DigitalHouse>

Certified Tech Developer
The Ultimate Degree

13 DE DICIEMBRE DE 2021 - [COM XX]

Testing I

Examen integrador

Les pedimos que lean atentamente las siguientes consignas y respondan a las preguntas de acuerdo a lo solicitado.
No se aceptarán links de Drive, solo documentos adjuntos. Caso contrario, el examen no será considerado para su corrección.

Nota aclaratoria: al enviar el formulario con el adjunto se debe esperar la confirmación del profesor **antes de salir de la sala de Zoom** para garantizar que se recibió correctamente para posterior corrección. Caso contrario, no se recibirá la evaluación y el alumno deberá recuperar esta instancia de evaluación. **Solo se recibirá 1 (un) documento por alumno**.

Duración: 1 hora 30 minutos.

Nombre y Apellido: Diego Andres Yepes Celis

Parte teórica

1. Mencionar **2 características** que debe tener **un buen caso de prueba**.

(Sugerencia: utilizar 2 a 5 renglones/líneas)

RT: El caso de prueba debe ser simple.
El caso de uso no se puede asumir la funcionalidad ni las características de la aplicación.

2. ¿Qué **ventajas** presentan las **pruebas automatizadas**?

(Sugerencia: utilizar 2 a 5 renglones/líneas)

RT: El tester se puede ahorrar tiempo de ejecución de las pruebas.
Se pueden realizar pruebas mucho más complejas.
Las pruebas están menos sujetas a errores del tester.
La automatización permite rehusar las scripts de una prueba en otra prueba.

3. Explicar la diferencia entre una **prueba funcional** y una **prueba no funcional**.

(Sugerencia: utilizar 2 a 5 renglones/líneas)

RT: Las pruebas funcionales permiten validar y verificar las funcionalidades principales de un producto.
Las pruebas no funcionales se encargan de comprobar la calidad del producto.

4. Explicar la **técnica de prueba de caja negra**.

(Sugerencia: utilizar 2 a 5 renglones/líneas)

RT: Las pruebas de caja negra se centran hacer pruebas sin requerir código o sin saber la estructura interna del software, solo se basan en el uso de documentación.

5. Explicar el **nivel de prueba de componente**.

(Sugerencia: utilizar 2 a 5 renglones/líneas)

RT: Pruebas unitarias o de componente, este tipo de pruebas son utilizadas por los desarrolladores para la búsqueda y corrección de defectos.

Parte práctica

6. ¿Este **test de Postman** es correcto para validar si el contenido devuelto es igual al esperado? Justificar tu respuesta.

```
pm.test("Nombre es Victor", function (){
    let nombre = pm.response.json().nombre;
    pm.response.to.have.status(200);
});

```

RT: No es correcta, ya que el código mostrado se encarga de validar si la respuesta ha sido recibida de manera correcta, no de validar si el contenido devuelto es igual al esperado.

7. Continuamos trabajando con nuestra app **Comida Ya!** Esta se conecta con un servicio back-end. Si realizo una petición **GET en Postman** a la siguiente URL: https://ctd-api-resto.herokuapp.com/v1/products. ¿Qué resultado arroja y por qué?

RT: Arroja que requiere autorización, por que, no se cuenta con las credenciales necesarias para ingresar al servidor.

8. Detallar **1 caso de prueba** que aplicarías en la página de Comida Ya!, solo explicando su descripción (**no utilizar el template**).

RT: Aplicaría un caso de prueba al módulo de Registro y en la sección de elegir un perfil, validará que los perfiles creados como administrador tenga permisos distinto a el perfil de usuario y que los módulos sean especiales para el administrador, adicionalmente verificará si se puede crear un perfil de cliente ya siendo un administrador y de manera contraria.

9. Mencionar **1 defecto** que encuentres en el sitio de Comida Ya! (**no utilizar el template**).

RT: El sitio tiene un defecto en el módulo de registro, ya que no valida el correo electrónico de manera correcta, deja registrar un correo sin @ y sin la terminación correcta de .es, .com etc.

10. Si estoy trabajando con Jest y quiero validar que el resultado devuelto sea **true**. ¿Qué matcher puedo utilizar? Dar un ejemplo de un posible test.

RT: Se podría utilizar toBeFalsy();

```
test('null', () => {
 const n = null;
expect(n).toBeFalsy();
});
```
