Taller: Implementación de un esquema de pruebas con Jest

Julian F. Latorre

12 de septiembre de 2024

Índice

1. Introducción

Jest es un framework de testing de JavaScript desarrollado por Facebook. Es especialmente útil para proyectos fullstack porque puede ser utilizado tanto en el backend (Node.js) como en el frontend (React, Vue, Angular, etc.). En este taller, aprenderemos cómo implementar Jest en ambas partes de un proyecto fullstack.

2. Prerequisitos

Antes de comenzar, asegúrate de tener lo siguiente:

- Node.js (versión 12 o superior) instalado

- Un proyecto fullstack existente (asumiremos un backend en Node.js/Express y un frontend en React)
- Un editor de código (como Visual Studio Code)

3. Configuración del Backend

3.1. Instalación de Jest en el Backend

1. Abre una terminal y navega hasta la carpeta de tu proyecto backend.
2. Ejecuta el siguiente comando para instalar Jest como dependencia de desarrollo:

```
npm install --save-dev jest
```

3. Abre el archivo `package.json` y añade el siguiente script en la sección "scripts":

```
"scripts": {
  "test": "jest"
}
```

3.2. Configuración de Jest en el Backend

1. Crea un archivo llamado `jest.config.js` en la raíz de tu proyecto backend con el siguiente contenido:

```
module.exports = {
  testEnvironment: 'node',
  coverageDirectory: 'coverage',
  collectCoverageFrom: ['src/**/*.js'],
};
```

2. Si estás utilizando ES6 modules, instala `@babel/preset-env`:

```
npm install --save-dev @babel/preset-env
```

3. Crea un archivo `.babelrc` en la raíz del proyecto con el siguiente contenido:

```
{
  "presets": ["@babel/preset-env"]
}
```

3.3. Escribiendo Tests para el Backend

1. Crea una carpeta llamada `__tests__` en la raíz de tu proyecto backend.
2. Dentro de esta carpeta, crea un archivo de test. Por ejemplo, si tienes un archivo `src/utils/math.js`, crea un archivo `__tests__/math.test.js`.
3. Escribe un test simple:

```
const { sum } = require('../src/utils/math');

describe('Math utility', () => {
  test('adds 1 + 2 to equal 3', () => {
    expect(sum(1, 2)).toBe(3);
  });
});
```

4. Ejecuta los tests con el comando:

```
npm test
```

4. Configuración del Frontend

4.1. Instalación de Jest en el Frontend

Si estás utilizando Create React App, Jest ya viene configurado. Si no:

1. Navega hasta la carpeta de tu proyecto frontend.
2. Instala Jest y las dependencias necesarias:

```
npm install --save-dev jest @testing-library/react @testing-
    library/jest-dom
```

3. Añade el script de test en `package.json`:

```
"scripts": {
  "test": "jest"
}
```

4.2. Configuración de Jest en el Frontend

1. Crea un archivo `jest.config.js` en la raíz de tu proyecto frontend:

```
module.exports = {
  testEnvironment: 'jsdom',
  setupFilesAfterEnv: ['<rootDir>/src/setupTests.js'],
  moduleNameMapper: {
    '\\.(css|less|sass|scss)$': '<rootDir>/__mocks__/styleMock
    .js',
```

```
    '\\.(gif|ttf|eot|svg)$': '<rootDir>/__mocks__/fileMock.js'
    ,
  },
};
```

2. Crea un archivo `src/setupTests.js`:

```
import '@testing-library/jest-dom';
```

3. Crea una carpeta `__mocks__` en la raíz y añade los archivos `styleMock.js` y `fileMock.js`:

```
// styleMock.js
module.exports = {};

// fileMock.js
module.exports = 'test-file-stub';
```

4.3. Escribiendo Tests para el Frontend

1. Crea una carpeta `__tests__` en la carpeta `src`.

2. Crea un archivo de test, por ejemplo `Button.test.js`:

```
import React from 'react';
import { render, fireEvent, screen } from '@testing-library/
    react';
import Button from '../components/Button';

test('renders button and responds to click', () => {
  const handleClick = jest.fn();
  render(<Button onClick={handleClick}>Click me</Button>);

  const button = screen.getByText('Click me');
  expect(button).toBeInTheDocument();

  fireEvent.click(button);
  expect(handleClick).toHaveBeenCalledTimes(1);
});
```

3. Ejecuta los tests:

```
npm test
```

5. Integración Continua

1. Crea un archivo `.github/workflows/test.yml` (si usas GitHub Actions):

```yaml
name: Run Tests

on: [push, pull_request]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - name: Use Node.js
      uses: actions/setup-node@v2
      with:
        node-version: '14'
    - name: Install dependencies
      run: npm ci
    - name: Run tests
      run: npm test
```

2. Configura tu repositorio para requerir que los tests pasen antes de permitir merges a la rama principal.

6. Mejores Prácticas

- Escribe tests para cada nueva funcionalidad.
- Mantén los tests pequeños y enfocados.
- Usa mocks para dependencias externas.
- Apunta a una cobertura de código de al menos 80 %.
- Ejecuta los tests antes de cada commit.

Recuerda que los tests son una parte crucial del desarrollo de software y te ayudarán a mantener la calidad de tu código a lo largo del tiempo.

Para más información, consulta la documentación oficial de Jest.