Sesión # 15 Componente Práctico
Testing

Realizar pruebas unitarias con Jest y Supertest

En este componente vamos a probar una API http de Node usando jest y supertest. Usaremos jest para probar la API usando un desarrollo basado en pruebas.

1. Crea un nuevo proyecto para el desarrollo de eeste componente, utiliza el siguiente template inicial https://github.com/Misiontic-Ciclo-4A/sesion15-template
2. Como puedes ver ya la plantilla de la solución cuenta con una estructura, en donde se configura un servidor express para escuchar peticiones por medio de un puerto específico el cual en este caso es 8080. Para ejecutarlo utiliza los comandos: npm install y luego npm start
3. Ahora es momento de implementar los tests para nuestro archivo app.js
4. Crea un nuevo archivo en tu proyecto y nombralo `app.test.js`
5. Ahora, desde la terminal ejecuta el siguiente comando
   `npm i -D jest supertest`
6. Replica el siguiente código en `app.test.js` para importar lo necesario

   ```
   import request from "supertest";
   import app from "./app.js";
   ```

7. En el archivo app.js crea un endpoint para crear nuevos usuarios recibiendo como parámetros el username y la password. Con esto nos aseguramos que nuestra API esté recibiendo una respuesta

```
import express from "express";

const app = express();

app.use(express.json());
app.post("/users", async (req, res) => {
  const { password, username } = req.body;
  if (!password || !username) {
    res.sendStatus(400);
    return;
  }

  res.send({ userId: 0 });
});
```

8. Una vez implementado nuestro endpoint, utiliza supertest en app.test.js para hacer un mock de los posibles casos de éxito o fallo que puede tener la petición creada en el paso anterior.

   Algunos casos pueden ser:

   - Se recibe un usuario y contraseña válidos - EXITO
   - Cuando no se recibe el usuario y contraseña - ERROR

9. Ejecuta el siguiente comando en tu terminal para añadir una variable de entorno al proyecto que permita ejecutar el archivo app.test.js con jest.

   ```
   NODE_OPTIONS=--experimental-vm-modules npx jest
   ```

```
(node:90626) ExperimentalWarning: VM Modules is an experimental feature. This feature could change at any time
(Use `node --trace-warnings ...` to show where the warning was created)
 PASS  ./app.test.js
  POST /users
    given a username and password
      ✓ should respond with a 200 status code (125 ms)
      ✓ should specify json in the content type header (7 ms)
      ✓ response has userId (7 ms)
    when the username and password is missing
      ✓ should respond with a status code of 400 (16 ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        1.436 s
Ran all test suites.
```

10. Replica la implementación de los tests para el caso de éxito. Ten en cuenta Mock Functions

```
describe("given a username and password", () => {

    test("should respond with a 200 status code", async () => {

      const response = await request(app).post("/users").send({

        username: "username",
```

```
      password: "password",
    });
    expect(response.statusCode).toBe(200);
  });


  test("should specify json in the content type header", async () => {
    const response = await request(app).post("/users").send({
      username: "username",
      password: "password",
    });
    expect(response.headers["content-type"]).toEqual(
      expect.stringContaining("json")
    );
  });

  test("response has userId", async () => {
    const response = await request(app).post("/users").send({
      username: "username",
      password: "password",
    });
    expect(response.body.userId).toBeDefined();
  });
});
```

11. Replica la implementación de los tests para el caso de error [Mock Functions]

```
describe("when the username and password is missing", () => {
  // Se  debe retornar un error
  test("should respond with a status code of 400", async () => {
    const bodyData = [{ username: "username" }, { password: "password" }, {}];
```

```
    for (const body of bodyData) {
      const response = await request(app).post("/users").send(body);
      expect(response.statusCode).toBe(400);
    }
  });
});
```

12. Con ayuda del instructor analiza la implementación de las pruebas unitarias y ejecuta:

    `NODE_OPTIONS=--experimental-vm-modules npx jest`

    para validar el resultado de tu solución.

13. Listo! Tenga en cuenta que hemos implementado solo una versión rápida de tests, pero estos dependen netamente de los endpoints con los que cuente nuestra aplicación y la maquetación de todos los posibles casos de éxito o error que estén asociados o apliquen a ellos.

Documentación sugerida:

- **Jest JS**
- **Supertests**
- **Understanding Jest Mocks**
- **Mock functions**