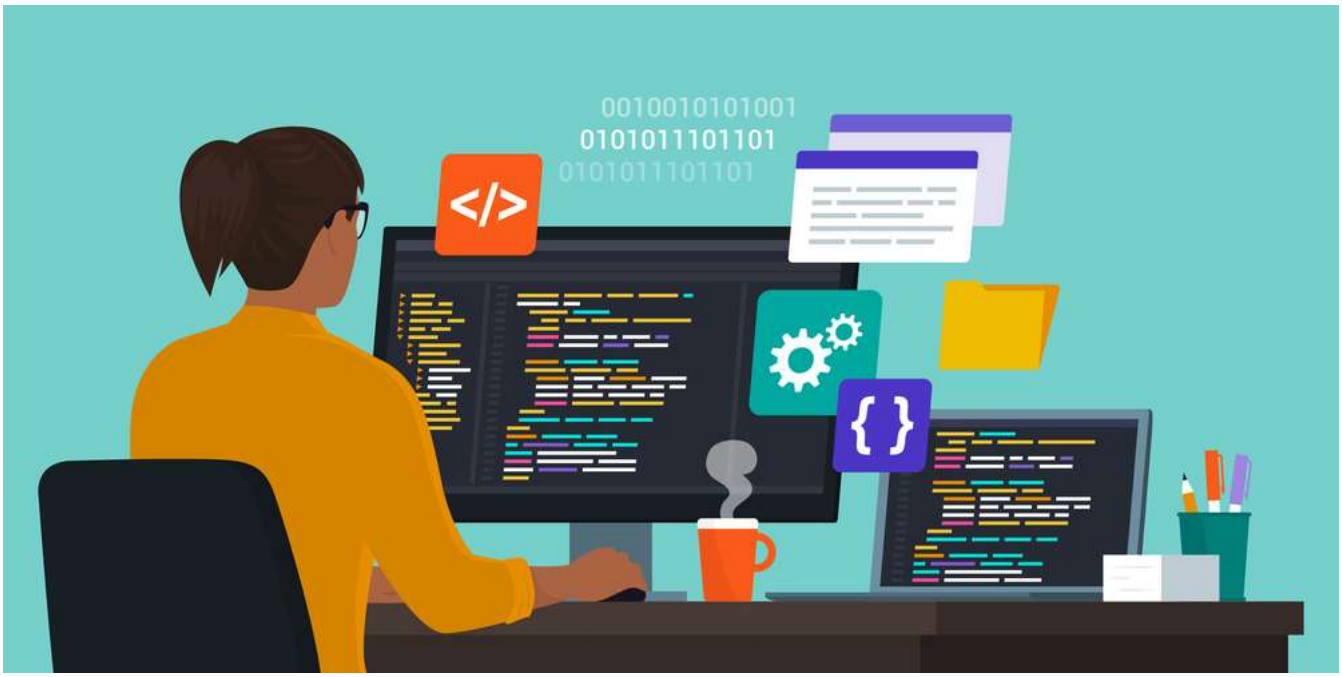


PRACTICA 3. PRUEBAS UNITARIAS EN VISUAL STUDIO CODE

Hecho por: Izan Navarro

ies serra perenxisa

INDICE

1.Introduccion:

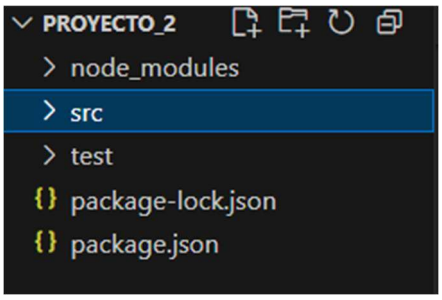
Como hemos visto en la parte teórica del módulo 1, existen diferentes aplicaciones software que permiten la automatización de pruebas. Dichas aplicaciones pueden variar dependiendo de los lenguajes de programación soportados, aunque casi todas presentan unas características similares.

En esta práctica nos centraremos en el estudio de herramientas de automatización de pruebas para aplicaciones desarrolladas en JavaScript.

Para la realización de esta práctica utilizaremos el entorno Visual Studio Code, dadas las enormes posibilidades que ofrece a nivel de pruebas para programadores y testers. También deberemos instalar el framework de pruebas de JavaScript, Mocha y la librería de aserciones, Chai.

2.Configuración e instalación librerías y Node.js:

En esta parte voy a omitir muchas capturas, simplemente os adjunto foto de mi estructura dentro de Visual Studio y foto de la versión de las dependencias instaladas.



Así se puede comprobar que se han instalado tanto el entorno como node.js y las librerías!

3.Segunda parte (5 puntos):

1) A continuación adjunto el código para realizar la actividad 1 y la actividad 2 y su respuesta por consola:

```
const Paciente = require("./Paciente");
const { expect } = require("chai");

// Creamos una instancia del paciente (ahora mismo estabas llamando constructor directamente, eso no lo hace bien)
const paciente = Object.create(Paciente);
paciente.constructor();

describe('Pruebas de la clase Paciente', () => {

  it('Debe tener el nombre "Pepe"', () => {
    expect(paciente.obtenerNombre()).to.equal('Pepe');
  });

  it('Debe tener los apellidos "Pepito"', () => {
    expect(paciente.obtenerApellidos()).to.equal('Pepito');
  });

  it('Debe tener la fecha de nacimiento "18/11/1999"', () => {
    expect(paciente.obtenerFechaNacimiento()).to.equal('18/11/1999');
  });

  it('El método saludar() debe devolver "Hola soy Antonio."', () => {
    expect(paciente.saludar()).to.equal('Hola soy Antonio');
  });
});      You, 1 second ago • Uncommitted changes
```

```
1) Pruebas de la clase Paciente
     Debe tener la fecha de nacimiento "18/11/1999":

    AssertionError: expected '23/09/2000' to equal '18/11/1999'
    + expected - actual

    -23/09/2000
    +18/11/1999

    at Context.<anonymous> (test\Prueba1.js:19:50)
    at process.processImmediate (node:internal/timers:483:21)
```

```
2) Pruebas de la clase Paciente
     El método saludar() debe devolver "Hola soy Antonio.":

    AssertionError: expected 'Hola soy Pepe' to equal 'Hola soy Antonio'
    + expected - actual

    -Hola soy Pepe
    +Hola soy Antonio

    at Context.<anonymous> (test\Prueba1.js:23:35)
    + expected - actual

    -Hola soy Pepe
    +Hola soy Antonio

    at Context.<anonymous> (test\Prueba1.js:23:35)
    -Hola soy Pepe
    +Hola soy Antonio
```

3) A continuación adjunto el código para realizar la actividad 3, añadiendo así el peso como nuevo atributo de la clase Paciente:

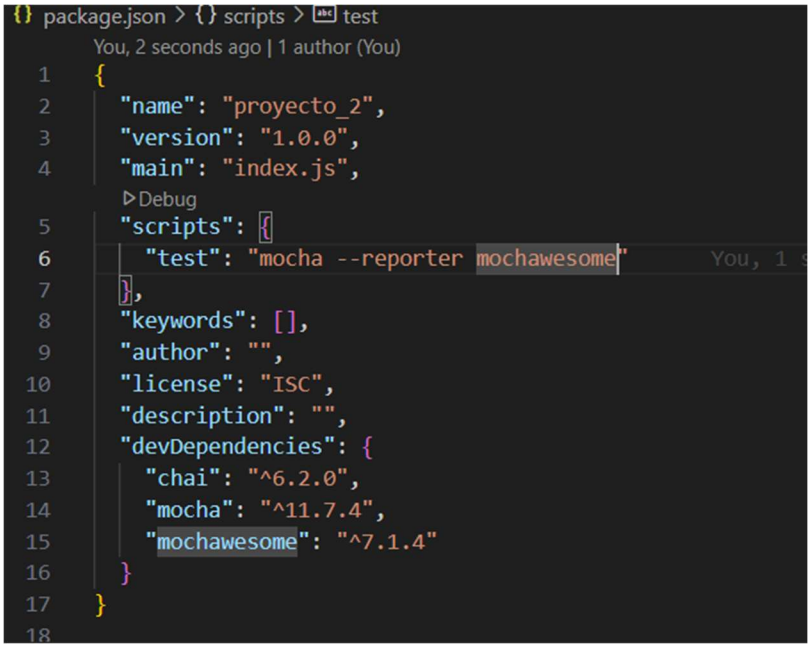
```
const Paciente = {
  constructor () {
    this.nombre = 'Pepe';
    this.apellidos = 'Pepito';
    this.fechaDeNacimiento = '23/09/2000';
    this.altura = '187';
    this.peso = '80';       You, 1 hour ago • subi


    return {
      nombre: this.nombre,
      apellidos: this.apellidos,
      fechaDeNacimiento: this.fechaDeNacimiento,
      altura: this.altura,
      peso: this.peso,
    }
  },
```

4) Código Modificado para que no de ningún error, modificando la fecha y poniendo la fecha que se añade en el objeto paciente por defecto y cambiando el “Hola soy Antonio” por el “Hola soy Pepe”:

```
const Paciente = require("./Paciente");
const { expect } = require("chai");

// Creamos una instancia del paciente (ahora mismo estabas llamando constructor directamente, eso no lo hace bien)
const paciente = Object.create(Paciente);
paciente.constructor();

describe('Pruebas de la clase Paciente', () => {

  it('Debe tener el nombre "Pepe"', () => {
    expect(paciente.obtenerNombre()).to.equal('Pepe');
  });

  it('Debe tener los apellidos "Pepito"', () => {
    expect(paciente.obtenerApellidos()).to.equal('Pepito');
  });

  it('Debe tener la fecha de nacimiento "23/09/2000"', () => {
    expect(paciente.obtenerFechaNacimiento()).to.equal('23/09/2000');
  });

  it('El método saludar() debe devolver "Hola soy Pepe."', () => {
    expect(paciente.saludar()).to.equal('Hola soy Pepe');
  });
});
```

4.Tercera parte (5 puntos):

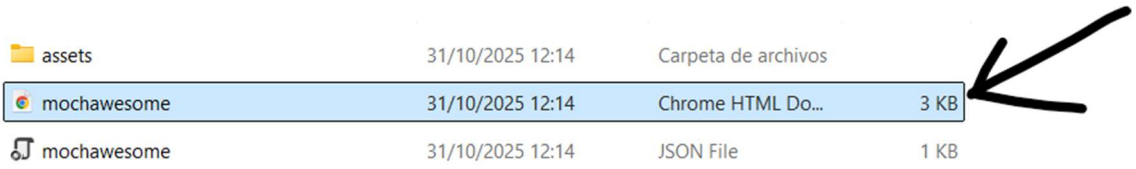
1 y 2) Adjunto imagen de la dependencia instalada y como configurarla para poder usarla



Acto seguido, ejecutamos de nuevo el comando “npm test” y nos saldrá algo así en la línea de comandos:

```
[mochawesome] Report JSON saved to C:\Users\Izan Navarro\Desktop\CIBERSEGURIDAD\CE_Ciberseguridad\PRODUCION_SEGURA\CODIGO\PROYECTO_2\mochawesome-report\mochawesome.json

[mochawesome] Report HTML saved to C:\Users\Izan Navarro\Desktop\CIBERSEGURIDAD\CE_Ciberseguridad\PRODUCION_SEGURA\CODIGO\PROYECTO_2\mochawesome-report\mochawesome.html
```

Se nos habrá generado estos archivos en la carpeta seleccionada y tendremos que entrar en el HTML:



El HTML nos muestra algo así: