NodeJS: Test unitarios y calidad de código

Gabriel Rodríguez Flores

October 28, 2025

ADA ITS
INSTITUTO TECNOLÓGICO SUPERIOR

  - Linting y formateo de código con ESLint 9 y Prettier
- Realización de test unitarios con Vitest, Jest o Ava
- Análisis de cobertura de código con Istanbul y SonarQube

Contents

1 Teoría

1.1 Introducción

El desarrollo de software de calidad requiere múltiples herramientas y prácticas:

1. **Linting y formateo**: ESLint y Prettier para mantener el código limpio y consistente
2. **Testing**: Frameworks como Vitest, Jest o Ava para ejecutar los test unitarios
3. **Mocking**: Librerías como Sinon para simular comportamientos externos
4. **Cobertura**: Istanbul (nyc) para medir la cobertura de código
5. **Análisis de calidad**: SonarQube para análisis estático y visualización de métricas

> Importante: Todas las librerías exclusivas para desarrollo han de ser instaladas en modo Desarrollo (`-D` o `--save-dev`).

1.1.1 Enlaces de interés

- Comparativa de frameworks de testing 2024
- State of JS 2023 - Testing
- ESLint 9 Migration Guide

1.2 Linting y formateo de código

1.2.1 ESLint 9

ESLint es una herramienta de análisis estático de código para identificar patrones problemáticos en código JavaScript/TypeScript. La versión 9 introduce cambios significativos en la configuración.

1.2.1.1 Instalación

```
npm i -D eslint
```

1.2.1.2 Configuración ESLint 9 (Flat Config) ESLint 9 usa un nuevo sistema de configuración plana (`eslint.config.js` o `eslint.config.mjs`) que reemplaza el antiguo `.eslintrc.*`.

Inicialización:

```
npm init @eslint/config@latest
```

Estructura del `eslint.config.js` (ESM):

```
import js from '@eslint/js';

export default [
  js.configs.recommended,
  {
    files: ['**/*.js'],
    languageOptions: {
      ecmaVersion: 'latest',
      sourceType: 'module',
      globals: {
        console: 'readonly',
        process: 'readonly'
      }
    },
    rules: {
      'no-unused-vars': 'warn',
      'no-console': 'off',
      'semi': ['error', 'always'],
      'quotes': ['error', 'single']
    }
  },
  {
    ignores: ['node_modules/**', 'dist/**', 'coverage/**']
  }
];
```

Configuración CommonJS (`eslint.config.cjs`):

```
const js = require('@eslint/js');

module.exports = [
  js.configs.recommended,
  {
    files: ['**/*.js'],
    languageOptions: {
      ecmaVersion: 2022,
      sourceType: 'commonjs',
      globals: {
        require: 'readonly',
        module: 'readonly',
        exports: 'writable',
        process: 'readonly'
      }
    },
    rules: {
      'no-unused-vars': 'warn',
      'semi': ['error', 'always']
    }
  }
];
```

1.2.1.3 Plugins populares para ESLint 9

```
# Node.js
npm i -D eslint-plugin-n

# Import/Export
npm i -D eslint-plugin-import

# Promesas
npm i -D eslint-plugin-promise

# Security
npm i -D eslint-plugin-security
```

Ejemplo con plugins:

```
import js from '@eslint/js';
import nodePlugin from 'eslint-plugin-n';
import promisePlugin from 'eslint-plugin-promise';

export default [
  js.configs.recommended,
  nodePlugin.configs['flat/recommended'],
  promisePlugin.configs['flat/recommended'],
  {
    files: ['**/*.js'],
    rules: {
      'n/no-missing-import': 'off'
    }
  }
];
```

1.2.1.4 Scripts en package.json

```
{
  "scripts": {
    "lint": "eslint .",
    "lint:fix": "eslint . --fix"
  }
}
```

1.2.1.5 Migración desde ESLint 8 Antes (`.eslintrc.json`):

```
{
  "extends": ["eslint:recommended"],
  "env": {
    "node": true,
    "es2022": true
  },
  "parserOptions": {
    "ecmaVersion": "latest"
```

```
  },
  "rules": {
    "semi": ["error", "always"]
  }
}
```

Después (`eslint.config.js`):

```
import js from '@eslint/js';
import globals from 'globals';

export default [
  js.configs.recommended,
  {
    languageOptions: {
      ecmaVersion: 'latest',
      sourceType: 'module',
      globals: {
        ...globals.node
      }
    },
    rules: {
      'semi': ['error', 'always']
    }
  }
];
```

1.2.2 Prettier

Alternativa a ESLint para formateo de código. También se puede integrar con ESLint.

1.2.2.1 Instalación

```
npm i -D prettier
npm i -D eslint-config-prettier  # Para evitar conflictos con ESLint
```

1.2.2.2 Configuración `.prettierrc.json`

```
{
  "semi": true,
  "trailingComma": "es5",
  "singleQuote": true,
  "printWidth": 100,
  "tabWidth": 2,
  "arrowParens": "avoid"
}
```

1.2.2.3 Integración con ESLint 9

```
import js from '@eslint/js';
```

```
import prettierConfig from 'eslint-config-prettier';

export default [
  js.configs.recommended,
  prettierConfig,  // Debe ir al final para sobrescribir reglas
  {
    files: ['**/*.js'],
    rules: {
      // Tus reglas personalizadas
    }
  }
];
```

1.2.2.4 Scripts en package.json

```
{
  "scripts": {
    "format": "prettier --write .",
    "format:check": "prettier --check ."
  }
}
```

1.3 Frameworks de testing

1.3.1 Comparativa de frameworks

Framework	Velocidad	Sintaxis	Ecosistema	Cobertura	Ideal para
Vitest	Alta	Similar Jest	Moderno	Integrada	Proyectos modernos, Vite
Jest	Media	Muy popular	Extenso	Integrada	React, proyectos grandes
Ava	Alta	Minimalista	Limitado	Externa (nyc)	APIs, proyectos pequeños
Node:test	Alta	Nativa	N/A	Externa	Sin dependencias

1.3.2 Vitest (Recomendado para proyectos modernos)

Framework de testing ultrarrápido compatible con Jest, diseñado para Vite pero funciona en cualquier proyecto.

1.3.2.1 Instalación

```
npm i -D vitest
```

1.3.2.2 Configuración `vitest.config.js`

```
import { defineConfig } from 'vitest/config';

export default defineConfig({
  test: {
    globals: true,
    environment: 'node',
    coverage: {
      provider: 'v8',
      reporter: ['text', 'html', 'lcov'],
      exclude: ['node_modules/', 'test/']
    }
  }
});
```

1.3.2.3 Ejemplo de test

```
import { describe, it, expect } from 'vitest';
import { fizzBuzz } from '../src/fizzbuzz.js';

describe('FizzBuzz', () => {
  it('should return 1 for input 1', () => {
    expect(fizzBuzz(1)).toBe(1);
  });

  it('should return fizz for multiples of 3', () => {
    expect(fizzBuzz(3)).toBe('fizz');
  });

  it('should return buzz for multiples of 5', () => {
    expect(fizzBuzz(5)).toBe('buzz');
  });

  it('should return fizzbuzz for multiples of 15', () => {
    expect(fizzBuzz(15)).toBe('fizzbuzz');
  });
});
```

1.3.2.4 Scripts en package.json

```
{
  "scripts": {
    "test": "vitest run",
    "test:watch": "vitest",
    "test:ui": "vitest --ui",
    "test:coverage": "vitest run --coverage"
  }
}
```

1.3.3 Jest

Framework de testing más popular, especialmente en proyectos React.

1.3.3.1 Instalación

```
npm i -D jest
```

1.3.3.2 Configuración `jest.config.js`

```
module.exports = {
  testEnvironment: 'node',
  coverageDirectory: 'coverage',
  collectCoverageFrom: [
    'src/**/*.js',
    '!src/**/index.js'
  ],
  coverageThreshold: {
    global: {
      branches: 80,
      functions: 80,
      lines: 80,
      statements: 80
    }
  },
  testMatch: [
    '**/test/**/*.test.js'
  ]
};
```

1.3.3.3 Ejemplo de test

```
const fizzBuzz = require('../src/fizzbuzz');

describe('FizzBuzz', () => {
  test('should return 1 for input 1', () => {
    expect(fizzBuzz(1)).toBe(1);
  });

  test('should return fizz for multiples of 3', () => {
    expect(fizzBuzz(3)).toBe('fizz');
  });

  test('should return buzz for multiples of 5', () => {
    expect(fizzBuzz(5)).toBe('buzz');
  });

  test('should return fizzbuzz for multiples of 15', () => {
    expect(fizzBuzz(15)).toBe('fizzbuzz');
  });
});
```

1.3.3.4 Scripts en package.json

```
{
  "scripts": {
    "test": "jest",
    "test:watch": "jest --watch",
    "test:coverage": "jest --coverage"
  }
}
```

1.3.4 AVA

Framework minimalista y rápido con ejecución concurrente de tests.

1.3.4.1 Instalación

```
npm i -D ava
```

1.3.4.2 Configuración en package.json

```
{
  "ava": {
    "files": [
      "test/**/*.test.js"
    ],
    "timeout": "30s",
    "verbose": true
  }
}
```

1.3.4.3 Ejemplo de test

```
const test = require('ava');
const fizzBuzz = require('../src/fizzbuzz');

test('should return 1 for input 1', t => {
  const result = fizzBuzz(1);
  t.is(result, 1);
});

test('should return fizz for multiples of 3', t => {
  const result = fizzBuzz(3);
  t.is(result, 'fizz');
});

test('should return buzz for multiples of 5', t => {
  const result = fizzBuzz(5);
  t.is(result, 'buzz');
});

test('should return fizzbuzz for multiples of 15', t => {
```

```
  const result = fizzBuzz(15);
  t.is(result, 'fizzbuzz');
});
```

1.3.4.4 Scripts en package.json

```
{
  "scripts": {
    "test": "ava",
    "test:watch": "ava --watch"
  }
}
```

1.3.5 Node:test (Nativo desde Node 18)

Sistema de testing nativo de Node.js, sin dependencias externas.

1.3.5.1 Sin instalación (requiere Node.js 18+)

```
import { describe, it } from 'node:test';
import assert from 'node:assert/strict';
import { fizzBuzz } from '../src/fizzbuzz.js';

describe('FizzBuzz', () => {
  it('should return 1 for input 1', () => {
    assert.equal(fizzBuzz(1), 1);
  });

  it('should return fizz for multiples of 3', () => {
    assert.equal(fizzBuzz(3), 'fizz');
  });

  it('should return buzz for multiples of 5', () => {
    assert.equal(fizzBuzz(5), 'buzz');
  });

  it('should return fizzbuzz for multiples of 15', () => {
    assert.equal(fizzBuzz(15), 'fizzbuzz');
  });
});
```

1.3.5.2 Scripts en package.json

```
{
  "scripts": {
    "test": "node --test",
    "test:watch": "node --test --watch"
  }
}
```

1.4 Mocking: Simular comportamientos externos

Cuando una función no es pura (no siempre tiene el mismo resultado cuando se invoca con los mismos parámetros), tenemos que probar las distintas variantes de respuestas que esta función nos puede dar. También cuando no queremos que se realice una petición real a servicios externos (APIs, bases de datos) durante los tests.

1.4.1 Comparativa de librerías de mocking

Librería	Framework	Características	Uso recomendado
Vitest Mock	Vitest	Integrado, compatible Jest	Proyectos con Vitest
Jest Mock	Jest	Integrado, muy completo	Proyectos con Jest
Sinon	Agnóstico	Independiente, flexible	Cualquier framework
testdouble	Agnóstico	Minimalista, opinado	Testing puro

1.4.2 Sinon

Librería independiente de mocking, spies y stubs.

1.4.2.1 Instalación

```
npm i -D sinon
```

1.4.2.2 Stubs: Controlar respuestas de funciones

```
const sinon = require('sinon');

// Stub básico con respuestas condicionales
const callback = sinon.stub();
callback.withArgs(42).returns(1);
callback.withArgs(1).throws(new Error('Invalid'));
callback.returns(23);

// Ejemplo en test
const test = require('ava');
const sinon = require('sinon');

test('should stub console.log', t => {
  const logStub = sinon.stub(console, 'log');

  console.log('test message');
```

```
  t.true(logStub.calledOnce);
  t.true(logStub.calledWithExactly('test message'));

  logStub.restore(); // Importante: restaurar
});
```

1.4.2.3 Spies: Observar llamadas sin modificar comportamiento

```
const sinon = require('sinon');

const spy = sinon.spy(myObject, 'myMethod');
myObject.myMethod('param');

console.log(spy.called); // true
console.log(spy.callCount); // 1
console.log(spy.calledWith('param')); // true

spy.restore();
```

1.4.2.4 Fake timers: Controlar el tiempo

```
const sinon = require('sinon');

const clock = sinon.useFakeTimers();

setTimeout(() => {
  console.log('Ejecutado');
}, 1000);

clock.tick(1000); // Avanza el tiempo 1 segundo
clock.restore();
```

1.4.3 Mocking nativo en frameworks

1.4.3.1 Vitest

```
import { describe, it, expect, vi } from 'vitest';

describe('Mocking con Vitest', () => {
  it('should mock a function', () => {
    const mockFn = vi.fn(() => 'mocked value');

    expect(mockFn()).toBe('mocked value');
    expect(mockFn).toHaveBeenCalled();
  });

  it('should spy on method', () => {
    const obj = { method: () => 'real' };
    const spy = vi.spyOn(obj, 'method');

    obj.method();
```

```
    expect(spy).toHaveBeenCalled();
  });
});
```

1.4.3.2 Jest

```
describe('Mocking con Jest', () => {
  test('should mock a function', () => {
    const mockFn = jest.fn(() => 'mocked value');

    expect(mockFn()).toBe('mocked value');
    expect(mockFn).toHaveBeenCalled();
  });

  test('should mock module', () => {
    jest.mock('../src/api');
    const api = require('../src/api');

    api.fetchData.mockResolvedValue({ data: 'test' });
  });
});
```

1.5 Cobertura de código

1.5.1 Istanbul (nyc)

Herramienta para medir la cobertura de código en frameworks sin cobertura integrada (como AVA).

1.5.1.1 Instalación

```
npm i -D nyc
```

1.5.1.2 Configuración en `package.json`

```
{
  "nyc": {
    "reporter": ["html", "text", "lcov"],
    "exclude": [
      "test/**",
      "node_modules/**",
      "coverage/**"
    ],
    "all": true,
    "check-coverage": true,
    "lines": 80,
    "functions": 80,
    "branches": 80,
    "statements": 80
  }
}
```

1.5.1.3 Uso con AVA

```
nyc ava
nyc --reporter=html ava
nyc --reporter=lcov --reporter=text ava
```

1.5.1.4 Scripts en package.json

```
{
  "scripts": {
    "test": "nyc ava",
    "test:coverage": "nyc --reporter=html --reporter=text ava"
  }
}
```

Directorios generados (añadir a `.gitignore`):

```
.nyc_output/
coverage/
```

> **Estándar de la industria**: Se suele requerir una cobertura de código mínima del 80%.

1.5.2 Cobertura integrada (Vitest/Jest)

Vitest y Jest incluyen cobertura de código nativa, no necesitan nyc.

Vitest con v8 (recomendado):

```
npm i -D @vitest/coverage-v8
```

Jest (incluida por defecto):

```
npm test -- --coverage
```

1.6 Análisis de calidad de código

1.6.1 SonarQube

Plataforma para análisis continuo de calidad de código, detección de bugs, vulnerabilidades y code smells.

1.6.1.1 Alternativas

- **SonarCloud**: Versión cloud de SonarQube (gratuita para proyectos open source)
- **CodeClimate**: Plataforma de análisis de código

- **Snyk**: Enfocado en seguridad

Directorios generados (añadir a `.gitignore`):

```
.scannerwork/
```

1.6.1.2 Docker Compose para SonarQube `docker-compose.sonar.yml`:

```yaml
version: "3.9"
services:
  sonarqube:
    image: sonarqube:10-community
    container_name: sonarqube
    ports:
      - "9000:9000"
    environment:
      - SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true
    volumes:
      - sonarqube_data:/opt/sonarqube/data
      - sonarqube_logs:/opt/sonarqube/logs
      - sonarqube_extensions:/opt/sonarqube/extensions

volumes:
  sonarqube_data:
  sonarqube_logs:
  sonarqube_extensions:
```

Arrancar SonarQube:

```
docker-compose -f docker-compose.sonar.yml up -d
```

Acceder: `http://localhost:9000` (usuario/password por defecto: `admin`/`admin`)

1.6.1.3 Integración con Node.js

```
npm i -D sonarqube-scanner
```

1.6.1.4 Configuración `sonar-project.properties`

```
sonar.projectKey=mi-proyecto
sonar.projectName=Mi Proyecto
sonar.projectVersion=1.0.0

sonar.sources=src
sonar.tests=test
sonar.exclusions=node_modules/**,coverage/**,dist/**

sonar.javascript.lcov.reportPaths=coverage/lcov.info
sonar.testExecutionReportPaths=test-report.xml
```

```
sonar.host.url=http://localhost:9000
sonar.token=tu_token_aqui
```

1.6.1.5 Configuración programática `test/sonar.js`

```
const sonarqubeScanner = require('sonarqube-scanner');

sonarqubeScanner({
  serverUrl: 'http://localhost:9000',
  token: process.env.SONAR_TOKEN || '',
  options: {
    'sonar.projectKey': 'mi-proyecto',
    'sonar.projectName': 'Mi Proyecto',
    'sonar.sources': 'src',
    'sonar.tests': 'test',
    'sonar.inclusions': 'src/**/*.js',
    'sonar.exclusions': 'node_modules/**,coverage/**',
    'sonar.javascript.lcov.reportPaths': 'coverage/lcov.info'
  }
}, () => {
  console.log('Análisis de SonarQube completado');
});
```

1.6.1.6 Procedimiento completo

1. Arrancar el servicio de SonarQube con Docker Compose
2. Configurar token de acceso en SonarQube
3. Ejecutar tests con cobertura: `npm run test:coverage`
4. Ejecutar análisis de SonarQube: `node test/sonar.js`
5. Acceder a SonarQube para visualizar resultados: `http://localhost:9000`

1.6.1.7 Scripts en package.json

```
{
  "scripts": {
    "sonar:start": "docker-compose -f docker-compose.sonar.yml up -d",
    "sonar:stop": "docker-compose -f docker-compose.sonar.yml down",
    "sonar:scan": "node test/sonar.js",
    "sonar:full": "npm run test:coverage && npm run sonar:scan"
  }
}
```

2 Ejemplos

2.1 Proyecto completo con Vitest (Moderno)

2.1.1 Estructura del proyecto

```
fizzbuzz/
|-- src/
|   |-- fizzbuzz.js
|-- test/
|   |-- fizzbuzz.test.js
|-- .gitignore
|-- eslint.config.js
|-- vitest.config.js
|-- package.json
```

2.1.2 `package.json` completo con Vitest

```
{
  "name": "fizzbuzz",
  "version": "1.0.0",
  "type": "module",
  "description": "FizzBuzz con testing completo",
  "main": "src/fizzbuzz.js",
  "scripts": {
    "dev": "node src/index.js",
    "lint": "eslint .",
    "lint:fix": "eslint . --fix",
    "format": "prettier --write .",
    "test": "vitest run",
    "test:watch": "vitest",
    "test:ui": "vitest --ui",
    "test:coverage": "vitest run --coverage",
    "sonar:start": "docker-compose -f docker-compose.sonar.yml up -d",
    "sonar:stop": "docker-compose -f docker-compose.sonar.yml down",
    "sonar:scan": "node test/sonar.js",
    "quality": "npm run lint && npm run test:coverage",
    "clean": "rm -rf coverage .nyc_output .scannerwork node_modules"
  },
  "keywords": ["testing", "fizzbuzz", "vitest"],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "@eslint/js": "^9.13.0",
    "@vitest/coverage-v8": "^2.1.0",
    "@vitest/ui": "^2.1.0",
    "eslint": "^9.13.0",
```

```
    "eslint-config-prettier": "^9.1.0",
    "prettier": "^3.3.0",
    "sonarqube-scanner": "^4.2.0",
    "vitest": "^2.1.0"
  }
}
```

2.1.3 .gitignore

```
node_modules/
coverage/
.nyc_output/
.scannerwork/
dist/
*.log
.env
```

2.2 Proyecto completo con AVA (Clásico)

2.2.1 package.json completo con AVA

```
{
  "name": "fizzbuzz",
  "version": "1.0.0",
  "description": "FizzBuzz con AVA y SonarQube",
  "main": "src/index.js",
  "scripts": {
    "dev": "node src/index.js",
    "lint": "eslint .",
    "lint:fix": "eslint . --fix",
    "test": "nyc ava",
    "test:watch": "ava --watch --verbose",
    "test:coverage": "nyc --reporter=html --reporter=lcov ava",
    "sonar:start": "docker-compose -f docker-compose.sonar.yml up -d",
    "sonar:stop": "docker-compose -f docker-compose.sonar.yml down",
    "sonar:scan": "node test/sonar.js",
    "sonar:full": "npm run test:coverage && npm run sonar:scan",
    "clean": "rm -rf .nyc_output .scannerwork coverage node_modules"
  },
  "keywords": ["testing", "fizzbuzz", "ava"],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "@eslint/js": "^9.13.0",
    "ava": "^6.1.0",
    "eslint": "^9.13.0",
```

```
    "nyc": "^17.1.0",
    "prettier": "^3.3.0",
    "sonarqube-scanner": "^4.2.0"
  },
  "ava": {
    "files": [
      "test/**/*.test.js"
    ],
    "timeout": "30s",
    "verbose": true
  },
  "nyc": {
    "reporter": ["html", "text", "lcov"],
    "exclude": [
      "test/**",
      "node_modules/**"
    ],
    "all": true,
    "check-coverage": true,
    "lines": 80,
    "functions": 80,
    "branches": 80,
    "statements": 80
  }
}
```

2.3 Ejemplos de código

2.3.1 FizzBuzz básico

src/fizzbuzz.js:

```
/**
 * Implementación del juego FizzBuzz
 * @param {number} n - Número a evaluar
 * @returns {string|number} - 'fizz', 'buzz', 'fizzbuzz' o el número
 */
function fizzBuzz(n) {
  let result = '';
  if (n % 3 === 0) result += 'fizz';
  if (n % 5 === 0) result += 'buzz';
  return result || n;
}

module.exports = fizzBuzz;
```

2.3.2 Tests con AVA

test/fizzbuzz.test.js:

```
const test = require('ava');
const fizzBuzz = require('../src/fizzbuzz');

test('should return 1 for input 1', t => {
  const result = fizzBuzz(1);
  t.is(result, 1);
});

test('should return 2 for input 2', t => {
  const result = fizzBuzz(2);
  t.is(result, 2);
});

test('should return fizz for multiples of 3', t => {
  t.is(fizzBuzz(3), 'fizz');
  t.is(fizzBuzz(6), 'fizz');
  t.is(fizzBuzz(9), 'fizz');
});

test('should return buzz for multiples of 5', t => {
  t.is(fizzBuzz(5), 'buzz');
  t.is(fizzBuzz(10), 'buzz');
  t.is(fizzBuzz(20), 'buzz');
});

test('should return fizzbuzz for multiples of 15', t => {
  t.is(fizzBuzz(15), 'fizzbuzz');
  t.is(fizzBuzz(30), 'fizzbuzz');
  t.is(fizzBuzz(45), 'fizzbuzz');
});
```

2.3.3 Tests con Vitest

test/fizzbuzz.test.js:

```
import { describe, it, expect } from 'vitest';
import { fizzBuzz } from '../src/fizzbuzz.js';

describe('FizzBuzz', () => {
  describe('números regulares', () => {
    it('should return 1 for input 1', () => {
      expect(fizzBuzz(1)).toBe(1);
    });

    it('should return 2 for input 2', () => {
```

```
      expect(fizzBuzz(2)).toBe(2);
    });
  });

  describe('múltiplos de 3', () => {
    it('should return fizz for multiples of 3', () => {
      expect(fizzBuzz(3)).toBe('fizz');
      expect(fizzBuzz(6)).toBe('fizz');
      expect(fizzBuzz(9)).toBe('fizz');
    });
  });

  describe('múltiplos de 5', () => {
    it('should return buzz for multiples of 5', () => {
      expect(fizzBuzz(5)).toBe('buzz');
      expect(fizzBuzz(10)).toBe('buzz');
      expect(fizzBuzz(20)).toBe('buzz');
    });
  });

  describe('múltiplos de 15', () => {
    it('should return fizzbuzz for multiples of 15', () => {
      expect(fizzBuzz(15)).toBe('fizzbuzz');
      expect(fizzBuzz(30)).toBe('fizzbuzz');
      expect(fizzBuzz(45)).toBe('fizzbuzz');
    });
  });
});
```

2.3.4 Tests con Jest

test/fizzbuzz.test.js:

```
const fizzBuzz = require('../src/fizzbuzz');

describe('FizzBuzz', () => {
  describe('números regulares', () => {
    test('should return 1 for input 1', () => {
      expect(fizzBuzz(1)).toBe(1);
    });

    test('should return 2 for input 2', () => {
      expect(fizzBuzz(2)).toBe(2);
    });
  });

  describe('múltiplos de 3', () => {
    test('should return fizz for multiples of 3', () => {
      expect(fizzBuzz(3)).toBe('fizz');
```

```
      expect(fizzBuzz(6)).toBe('fizz');
      expect(fizzBuzz(9)).toBe('fizz');
    });
  });

  describe('múltiplos de 5', () => {
    test('should return buzz for multiples of 5', () => {
      expect(fizzBuzz(5)).toBe('buzz');
      expect(fizzBuzz(10)).toBe('buzz');
    });
  });

  describe('múltiplos de 15', () => {
    test('should return fizzbuzz for multiples of 15', () => {
      expect(fizzBuzz(15)).toBe('fizzbuzz');
      expect(fizzBuzz(30)).toBe('fizzbuzz');
    });
  });
});
```

2.4 Ejemplo con mocking

`src/api.js`:

```
async function fetchUser(id) {
  const response = await fetch(`https://api.example.com/users/${id}`);
  return response.json();
}

module.exports = { fetchUser };
```

`test/api.test.js` (con Vitest):

```
import { describe, it, expect, vi, beforeEach } from 'vitest';
import { fetchUser } from '../src/api.js';

describe('API Tests', () => {
  beforeEach(() => {
    global.fetch = vi.fn();
  });

  it('should fetch user data', async () => {
    const mockUser = { id: 1, name: 'John Doe' };

    global.fetch.mockResolvedValue({
      json: async () => mockUser
    });

    const result = await fetchUser(1);
```

```
    expect(result).toEqual(mockUser);
    expect(global.fetch).toHaveBeenCalledWith('https://api.example.com/
       users/1');
  });
});
```

3 Ejercicios

3.1 Ejercicios básicos

1. **Configurar ESLint 9**: Crear un proyecto Node.js y configurar ESLint 9 con la nueva configuración flat config. Añadir reglas personalizadas y plugins recomendados.
2. **FizzBuzz con tests**: Implementar el ejercicio *fizzbuzz* con tests completos usando el framework que prefieras (Vitest/Jest/AVA).
   - Programa funcionando correctamente
   - Tests con cobertura de código 100%
   - Configuración de ESLint y Prettier
3. **Comparación de fechas**: Crear una función `dateCompare` que:
   - Al recibir dos fechas, devuelva cuál es anterior y cuál es posterior: `{ startDate: 'ISODateString', endDate: 'ISODateString'}`
   - Si solo recibe una fecha, la compare con el momento actual
   - Tests completos con diferentes casos de uso
   - Usar librería `luxon` o nativa `Date`
4. **FizzBuzz flexible**: Modificar *fizzbuzz* para recibir condiciones dinámicas:

```
const n = 100;
const conditions = {
  2: 'poo',
  3: 'fizz',
  5: 'buzz',
  7: 'bar'
};

// Debe devolver combinaciones: 'poofizz' para 6, 'fizzbuzz' para 15, etc.
```

3.2 Ejercicios avanzados

5. **Testing con mocks**: Crear una función que consulte una API externa (usa `axios` o `fetch`) y escribir tests usando mocks para simular las respuestas.
6. **Calculadora con TDD**: Desarrollar una calculadora usando TDD (Test-Driven Development):
   - Escribir los tests primero
   - Implementar las funciones después
   - Operaciones: suma, resta, multiplicación, división
   - Manejo de errores (división por cero, parámetros inválidos)
7. **Testing de funciones asíncronas**: Crear funciones asíncronas y testearlas:
   - Promesas
   - Async/await
   - Manejo de errores
   - Timeouts

3.3 Ejercicios de integración

8. **Proyecto completo con calidad de código**:
   - ESLint 9 configurado con plugins
   - Prettier integrado
   - Tests con Vitest o Jest
   - Cobertura mínima del 80%
   - SonarQube configurado con Docker
   - GitHub Actions o GitLab CI para ejecutar tests automáticamente
9. **Comparar frameworks**: Implementar los mismos tests en:
   - Vitest
   - Jest
   - AVA
   - Documentar diferencias, ventajas y desventajas de cada uno

4 Entregables

4.1 En clase

Ejercicios 1, 2 y 3

Entrega del proyecto *fizzbuzz* con:

- Programa funcionando correctamente
- ESLint 9 configurado con flat config
- Prettier configurado e integrado con ESLint
- Tests realizados (elegir Vitest, Jest o AVA)
- Cobertura de código del 100%
- Scripts en `package.json` para:
  - Ejecutar linting: `npm run lint`
  - Ejecutar tests: `npm run test`
  - Ejecutar tests en modo watch: `npm run test:watch`
  - Generar reporte de cobertura: `npm run test:coverage`
- Fichero `.gitignore` adecuado
- README.md con instrucciones de instalación y uso

4.2 Tarea

Ejercicios 4 y 5

- **Ejercicio 4**: FizzBuzz flexible con condiciones dinámicas. Tests completos con diferentes configuraciones.
- **Ejercicio 5**: Testing con mocks. Implementar al menos 3 funciones que dependan de servicios externos y testearlas con mocks.

4.3 Trabajo (opcional)

Ejercicios 6, 7 y 8

Proyecto completo con:

- Docker Compose para SonarQube
- Configuración de SonarQube
- Scripts del `package.json` para:
  - Arrancar/parar SonarQube: `npm run sonar:start` / `npm run sonar:stop`
  - Ejecutar análisis completo: `npm run sonar:full`
  - Limpiar artefactos: `npm run clean`
- Documentación completa en README.md
- Integración continua (GitHub Actions o GitLab CI) - opcional

4.4 Recursos adicionales

4.4.1 Documentación oficial

- ESLint 9 Documentation
- Vitest Documentation
- Jest Documentation
- AVA Documentation
- Prettier Documentation
- SonarQube Documentation

4.4.2 Tutoriales y guías

- ESLint Flat Config Migration
- Testing JavaScript Applications
- Test-Driven Development with Node.js