DigitalHouse>

Certified Tech Developer
The Ultimate Degree

21 DE ABRIL DE 2022 [0222TDTE1M1C1LAED1021FT]

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**.
Recordá que cada ejercicio vale 1 punto.

Duración: 1 hora 30 minutos.

Nombre y Apellido: R. Indira Valentina Réquiz Molina

Parte teórica

1. ¿Cuál es la diferencia entre las pruebas de **humo** y las pruebas **regresión**?

   Las suites de humo son pruebas que nos ayudan a verificar un sistema; mientras que las pruebas de regresión buscan asegurar el funcionamiento de un sistema al ingresar cambios o actualizaciones y así verificar que estos cambios no han introducido defectos nuevos.

2. Explique con un ejemplo un tipo de **técnica de prueba de caja blanca**.

   Las pruebas que implementan la técnica de caja blanca son aquellas que se basan en el análisis del código o estructura de un sistema. Ejemplo de ello puede ser una prueba de componente.

3. Mencionar las principales diferencias entre **debugging** y **testing**.

   El testing nos ayuda a identificar las fallas de un sistema, mientras que el debugging nos ayuda a solucionar esas fallas en el código.
   El testing puede ser manual o automatizado pero el debugging debe hacerse manualmente.
   El testing lo puede realizar cualquier persona, ya sea que haya desarrollado el proyecto o sea un usuario, pero el debugging lo realiza personal interno, ligado al desarrollo y con acceso al código.
   Por último, el testing es una etapa del desarrollo, por ello puede realizarse antes de escribir el código; por el contrario, el debugging no lo es y solo puede realizarse una vez ejecutado el código de un sistema.

4. ¿En qué ambiente aplicarías **pruebas unitarias**? Justifique su respuesta.
   Las pruebas unitarias se realizan en el ambiente de desarrollo, Pues es en este momento en el que se realiza el código. Además, si el sistema a desarrollar se basa en TDD, el proceso de desarrollo y ejecución de pruebas de componente van de la mano.

5. ¿Cuál es el objetivo de una prueba de **confirmación**?

   Estas pruebas se ejecutan una vez se ha corregido un defecto, de forma que sirven para testear los casos que resultaron fallidos y confirmar que los defectos encontrados fueron solucionados.

Parte práctica

6. ¿Qué nos permite validar el siguiente **test de Postman**?

```
pm.test("Prueba", function () {
    let nombre = pm.response.json().nombre;
    pm.expect(nombre).to.equal("Virginia");
});
```

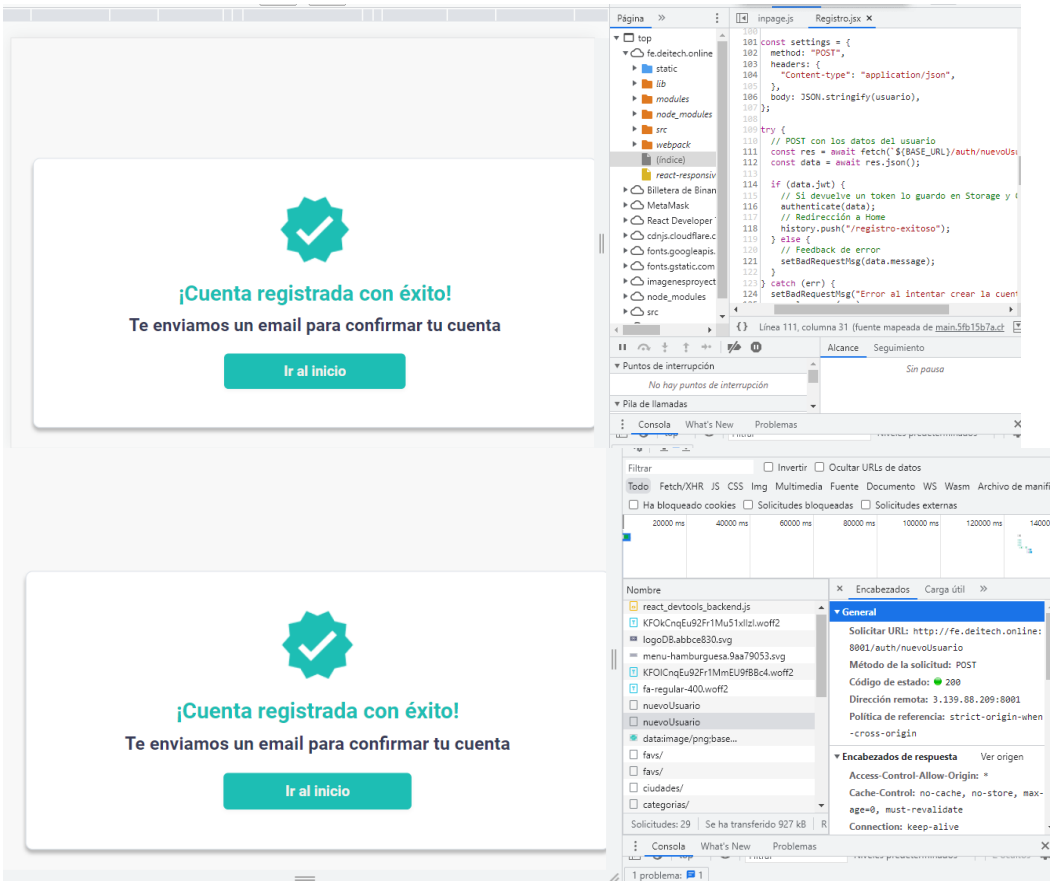
   Este test busca validar que el valor de la variable 'nombre' sea igual a la cadena de texto ' Virginia'.

7. Identificar una petición GET y una POST de nuestra app **Digital Booking!**, (Recomendación: Utilizar la herramienta de desarrollo > DevTools) Explicar brevemente de qué se trata una petición POST.

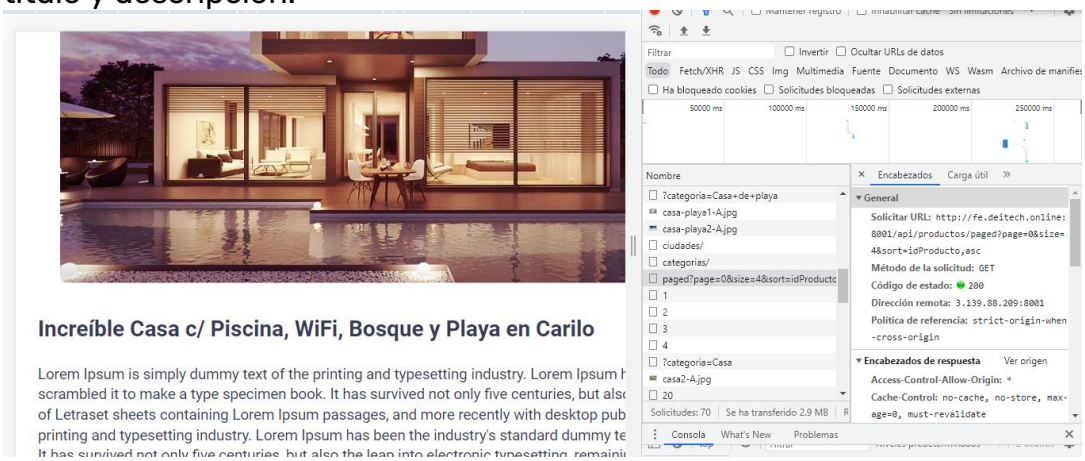
   El método POST nos sirve para enviar un dato al server de forma más segura que un GET.

   Método POST:

   Se crea el usuario



Método GET:
Se accede a una de las propiedades y retornan los elementos como imagen, título y descripción.



8. Redactar brevemente **1 caso de prueba negativo** que aplicarías en la página de **Digital Booking!** (**No se requiere escribirlos en formato de template**).

   Caso de prueba:
   Se verificará que los usuarios no puedan hacer una reserva si no están habilitados.

9. Mencionar **1 defecto** que encuentres en la página de **Digital Booking!** (**No se requiere escribirlos en formato de template**).

   Defecto:
   El sistema no permite que el usuario al ingresar en una propiedad pueda calificarla con las estrellas.

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

    Para validar resultado falsos, debe usarse el matcher .toBeFalsy()