Second Partial Exam. TC2026: Web Applications Development. Jul-Aug 2020

NAME: Tomas Alejandro Lugo Salinas
GITHUB REPOSITORY: https://github.com/tomvik/Web_Parcial_2

Instructions:

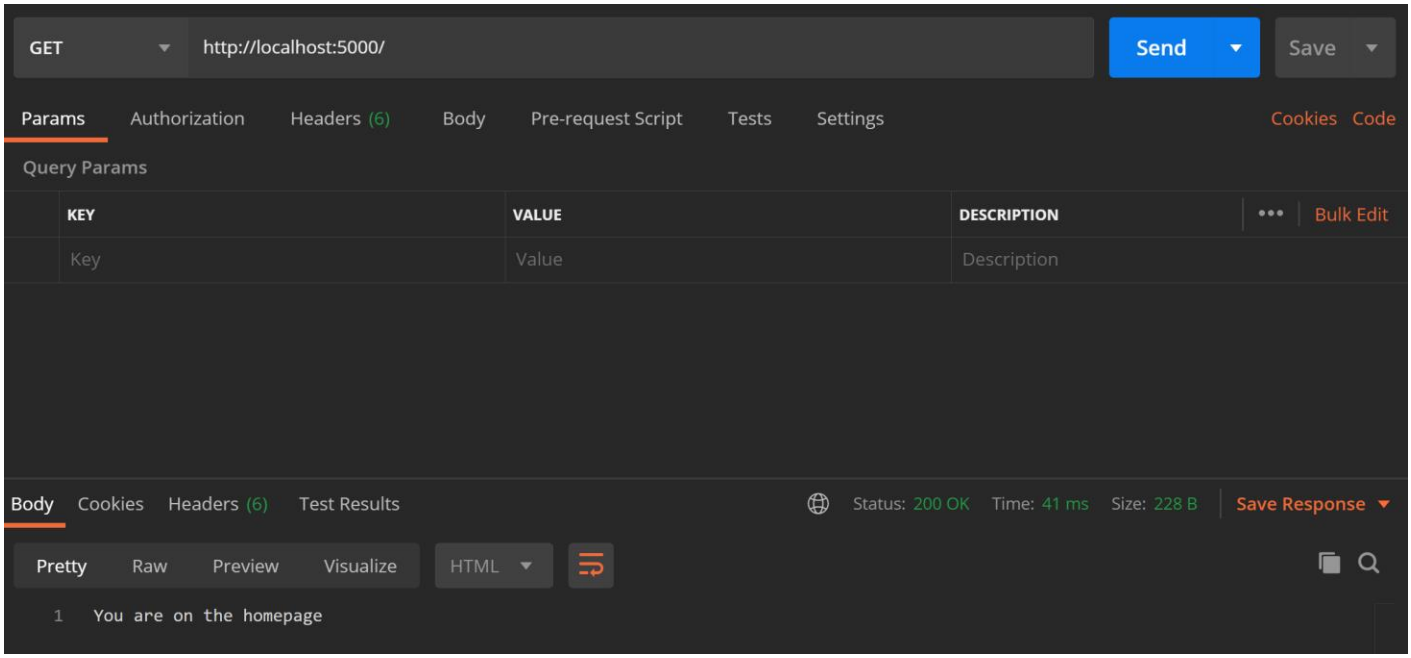
- You must have your webcam turned on.
- **Turn off your cellphone and close any social media site.**
- You are allowed to use the following during the exam:
  - Command line / Terminal / Gitbash.
  - The editor of your preference to write the coding solutions.
  - Material from class is allowed.
- Code everything from zero.
- When you finish the exam, you will need to upload to CANVAS this file.
- You cannot ask any classmate for anything. This exam is a test of how much you have learned.

Part 1 - Express

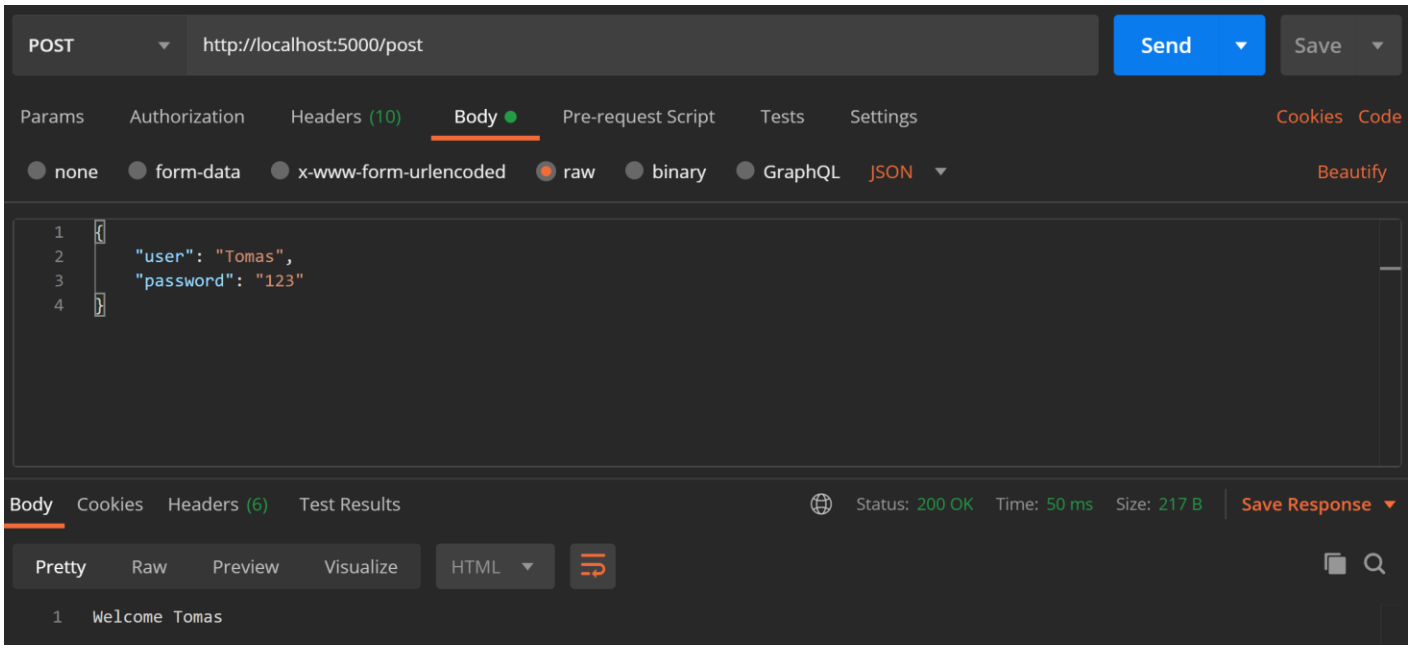
1. Create file called "server.js" and write all necessary code to have a node-express server which runs on port 5000. (Don't forget to initialize your project).

2. Create the following routes using the necessary types of http requests. You must include a screenshot of each request with the corresponding response using any of: Postman, Rest Client, Insomnia.

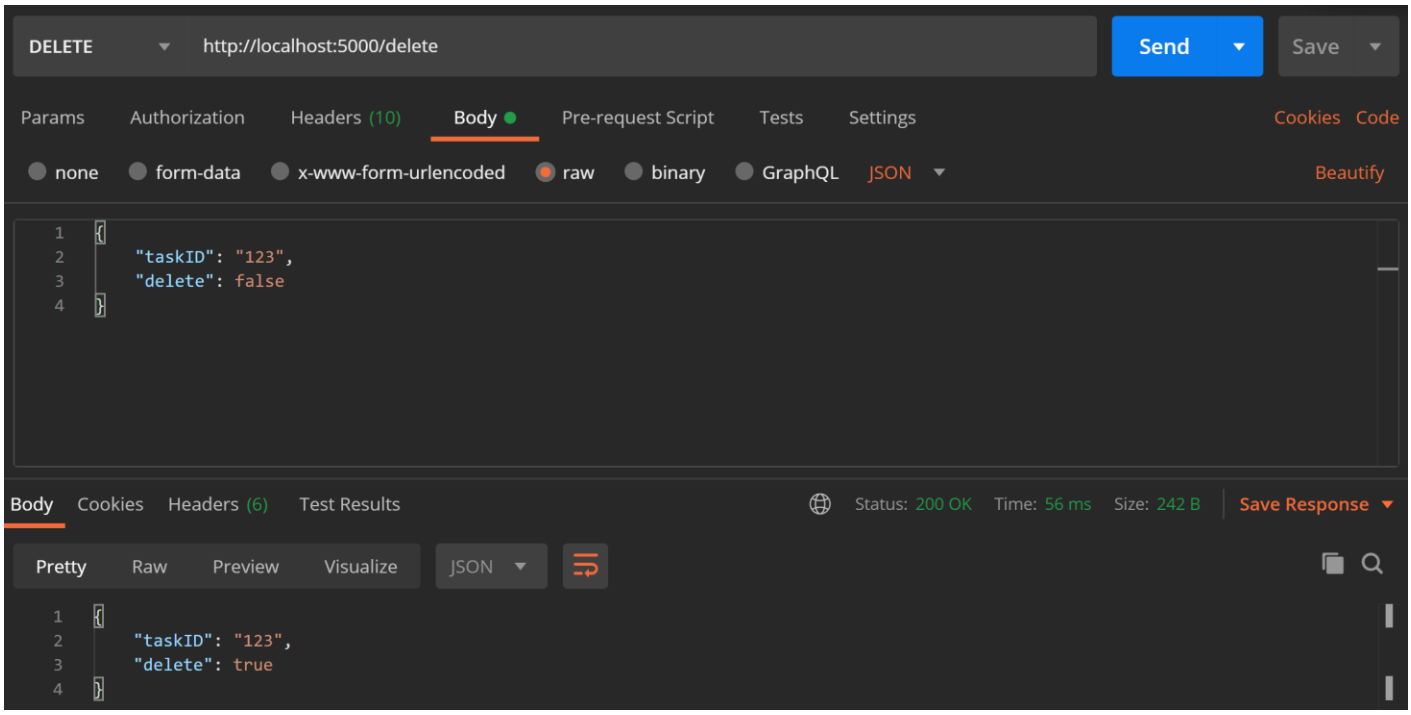
   a. ***GET*** *listening on url: "/" which replies to the client a text: "You are on the homepage".*



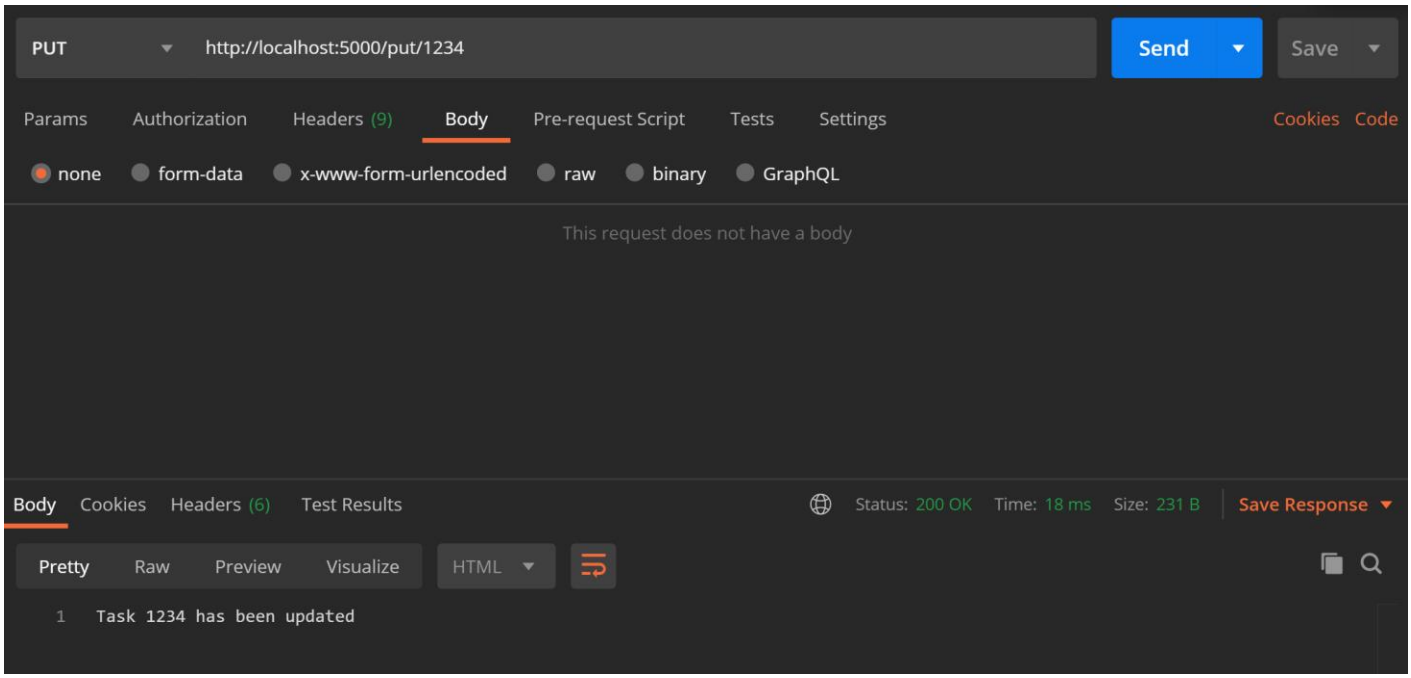
b. ***POST*** *listening on url: "/post" which must send to the server a json object with the following fields:* ***user, password.***
Send a request (with random data) and reply back to the user "Welcome {user}" (where user must be the value sent in the request). ***HINT:*** *You need to include a middleware so the server understands the json data it is receiving.*



c. ***DELETE*** listening on url: "/delete" which must send a json object with the following field: **taskId**
Send a request with a **taskId** and reply back to the user: "{delete: true}



d. ***PUT*** listening on url "/put/**{ID}**" which does not send anything in the body.
Send a request like: "/put/123 and reply back to the user: "Task 123 has been updated"

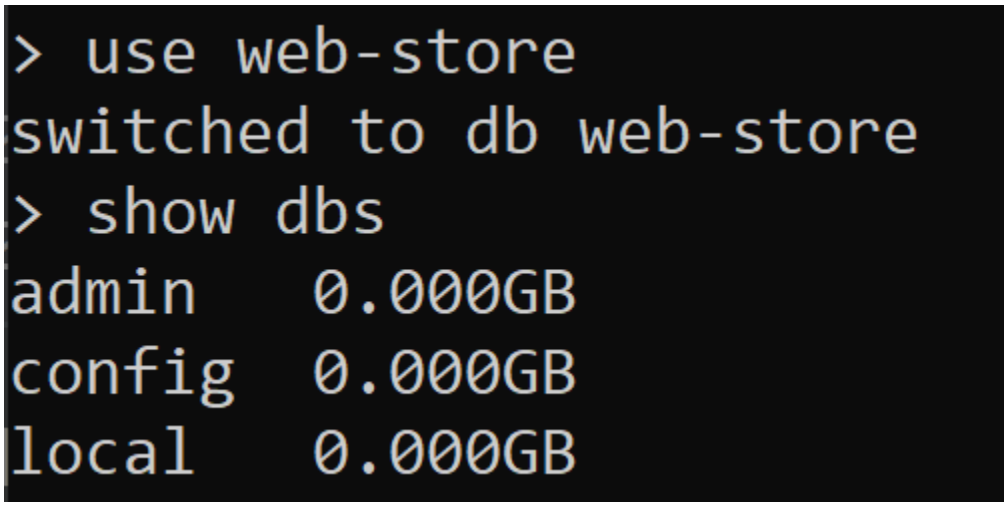


After you are done create a new github repository and upload your project. Be sure to paste your github repo in the top of this file.

Part 2 – Mongo

In this section you need to write the requested queries. You must attach a screenshot that contains the query and the result of each query. If you have issues with your local MongoDB , you can use the following web-shell:
https://docs.mongodb.com/manual/tutorial/getting-started/

1. Write a query to: Create a Database called: **web-store**



2. Write a query to: Create a Collection called: **products**

```
> db.createCollection('products')
{ "ok" : 1 }
```

```
> show dbs
admin      0.000GB
config     0.000GB
local      0.000GB
web-store  0.000GB
```

3. Write a query to: show the list of available collections.

```
> show collections
products
```

4. Write a query to insert a document to a Collection with the following fields/data:

   name: “shoes”
   cost: 199.99
   stock: 10
   date_added: [current Date]

```
> db.products.insert( { name: "shoes", cost: 199.99, stock: 10, date_added: Date()  } )
WriteResult({ "nInserted" : 1 })
```

5. Write a query that: shows all available products in the **products** Collection.

```
> db.products.find()
{ "_id" : ObjectId("5f20b85c0a6faf157cac2e3f"), "name" : "shoes", "cost" : 199.99, "stock" : 10, "date_added" : "Tue Jul
 28 2020 18:44:28 GMT-0500 (Central Daylight Time (Mexico))" }
```

6. Add a second document to the **products** Collection with the following data:

   name: “sun-glasses”
   cost: 500
   stock: 2

```
> db.products.insert( { name: "sun-glasses", cost: 500, stock: 2 } )
WriteResult({ "nInserted" : 1 })
> db.products.find()
{ "_id" : ObjectId("5f20b85c0a6faf157cac2e3f"), "name" : "shoes", "cost" : 199.99, "stock" : 10, "da
 28 2020 18:44:28 GMT-0500 (Central Daylight Time (Mexico))" }
{ "_id" : ObjectId("5f20b9020a6faf157cac2e40"), "name" : "sun-glasses", "cost" : 500, "stock" : 2 }
```

7. Write a query that: shows all available products in the **products** Collection using the “pretty” mode.

```
> db.products.find().pretty()
{
        "_id" : ObjectId("5f20b85c0a6faf157cac2e3f"),
        "name" : "shoes",
        "cost" : 199.99,
        "stock" : 10,
        "date_added" : "Tue Jul 28 2020 18:44:28 GMT-0500 (Central Daylight Time (Mexico))"
}
{
        "_id" : ObjectId("5f20b9020a6faf157cac2e40"),
        "name" : "sun-glasses",
        "cost" : 500,
        "stock" : 2
}
```

8. Write a query to update the “stock” of the “sun-glasses” from the current stock to 20. Also write another query to show that the update worked showing the new stock. (In total here 2 queries)

```
> db.products.update ( { name: "sun-glasses" }, { $set: { stock: 20 } } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.products.find().pretty()
{
        "_id" : ObjectId("5f20b85c0a6faf157cac2e3f"),
        "name" : "shoes",
        "cost" : 199.99,
        "stock" : 10,
        "date_added" : "Tue Jul 28 2020 18:44:28 GMT-0500 (Central Daylight Time (Mexico))"
}
{
        "_id" : ObjectId("5f20b9020a6faf157cac2e40"),
        "name" : "sun-glasses",
        "cost" : 500,
        "stock" : 20
}
```

9. Write a query to delete product with name “shoes” and also write a query to visualize that the product is no longer present. (In total here 2 queries)

```
> db.products.deleteOne(
... {
... name : "shoes"
... }
... )
{ "acknowledged" : true, "deletedCount" : 1 }
> db.products.find().pretty()
{
        "_id" : ObjectId("5f20b9020a6faf157cac2e40"),
        "name" : "sun-glasses",
        "cost" : 500,
        "stock" : 20
}
```

10. Write a query to drop the current database.

```
> db.products.drop()
true
> show collections
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
```

“If you try and Fail, Congratulations.

Most People won’t even try”