**EXAMEN INTEGRADOR - TESTING I (CTD) - 05 DE NOVIEMBRE DE 2021 [0921TDTE1M1C1LAED0621FT]**

****

Leer atentamente las siguientes consignas, y responder a las preguntas de acuerdo a lo solicitado.

**No se aceptarán links de Drive, sólo documentos adjuntos. Caso contrario, el examen no será considerado para su corrección.**

*Nota aclaratoria:* al enviar el adjunto en el [formulario](https://forms.gle/XuJqXFQu7ss9kePv9), 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. **Sólo se recibirá 1 (un) documento por alumno**.

**Duración:** 1 hora 45 minutos.

**Parte teórica**

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

(2 renglones)

Un buen caso de prueba no asume ni la funcionalidad ni las características de aplicación mientras se prepara el caso siendo fiel a la documentación. Otra característica es que trate de asegurar la mayor cobertura posible, para todos los requisitos especificados.

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

(2 renglones)

La automatización de la prueba tiene la ventaja de disminuir la probabilidad de errores por acción humana, además de abaratar costos al disminuir empleo de recursos humanos y tiempo, permitiendo que el personal se encargue de otras tareas prioritarias.

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

(2 renglones)

La diferencia se centra en que una prueba funcional evalúa las funciones que el sistema debe realizar, mientras que una no funcional evalúa qué tan bien se llevan a cabo esas funciones en el sistema.

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

(2 renglones)

La técnica de prueba de caja negra es aquella donde se evaluan las entradas y salidas del sistema, limitándose al cumplimiento de esa funcionalidad sin reparar en las características del código o el funcionamiento interno del mismo.

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

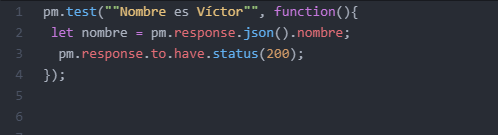
(2 renglones)

El nivel de prueba de componente es aque donde la prueba se centra en los componentes individuales del software. En este nivel se crean pequeños tests para cubrir todos los requisitos del código y verificar los resultados obtenidos usando técnicas de caja blanca para ello.

**Parte práctica**

**Consignas**

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



Este test no es correcto para validar si el contenido devuelto es igual al esperado ya que para esto debemos utilizar el método “pm.expect”, además de corregir la segunda línea y borrar el “.nombre”. Siendo el formato correcto el siguiente:

Pm.test(“Nombre es Victor”, function(){

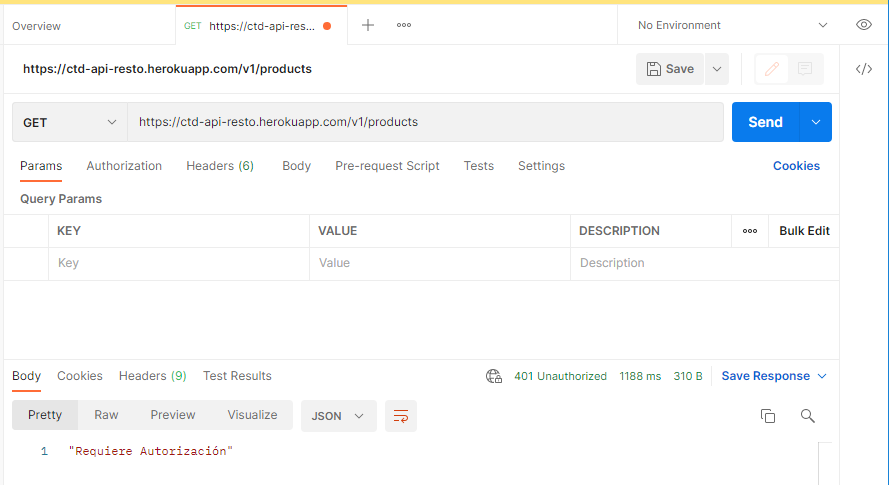
Var jsonData = pm.response.json();

Pm.expect(jsonData.name).to.eql(“Victor”);

});

1. 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?

Arroja el siguiente resultado:



Como podemos ver, la solicitud realizada al servidor lanzó un error “401 Unauthized”, lo que nos indica que el estado de nuestra solicitud falló dado que necesitamos autorización para acceder. Debido a este error es que en nuestro response no podemos observar los resultados esperados, ya que el servidor no puede devolvernos la información solicitada. El tiempo y el tamaño de respuesta fueron de 1188 ms y 310 B, respectivamente.

1. Detallar **1 caso de prueba** que aplicarías en la página de [Comida Ya!](https://ctd-app-resto.herokuapp.com/admin/dashboard/), solo explicando su descripción (**no utilizar el template**).

Un posible caso de prueba a aplicar puede ser el comprobar que la plataforma impide el registro de usuarios con correos electrónicos inexistentes y no permitir el acceso sin la validación previa. Al intentar registrarnos con un correo inexistente debería aparecer un cartel que comunique esa situación o a lo sumo solicitar la validación del mismo (cosa que no podría hacerse) previo al acceso a la plataforma. Podemos decir que este caso de prueba sería del tipo negativo.

1. Mencionar **1 defecto** que encuentres en el sitio de [Comida Ya!](https://ctd-app-resto.herokuapp.com/admin/dashboard/) (**no utilizar el template**).

Haciendo referencia al caso de prueba antes mencionado, aplicándolo podemos observar que no existe aviso de que el correo electrónico utilizado para el registro es inexistente ni tampoco solicita validación previo a permitir el acceso al sistema, lo que constituye una vulnerabilidad de severidad y prioridad alta ya que permitiría el acceso de cualquier persona que proporcionase datos falsos.

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

Para validad que un resultado devuelto sea **true**, podemos utilizar el método “expect”, introduciéndole por parámetro al valor de la función que deseamos evaluar, usando luego el matcher “.toBe()”, introduciéndole como parámetro el valor que esperamos. Un ejemplo sería:

Const {esAfiliado, nombre} = require(‘../util.js’);

Describe(‘Prueba de salidad de datos’, ()=> {

Const estado = esAfiliado (“Jorge”);

Expect(estado).toBe(true)

})