Plan de pruebas de la aplicación Seguros

Los niveles de prueba que se van a aplicar son los siguientes:

- Pruebas de aceptación. Las pruebas de aceptación se definirán siguiendo una estrategia basada en casos de uso y se ejecutarán de forma manual.
- Pruebas de integración. La estrategia para la definición del orden de las pruebas de integración será jerárquica. Se probará:
    - La integración entre la capa de negocio y la de persistencia. En este caso, para la definición de los casos de prueba se utilizarán técnica de métodos y caja negra y se utilizará JUnit.
    - La integración entre las tres capas. En este caso, para la definición de los casos de prueba se utilizarán técnica de casos de uso y se utilizarán Junit y FEST.
- Pruebas unitarias. Se utilizará la técnica de prueba de métodos, usando técnicas de caja negra (partición equivalente y AVL) para la definición de los casos de prueba de cada método de cada clase o componente. Será necesaria la utilización de JUnit, Mockito y FEST.

A continuación, se muestra una especificación detallada de los casos de prueba a aplicar en cada nivel mencionado anteriormente.

PRUEBAS DE ACEPTACIÓN

En base a los casos de uso se identifican los siguientes escenarios:

A1. CU: Nuevo Cliente
- a. Alta válida (nuevo cliente)
- b. Alta no válida (cliente repetido)

A2. CU: Baja cliente
- a. Baja válida (cliente existente sin seguros a su nombre)
- b. Baja no válida (cliente existente con seguros a su nombre)
- c. Baja no válida (cliente no existente)

A3. CU: Alta seguro
- a. Alta válida
- b. Alta no válida (cliente no existente)
- c. Alta no válida (seguro ya existente)

A5. CU: Consulta cliente
- a. Consulta válida (cliente existente)
- b. Consulta no válida (cliente no existente)

A6. CU: Consulta seguro
- a. Consulta válida (seguro existente)
- b. Consulta no válida (seguro no existente)

A7. CU: Baja seguro

a. Baja válida (cliente existente y seguro existente)
b. Baja no válida (cliente existente y seguro no existente)
c. Baja no válida (cliente no existente y seguro existente)
d. Baja no válida (cliente no existente y seguro no existente)
e. Baja no válida (el seguro no pertenece a ese cliente)

Los casos de prueba definidos para cada uno de estos escenarios, suponiendo que el estado inicial del sistema es el reflejado en el fichero aseguradora.xml, son los que se muestran en la Tabla 1. Los casos de prueba deberían ser ejecutados en el orden indicado, para que el estado final del sistema sea igual al inicial.

Tabla 1. Casos de prueba de aceptación

Identificador	Entrada	Resultado
A1.a	Lidia López Revuelta, 77777777G, true	Alta válida
A1.b	12345678S	Cliente ya existe
A2.a	11111111ª	Baja correcta
A2.b	12345678S	Tiene seguros asociados
A2.c	77777777G	Cliente no existe
A3.a	(“2020AAA”, TODORIESGO, 100, hoy) para el usuario con DNI 12345678S	Alta correcta
A3.b	(“2020AAA”, TODORIESGO, 100, hoy) para el usuario con DNI 77777777G	Cliente no existe
A3.c	(“PLL9597”, TODORIESGO, 100, 2022-01-01) para el usuario con DNI 12345678S	Seguro ya existe
A4.d	PLX9597 para el usuario con DNI 12345678S	El seguro no pertenece al cliente
A5.a	12345678S	Andrés Ortega . . .
A5.b	77777777G	Cliente no existe
A6.a	PLL9597	PLL9597 TODORIESGO 75 2022-01-01
A6.b	1111AAA	Seguro no existe
A7.a	“PLL9597”, “12345678S”	Baja correcta
A7.b	“2020AAA”, “12345678S”	Seguro no existente
A7.c	“PLL9597”, 77777777G	Cliente no existente
A7.d	“2020AAA”, “77777777G”	Seguro y cliente no existente
A7.e	“PLX9597”, “11111111A”	El seguro no pertenece a ese cliente

PRUEBAS DE INTEGRACIÓN

El orden de las pruebas y los casos de prueba a realizar serían los siguientes:

1. GestionSeguros con ClientesDAO y SegurosDAO. Se usarían los mismos casos de prueba definidos como UGIC.x en la sección de pruebas unitarias, aquí renombrados como IGIC.x.
2. VistaAgente con GestionSeguros, ClientesDAO y SegurosDAO. Estas pruebas coincidirían con las pruebas de aceptación, aunque en este caso se automatizarían utilizando la librería FEST. Podría hacerse una prueba previa de integración VistaAgente y GestionSeguros usando objetos Mock de los componentes DAO, pero no se considera necesario.

En este ejemplo, la prueba de la integración de los componentes DAO con la BBDD no tiene sentido pues se implementa con un fichero.

PRUEBAS UNITARIAS

Pruebas unitarias de las clases de dominio

Se aplican técnicas de clases de equivalencia y AVL para la definición de los casos de prueba de cada método. Además, se deberá garantizar la cobertura de condición/decisión. A modo de ejemplo, los casos de prueba válidos para el método precio() de la clase Seguro serían los que se muestran en la Tabla 2. La definición completa de estas pruebas y su ejecución se realizó en la práctica 3 de Ingeniería del Software II en el curso anterior.

Tabla 2. Casos de prueba válidos para el método precio de la clase Turismo

Identificador	Entrada	Valor esperado
UT.1ª	(45, TODORIESGO, hoy)	800
UT.1b	(100, TERCEROS, hoy – 30 días)	340
UT.1c	(300, TERCEROSLUNAS, hoy-(1 año+1dia))	600
UT.1d	(90, TERCEROS, hoy-1año)	380
UT.1e	(110, TODORIESGO, hoy-550días)	950
UT.1f	(111, TODORIESGO , hoy- hoy-(2años +1 día))	1100
UT.1g	(200, TERCEROSLUNAS , hoy-2años)	720
UT.1h	(55, TERCEROS , hoy – 965)	400

Pruebas unitarias de la capa de persistencia

Se aplica prueba de métodos, siendo los casos de prueba definidos para cada método los que se exponen a continuación. Los casos expuestos para cada método suponen como punto de partida el fichero aseguradora.xml (al codificar las pruebas deberá tenerse cuidado para no modificar el estado final del fichero).

- Método creaCliente

Identificador	Entrada	Valor esperado
UCD.1a	Lidia López Revuelta, 77777777G, true (Cliente nuevo)	Lidia López Revuelta, 77777777G
UCD.1b	12345678S (Cliente ya existe)	Null

- Método eliminaCliente

Identificador	Entrada	Valor esperado
UCD.2a	12345678S (Cliente ya existe)	Andrés Ortega, 12345678S
UCD.2b	77777777G (Cliente no existe)	null

- Método cliente()

Identificador	Entrada	Valor esperado
UCD.2ª	12345678S (Cliente ya existe)	Andrés Ortega, 12345678S
UCD.2b	77777777G (Cliente no existe)	Null

- Método clientes()

Identificador	Entrada	Valor esperado
UCD.4a		Listado con 3 clientes
UCD.4b		Lista vacía (sería necesario probar con un fichero aseguradora.xml vacío)

- Método actualizaCliente()

Identificador	Entrada	Valor esperado
UCD.5a	Andrés Ortega, 12345678S, false (Cliente existe)	Andrés Ortega, 12345678S, false
UCD.5b	Lidia López Revuelta, 77777777G (Cliente no existe)	Null

(Similares casos de prueba para los métodos de la interfaz ISegurosDAO).

Pruebas unitarias de la capa de negocio

Para poder llevar a cabo estas pruebas, será necesario el uso de objetos Mock para las interfaces IClientesDAO e ISegurosDAO. Se aplica prueba de métodos, siendo los casos de prueba definidos para cada método los siguientes.

- Método nuevoSeguro: Conceptualmente se trata de los mismos casos identificados para el caso de uso 3.

Identificador	Entrada	Valor esperado
UGIC.1a	(“2020AAA”, TODORIESGO, 100, hoy), 12345678S	(“2020AAA”, TODORIESGO, 100, hoy)
UGIC.1b	(“2020AAA”, TODORIESGO, 100, hoy), 77777777G	null
UGIC.1c	(“PLL9597”, TODORIESGO, 100, 2022-01-01), 12345678S	OperacionNoValida()

- Método bajaSeguro: Conceptualmente se trata de los mismos casos identificados para el caso de uso 3.

Identificador	Entrada	Valor esperado
UGIC.2a	"PLL9597", "12345678S"	(75, "PLL9597", TODORIESGO, "2022-01-01")
UGIC.2b	"2020AAA", "12345678S"	Null
UGIC.2c	"PLL9597", "77777777G"	Null
UGIC.2d	"2020AAA", "77777777G"	Null
UGIC.2e	"PLX9597", "12345678S"	OperacionNoValida()

- Método seguro: Conceptualmente se trata de los mismos casos identificados para el caso de uso 6.

Identificador	Entrada	Valor esperado
UGIC.3a	PLL9597	("PLL9597", TODORIESGO, 75, 2022-01-01)
UGIC.3b	1111AAA	null

- Método nuevoCliente: Conceptualmente se trata de los mismos casos identificados para el caso de uso 1.

Identificador	Entrada	Valor esperado
UGIC.4ª	Lidia López Revuelta, 77777777G, true (Cliente nuevo)	Lidia López Revuelta, 77777777G
UGIC.4b	12345678S (Cliente ya existe)	Null

- Método cliente: Conceptualmente se trata de los mismos casos identificados para el caso de uso 5.

Identificador	Entrada	Valor esperado
UGIC.5ª	12345678S	Andrés Ortega, …
UGIC.5b	77777777G	Null

- Método bajaCliente(): Conceptualmente se trata de los mismos casos identificados para el caso de uso 2.

Identificador	Entrada	Valor esperado
UGIC.8a	11111111A	Patricia López …
UGIC.8b	12345678S	OperacionNoValida
UGIC.8c	77777777G	null

Pruebas unitarias de la capa de presentación

Para poder llevar a cabo estas pruebas, será necesario el uso de objetos Mocks para las interfaces IGestionClientes e IGestionSeguros.

En este caso se aplica la técnica basada en casos de uso para la definición de las pruebas a realizar. Los casos de prueba definidos serán los mismos que los de las pruebas de aceptación (renombrados como UVF.X) pero automatizados a través de JUnit y FEST.

Informe de pruebas

Durante las pruebas realizadas en el código se encontraron múltiples fallos en este. Al aplicar las pruebas unitarias definidas en el plan de pruebas para los métodos bajaSeguro y altaSeguro. Para altaSeguro en el caso de prueba de que el seguro no exista no lanzaba el throw OperacionNoValida para ello ha sido necesario añadir la comprobación al método y lanzar el throw. Por la parte de bajaSeguro en el caso de prueba de que el seguro no exista es necesario que retorne null, anteriormente no pasaba ya que no se comprobaba que el seguro fuese distinto de null para seguir con el código. Ahora se comprueba y se retorna null en el caso que no exista el seguro.

En las pruebas unitarias de la capa de presentación de la aplicación también se encontraron fallos. Para el caso de prueba de probar el caso valido de que el DNI esta registrado, no se mostraba el seguro del cliente, esto era debido a que la listSeguros no tenia un setName y no se podía encontrar la lista por su nombre. También, a la hora de mostrar el pago total que tenia que hacer el cliente fallaba, ya que el setName de este textBox estaba puesto el nombre del textBox del nombre del cliente, para arreglarlo se le cambio el setName al suyo correspondiente. Una vez solucionado ese fallo el precio no se mostraba correctamente, para solucionarlo se comprobó que el método precio de la clase Seguro tenía un “;” después del primer else if, lo cual provocaba que no entrase en la siguiente línea entonces se borró. Luego, en el caso de prueba de que el DNI no exista se decidió quitar la tilde de válido, ya que según la codificación se muestran caracteres raros.

Para terminar, en las pruebas de integración no se encontraron más fallos.