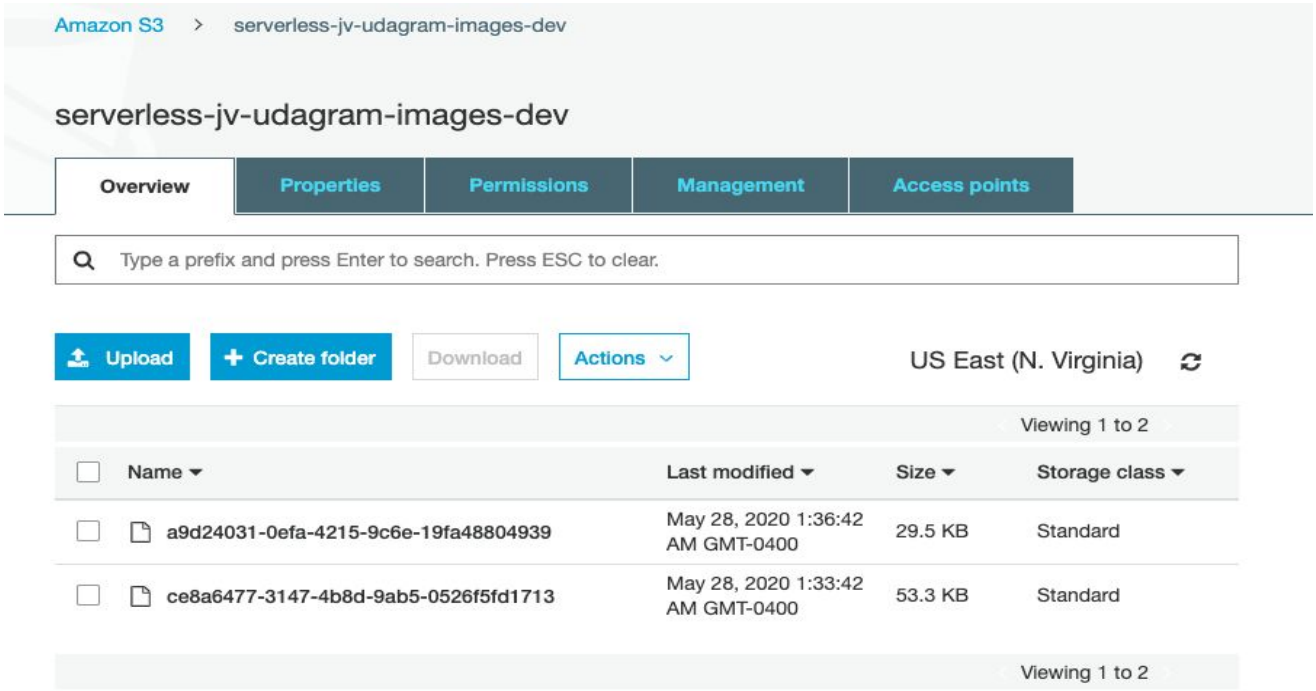
User: anotheruser@vaduinc.com
Password: Anotheruser123

# 1. Functionality

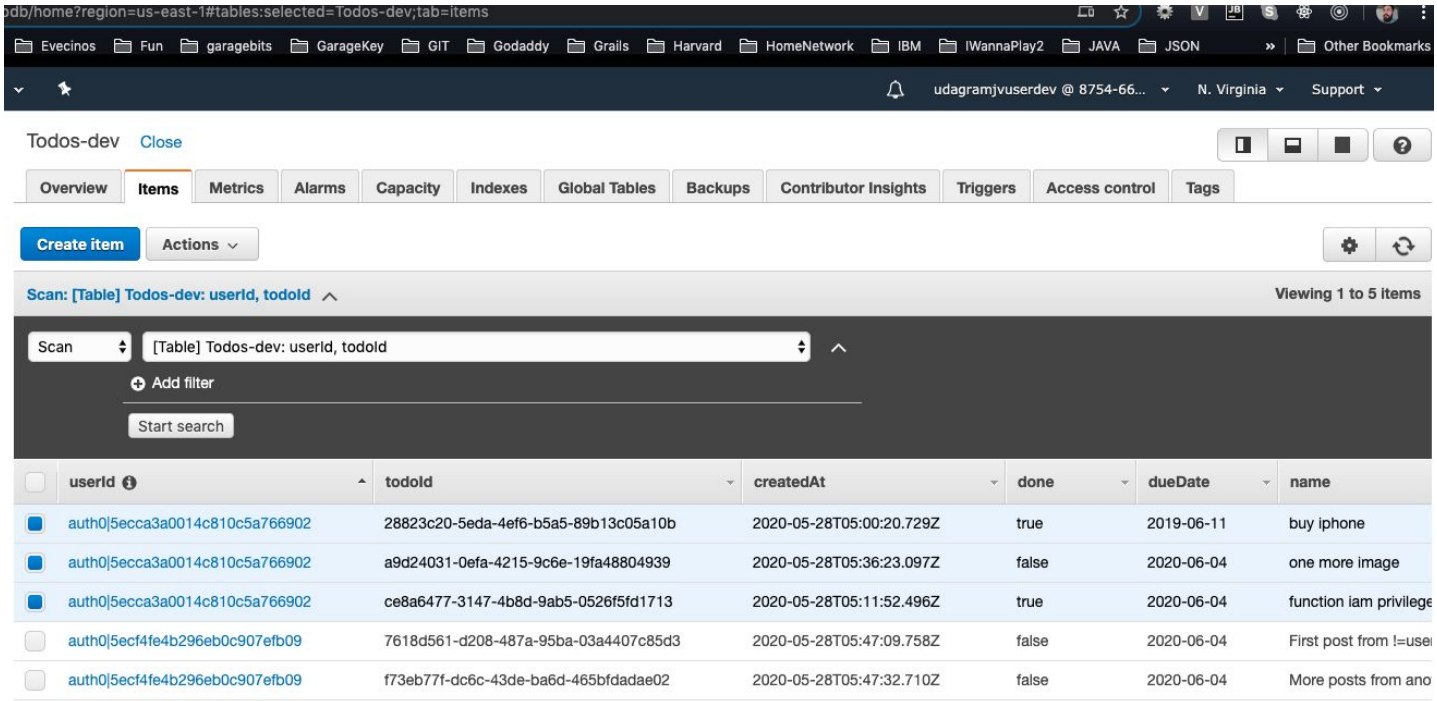
a. The application allows users to create, update, delete TODO items



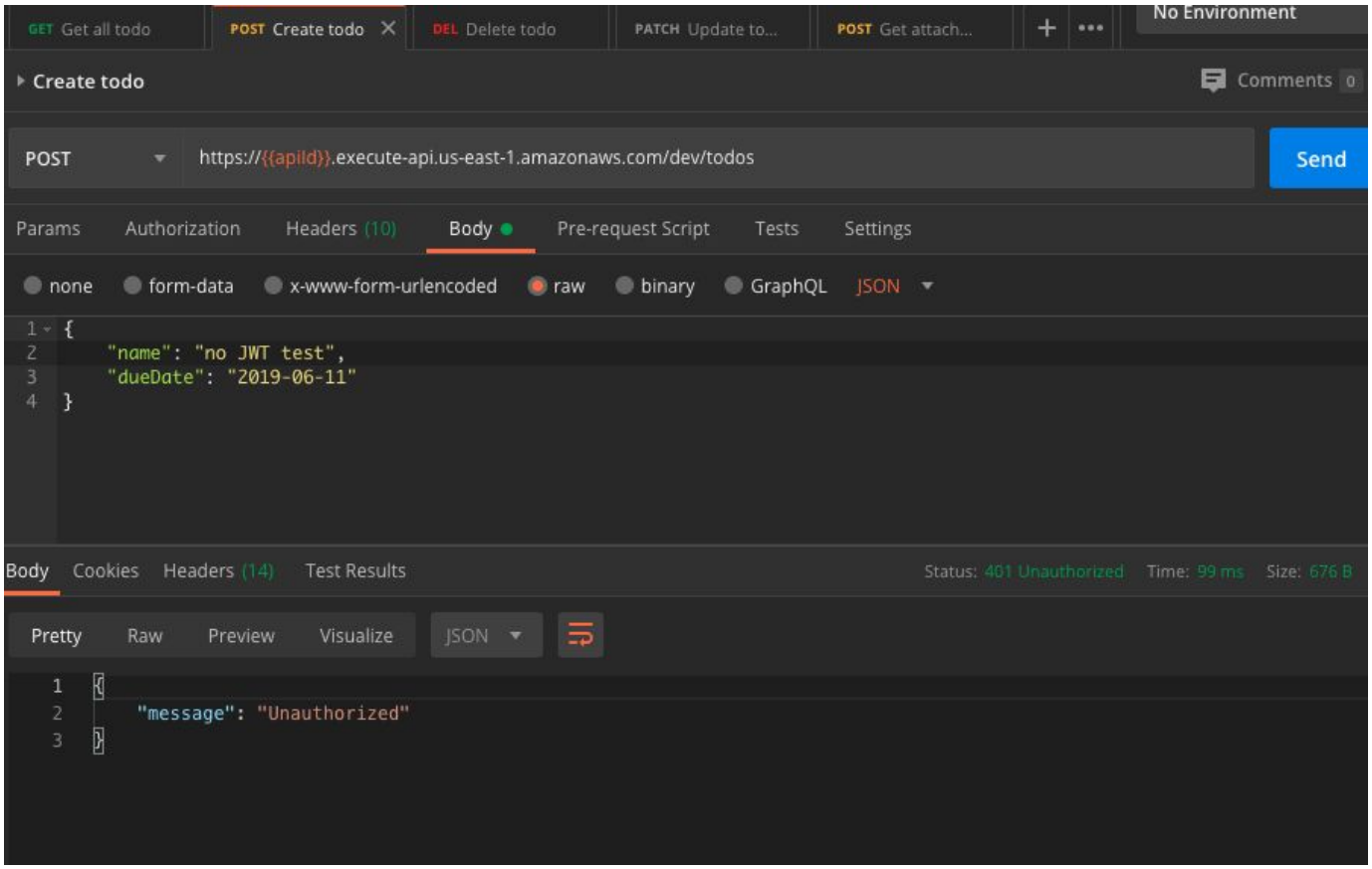
b. The application allows users to upload a file: see next picture of S3 with uploaded images using the FE application. The images names are the todoId of the items.

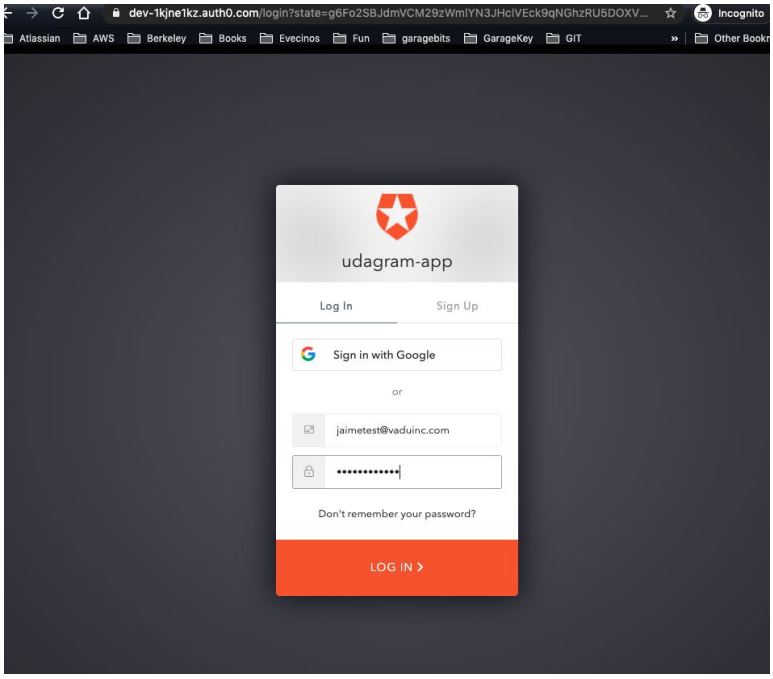



c. The application only displays TODO items for a logged in user: there are 5 todos in the dynamoDB but only 3 are displayed in the FE (see bullet point a. picture)



d. Authentication is implemented and does not allow unauthenticated access: JWT token is required to access the API end-point and Auth0 was set to login into the FE

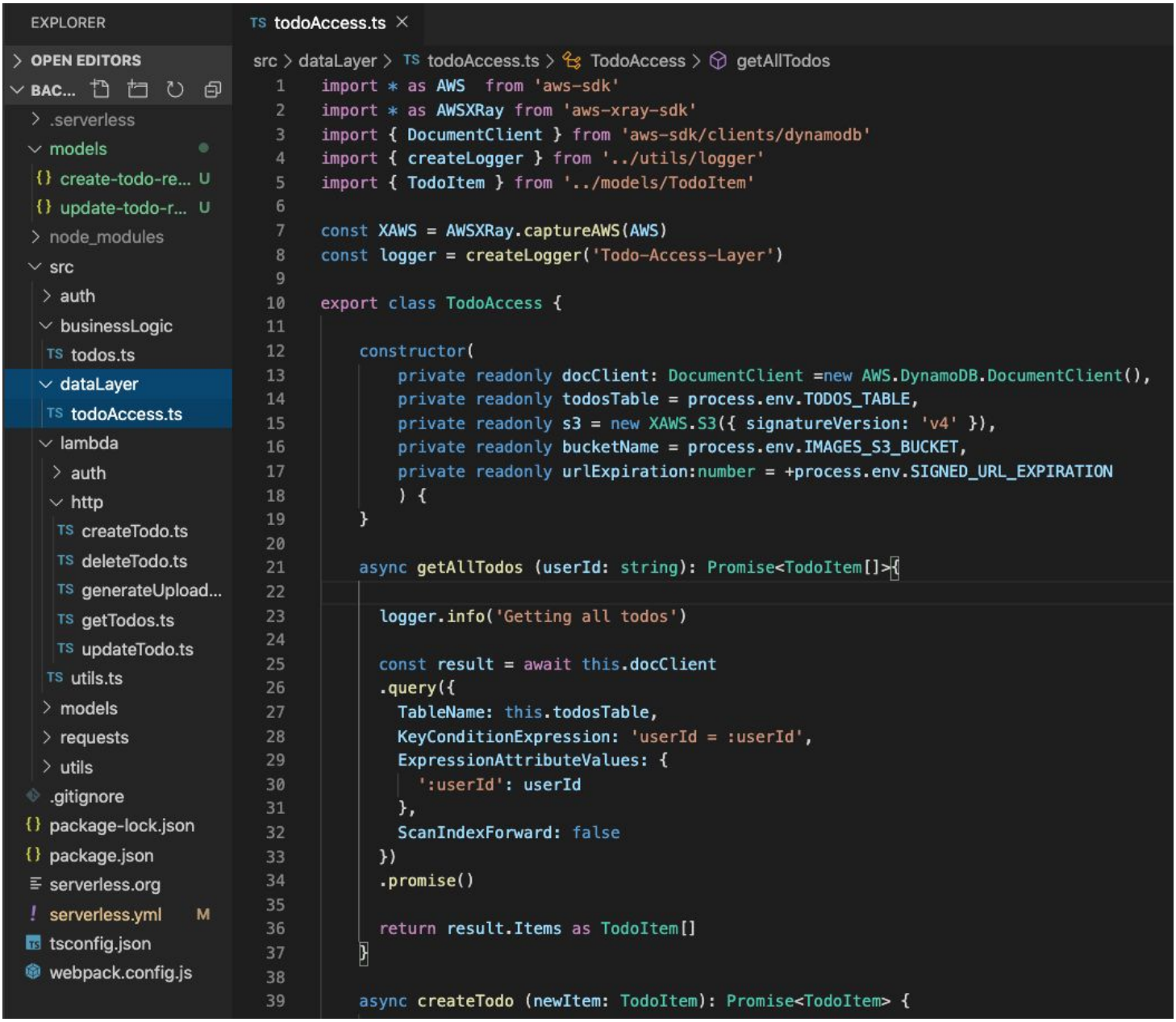


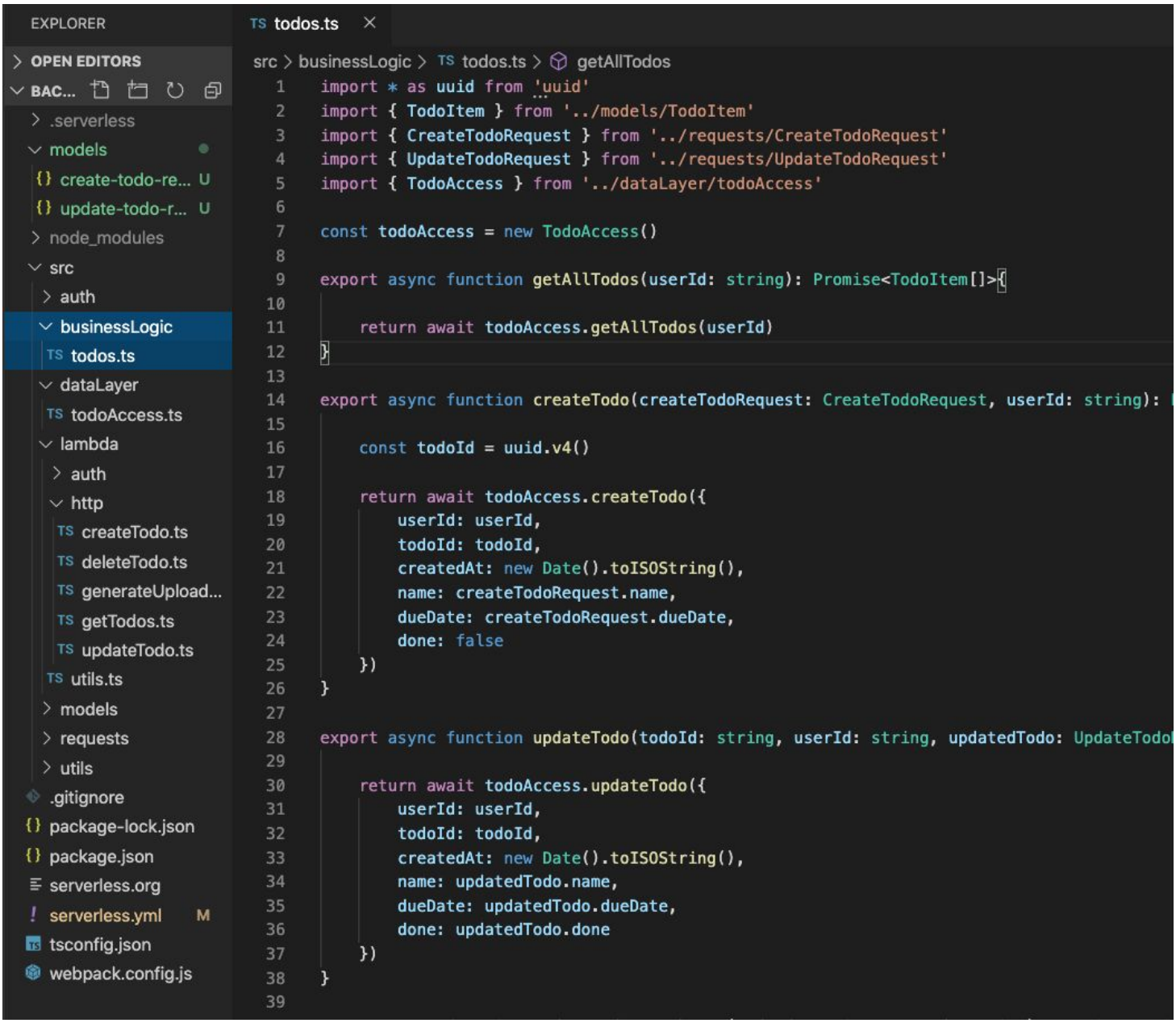
# 2. Code Base

a. The code is split into multiple layers separating business logic from I/O related code:

- Data Layer



- Business Layer



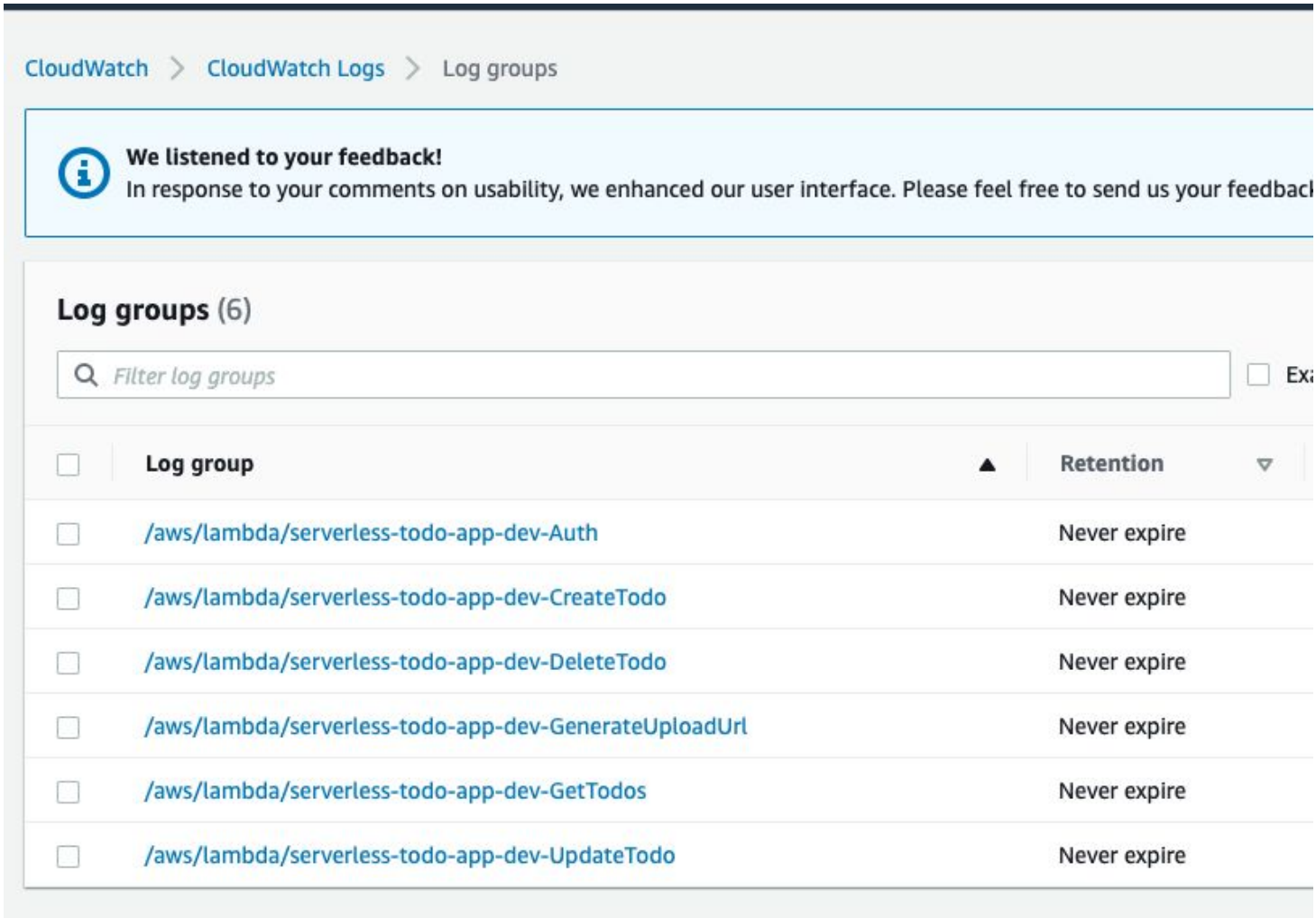

b. Code is implemented using async/await and Promises without using callbacks: I used *nodejs10.x* in the YAML file. There are no callbacks in the code and I used async/await in most of the functions. See above pictures.

# 3. Best Practices

a. All resources in the application are defined in the "serverless.yml" file: All resources needed by an application are defined in the "serverless.yml". See file in the code.

b. Each function has its own set of permissions: See “serverless.yml” file in the code

c. Application has sufficient monitoring. It has a sufficient amount of log statements



Details about CreateTodo lambda

CloudWatch > CloudWatch Logs > Log groups > /aws/lambda/serverless-todo-app-dev-CreateTodo

# /aws/lambda/serverless-todo-app-dev-CreateTodo

## Log group details

| Retention | Creation time | Stored bytes |
|---|---|---|
| Never expire | 17 hours ago | - |
| **KMS key ID** | **Metric filters** | **Subscriptions** |
| - | 0 | - |

Log streams | Metric filters | Contributor Insights

## Log streams (4)

Filter log streams

| Log stream |
|---|
| 2020/05/28/[$LATEST]934e0521372a4a9992f0417811c5dd04 |
| 2020/05/28/[$LATEST]cfdbb8c0123c478c80560c74e6e1100f |
| 2020/05/28/[$LATEST]1b16cf76aafe403d844c310c9cad698e |
| 2020/05/28/[$LATEST]e342c668aeb34a0b9f378146b32b7148 |

d. HTTP requests are validated. See next validation schemas created for requests:

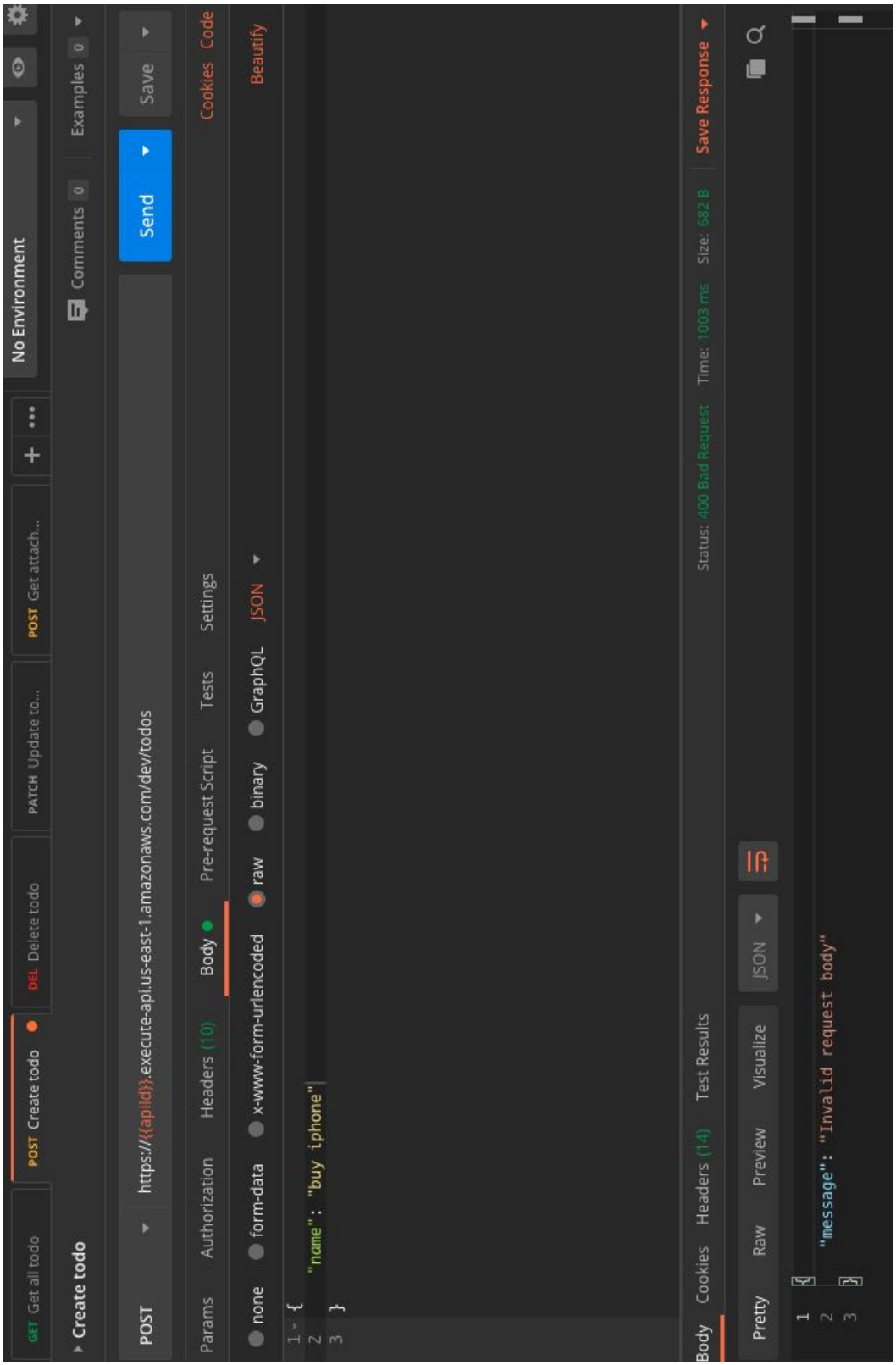
## *Create-todo-request.json*

```
{
   "$schema": "http://json-schema.org/draft-04/schema#",
   "title": "createTodo",
   "type": "object",
   "properties": {
     "name": {
       "type": "string"
     },
     "dueDate": {
       "type": "string"
     }
   },
   "required": [
     "name",
     "dueDate"
   ],
   "additionalProperties": false
}
```

## *Update-todo-request.json*

```
{
   "$schema": "http://json-schema.org/draft-04/schema#",
   "title": "updateTodo",
   "type": "object",
   "properties": {
     "name": {
       "type": "string"
     },
     "dueDate": {
       "type": "string"
     },
     "done": {
       "type": "boolean"
     }
   },
   "required": [
     "name",
     "dueDate",
     "done"
   ],
   "additionalProperties": false
}
```

See next image. It is a test to add a TODO item without a *dueDate.* A return 400 error is received.



Then tried it again adding the dueDate field. A successful response is received. See next image.

GET Get all todo | POST Create todo | DEL Delete todo | PATCH Update to... | POST Get attach...

No Environment

Create todo

Comments 0

POST https://{{apiId}}.execute-api.us-east-1.amazonaws.com/dev/todos

Send

Params | Authorization | Headers (10) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON

```
{
    "name": "buy iphone",
    "dueDate": "2019-06-11"
}
```

Body | Cookies | Headers (12) | Test Results

Status: 201 Created | Time: -- | Size: 720 B

Pretty | Raw | Preview | Visualize | JSON

```
{
    "item": {
        "userId": "auth0|5ecca3a0014c810c5a766902",
        "todoId": "28823c20-5eda-4ef6-b5a5-89b13c05a10b",
        "createdAt": "2020-05-28T05:00:20.729Z",
        "name": "buy iphone",
        "dueDate": "2019-06-11",
        "done": false
    }
}
```

# 4. Architecture

a. Data is stored in a table with a composite key. See the following TABLE definition from the *serverless.yml* file definition.

```
  TodosTable:
    Type: AWS::DynamoDB::Table
    Properties:
      AttributeDefinitions:
        - AttributeName: userId
          AttributeType: S
        - AttributeName: todoId
          AttributeType: S
        - AttributeName: dueDate
          AttributeType: S
      KeySchema:
        - AttributeName: userId
          KeyType: HASH
        - AttributeName: todoId
          KeyType: RANGE
      BillingMode: PAY_PER_REQUEST
      TableName: ${self:provider.environment.TODOS_TABLE}
      LocalSecondaryIndexes:
        - IndexName: ${self:provider.environment.INDEX_NAME}
          KeySchema:
            - AttributeName: userId
              KeyType: HASH
            - AttributeName: dueDate
              KeyType: RANGE
          Projection:
            ProjectionType: ALL
```

b. Scan operation is not used to read data from a database. Only a query() call was used.