Universidad Nacional de Mar del Plata
Ingeniería en Informática
Taller de Programación I

Trabajo Práctico Final

Testing del software

“Subí que te llevo”

Grupo 2:
Carron, Lautaro
Cerdá, Agustin
Gracia, Manuel
Ontiveros, Lucas Nahuel

Determinación datos de prueba y casos de pruebas unitarias de caja negra

Las pruebas unitarias de caja negra fueron clave para comprobar que las funcionalidades del sistema cumplieran con lo que se esperaba, y también nos ayudaron a detectar errores en los resultados.

Capa de datos :
https://docs.google.com/spreadsheets/d/1u6vO0bmDcJUrCIjqylwvdMcA-Pto0fDU0A6JpJt0RUk/edit?usp=drive_link
Aclaraciones:
- El color verde indica que se cumplió con lo esperado.

Capa de negocios :
https://docs.google.com/spreadsheets/d/1skkIO5zljUMgt2RqHP_tyGVxBYITH5Yf8ghd4dnchSk/edit?usp=drive_link

Aclaraciones:
- El color verde indica que se cumplió con lo esperado.
- El color rojo indica que no se ha cumplido con lo esperado, a la derecha del color se explica que arrojó.
- Lo determinamos para el caso donde la instancia de la empresa es null

. En los casos donde no hay presente un escenario redactado fue debido a que no depende el método de otras listas con su estado inicial, por lo tanto consideramos que esas listas están vacías.

Indicaciones extras:
. En la primera fila se indica el método a testear, con su cabecera, precondiciones y excepciones del cual trata.
. Próximo a esto visualizamos el escenario previo a la ejecución de la batería de pruebas.

Determinación de casos de uso y casos de prueba para test de integración

ENLACE:
https://docs.google.com/spreadsheets/d/1thA9IN27oQIQuQm6sTTFvOGw6GC-SVh6lggkgQ5TaDw/edit?usp=sharing

Las pruebas de integración son esenciales para verificar que los diferentes módulos del sistema no solo funcionaran de manera individual, sino también que interactúen

correctamente entre sí. Esto permite asegurarse de que, al combinar los módulos, el sistema como un todo siguiera comportándose de acuerdo a lo esperado, haciendo las aclaraciones correspondientes en los casos donde el sistema no funciona como se esperaba.

Aclaración:

En nuestro caso no pudimos ejecutar estos test debido a que el proyecto no reconoció los mocks.

Test de Persistencia.

ENLACE:
https://docs.google.com/spreadsheets/d/17orTRHUCT1sOxlSa7b9-hNe99VUFO3XFDxwyDIg2fpk/edit?gid=0#gid=0

Aclaraciones:

- El color verde indica que se cumplió con lo esperado.
- El color rojo indica que no se ha cumplido con lo esperado.

El test de persistencia fue clave para asegurarnos de que el sistema manejara correctamente la lectura y escritura de datos. Probamos casos donde el archivo persistido existía y contenía datos válidos, y también situaciones donde no existía o estaba dañado. En todos los casos, verificamos que el sistema reaccionara adecuadamente, creando un nuevo archivo o manejando los errores sin afectar su funcionamiento, haciendo las aclaraciones correspondientes en los casos donde el sistema no funcionaba como se esperaba. Esto garantizó que los datos se guardaran y recuperaran de forma segura, asegurando la confiabilidad del sistema incluso en escenarios inesperados.

Test de GUI para las interfaces de usuario.

También le prestamos mucha atención a las pruebas de interfaz gráfica (GUI), porque son clave para asegurarnos de que el usuario tenga una buena experiencia. La interfaz es lo que conecta directamente al sistema con el usuario, por lo que tiene que ser fácil de usar, intuitiva y responder bien a lo que el usuario haga. Durante las pruebas, verificamos que todos los elementos visuales, como botones, menús y campos de entrada, funcionaran correctamente, asegurándonos de que no hubiera errores en su comportamiento o en su apariencia, y registrando las aclaraciones cuando algo no funcionaba como esperábamos.

Enlace:

https://docs.google.com/document/d/1yeBGC-EzZN8-sSv_StB6GSYIlmNzfPwDV3Nl5T2nEAQ/edit?usp=sharing

Aclaraciones:

El test de GUI fue realizado basándose con las condiciones señaladas en el documento “Anexo,Ventana” realizado por la cátedra.

- El color verde indica que se cumplió con lo esperado.
- El color rojo indica que no se ha cumplido con lo esperado, a la derecha del color se explica que arrojó.
- Se escribe en color amarillo para aclarar que se utilizó en el modelo de Enabled Disabled.

Testeo de excepciones

El manejo de excepciones fue otro punto importante: verificamos que el sistema reaccionara bien ante situaciones fuera de lo común, lo que hizo que fuera más estable. Durante este proceso, pudimos detectar que algunas excepciones no cumplían con lo que se esperaba de ellas. Todo esto nos mostró lo importante que es hacer un testing completo, no solo para encontrar errores, sino también para que el software sea más confiable.

Documentación de funcionamiento de excepciones:

ChoferNoDisponibleException -> funciona correctamente.
ChoferRepetidoException -> no se cumple el lanzamiento esperado
ClienteConPedidoPendienteException -> funciona correctamente.
ClienteConViajePendienteException -> no se cumple el lanzamiento esperado
ClienteNoExisteException -> funciona correctamente.
ClienteSinViajePendienteException -> funciona correctamente.
PasswordErroneaException -> no se cumple el lanzamiento esperado
PedidoInexistenteException -> funciona correctamente.
SinVehiculoParaPedidoException -> funciona correctamente.
SinViajesException -> no se cumple el lanzamiento esperado
UsuarioNoExisteException -> funciona correctamente.
UsuarioYaExisteException -> funciona correctamente.
VehiculoNoDisponibleException -> no se cumple el lanzamiento esperado
VehiculoNoValidoException -> funciona correctamente.
VehiculoRepetidoException -> funciona correctamente.

HALLAZGO:

Una de las razones de errores encontrados tanto en la persistencia como en el test de GUI es que el pedido no se puede crear desde la ventana, ni se puede obtener. Se puede obtener únicamente en el caso donde el pedido fue instanciado manualmente.
Este hallazgo fue el más significativo encontrado por eso quisimos hacer esa aclaración, por qué afecta a todo el sistema.

CONCLUSIÓN:

Este proyecto fue una gran oportunidad para aprender sobre testing y darnos cuenta de lo importante que es asegurarnos de que todo en el sistema funcione bien. Aunque organizar las pruebas y cubrir todos los aspectos del sistema fue un desafío, al final valió la pena. Aprendimos a encontrar y corregir errores de manera más eficiente y mejoramos nuestras habilidades técnicas. Nos dimos cuenta de lo clave que es probar cada parte del sistema para que todo funcione de forma estable y confiable.

Lo que más aprendimos fue lo importante que es documentar todo lo que probamos. No solo para tener un registro de lo que hicimos, sino también para justificar por qué hicimos ciertas pruebas y asegurarnos de que el software sea fácil de mantener. Este proyecto nos mostró que hacer testing es fundamental para mejorar la calidad del software y asegurarnos de que funcione bien en diferentes situaciones. Además, nos dio una mejor idea de lo que se necesita para desarrollar un sistema sólido y bien hecho.