INFORME DE TEST UNITARIOS SPRINT II

El test es parte de la clase CategoriaServiceTest y se llama testFindByIdNonExistingCategoria. Este test tiene como objetivo probar el escenario en el que se busca una categoría por su ID y no se encuentra en el repositorio.

En el test se realiza lo siguiente:

Se establece la configuración inicial utilizando la anotación @BeforeEach.

Se crea una variable categoryId con el valor 2L, que representa el ID de la categoría a buscar.

Se simula el comportamiento del repositorio utilizando el método when de Mockito. Se configura que al llamar al método findById del repositorio con el categoryId, retorne un objeto Optional vacío.

Se utiliza el método assertThrows de la clase Assertions para verificar que se lance una excepción del tipo ResourceNotFoundException al llamar al método findById del servicio de categoría (categoriaService.findById(categoryId)).

Se verifica que el método findById del repositorio haya sido llamado una vez con el argumento categoryId, utilizando el método verify de Mockito.

Este test se encarga de probar el manejo adecuado de la excepción ResourceNotFoundException cuando se intenta buscar una categoría que no existe en el repositorio.

Clase CategoriaServiceTest:

Test: testFindByIdNonExistingCategoria

Descripción: Este test verifica el comportamiento del servicio CategoriaService cuando se busca una categoría por su ID y no se encuentra en el repositorio.

Resultado esperado: Se espera que el servicio arroje una excepción del tipo ResourceNotFoundException al buscar una categoría inexistente.

Mocking: Se utiliza la biblioteca Mockito para simular el comportamiento del repositorio. Se configura que el método findById del repositorio retorne un objeto Optional vacío cuando se le pase el ID de la categoría.

Clase ClienteServiceTest:

Test: findById_NonExistingId_ShouldThrowResourceNotFoundException

Descripción: Este test verifica el comportamiento del servicio ClienteService cuando se busca un cliente por su ID y no se encuentra en el repositorio.

Resultado esperado: Se espera que el servicio arroje una excepción del tipo ResourceNotFoundException al buscar un cliente inexistente.

Mocking: Se utiliza la biblioteca Mockito para simular el comportamiento del repositorio. Se configura que el método findById del repositorio retorne un objeto Optional vacío cuando se le pase el ID del cliente.

Test: save_DuplicateEmail_ShouldThrowBadRequestException

Descripción: Este test verifica el comportamiento del servicio ClienteService cuando se intenta guardar un cliente con un correo electrónico duplicado.

Resultado esperado: Se espera que el servicio arroje una excepción del tipo BadRequestException al intentar guardar un cliente con un correo electrónico duplicado.

Mocking: Se utiliza la biblioteca Mockito para simular el comportamiento del repositorio. Se configura que el método findByEmail del repositorio retorne un objeto Optional que representa la existencia de un usuario con el mismo correo electrónico.

Test: update_NonExistingId_ShouldThrowBadRequestException

Descripción: Este test verifica el comportamiento del servicio ClienteService cuando se intenta actualizar un cliente que no existe en el repositorio.

Resultado esperado: Se espera que el servicio arroje una excepción del tipo BadRequestException al intentar actualizar un cliente inexistente.

Mocking: Se utiliza la biblioteca Mockito para simular el comportamiento del repositorio. Se configura que el método findById del repositorio retorne un objeto Optional vacío cuando se le pase el ID del cliente.

Test: deleteById_NonExistingId_ShouldThrowResourceNotFoundException

Descripción: Este test verifica el comportamiento del servicio ClienteService cuando se intenta eliminar un cliente por su ID y no se encuentra en el repositorio.

Resultado esperado: Se espera que el servicio arroje una excepción del tipo ResourceNotFoundException al intentar eliminar un cliente inexistente.

Mocking: Se utiliza la biblioteca Mockito para simular el comportamiento del repositorio. Se configura que el método findById del repositorio retorne un objeto Optional vacío cuando se le pase el ID del cliente.

Estos tests se realizaron con el objetivo de asegurar el correcto funcion

En el proyecto Hairphoria, se realizó un test en la clase ServicioServiceTest. A continuación, se presenta un resumen del test realizado:

Clase ServicioServiceTest:

Test: save_InvalidEspecialidad_ShouldThrowResourceNotFoundException

Descripción: Este test verifica el comportamiento del servicio ServicioService cuando se intenta guardar un servicio con una especialidad inválida.

Resultado esperado: Se espera que el servicio arroje una excepción del tipo ResourceNotFoundException al intentar guardar un servicio con una especialidad inválida.

Mocking: Se utiliza la biblioteca Mockito para simular el comportamiento de los repositorios. Se configura que el método findByEspecialidad del repositorio de categorías retorne un objeto Optional vacío cuando se le pase la especialidad del servicio.

Este test se realizó con el objetivo de asegurar que el servicio ServicioService maneje correctamente el escenario en el que se intenta guardar un servicio con una especialidad inválida.