IIC2513 - Aplicaciones y Tecnologías Web

Testing

Ayudantía 5

Francisca Otero & Diego Solari

mfotero@uc.cl dasolari@uc.cl

¿Qué es Testing?

Software testing es un proceso usado para evaluar la correctitud, completitud y la calidad de un programa de computador.

¿Qué es Testing?

Software testing es un proceso usado para evaluar la correctitud, completitud y la calidad de un programa de computador.

¿Qué es Testing?

Software testing es un proceso usado para evaluar la correctitud, completitud y la calidad de un programa de computador.

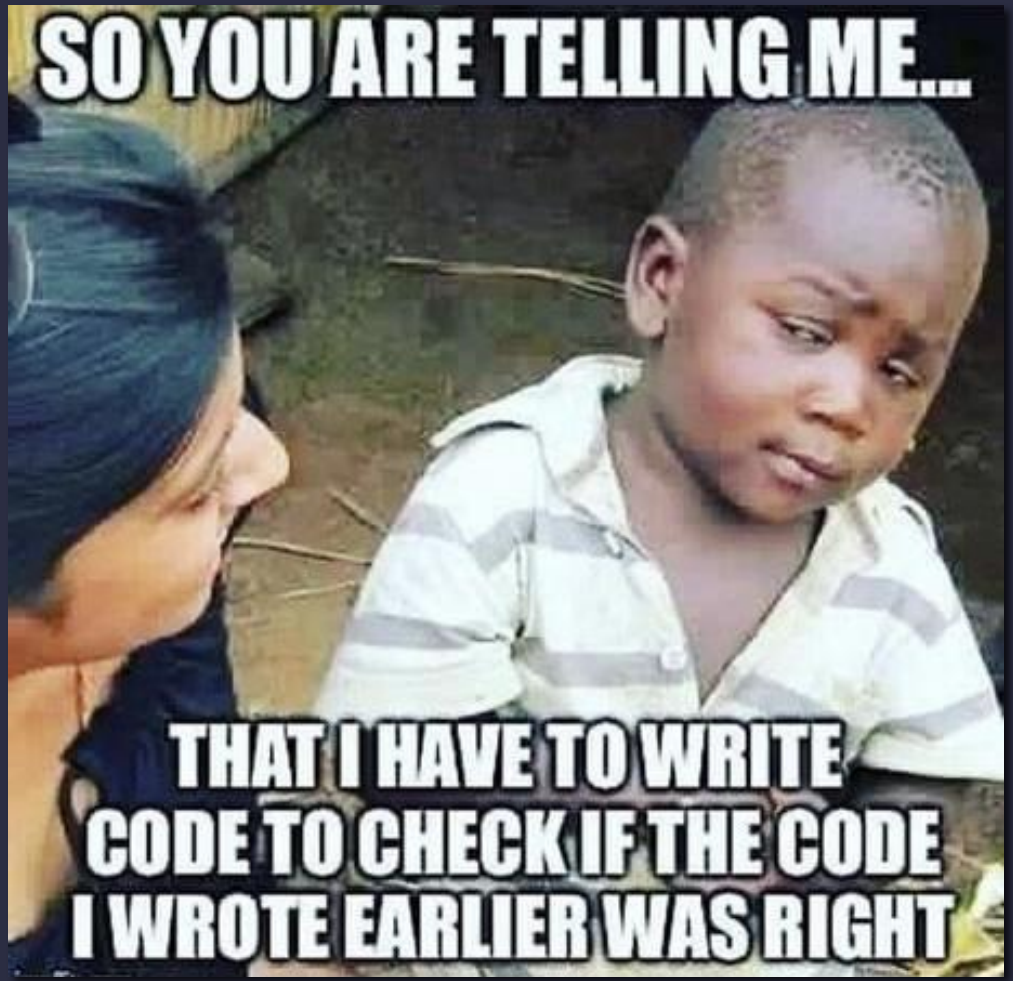
Nos permite principalmente...

Encontrar defectos o bugs en nuestro código

Aumentar la confianza en la calidad de nuestro código

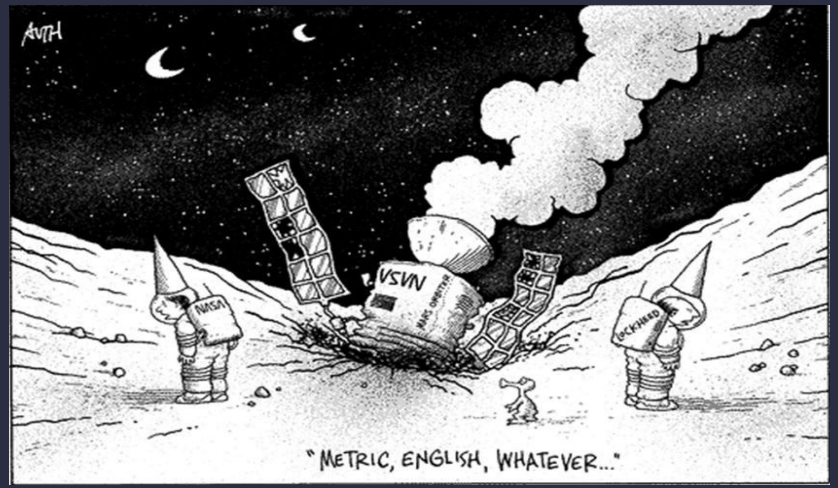
Facilitar las tomas de decisiones que involucren pasos a producción

Evitar la aparición de nuevos defectos


SO YOU ARE TELLING ME...
THAT I HAVE TO WRITE
CODE TO CHECK IF THE CODE
I WROTE EARLIER WAS RIGHT

¿Por qué testear software es importante?

Ariane 5 Flight 501



NASA's Mars Climate Orbiter

Tipos de pruebas

Según ejecución:

- Estáticas
- Dinámicas

Tipos de pruebas

Según ejecución:

- Estáticas
- Dinámicas

Según herramientas:

- Manuales
- Automáticas

Tipos de pruebas

Según ejecución:

- Estáticas
- Dinámicas

Según herramientas:

- Manuales
- Automáticas

Según lo que verifica:

- Funcionales
  - Unitarias
  - De integración
  - De sistemas
  - Otras
- No funcionales
  - Seguridad
  - Usabilidad
  - Rendimiento
  - Carga

Sistema

Integración

Unitarios

¿Qué vamos a usar?

Jest

Setup de entorno de testing

Jest es un framework de JavaScript diseñado para garantizar la corrección de cualquier base de código JavaScript.

https://jestjs.io/

Setup de entorno de testing

El template del curso ya viene con Jest instalado.

Además, también utilizaremos la librería supertest, la cual ofrece una abstracción para testear endpoints http, lo que nos permitirá testear nuestros endpoints.

Setup de entorno de testing

Pero antes...

```
"scripts": {
  "start": "node index.js",
  "dev": "nodemon index.js",
  "lint": "eslint ./src index.js",
  "lint-fix": "eslint --fix ./src index.js",
  "build-assets": "yarn run clean-assets && NODE_ENV=production webpack -p",
  "clean-assets": "rm -rf build/assets",
  "heroku-postbuild": "yarn run build-assets && sequelize db:migrate",
  "test": "jest"
},
```

package.json

```
"scripts": {
  "start": "node index.js",
  "dev": "nodemon index.js",
  "lint": "eslint ./src index.js",
  "lint-fix": "eslint --fix ./src index.js",
  "build-assets": "yarn run clean-assets && NODE_ENV=production webpack -p",
  "clean-assets": "rm -rf build/assets",
  "heroku-postbuild": "yarn run build-assets && sequelize db:migrate",
  "test": "jest"
},
```

package.json

```
"scripts": {
  "start": "node index.js",
  "dev": "nodemon index.js",
  "lint": "eslint ./src index.js",
  "lint-fix": "eslint --fix ./src index.js",
  "build-assets": "yarn run clean-assets && NODE_ENV=production webpack -p",
  "clean-assets": "rm -rf build/assets",
  "heroku-postbuild": "yarn run build-assets && sequelize db:migrate",
  "test": "jest --runInBand --verbose"
},
```

package.json

Estructura de un test

```
describe('Example suite', () => {
 /* Do necessary setup before tests */
 test('Example test description', () => {
    /* Do a request or call function from another
    part of your program expect your request or
    function return to be equal to something you know
    */
    expect(something).matcher(expected response);

 });
});
```

Matchers más comunes

```
expect(something).toBe(primitive);
expect(something).toEqual(object);
expect(something).toBeNull();
expect(something).toBeFalsey();
expect(something).toContain(item);
expect(something).toBeDefined();
expect(something).toMatchSnapshot();
```

Funciones de Setup y Teardown

```
beforeAll(() => {
  /* Setup before running all tests in the scope of the suite */
});
afterAll(() => {
  /* Teardown after running all tests in the scope of the suite */
});
beforeEach(() => {
  /* Setup before every test in the scope of the suite */
});
afterEach(() => {
  /* Teardown after every test in the scope of the suite */
});
```

Estructura de un test

```
describe('Example suite', () => {
 const db = dbConnection();
 beforeAll(async () => {
   await db.connect();
 });

 afterAll(async () => {
   await db.disconnect();
 });

 test('Example test description, async () => {
   const something = await db.someRequest();
   expect(something).matcher(expected response);
 });
});
```

Pasemos a ver código...