Ingeniería de Software

Proceso de Pruebas
Entrega 3

Crístofer Canosa Domínguez
Silvia Rodríguez Alcaraz
Orquídea Seijas Salinas
Samuel Soutullo Sobral

FECHA DE ENTREGA: 19/05/2017

CONTROL DE VERSIONES		
Edición	Fecha	Descripción del cambio
1.0	02/04/2017	Diseño y casos de prueba de caja negra.
2.0	19/05/2017	Modificaciones y casos de prueba de caja blanca. Realización de las pruebas y obtención de resultados.

ÍNDICE

1 Plan de pruebas

1.1 Identificador único del documento

PDP_v1

1.2 Introducción

Este documento se corresponde con el plan de pruebas correspondiente al proyecto de ventas de la USC. El software se basa en un portal web mediante el cual los usuarios pueden solicitar pedidos, siendo tramitados posteriormente por la universidad.

El objetivo de este plan es marcar las pautas y definir la estrategia que se seguirá para certificar el software. En este caso, los casos de prueba y procedimientos tan sólo están generados mediante métodos de caja negra contra la especificación de las interfaces del software.

Por tanto, los elementos sobre los que se aplicarán las pruebas son las interfaces correspondientes a los módulos de importación, control, estadística y acceso a base de datos (DAO). Dado que las pruebas son de caja negra, se centrarán en el estudio de las especificaciones: testear las entradas para comprobar si se obtienen las salidas esperadas. Para la selección de casos de prueba se utilizó un enfoque sistemático basado en el método de partición o clases de equivalencia.

1.3 Clases software a probar

Se ha decidido probar las clases con más probabilidad de tener algún fallo: ImportsModule, ItemDAO, PurchaseDAO, UserDAO y StatisticsModule. Las clases dentro del paquete Model se han probado a través de los métodos contenidos en las clases citadas. Cabe decir que el controlador no se ha probado debido a que si funcionan todas las clases anteriores se sobreentiende que éste funcionará de forma correcta.

1.4 Funcionalidades a probar

Se han probado todas las funcionalidades implementadas por el equipo autor del código que se nos ha adjudicado. Las funcionalidades probadas han sido: para el requisito 1, insertar y modificar usuarios; para el requisito 2, insertar y modificar ítems; y para el requisito 3, insertar ventas.

1.5 Funcionalidades que no se prueban

No se han probado las funcionalidades no implementadas por el otro equipo. Éstas serían las no citadas en el apartado anterior: dar de baja a usuarios y mostrar productos como no disponibles.

1.6 Enfoque general de la prueba

Las pruebas se ejecutarían siguiendo una jerarquía: en primer lugar, se ejecutaría la interfaz de importación. En segundo lugar, las pruebas que se realizarían serían las de inserción y lectura, que se aplican sobre la interfaz DAO y, finalmente, se prueba la interfaz estadística. Ésta se encarga de realizar cálculos con los datos anteriormente introducidos en la base de datos.

1.7 Criterios de paso/fallo para cada elemento

La aprobación o rechazo de una funcionalidad, sobre la que se han aplicado pruebas, dado que nos encontramos en un contexto de una aplicación en producción, se basará en un criterio basado en la funcionalidad de las mismas. Para cada funcionalidad, el criterio de paso/fallo será que funcionen todos los casos válidos y el 99% de casos no válidos se comporten como deben.

1.8 Criterios de suspensión y requisitos de reanudación

En ocasiones cuando el número o tipo de errores llega a un punto en el que el seguimiento de la prueba no tiene valor, es mejor barajar el hecho de frenar el proceso de pruebas. A continuación, se indica en qué situación se frenarían las pruebas para no desperdiciar recursos:

- **Error en método simple:** tras detectar un error al realizar pruebas sobre un método simple de un módulo concreto, se detendrá la ejecución de pruebas sobre el resto de métodos más complejos. Las pruebas se reanudarán cuando el equipo de implementación haya arreglado estos errores.

1.9 Documentos a entregar

- Plan de pruebas
- Informe de ejecución

1.10 Actividades de preparación y ejecución de pruebas

Para el correcto desarrollo de las pruebas se necesita contar con equipos que contengan el software que se utilizará para su realización. En este caso, Mockito y JUnit, además del correspondiente IDE de trabajo que compartirán todos los programadores de pruebas a la hora de codificar: Eclipse. Es obvio que, además, todo el personal implicado en la ejecución de las pruebas tendrá acceso al código de las implementado, el código de los elementos a probar y el plan de pruebas a seguir.

1.11 Necesidades de entorno

Para la realización de las pruebas será necesario tener una base de datos funcional con los datos necesarios para cada uno de los casos de prueba. Dado que los casos serán idempotentes siempre que no se indique lo contrario, serán ellos los encargados de inicializar la base de datos a sus valores necesarios para la correcta ejecución.

1.12 Responsabilidades en la organización y realización de las pruebas

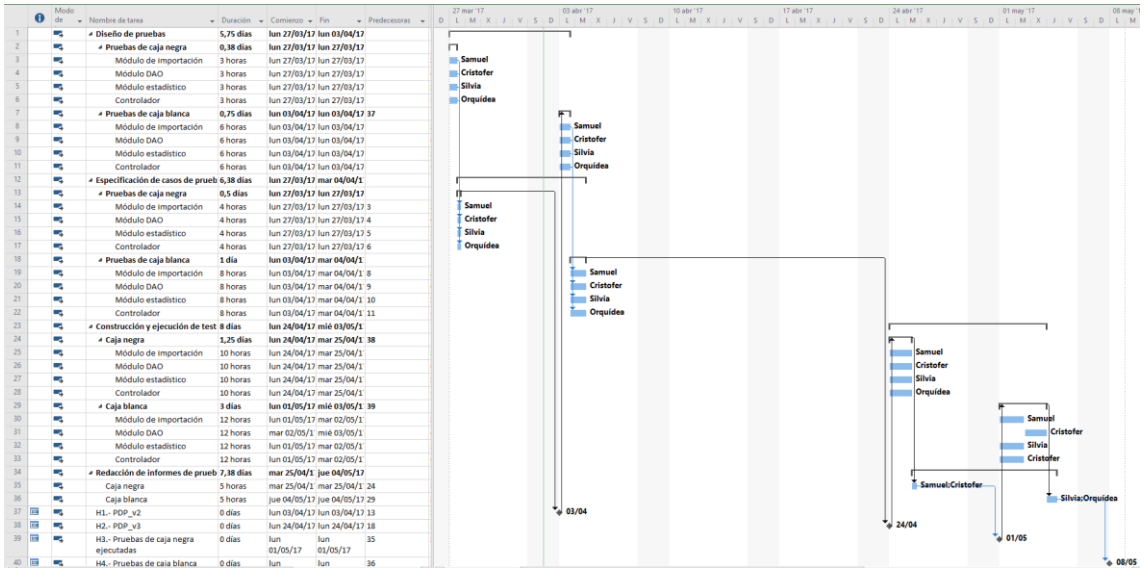
Para la realización de las pruebas se precisará un equipo de pruebas que está formado por personas que cumplen los siguientes roles:

Rol	Responsabilidad
Programador de pruebas	Responsable de la generación del código pertinente a las pruebas planteadas.
Gestor documental	Responsable de generar el documento de plan de pruebas y de redactar el informe de ejecución una vez finalicen las pruebas.

1.13 Necesidades de personal y de formación

El personal debe poseer los conocimientos técnicos informáticos suficientes como para poder implementar las pruebas que se van a realizar y comprender el código a probar, además de saber manejar las herramientas planteadas para el desarrollo de las pruebas (JUnit y Mockito).

1.14 Esquema de tiempos



1.15 Riesgos asumidos por el plan y planes de contingencias para cada riesgo

Riesgo	Descripción	Contingencia
Fuerte limitación temporal para las pruebas.	Intervalo de tiempo escaso para realizar las pruebas.	Planteamiento de pruebas con escaso nivel de detalle y exclusión de algunos casos de prueba a la hora de implementar las pruebas.
Recursos con escaso tiempo para asignar a las pruebas.	Recursos con poca disponibilidad.	Redistribución de la carga entre recursos más disponibles.
Modificación del código sobre el que se hizo el plan de pruebas.	El código sobre el que se realizó el plan de pruebas fue alterado.	Corregir lo antes y mejor posible el plan de pruebas y proceder a su implementación tras esto.

1.16 Aprobaciones y firmas con nombre y puesto desempeñado

La aprobación para definir el proceso como completo y permitir que avance al siguiente nivel es una tarea cooperativa entre dos departamentos. Mediante una reunión formal en la que se encuentren el jefe de desarrollo y el analista a cargo del proyecto se decidirá si el software está preparado. Se precisa de la aprobación de ambos ya que es necesario contar con un aprobado a nivel técnico y otro más enfocado a un nivel empresarial. De esta forma, mediante el debate y el estudio de los informes de pruebas se considera que el software pasa a la siguiente fase si se aprueba por unanimidad.

La firma del analista y el jefe de desarrollo supondrían la aprobación del software y, por tanto, el pase a la siguiente fase del proyecto.

2 Diseño de pruebas

2.1 Pruebas importación

2.1.1 Prueba I-01

2.1.1.1 Objetivo

Comprobar qué sucede al introducir un archivo en el programa variando el contenido del mismo.

2.1.1.2 Técnicas de caja negra

Aplicando las reglas especificadas a continuación se obtienen las siguientes clases de equivalencia:

- **R1**: incluir una línea con campos de menos (todos ellos con valores válidos), otra línea con campos de más (los que no se hayan incluido a mayores deben tener valores válidos) y una línea con el número de campos correctos (todos ellos con valores válidos).
- **R3**: incluir una línea en blanco y otra completa y correcta.
- **R1**: para todos los campos con longitud máxima delimitada, se probará introduciendo el campo vacío, el campo con una longitud correcta, y el campo con una longitud superior a la correcta.
- **R3**: para cada uno de los campos sujetos a restricción de formato, se probará un caso con formato correcto y otro con formato incorrecto (por ejemplo: fecha con formato adecuado, unidades, cantidad y precio deben ser numéricos, etc.).
- **R3**: para cada uno de los campos identificadores, se probará un caso con un identificador ya presente en la base de datos, y otro no presente en la base de datos. En otras palabras, se probará a importar un elemento ya presente en la base de datos y otro que no lo esté.
- **R3:** para los campos de referencia a ítem y usuario en las líneas de venta, se probará un caso en el que el identificador del ítem o usuario referenciado se encuentre previamente en la base de datos y otro caso en el que no.

Dada su relación directa con el tipo de línea en cuestión, las reglas que se listan a continuación sí se aplican de manera diferente a cada método:

- importarUsuarios
  - **R3:** un caso en el que el primer campo sea *U* y otro en el que sea distinto de *U*.
- importarProducto
  - **R3:** un caso en el que el primer campo sea *I* y otro en el que sea distinto de *I*.
- importarCompra
  - **R3:** un caso en el que el primer campo sea *V* y otro en el que sea distinto de *V*.

- **R3:** para los campos de referencia a ítem y usuario en las líneas de venta, se probará un caso en el que el identificador del ítem o usuario referenciado se encuentre previamente en la base de datos y otro caso en el que no.

Aplicando las técnicas de análisis por valores límite se obtiene lo siguiente:

- **R1**: para todos los campos con longitud máxima *m*, se probará introduciendo valores con longitud 0, 1, *m* y *m + 1*.

Finalmente, aplicando la técnica de conjetura de errores, se obtiene lo siguiente:

- Se debe comprobar si la existencia de varias líneas de venta con el mismo identificador provoca que se añadan varios ítems a la misma venta. Si esto no sucede, el comportamiento es incorrecto.

Dado que las clases de equivalencia se obtienen en base al contenido del archivo, se aplicarán a los tres métodos de la interfaz (*importarUsuarios, importarProducto* e *importarCompra*).

2.1.1.2.1 Resultado de aplicar las técnicas de caja negra

Información	Tipo de dato	Regla	Clase válida	Clase no válida
Línea	String[]	R1	{1} Número de campos correcto.	{2} Número de campos inferior al correcto.
				{3} Número de campos superior al correcto.
		R3	{4} Línea correcta.	{5} Línea en blanco.
Nombre	String	R1	{6} Longitud entre 1 y 255 caracteres.	{7} Longitud de cero caracteres.
Apellidos				{8} Longitud superior a 255 caracteres.
Categoría				
Descripción	String	R1	{9} Longitud entre 1 y 1024 caracteres.	{10} Longitud de cero caracteres.
				{11} Longitud superior a 1024 caracteres.
Fecha	Date	R3	{12} Fecha con formato dd/mm/aaaa	{13} Fecha formato distinto a dd/mm/aaaa
Unidades	Integer	R3	{14} Formato de número entero.	{15} Formato distinto a número entero.
Cantidad				
PrecioUnidad	Float	R3	{16} Formato de número con dígitos decimales.	{17} Formato distinto de número con dígitos decimales.
IdxUser	String	R3	{18} Identificador no presente previamente en la base de datos.	{19} Identificador previamente presente en la base de datos.
ItemRef				
VRef		R3	{20} Primer campo con el valor adecuado para el método llamado	{21} Primer campo con un valor incorrecto para el método llamado
IdxUser	String	R3		

ItemRef			{22} Identificador referenciado previamente en base de datos.	{23} Identificador referenciado no previamente en base de datos.

2.1.1.3 Técnicas de caja blanca

2.1.1.3.1 importarUsuarios

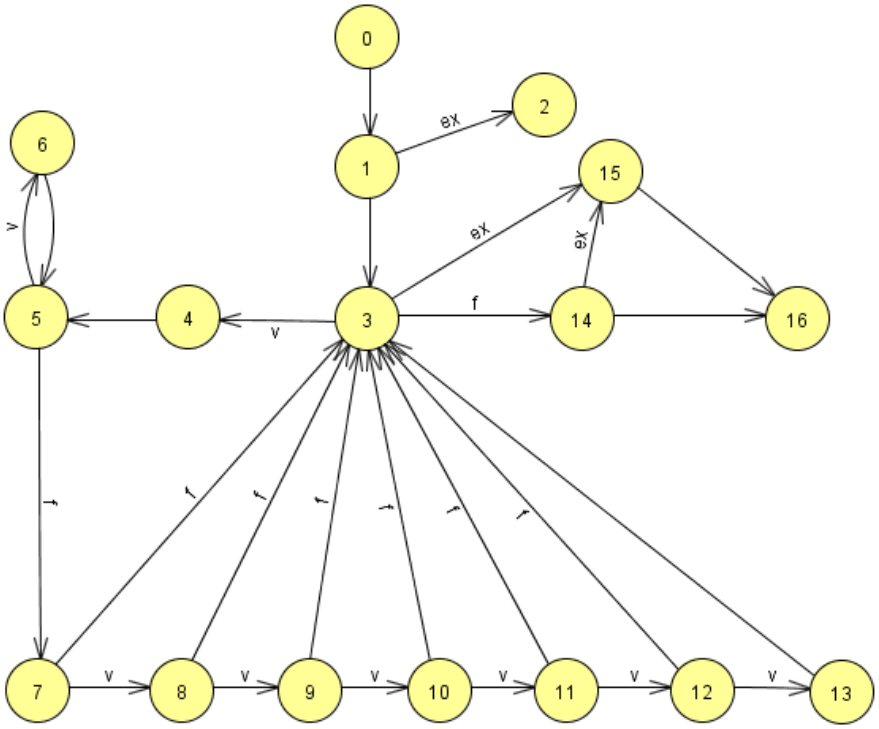


Figura 1. Grafo del método importarUsuario

2.1.1.3.1.1 Definición de nodos

- 0: sentencias previas al primer ‘try‘
- 1: sentencias dentro del primer ‘try‘
- 2: sentencias dentro del ‘catch(FileNotFoundException e)‘
- 3: condición del bucle ‘while‘
- 4: primera sentencia dentro del bucle ‘while‘
- 5: condición del bucle ‘for‘
- 6: sentencia dentro del bucle ‘for‘
- 7: primera condición del ‘if‘
- 8: segunda condición del ‘if‘
- 9: tercera condición del ‘if‘
- 10: cuarta condición del ‘if‘
- 11: quinta condición del ‘if‘
- 12: sexta condición del ‘if‘
- 13: sentencias dentro del ‘if‘
- 14: sentencia ‘lectura.close();‘
- 15: sentencias dentro del ‘catch(IOException ioe)‘
- 16: sentencia ‘return numero;

2.1.1.3.1.2 Complejidad ciclomática

V(G) = a - n + 2 = 25 - 17 + 2 = 10

2.1.1.3.1.3 Selección de caminos

- *Camino 1:0 - 1 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 12 - 13 - 3 - 14 - 16*
  Es el camino del caso base. Se lee un archivo con una línea.
- *Camino 2: 0 - 1 - 2*
  Se lanza una excepción porque el archivo no existe.
- *Camino 3: 0 - 1 - 3 - 15 - 16*
  Se lanza una excepción porque se produce un error de E/S al leer una línea del archivo.
- *Camino 4: 0 - 1 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 12 - 13 - 3 - 14 - 15 - 16*
  Se lanza una excepción porque se produce un error de E/S al cerrar el *BufferedReader*.
- *Camino 5: 0 - 1 - 3 - 4 - 5 - 6 - 7 - 3 - 14 - 16*
  El *if* falla en la primera condición.
- *Camino 6: 0 - 1 - 3 - 4 - 5 - 6 - 7 - 8 - 3 - 14 - 16*
  El *if* falla en la segunda condición.
- *Camino 7: 0 - 1 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 3 - 14 - 16*
  El *if* falla en la tercera condición.
- *Camino 8: 0 - 1 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 3 - 14 - 16*
  El *if* falla en la cuarta condición.
- *Camino 9: 0 - 1 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 3 - 14 - 16*
  El *if* falla en la quinta condición.
- *Camino 10: 0 - 1 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 12 - 3 - 14 - 16*
  El *if* falla en la sexta condición.

2.1.1.3.2 ImportarCompra

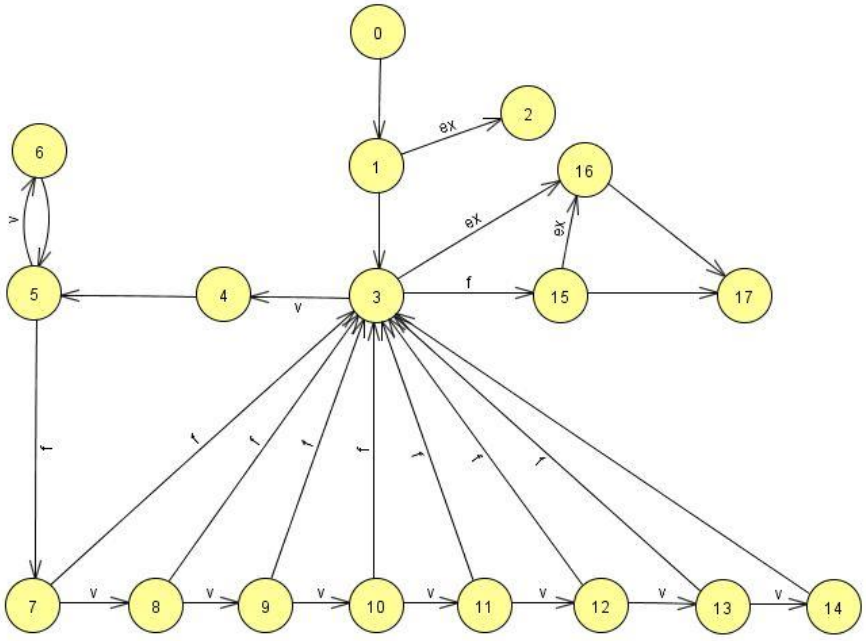


Figura 2. Grafo del método importarCompra

2.1.1.3.2.1 Definición de nodos

- 0: sentencias previas al primer ‘try‘
- 1: sentencias dentro del primer ‘try‘
- 2: sentencias dentro del ‘catch(FileNotFoundException e)‘
- 3: condición del bucle ‘while‘
- 4: primera sentencia dentro del bucle ‘while‘
- 5: condición del bucle ‘for‘
- 6: sentencia dentro del bucle ‘for‘
- 7: primera condición del ‘if‘
- 8: segunda condición del ‘if‘
- 9: tercera condición del ‘if‘
- 10: cuarta condición del ‘if‘
- 11: quinta condición del ‘if‘
- 12: sexta condición del ‘if‘
- 13: séptima condición del ‘if‘
- 14: sentencias dentro del ‘if‘
- 15: sentencia ‘lectura.close();‘
- 16: sentencias dentro del ‘catch(IOException ioe)‘
- 17: sentencia ‘return numero;‘

2.1.1.3.2.2 Complejidad ciclomática

V(G) = a - n + 2 = 27 - 18 + 2 = 11

2.1.1.3.2.3 Selección de caminos

- *Camino 1: 0 - 1 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 12 - 13 - 14 - 3 - 15 - 17*
  Camino del caso base. Se lee un archivo con una línea.
- *Camino 2: 0 - 1 - 2*
  Se lanza una excepción porque el archivo no existe
- *Camino 3: 0 - 1 - 3 - 16 - 17*
  Se lanza una excepción porque se produce un error de E/S al leer una línea del archivo
- *Camino 4: 0 - 1 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 12 - 13 - 14 - 3 - 15 - 16 - 17*
  Se lanza una excepción porque se produce un error de E/S al cerrar el ‘BufferedReader‘
- *Camino 5: 0 - 1 - 3 - 4 - 5 - 6 - 7 - 3 - 15 - 17*
  El ‘if‘ falla en la primera condición
- *Camino 6: 0 - 1 - 3 - 4 - 5 - 6 - 7 - 8 - 3 - 15 - 17*
  El ‘if‘ falla en la segunda condición
- *Camino 7: 0 - 1 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 3 - 15 - 17*
  El ‘if‘ falla en la tercera condición
- *Camino 8: 0 - 1 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 3 - 15 - 17*
  El ‘if‘ falla en la cuarta condición
- *Camino 9: 0 - 1 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 3 - 15 - 17*
  El ‘if‘ falla en la quinta condición
- *Camino 10: 0 - 1 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 12 - 3 - 15 - 17*
  El ‘if‘ falla en la sexta condición
- *Camino 11: 0 - 1 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 12 - 13 - 3 - 15 - 17*
  El ‘if‘ falla en la séptima condición.

2.1.1.4 Criterios de paso/fallo

En ninguno de los casos se debe generar una excepción no controlada como consecuencia del formato inesperado de la línea del fichero o del campo. Cuando un archivo contenga líneas o campos incorrectos se debe cancelar la importación informando al usuario del error. Por otra parte, un archivo con líneas y campos correctos debe ser correctamente procesado.

2.1.2 Prueba I-02

2.1.2.1 Objetivo

Comprobar qué pasa al variar los argumentos que se introducen en los métodos de importación.

2.1.2.2 Técnicas de caja negra

Aplicando las reglas especificadas a continuación se obtienen las siguientes clases de equivalencia:

- **R3**: se probará, para los tres métodos de importación, un caso con un *path* con formato válido y otro caso con formato inválido.
- **R3**: se probará, para los tres métodos de importación, un caso con un *path* que referencia a un archivo existente, y otro caso que referencia a un archivo inexistente.

2.1.2.2.1 Resultado de aplicar las técnicas de caja negra

Información	Tipo de dato	Regla	Clase válida	Clase no válida
Path	String	R3	{1} El *path* tiene un formato válido.	{2} El *path* no tiene un formato válido.
		R3	{3} El *path* referencia a un fichero existente.	{4} El *path* referencia a un fichero inexistente.

2.1.2.3 Criterios de paso/fallo

En ninguno de los casos se debe generar una excepción no controlada como consecuencia de que los métodos reciban argumentos inválidos. Tanto si el fichero no existe como si la ruta tiene un formato incorrecto, se debe informar al usuario mediante un error. Por otra parte, si se pasa como argumento la ruta de un fichero de importación válido, la importación se debe realizar correctamente.

2.2 Pruebas de inserción y lectura sobre la base de datos

2.2.1 Prueba D-01

2.2.1.1 Objetivo

Comprobar que las inserciones o modificaciones de usuarios se reflejan correctamente en la base de datos.

2.2.1.2 Técnicas de caja negra

2.2.1.2.1 Generación de clases de equivalencia

Centrándose en los métodos de inserción y modificación de usuarios y haciendo uso de las reglas especificadas, se definen las siguientes clases:

- **Insertar usuario**
  - **R3**: Probar un usuario con ID válido y otro sin ID.
  - **R5**: Los Id de usuario de forma N-XXXXXX-000.
    - R1: Cadena de letras con 5 y 7 caracteres (incorrecto) y una correcta. Clase equivalente para la cadena numérica.
    - R3: Eliminar uno de los campos del ID.
    - R3: Que N sea U u otro caracter.
- **Modificar usuario**
  - Se supone que la introducción del usuario modificado en la base de datos será equivalente a la usada en el método anterior y, por tanto, ya está probada.
  - **R3**: Insertar datos modificados de un usuario existente (caso correcto) e insertar datos de un usuario que no existe.

2.2.1.2.2 Resultado de aplicar las técnicas de caja negra

Información	Tipo de dato	Regla	Clase válida	Clase no válida
insertUser	Usuario	R3	{1} Id válido.	{2} Id inválido
		R5,R1	{3} Cadena de 5 letras en el Id	{4} Cadena de 4 letras
				{5} Cadena de 6 letras
		R5,R1	{6} Cadena de 3 dígitos en el ID	{7} Cadena de 2 dígitos
				{8} Cadena de 4 dígitos
		R5,R3	{9} Uno de los campos del ID existe.	{10} Uno de los campos del ID no existe.
		R5,R3	{10} El valor N del ID es U	{11} Otro valor.
modUsuario	Usuario	R3	{12} Usuario existente modificado.	{13} Usuario correcto no existente.

2.2.1.3 Criterios de paso/fallo

Los usuarios introducidos o modificados usando datos correctos deben verse reflejados en la base de datos.

En los casos de incluir usuarios de forma errónea debe notificarse del error y en ningún caso reflejarse en la base de datos.

2.2.2 Prueba D-02

2.2.2.1 Objetivo

Comprobar que las inserciones o modificaciones de items y ventas se reflejan correctamente en la base de datos. Dada su similitud con las pruebas para usuarios no se

hará una prueba exhaustiva excepto en el caso de que se encuentren fallos en las pruebas anteriores.

2.2.2.2 Técnicas de caja negra

2.2.2.2.1 Generación de clases de equivalencia

Centrándose en los métodos de inserción y modificación de items y ventas y haciendo uso de las reglas especificadas, se definen las siguientes clases:

- **Insertar item / venta / Modificar item**
  - **R3:** Id válido para la inserción y uno inválido.
  - **R3:** Id existente y no existente para la modificación.

2.2.2.2.2 Resultado de aplicar las técnicas de caja negra

Información	Tipo de dato	Regla	Clase válida	Clase no válida
insertItem	Item	R3	{1} Id válido.	{2} Id inválido
insertVenta	Venta	R3	{3} Id válido.	{4} Id inválido
updateItem	Item	R3	{5} Id de ítem existente	{6}Id de ítem no existente

2.2.2.3 Criterios de paso/fallo

Los items o ventas introducidos o modificados usando datos correctos deben verse reflejados en la base de datos. En los casos de incluir items o ventas de forma errónea debe notificarse del error y en ningún caso reflejarse en la base de datos.

2.2.3 Prueba D-03

2.2.3.1 Objetivo

Comprobar la correcta validación de un pedido. Para la correcta validación de un pedido este se confirma como compra en la base de datos.

Este método fue cambiado en un momento posterior a la realización del caso de pruebas inicial, por lo que no se probará exhaustivamente el argumento purchase, siendo éste siempre correcto, debido a restricciones temporales.

2.2.3.2 Técnicas de caja negra

2.2.3.2.1 Generación de clases de equivalencia

Solo el metodo validateOrder gestiona esta acción.

- **validateOrder**
  - **R3**: Decisión afirmativa o no.

2.2.3.2.2 Resultado de aplicar las técnicas de caja negra

Información	Tipo de dato	Regla	Clase válida	Clase no válida
validateOrder	boolean	R3	{1} decision = true	{2} decision = false

2.2.3.3 Técnicas de caja blanca

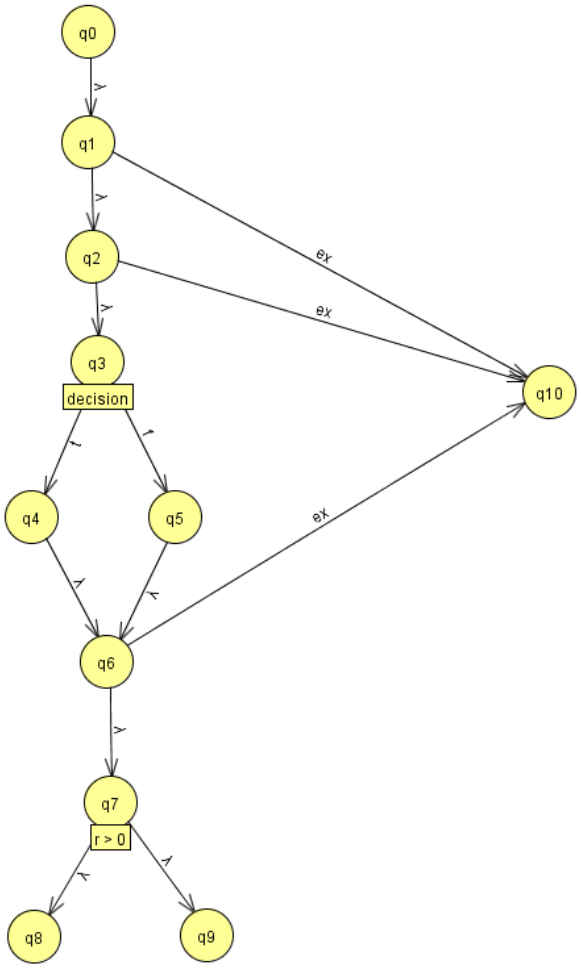


Figura 3. Grafo del método validateOrder

2.2.3.3.1 Definición de nodos

- 0: Declaraciones previas al try.
- 1: Apertura de la conexión.
- 2: Declaraciones de sql y ejecución de la inserción.
- 3: Condición para la variable decisión.
- 4: Asignación para la anterior condición si es correcta.
- 5: Sentencia vacía para q3 si es false. Implica que state = Order.DENEGATED.
- 6: Ejecución de la actualización y cierre de conexión.
- 7: Último if.
- 8: return true.
- 9: return false.
- 10: Ejecución del código de control de excepciones.

2.2.3.3.2 Complejidad ciclomática

V(G) = a - n + 2 = 13 - 11 + 2 = 4

2.2.3.3.3 Selección de caminos

- *Camino 1 (base): 0 - 1 - 2 - 3 - 4 - 6 - 7 - 8*
  Camino común aceptando el order.

- *Camino 2: 0 - 1 - 2 - 3 - 5 - 6 - 7 - 8*
  Camino común rechazando el order.
- *Camino 3: 0 - 1 - 10 - 7 - 9*
  Error al abrir la conexión.
- *Camino 4: 0 - 1 - 2 - 10 - 7 - 9*
  Error en la inserción
- *Camino 5: 0 - 1 - 2 - 3 - 4 - 6 - 10 - 7 - 9*
  Error en la actualización del estado o al cerrar la conexión.

2.2.3.4 Criterios de paso/fallo

La prueba será satisfactoria en caso de que el pedido pase al estado declarado siempre que el usuario tenga los privilegios para ello. En caso de que un usuario no esté autorizado deberá indicarse dicho error. Ninguna entrada incorrecta debe reflejarse en la base de datos.

2.2.4 Prueba D-04

2.2.4.1 Objetivo

Comprobar que el historial de compra de los usuarios devuelto se corresponde con el de la base de datos.

Modificación del diseño: Se ha modificado este método para aceptar un objeto User.

2.2.4.2 Técnicas de caja negra

2.2.4.2.1 Generación de clases de equivalencia

Se comprobará el método getHistorialUser que devuelve una lista de compras.

- **getHistorialUser**
  - **R3**: Id válido de usuario e Id inválido.
  - **R3**: Id de usuario existente e inexistente.

2.2.4.2.2 Resultado de aplicar las técnicas de caja negra

Información	**Tipo de dato**	**Regla**	**Clase válida**	**Clase no válida**
getHistorialUsuario	User	R3	{1} Id válido	{2} Id inválido
		R3	{3} Id de usuario existente	{4} Id de usuario no existente

2.2.4.3 Criterios de paso/fallo

Siempre que se introduzca un usuario correcto se devuelve un historial correspondiente a dicho usuario. En caso de introducir un usuario incorrecto o cuyo historial de compras es vacío se debe informar de dichos errores.

2.2.5 Prueba D-05

2.2.5.1 Objetivo

Comprobar la recuperación correcta de un ítem en función de su id.

2.2.5.2 Técnicas de caja negra

2.2.5.2.1 Generación de clases de equivalencia

Se comprueba el método getItemById, que recibe un id y devuelve el objeto Item con los datos de la base.

- **getItemById:**
  - **R3:** Id válido e inválido.

2.2.5.2.2 Resultado de aplicar las técnicas de caja negra

Información	**Tipo de dato**	**Regla**	**Clase válida**	**Clase no válida**
getItemById	String	R3	{1} Id válido	{2} Id inválido
		R3	{3} Id de item existente	{4} Id de ítem no existente

2.2.5.3 Criterios de paso/fallo

Cuando se introduce in id correcto, se deben visualizar los datos del ítem coherentes con aquellos en la base de datos.

En caso contrario se debe notificar del error, sin parar la ejecución correcta del programa.

2.2.6 Prueba D-06

2.2.6.1 Objetivo

Comprobar si se puede insertar un pedido correctamente.

2.2.6.2 Técnicas de caja blanca

2.2.6.2.1 insertOrder

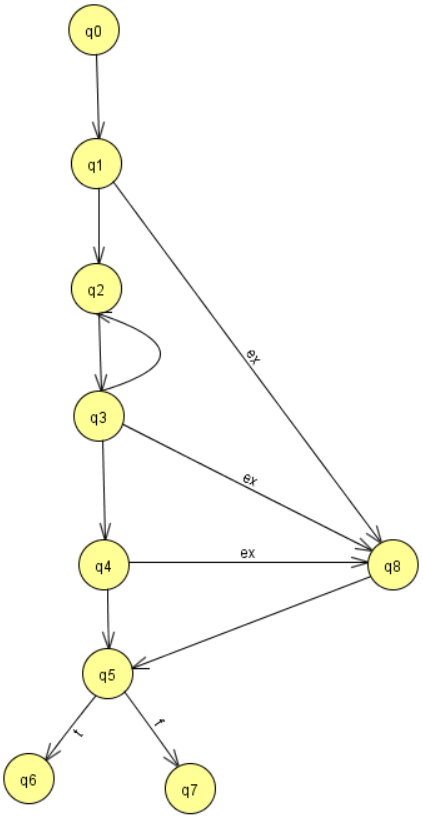


Figura 4. Grafo del método insertOrder

2.2.6.2.2 Definición de nodos.

- 0: Sentencias previas al **try**.
- 1: Desde abrir la conexión (susceptible de lanzar una excepción) hasta la ejecución de la primera inserción.
- 2: Condición del for.
- 3: Código interno del for.
- 4: Cierre de la conexión.
- 5: Último if.
- 6: return true.
- 7: return false.
- 8: Manejo de la excepción.

2.2.6.2.3 Complejidad ciclomática

V(G) = a - n + 2 = 11 - 9 + 2 = 4

2.2.6.2.4 Selección de caminos

- *Camino 1 (base): 0 - 1 - 2 - 3 - 2 - 3 - 4 - 5 - 6*
  Camino común con al menos una iteración del for (2-3).
- *Camino 2: 0 - 1 - 8 - 5 - 7*
  Error al abrir la conexión o en la primera sentencia.
- *Camino 3: 0 - 1 - 2 - 4 - 5 - 6*
  La venta no contiene líneas
- *Camino 4: 0 - 1 - 2 - 3 - 8 - 5 - 7*
  Error en la inserción de alguna de las líneas.

2.2.7 Prueba D-07

2.2.7.1 Objetivo

Comprobar que es posible obtener a todos los usuarios con toda su información a través de *getAllUsers*.

2.2.7.2 Técnicas de caja blanca

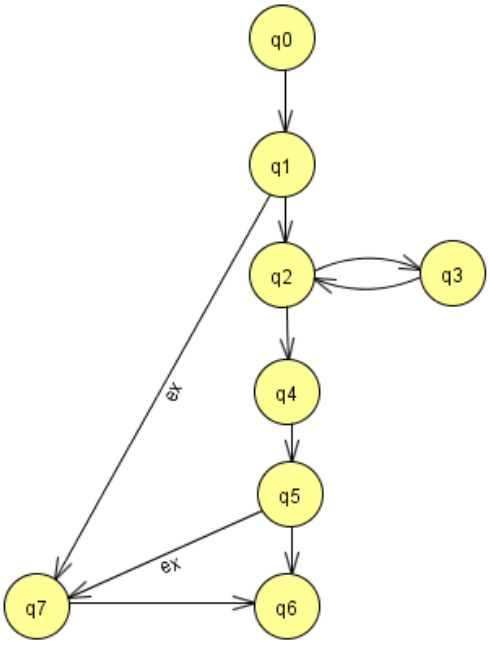


Figura 5. Grafo del método getAllUsers

2.2.7.2.1 Definición de nodos

- q0: Sentencias previas al **try**.
- q1: Desde abrir la conexión, acción susceptible de lanzar una excepción, hasta la ejecución de la consulta.
- q2: Condición del while.
- q3: Código dentro del while.
- q4: Cuando se deja de cumplir la condición del while.
- q5: Cierre de la conexión.
- q6: return usuarios.
- q7: Manejo de la excepción.

2.2.7.2.2 Complejidad ciclomática

V(G) = a - n + 2 = 10 - 8 + 2 = 4

2.2.7.2.3 Selección de caminos

- *Camino 1 (base): 0 - 1 - 2 - 3 - 4 - 5 - 6*
  Camino común con al menos una iteración del bucle while (2 - 3)
- *Camino 2: 0 - 1 - 7 - 6*
  Error al abrir la conexión.
- *Camino 3: 0 - 1 - 2 - 4 - 5 - 6*
  No se obtiene nada de la ejecución de la sentencia SQL y por tanto no se ejecuta el código interno del bucle while.
- *Camino 4: 0 - 1 - 2 - 4 - 5 - 7 - 6*
  Error al cerrar la conexión

2.2.7.3 Criterio de paso/fallo

La salida de este método no debe provocar una salida no controlada en el programa.

2.3 Pruebas sobre el módulo Estadistico

2.3.1 Prueba E-01

2.3.1.1 Objetivo

Comprobar qué sucede al introducir varios tipos de argumentos en los métodos getHistogramas y getPorcentajes del módulo estadístico.

2.3.1.2 Técnicas de caja negra

2.3.1.2.1 Generación de clases de equivalencia

Se ha aplicado la siguiente regla a los métodos getHistogramas y getPorcentajes de la interfaz ya que todos reciben la misma entrada: un entero en el rango [1,3] para referenciar respectivamente a días, semanas y meses.

- getHistogramas / getPorcentajes
  - **R4**: Introducir cada uno de los modos correctos {1,2,3} y uno incorrecto.

2.3.1.2.2 Resultado de aplicar las técnicas de caja negra

Información	Tipo de dato	Regla	Clase válida	Clase no válida
getHistos / getPorcentajes	Int	R4	{1} Modo 1	{2} Nº fuera del rango [1,3]
			{3} Modo 2	
			{4} Modo 3	

2.3.1.3 Técnicas de caja blanca

2.3.1.3.1 getPorcentajes

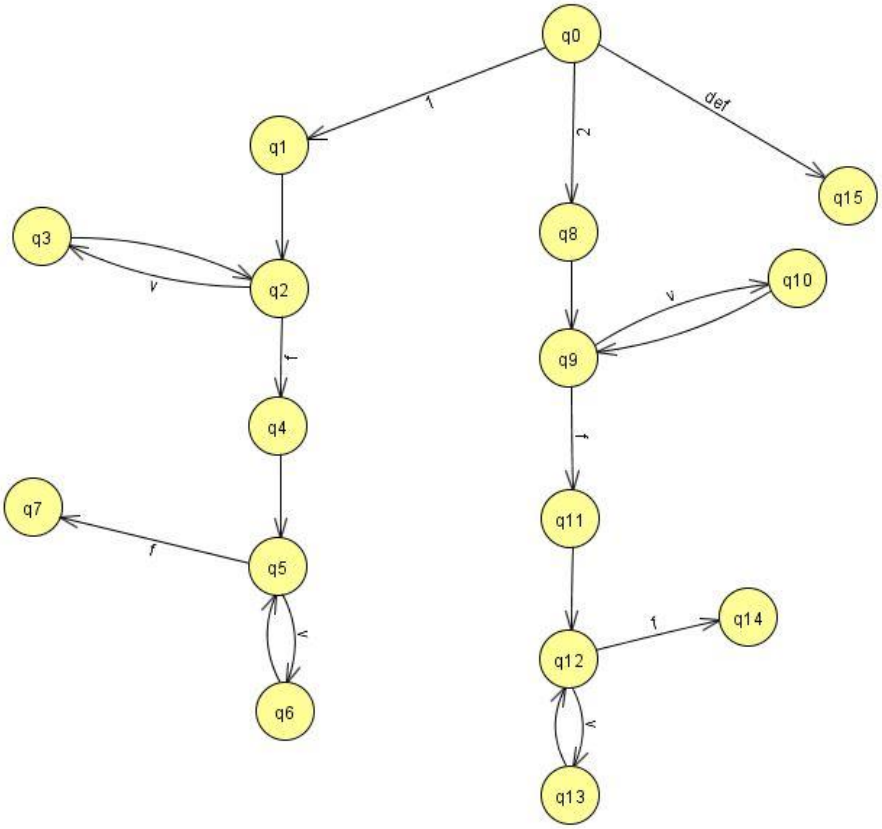


Figura 6. Grafo del método getPorcentajes

2.3.1.3.1.1 Definición de nodos

Sus bucles se tratan como concatenados ya que los valores del segundo dependen del primero en ambos case.

- 0: condición switch case.
- 1: Si mode=1.
- 2: Condición bucle 1 (case 1).
- 3: Código que contiene el bucle 1.
- 4: Inicialización del array de floats.
- 5: Condición bucle 2 (case 1).
- 6: Código del bucle 2 (case 1).
- 7: Return en caso de que no se cumpla la condición del bucle 2 (case 1).
- 8: Si mode=2.
- 9: Condición bucle 1 (case 2).
- 10: Código que contiene el bucle 1 (case 2).
- 11: Inicialización del array de floats.
- 12: Condición bucle 2 (case 2).
- 13: Código del bucle 2 (case 2).
- 14: Return en caso de que no se cumpla la condición del bucle 2 (case 2).
- 15: Si mode no es ni 1, ni 2 (default).

2.3.1.3.1.2 Complejidad ciclomática

V(G) = r = 5

2.3.1.3.1.3 Selección de caminos

- *Camino 1: 0-1-2-3-2-4-5-6*
  Modo = 1, ejecución siguiendo el flujo normal.
- *Camino 2: 0-1-2-3-2-4-5-7*
  Modo = 1, ejecución con condición no cumplida en el segundo bucle.
- *Camino 3: 0-8-9-10-9-11-12-13*
  Modo=2, ejecución siguiendo el flujo normal.
- *Camino 4: 0-8-9-10-9-11-12-14*
  Modo = 2, ejecución con condición no cumplida en el segundo bucle.
- *Camino 5: 0-15*
  Modo!=1 && Modo!=2

2.3.1.4 Criterios de paso/fallo

Dada una entrada no esperada en un método, ésta no puede provocar un comportamiento no controlado. De ocurrir esto, la consulta realizada se omitirá, devolviendo un mensaje de error indicando que el método no se ha utilizado de manera apropiada.

2.3.2 Prueba E-02

2.3.2.1 Objetivo

Comprobar qué sucede al introducir varios tipos de argumentos en los métodos getValoresBrutos y getMedias del módulo estadístico.

2.3.2.2 Técnicas de caja negra

2.3.2.2.1 Generación de clases de equivalencia

Se ha aplicado la siguiente regla a los métodos getValoresBrutos y getMedias ya que todos reciben la misma entrada: un entero para referenciar una cantidad de días sobre los que se desea obtener información.

- ***R3:*** Introducir un valor menor que 0 o un valor mayor o igual que 0.

2.3.2.2.3 Resultado de aplicar las técnicas de caja negra

Información	Tipo de dato	Regla	Clase válida	Clase no válida
getValoresBrutos / getMedias	Int	R3	{1} Nº días > 0	{2} Nº días <= 0

2.3.2.4 Técnicas de caja blanca

2.3.2.4.1 GetMedias

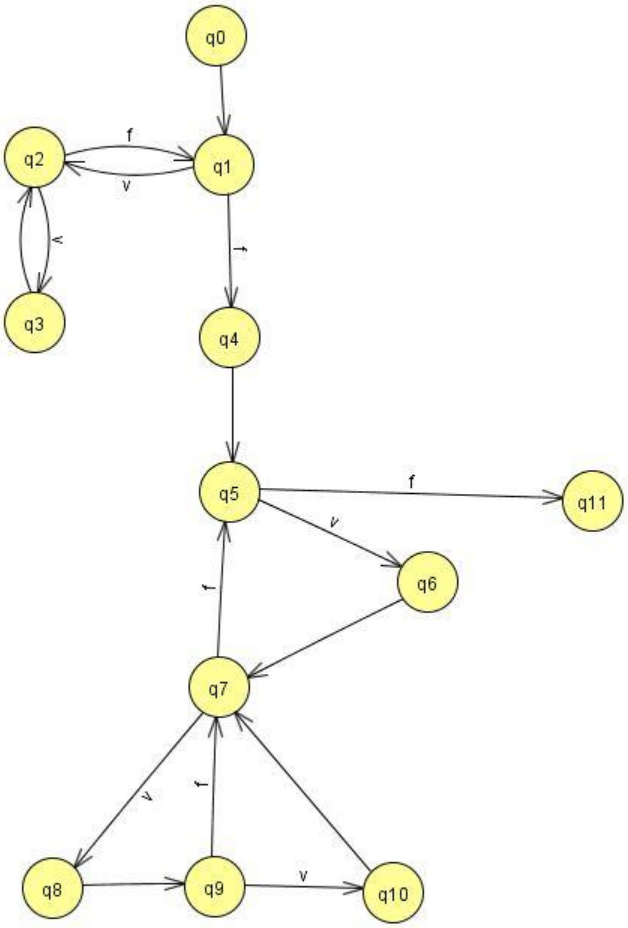


Figura 7. Grafo del método getMedias

2.3.2.4.1.1 Definición de nodos

- 0: Sentencias previas al for.
- 1: Condición del primer for.
- 2: Código interno del primer for y condición del for interno.
- 3: Código interno del for interno.
- 4: Código secuencial.
- 5: Condición del segundo for.
- 6: Código interno del segundo for
- 7: Condición del for interno.
- 8: Código interno del for interno.
- 9: Ejecución del if.
- 10: Condición verdadera y código asociado a la misma.
- 11: return valores.

2.3.2.4.1.2 Complejidad ciclomática

V(G) = a - n + 2 = 16 - 12 + 2 = 6

2.3.2.4.1.3 Selección de caminos

- *Camino 1 (base): 0 - 1 - 2 - 3 - 2 - 1 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 7 - 5 - 11*
  Camino común con al menos una iteración de cada bucle y la condición verdadera.
- *Camino 2: 0 - 1 - 4 - 5 - 6 - 7 - 5 - 11*
  La tabla de usuarios está vacia por lo que el primer conjunto de bucles no se ejecuta y el for interno del segundo tampoco.
- *Camino 3: 0 - 1 - 2 - 3 - 2 - 1 - 4 - 5 - 11*
  Se introduce 0 como valor de dias.
- *Camino 4: 0 - 1 - 2 - 1 - 4 - 5 - 6 - 7 - 5 - 11*
  La tabla de purchases está vacía, por lo que no se ejecuta el primer for interno ni el segundo.
- *Camino 5: 0 - 1 - 2 - 3 - 2 - 1 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 7 - 5 - 11*
  La tabla de purchases tiene más de un valor, la de usuarios tiene más de un valor y se introduce más de un día al método.
- *Camino 6: 0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 9 - 10*
  El if nunca se cumple.

2.3.2.4.2 GetValoresBrutos

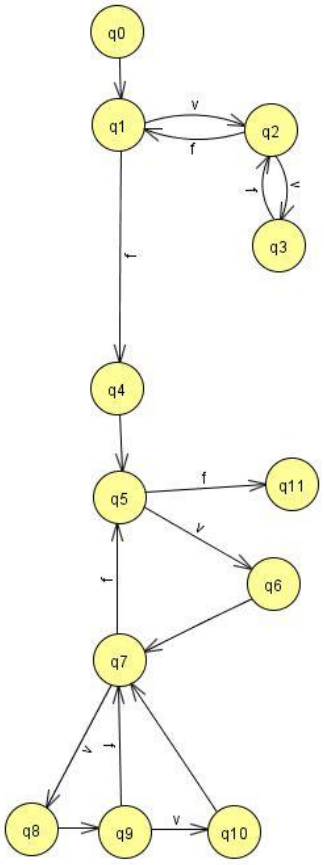


Figura 8. Grafo del método getValoresBrutos

2.3.2.4.2.1 Definición de nodos

- 0: Sentencias previas al for.
- 1: Condición del primer for.
- 2: Código interno del primer for y condición del for interno.
- 3: Código interno del for interno.
- 4: Código secuencial.

- 5: Condición del segundo for.
- 6: Código interno del segundo for
- 7: Condición del for interno.
- 8: Código interno del for interno.
- 9: Ejecución del if.
- 10: Condición verdadera y código asociado a la misma.
- 11: return valores.

2.3.2.4.2.2 Complejidad ciclomática

V(G) = a - n + 2 = 16 - 12 + 2 = 6

2.3.2.4.2.3 Selección de caminos

- *Camino 1 (base): 0 - 1 - 2 - 3 - 2 - 1 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 7 - 5 - 11*
  Camino común con al menos una iteración de cada bucle y la condición verdadera.
- *Camino 2: 0 - 1 - 4 - 5 - 6 - 7 - 5 - 11*
  La tabla de usuarios está vacia por lo que el primer conjunto de bucles no se ejecuta y el for interno del segundo tampoco.
- *Camino 3: 0 - 1 - 2 - 3 - 2 - 1 - 4 - 5 - 11*
  Se introduce 0 como valor de dias.
- *Camino 4: 0 - 1 - 2 - 1 - 4 - 5 - 6 - 7 - 5 - 11*
  La tabla de purchases está vacía, por lo que no se ejecuta el primer for interno ni el segundo.
- *Camino 5: 0 - 1 - 2 - 3 - 2 - 1 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 7 - 5 - 11*
  La tabla de purchases tiene más de un valor, la de usuarios tiene más de un valor y se introduce más de un día al método.
- *Camino 6: 0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 9 - 10*
  El if nunca se cumple.

2.3.2.5 Criterios de paso/Fallo

Una vez se introduzca un valor mayor o igual que 0, debería ejecutarse correctamente. Cuando se introduzca un valor menor, no debe producirse una excepción no controlada.

2.4 Pruebas sobre el controlador.

2.4.1 Nota

Los métodos *insertarVenta, logIn* están probados a través de la interfaz DAO, por lo que no se realizarán pruebas exhaustivas sobre este método a menos que se encuentren errores en las pruebas realizadas.

Los métodos insertarItem, insertarUsuario, importarUsuarios, importarProducto, importarCompra, getMedias, getHistogramas, getPorcentajes, getValoresBruto. getItermById, insertOrder cuentan con métodos análogos en otras interfaces. Por tanto, no se realizarán pruebas exhaustivas sobre ellos ya que se considera que estas han sido correctamente realizadas.

3 Casos de prueba de caja negra

3.1 Pruebas de importación de datos

3.1.1 Caso de prueba I-01-P-01: importarUsuarios

En este caso se valida la correcta interpretación y procesamiento de una línea que contenga todos los campos necesarios con valores correctos.

Necesidades del entorno	Entradas	Salidas	Clases que valida
En la base de datos no debe existir un usuario con el identificador *U-aaaaaa-000*.	U; U-aaaaaa-000; 10/10/2010; Samuel; Soutullo Sobral; 77013889E	Importación del usuario con ID *U-aaaaaa-000*.	1, 4, 6, 9, 12, 13, 15, 17

3.1.2 Caso de prueba I-01-P-02: importarUsuarios

En este caso se valida la correcta interpretación de una línea que contenga una cantidad de campos inferior a la correcta.

Necesidades del entorno	Entradas	Salidas	Clases que valida
En la base de datos no debe existir un usuario con el identificador *U-aaaaaa-000*.	U; U-aaaaaa-000; 10/10/2010; Samuel; Soutullo Sobral	Notificación de error por parte de la aplicación, abortando toda la importación. No se modifica la base de datos.	2

3.1.3 Caso de prueba I-01-P-03: importarUsuarios

En este caso se valida la correcta interpretación de una línea que contenga una cantidad de campos superior a la correcta.

Necesidades del entorno	Entradas	Salidas	Clases que valida
En la base de datos no debe existir un usuario con el identificador *U-aaaaaa-000*.	U; U-aaaaaa-000; 10/10/2010; Samuel; Soutullo Sobral; 77013889E; asdfg	Notificación de error por parte de la aplicación, abortando toda la importación. No se modifica la base de datos.	3

3.1.4 Caso de prueba I-01-P-04: importarUsuarios

En este caso se valida la correcta interpretación de una línea en blanco.

Necesidades del entorno	Entradas	Salidas	Clases que valida
En la base de datos no debe existir un usuario con el identificador *U-aaaaaa-000*.	Línea en blanco	No se notifica de ningún error, simplemente se salta la línea en blanco.	5

3.1.5 Caso de prueba I-01-P-05: importarCompra

En este caso se valida el correcto comportamiento de la aplicación cuando el método llamado no se corresponde con el tipo de líneas del archivo. En caso de que se detecte un fallo al ejecutar la prueba, se realizará para todas las combinaciones de métodos y tipos de línea.

Necesidades del entorno	Entradas	Salidas	Clases que valida
En la base de datos debe existir una compra con el identificador 77013889E.	U; U-aaaaaa-000; 10/10/2010; Samuel; Soutullo Sobral; 77013889E	La base de datos no se verá modificada dado que el archivo no contenía ninguna línea de venta. Opcionalmente, la aplicación notificará al usuario de la situación que se acaba de dar.	19

3.1.6 Caso de prueba I-01-P-06: importarCompra

En este caso se valida la correcta interpretación y procesamiento de una línea de venta cuando los identificadores de usuario e ítem que ésta referencia ya se encuentran en la base de datos.

Necesidades del entorno	Entradas	Salidas	Clases que valida
En la base de datos no existe la venta *V-aaaaaa-000*, pero sí el usuario *U-aaaaaa-000* y el ítem *I-aaaaaa-000*.	V; V-aaaaaa-000; 10/10/2010; U-aaaaaa-000; I-aaaaaa-000; 1; 1.53	Se importará correctamente la nueva venta. La base de datos se verá modificada, de forma que la nueva venta sea insertada, referenciando correctamente al usuario e ítem correspondientes.	21

3.1.7 Caso de prueba I-01-P-09: importarUsuarios

En este caso se valida la correcta interpretación de una línea que contiene un campo con un tamaño inferior al necesario. Si la prueba falla se realizarán pruebas sobre los campos *apellidos*, *categoría* y *descripción*, validando así a mayores las clases 7, 8,9, 10 y 11.

Necesidades del entorno	Entradas	Salidas	Clases que valida
Debe existir un usuario en la base de datos con identificador U-aaaaaa-000.	U; U-aaaaaa-000; 10/10/2010; ; Soutullo Sobral; 77013889E	La aplicación notificará del error correspondiente. La base de datos no se verá modificada.	7

3.1.8 Caso de prueba I-01-P-10: importarUsuarios

En este caso se valida la correcta interpretación de una línea que contiene un campo con un tamaño superior al necesario. Si la prueba falla se realizarán pruebas sobre los campos *apellidos*, *categoría* y *descripción*, validando así a mayores las clases 7, 8, 9, 10 y 11.

Necesidades del entorno	Entradas	Salidas	Clases que valida
Debe existir un usuario en la base de datos con identificador U-aaaaaa-000.	U; U-aaaaaa-000; 10/10/2010; aaaaa[…(x255)…]aaaa ; Soutullo Sobral; 77013889E	La aplicación notificará del error correspondiente. La base de datos no se verá modificada.	8

3.1.9 Caso de prueba I-01-P-11: importarUsuarios

En este caso se valida la correcta interpretación de una línea que contiene un campo de fecha con un formato incorrecto. Si la prueba falla se realizarán pruebas sobre los campos de fecha de todos los tipos de línea.

Necesidades del entorno	Entradas	Salidas	Clases que valida
Debe existir un usuario en la base de datos con identificador U-aaaaaa-000.	U; U-aaaaaa-000; 10/10/2010; aaaaa[…(x255)…]aaaa ; Soutullo Sobral; 77013889E	La aplicación notificará del error correspondiente. La base de datos no se verá modificada.	13

3.1.10 Caso de prueba I-01-P-12: importarCompra

En este caso se valida la correcta interpretación de una línea que contiene un campo de precio con un formato incorrecto. Si la prueba falla se realizarán pruebas sobre los campos de unidades y cantidad.

Necesidades del entorno	Entradas	Salidas	Clases que valida
Debe existir un usuario con identificador U-aaaaaa-000 y un *ítem* con identificador I-aaaaaa-000 en la base de datos.	V; V-aaaaaa-000; 10/10/2010; U-aaaaaa-000; I-aaaaaa-000; 1; 125ª	La aplicación notificará del error correspondiente. La base de datos no se verá modificada.	16

3.1.11 Caso de prueba I-01-P-13: importarUsuarios

En este caso se valida la correcta interpretación de una línea cuyo identificador ya se encuentra en la base de datos. Si la prueba falla, se efectuarán también pruebas para cubrir la clase 20.

Necesidades del entorno	Entradas	Salidas	Clases que valida
El usuario *U-aaaaaa-000* se encuentra previamente en la base de datos.	U; U-aaaaaa-000; 10/10/2010; Samuel; Soutullo Sobral; 77013889E	La aplicación notificará del error correspondiente. La base de datos no se verá modificada.	18

3.1.12 Caso de prueba I-01-P-14: importarCompra

En este caso se valida la correcta interpretación de una línea de venta cuando los identificadores de usuario e ítem que ésta referencia no se encuentran en la base de datos.

Necesidades del entorno	Entradas	Salidas	Clases que valida
Los identificadores de usuario (; U-trewq-000) e *ítem* (I- trewq-000) no se encuentran en la base de datos.	V; V-aaaaa-000; 10/10/2010; U- trewq-000; I- trewq-000; 1; 1.53	La aplicación notificará del error correspondiente. La base de datos no se verá modificada.	22

3.1.13 Caso de prueba I-01-P-15: importarCompra

En este caso se valida la correcta interpretación de un conjunto de líneas de venta cuando dichas líneas tienen el mismo valor para el campo *VRef*.

Necesidades del entorno	Entradas	Salidas	Clases que valida
La base de datos contiene el usuario *U-aaaaaa-000* y los productos *I-abcde-000* y *I-abcde-001*.	V; V-aaaaaa-000; 10/10/2010;U-aaaaaa-000; I-abcde-000; 1; 1.53 V; V-aaaaaa-000; 10/10/2010; U-aaaaaa-000; I-abcde-001; 1; 2.64	Se añadirá a la base de datos una venta correspondiente al usuario *U-aaaaaa-000*, con los productos *I-aaaaaa-000* y *I-aaaaaa-001*.	Este caso de prueba se ha obtenido en base a conjetura de errores.

3.1.14 Caso de prueba I-02-P-01: importarUsuarios

Se valida el correcto comportamiento del módulo de importación cuando se le pasa la ruta de un archivo existente. Si la prueba falla, se ejecutará contra todos los métodos de la interfaz.

Necesidades del entorno	Entradas	Salidas	Clases que valida
El siguiente fichero existe y es un fichero válido: /*home/usuario/fichero.csv*	Path: "/*home/usuario/fichero.csv*"	Se notificará al usuario del error. No se generará ninguna excepción sin controlar.	1, 3

3.1.15 Caso de prueba I-02-P-02: importarUsuarios

Valida el correcto comportamiento del módulo de importación cuando se le pasa la ruta de un archivo con un formato inválido.

Necesidades del entorno	Entradas	Salidas	Clases que valida
El siguiente fichero existe pero es un fichero inválido: /*home/usuario/fichero.csv*	Path: "!·$%&/()=?¿"	Se notificará al usuario del error. No se generará ninguna excepción sin controlar.	2

3.1.16 Caso de prueba I-02-P-03: importarUsuarios

Valida el correcto comportamiento del módulo de importación cuando se le pasa la ruta de un archivo que no existe.

Necesidades del entorno	Entradas	Salidas	Clases que valida
El siguiente fichero no existe: "*/home/usuario/ficheroInexistente.csv*".	Path: "/home/usuario/ficheroInexistente.csv".	Se notificará al usuario del error. No se generará ninguna excepción sin controlar.	4

3.2 Pruebas de lectura e inserción en la base de datos

3.2.1 Caso de prueba D-01-P-01: insertUser

Se valida la inserción correcta de un usuario.

Necesidades del entorno	Entradas	Salidas	Clases que valida
No debe existir un usuario con identificador U-abcde-000 en la BBDD.	Usuario: | U-abcde-000 | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |	Inserción del usuario con id: U-abcdef-000 en la base de datos.	1,3,6,9,10.

3.2.2 Caso de prueba D-01-P-02: insertUser

Se valida la inserción correcta de un usuario.

Necesidades del entorno	Entradas	Salidas	Clases que valida
No debe existir un usuario con igual identificador en la BBDD.	Usuario: | x | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |	Notificación de error por parte de la aplicación. Sin consecuencias en la base de datos.	2

3.2.3 Caso de prueba D-01-P-03: insertUser

Valida la inserción correcta de un usuario.

Necesidades del entorno	Entradas	Salidas	Clases que valida
No debe existir un usuario con igual identificador en la BBDD.	Usuario: |U-abcd-000 | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |	Notificación de error por parte de la aplicación. Sin consecuencias en la base de datos.	4

3.2.4 Caso de prueba D-01-P-04: insertUser

Valida la inserción correcta de un usuario.

Necesidades del entorno	Entradas	Salidas	Clases que valida

No debe existir un usuario con igual identificador en la BBDD.	Usuario: |U-abcde-0000 | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |	Notificación de error por parte de la aplicación. Sin consecuencias en la base de datos.	8

3.2.5 Caso de prueba D-01-P-05: insertUser

Valida la inserción correcta de un usuario.

Necesidades del entorno	Entradas	Salidas	Clases que valida
No debe existir un usuario con igual identificador en la BBDD.	Usuario: |U-abcdef-| Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |	Notificación de error por parte de la aplicación. Sin consecuencias en la base de datos.	10

3.2.6 Caso de prueba D-01-P-06: modUsuario

Valida la modificación correcta de un usuario.

Necesidades del entorno	Entradas	Salidas	Clases que valida
La base de datos solo cuenta con el usuario "U-aaaaa-000", nombre Limón Novoa, fecha 10-10-2010 y tipo estudiante.	Usuario: |U-abcde-000 | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |	En la base de datos el usuario cuyo id es U-aaaaa-000 pasa a tener el nombre Manuel Soutoullo.	12

3.2.7 Caso de prueba D-01-P-07: modUsuario

Valida la modificación correcta de un usuario.

Necesidades del entorno	Entradas	Salidas	Clases que valida
La base de datos solo cuenta con el usuario "U-aaaaa-000", nombre Limón Novoa, fecha 10-10-2010 y tipo estudiante.	Usuario: |U-ddddd-111 | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |	Se debe indicar que el usuario no existe y, por tanto, no se puede modificar. Como alternativa puede decidir insertar el nuevo usuario en la base.	13

3.2.8 Caso de prueba D-02-P-01: insertItem

Valida la inserción correcta de un item.

Necesidades del entorno	Entradas	Salidas	Clases que valida
No debe haber un *ítem* con el mismo identificador (I-abcde-000) en la base de datos.	Item: |I-abcde-000 | Robot limpiapiscinas | Limpia piscinas | Exteriores | 50 |10/10/2010|	Inserción del item con id: I-abcde-000 en la base de datos.	1

3.2.9 Caso de prueba D-02-P-02: updateItem

Valida la actualización correcta de un *item*.

Necesidades del entorno	Entradas	Salidas	Clases que valida
El único *item* que existe en la base de datos es |I-abcde-000 | Robot limpiapiscinas | Limpia piscinas | Exteriores | 50 |10/10/2010|	*Item*: |I-abcde-000 | Robot limpiapiscinas | Limpia piscinas de forma eficiente | Exteriores | 50 |10-10-2010|	Actualización correcta del *item* anterior con la descripción actualizada.	5

3.2.10 Caso de prueba D-02-P-03: insertItem

Valida la inserción correcta de un *item*.

Necesidades del entorno	Entradas	Salidas	Clases que valida
No debe existir un *ítem* con el mismo identificador en la base de datos.	Item: | x | Robot limpiapiscinas | Limpia piscinas de forma eficiente | Exteriores | 50 |10-10-2010|	Muestra un error y no se pierde la consistencia de la base de datos.	2

3.2.11 Caso de prueba D-02-P-04: insertVenta

Se prueba el método *insertOrder()* introduciendo un id de order incorrecto. Venta:

```
new Order() {
    ID_Order = "x",
    state = Order.WAITTING,
    user = new User() {
        ID_User = "U-abcde-000",
        name = "Usuario",
        surname = "Usuario1",
        NIF = "12213428H",
        date = "24-04-2017",
        tipe = User.PID
},
validator = "U-EFTGK-234",
lines = new[] {
      new Line() {
            quantity = 2,
            price = 19.99,
            item = new Item() {
                  itemRef = "I-abcde-000",
                  name = "producto",
                  description = "Descripción del producto",
                  category = "Cosas",
                  stock = 50,
                  availableDate = "01-01-1970"
            }
      }
}
```

Necesidades del entorno	Entradas	Salidas	Clases que valida
El usuario "U-abcde-000" y el producto "I-abcde-000" existen en la base de datos.	Item: | I-abcde-999 | Maletín de cuero | Maletín de portátil | Accesorios | 50 |10-10-2010 |	Muestra un error y no se pierde la consistencia de la base de datos.	4

3.2.12 Caso de prueba D-02-P-05: updateItem

Valida la actualización correcta de un item.

Necesidades del entorno	Entradas	Salidas	Clases que valida
El único item que existe en la base de datos es |I-abcde-000 | Robot limpiapiscinas | Limpia piscinas | Exteriores | 50 |10/10/2010|	Item: | I-abcde-999 | Maletín de cuero | Maletín de portátil | Accesorios | 50 |10-10-2010 |	Muestra un error indicando que el elemento a editar no existe o, alternativamente, inserta en la base el elemento nuevo.	6

3.2.13 Caso de prueba D-03-P-01: validateOrder

Comprueba la validación de un pedido.

```
new Purchase() {
ID_Purchase = "V-AAAAA-000",
order = new Order() {
      ID_Order = "O-AAAAA-000",
      state = Order.WAITTING,
      user = new User() {
            ID_User = "U-AAAAA-000",
            name = "Usuario",
            surname = "Usuario1",
            NIF = "12213428H",
            date = "24-04-2017",
            tipe = User.PID
        },
        validator = "U-EFTGK-234",
        lines = new[] {
                    new Line() {
                                quantity = 2,
                                price = 19.99,
                                item = new Item() {
                                            itemRef = "I-AAAAA-000",
                                            name = "producto",
                                            description = "Descripción del producto",
                                  category = "Cosas",
                                  stock = 50,
                                  availableDate = "01-01-1970"
                    }
                }
        }
        }
        date = "04-05-2017",
        discount = 0.2
},
        decision = true;
```

Necesidades del entorno	Entradas	Salidas	Clases que valida
Existencia del pedido en la base de datos.		El pedido pasa a ser una compra en la base de datos y se marca como aceptado.	1

3.2.14 Caso de prueba D-03-P-02: validateOrder

Comprueba la validación de un pedido.

```
new Purchase() {

ID_Purchase = "V-AAAAA-000",
order = new Order() {
      ID_Order = "O-AAAAA-000",
      state = Order.WAITTING,
```

```
user = new User() {
    ID_User = "U-AAAAA-000",
    name = "Usuario",
    surname = "Usuario1",
    NIF = "12213428H",
    date = "24-04-2017",
    tipe = User.PID
                },
                                                validator = "U-EFTGK-234",
                                                lines = new[] {
    new Line() {
quantity = 2,
price = 19.99,
item = new Item() {
    itemRef = "I-AAAAA-000",
    name = "producto",
    description = "Descripción del producto",
    category = "Cosas",
    stock = 50,
    availableDate = "01-01-1970"
                    }
                }
            }
        }
                                                date = "04-05-2017",
                                                discount = 0.2
            },
    decision = false;
```

Necesidades del entorno	Entradas	Salidas	Clases que valida
Existencia del pedido en la base de datos.		El pedido se marca como rechazado o bien se elimina de la base de datos	2

3.2.15 Caso de prueba D-04-P-01: getHistorialUsuario

Comprueba el correcto funcionamiento de la recuperación del historial de compra de un usuario.

Necesidades del entorno	Entradas	Salidas	Clases que valida
El usuario U-abcde-000 existe y tiene la compra introducida en el caso de prueba "caja blanca 4.5.1".	Id usuario: |U-abcde-000 | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |	Se recupera el historial de compras del usuario.	1,3

3.2.16 Caso de prueba D-04-P-02: getHistorialUsuario

Comprueba el correcto funcionamiento de la recuperación del historial de compra de un usuario.

Necesidades del entorno	Entradas	Salidas	Clases que valida
El usuario U-abcde-000 existe y tiene la compra introducida en el caso de prueba "caja blanca 4.5.1".	Id usuario: | U-a-000 | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |	Se muestra un error indicando que el id proporcionado no es válido.	2

3.2.17 Caso de prueba D-04-P-03: getHistorialUsuario

Comprueba el correcto funcionamiento de la recuperación del historial de compra de un usuario.

Necesidades del entorno	Entradas	Salidas	Clases que valida
El usuario U-abcde-000 existe y tiene la compra introducida en el caso de prueba "caja blanca 4.5.1".	Id usuario: | U-abcde-999 | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |	Se muestra un error indicando que el usuario no existe.	4

3.2.18 Caso de prueba D-04-P-04: getHistorialUsuario

Comprueba el correcto funcionamiento de la recuperación del historial de compra de un usuario.

Necesidades del entorno	Entradas	Salidas	Clases que valida
El usuario U-abcde-123 existe y no tiene compras en su historial.	Id usuario: | U-abcde-123 | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |	Se devuelve una lista vacía de compras de forma que sea entendible por el usuario que el historial está vacío. Si no se indica de ninguna forma que la operación ya ha acabado y que no hay resultados se considera que no pasa la prueba.	Este caso se infiere del estudio por valores límite.

3.2.19 Caso de prueba D-05-P-01: getItemById

Comprueba la recuperación de un Item.

Necesidades del entorno	Entradas	Salidas	Clases que valida
El único ítem existente en la base es: |I-abcde-000 | Robot limpiapiscinas | Limpia piscinas | Exteriores | 50 |10/10/2010|.	Id: I-abcde-000	Se devuelve |I-abcde-000 | Robot limpiapiscinas | Limpia piscinas | Exteriores | 50 |10/10/2010|	1, 3

3.2.20 Caso de prueba D-05-P-02: getItemById

Comprueba la recuperación de un Item.

Necesidades del entorno	Entradas	Salidas	Clases que valida
El único ítem existente en la base es: |I-abcde-000 | Robot limpiapiscinas | Limpia piscinas | Exteriores | 50 |10/10/2010|.	Id: I-s-000	La salida será una indicación de que el id es incorrecto.	2

3.2.21 Caso de prueba D-05-P-03: getItemById

Comprueba la recuperación de un Item.

Necesidades del entorno	Entradas	Salidas	Clases que valida
El único ítem existente en la base es: |I-abcde-000 | Robot limpiapiscinas | Limpia piscinas | Exteriores | 50 |10/10/2010|.	Id: I-abcde-999	La salida será una indicación de que el ítem no existe.	4

3.3 Pruebas sobre el módulo Estadistico

3.3.1 Nota

Dado que realizando todos los casos de prueba teniendo en cuenta las distintas entradas y métodos de la interfaz serían 12 casos, se ha decido probar un nº correcto de días sobre *getMedias*, un nº de días incorrecto sobre *getValoresBrutos*, dos modos correctos sobre *getPorcentajes* y un modo correcto y uno incorrecto sobre *getHistogramas*. Así, se realizarían 6 casos de prueba en lugar de 12.

3.3.2 Caso de prueba E-01-P-01: getHistogramas

Valida la consulta del histograma de ventas anuales.

Necesidades del entorno	Entradas	Salidas	Clases que valida
Deben existir en la base de datos el siguiente pedido, usuario, validador, compra e ítem con los siguientes datos: | U-AAAAA-000 | Nombre1 | Apellido1 | 11111111-A | 10-10-2010 | User.PID | | P-abcde-001 | aceptado | U-AAAAA-000 | O-AAAAA-000 | | I-abcde-001 | ItemPrueba | Esto es un ítem de prueba | categoria1 | 1 | 01-01-1970 | | V-abcde-001 | P-abcde-001 | fecha* | 0.0 | *Se introducirá una fecha 7 días anterior a la actual. Este dato se concreta de esta forma para que las pruebas puedan ser repetibles.	Modo: 1	La salida será un array de *int* con el nº de ventas para cada día del mes. La posición correspondiente al día 7 días anterior al actual deberá tener un 1 ya que hay una venta.	1

3.3.3 Caso de prueba E-01-P-02: getHistogramas

Valida la consulta del histograma de ventas semanales.

Necesidades del entorno	Entradas	Salidas	Clases que valida
Deben existir en la base de datos el siguiente pedido, usuario, validador, compra e ítem con los siguientes datos: | U-AAAAA-000 | Nombre1 | Apellido1 | 11111111-A | 10-10-2010 | User.PID | | P-abcde-001 | aceptado | U-AAAAA-000 | O-AAAAA-000 | | I-abcde-001 | ItemPrueba | Esto es un ítem de prueba | categoria1 | 1 | 01-01-1970 | | V-abcde-001 | P-abcde-001 | fecha* | 0.0 | *Se introducirá una fecha 7 días anterior a la actual. Este dato se concreta de esta forma para que las pruebas puedan ser repetibles.	Modo: 2	La salida será un array de *int* con el nº de ventas para cada semana del año. La posición correspondiente a la semana de la fecha introducida (que es una semana antes de la fecha actual) deberá tener un 1, ya que hay una venta.	3

3.3.4 Caso de prueba E-01-P-03: getPorcentajes

Valida la consulta de los porcentajes de las ventas mensuales.

Necesidades del entorno	Entradas	Salidas	Clases que valida
Deben existir en la base de datos el siguiente pedido, usuario, validador, compra e ítem con los siguientes datos: | U-AAAAA-000 | Nombre1 | Apellido1 | 11111111-A | 10-10-2010 | User.PID | | P-abcde-001 | aceptado | U-AAAAA-000 | O-AAAAA-000 | | I-abcde-001 | ItemPrueba | Esto es un ítem de prueba | categoria1 | 1 | 01-01-1970 | | V-abcde-001 | P-abcde-001 | fecha* | 0.0 | *Se introducirá una fecha 7 días anterior a la actual. Este dato se concreta de esta forma para que las pruebas puedan ser repetibles.	Modo: 2	La salida será un *array* de *floats* con los porcentajes de ventas semanales para cada semana del año, mostrando un 100.0 en la semana previa a la actual, ya que sólo existe una venta en la BBDD que data de esa fecha.	3

3.3.5 Caso de prueba E-01-P-04: getPorcentajes

Valida la consulta de los porcentajes de las ventas en un modo incorrecto.

Necesidades del entorno	Entradas	Salidas	Clases que valida
No es necesario ningún prerrequisito.	Modo: 0	La salida será una excepción.	2

3.3.6 Caso de prueba E-02-P-01: getValoresBrutos

Valida la consulta de ventas totales en una cantidad de días dados.

Necesidades del entorno	Entradas	Salidas	Clases que valida
No es necesario ningún prerrequisito.	Días: -3	La salida esperada será una excepción.	2

3.3.7 Caso de prueba E-02-P-02: getMedias

Valida la consulta de la media de ventas en un conjunto de días dado, en este caso 5.

Necesidades del entorno	Entradas	Salidas	Clases que valida
Deben existir en la base de datos el siguiente pedido, usuario, validador, compra e ítem con los siguientes datos: | U-AAAAA-000 | Nombre1 | Apellido1 | 11111111-A | 10-10-2010 | User.PID | | P-abcde-001 | aceptado | U-AAAAA-000 | O-AAAAA-000 | | I-abcde-001 | ItemPrueba | Esto es un ítem de prueba | categoria1 | 1 | 01-01-1970 | | V-abcde-001 | P-abcde-001 | fecha* | 0.0 | *Se introducirá una fecha 7 días anterior a la actual. Este dato se concreta de esta forma para que las pruebas puedan ser repetibles.	Días: 5	La salida será la media de de los 5 días previos a la fecha actual. Debe devolver 0.0 ya que la venta se ha introducido 7 días antes de la fecha actual.	1

4 Casos de prueba de caja blanca

4.1 importarUsuarios

4.1.1 Camino 1

Necesidades del entorno	Entrada	Salida
Existe el fichero *users.csv*.	*path: "users.csv"*	El método devuelve *1* y el usuario se importa.
Se tienen permisos de lectura sobre el fichero *users.csv*.		
El fichero contiene la siguiente línea: *U; U-AAAAA-000; 10/10/2010; Samuel; Soutullo Sobral; 12345678E*		

4.1.2 Camino 2

Necesidades del entorno	Entrada	Salida
No existe el fichero *users.csv*	*path:"users.csv"*	El método devuelve *-1*.

4.1.3 Camino 3

Necesidades del entorno	Entrada	Salida
Existe el fichero *users.csv*	*path:"users.csv"*	El método devuelve *0* y se imprime por pantalla un mensaje de error.
No se tienen permisos de lectura sobre el fichero *users.csv*		

4.1.4 Camino 4

Necesidades del entorno	Entrada	Salida
Existe el fichero *users.csv*	*path: "users.csv"*	El método devuelve *0* y se imprime por pantalla un mensaje de error.
Se bloqueará el fichero *users.csv* justo antes de que se ejecute la línea *lectura.close();*		

4.1.5 Camino 6

Necesidades del entorno	Entrada	Salida
Existe el fichero *users.csv*.	*path: "users.csv"*	El método devuelve *0* y no se realiza ningún cambio en la base de datos.
Se tienen permisos de lectura sobre el fichero *users.csv*.		
El fichero contiene la siguiente línea: *A; U-AAAAA-000; 10/10/2010; Samuel; Soutullo Sobral; 12345678E*		

4.1.6 Camino 6

Necesidades del entorno	Entrada	Salida
Existe el fichero *users.csv*.	*path: "users.csv"*	El método devuelve *0* y no se realiza ningún cambio en la base de datos.
Se tienen permisos de lectura sobre el fichero *users.csv*.		
El fichero contiene la siguiente línea*: U; U-AAAAA-00; 10/10/2010; Samuel; Soutullo Sobral; 12345678E*		

4.1.7 Camino 7

Necesidades del entorno	Entrada	Salida
Existe el fichero *users.csv*.	*path: "users.csv"*	El método devuelve *0* y no se realiza ningún cambio en la base de datos.
Se tienen permisos de lectura sobre el fichero *users.csv*.		
El fichero contiene la siguiente línea*: U; U-AAAAA-000; 3/10/2010; Samuel; Soutullo Sobral; 12345678E*		

4.1.8 Camino 8

Necesidades del entorno	Entrada	Salida
Existe el fichero *users.csv*.	*path: "users.csv"*	El método devuelve *0* y no se realiza ningún cambio en la base de datos.
Se tienen permisos de lectura sobre el fichero *users.csv*.		
El fichero contiene la siguiente línea*: U; U-AAAAA-000; 10/10/2010;; Soutullo Sobral; 12345678E*		

4.1.9 Camino 9

Necesidades del entorno	Entrada	Salida
Existe el fichero *users.csv*.	*path: "users.csv"*	El método devuelve *0* y no se realiza ningún cambio en la base de datos.
Se tienen permisos de lectura sobre el fichero *users.csv*.		
El fichero contiene la siguiente línea*: U; U-AAAAA-000; 10/10/2010; Samuel;; 12345678E*		

4.1.10 Camino 10

Necesidades del entorno	Entrada	Salida
Existe el fichero *users.csv*.	*path: "users.csv"*	El método devuelve *0* y no se realiza ningún cambio en la base de datos.
Se tienen permisos de lectura sobre el fichero *users.csv*.		
El fichero contiene la siguiente línea: *U; U-AAAAA-000; 10/10/2010; Samuel; Soutullo Sobral; 12345678*		

4.2 importaCompra

4.2.1 Camino 1

Necesidades del entorno	Entrada	Salida
Existe el fichero *sells.csv*.	*path: "sells.csv"*	El método devuelve *1* y la compra se importa.
Se tienen permisos de lectura sobre el fichero *sells.csv*.		
El usuario *U-AAAAA-000* existe en la base de datos.		
El ítem I-AAAAA-000 existe en la base de datos.		
El fichero contiene la siguiente línea*: V; V-AAAAA-000; 10/10/2010; U-AAAAA-000; I-AAAAA-000; 1; 10*		

4.2.2 Camino 2

Necesidades del entorno	Entrada	Salida
No existe el fichero *sells.csv*.	*path: "sells.csv"*	El método devuelve *-1*.

4.2.3 Camino 3

Necesidades del entorno	**Entrada**	**Salida**
Existe el fichero *sells.csv*.	*path: "sells.csv"*	El método devuelve *0* y se imprime por pantalla un mensaje de error.
Ne tienen permisos de lectura sobre el fichero *sells.csv*.		

4.2.4 Camino 4

Necesidades del entorno	**Entrada**	**Salida**
Existe el fichero *sells.csv*.	*path: "sells.csv"*	El método devuelve *0* y se imprime por pantalla un mensaje de error.
Se bloqueará el fichero *sells.csv* justo antes de que se ejecute *lectura.close();*		

4.2.5 Camino 5

Necesidades del entorno	**Entrada**	**Salida**
Existe el fichero *sells.csv*.	*path: "sells.csv"*	El método devuelve *0* y no se realiza ningún cambio en la base de datos.
Se tienen permisos de lectura sobre el fichero *sells.csv*.		
El fichero contiene la siguiente línea: *A; V-AAAAA-000; 10/10/2010; U-AAAAA-000; I-AAAAA-000; 1; 10*		

4.2.6 Camino 6

Necesidades del entorno	**Entrada**	**Salida**
Existe el fichero *sells.csv*.	*path: "sells.csv"*	El método devuelve *0* y no se realiza ningún cambio en la base de datos.
Se tienen permisos de lectura sobre el fichero *sells.csv*.		
El fichero contiene la siguiente línea: *V; V-AAAAA-00; 10/10/2010; U-AAAAA-000; I-AAAAA-000; 1; 10*		

4.2.7 Camino 7

Necesidades del entorno	**Entrada**	**Salida**
Existe el fichero *sells.csv*.	*path: "sells.csv"*	El método devuelve *0* y no se realiza ningún cambio en la base de datos.
Se tienen permisos de lectura sobre el fichero *sells.csv*.		
El fichero contiene la siguiente línea: *V; V-AAAAA-000; 3/10/2010; U-AAAAA-000; I-AAAAA-000; 1; 10*		

4.2.8 Camino 8

Necesidades del entorno	**Entrada**	**Salida**
Existe el fichero *sells.csv*.	*path: "sells.csv"*	El método devuelve *0* y no se realiza ningún cambio en la base de datos.
Se tienen permisos de lectura sobre el fichero *sells.csv*.		
El fichero contiene la siguiente línea: *V; V-AAAAA-000; 10/10/2010; U-AAAAA-00; I-AAAAA-000; 1; 10*		

4.2.9 Camino 9

Prerrequisitos	**Entrada**	**Salida**
Existe el fichero *sells.csv*.	*path: "sells.csv"*	El método devuelve *0* y no se realiza ningún cambio en la base de datos.
Se tienen permisos de lectura sobre el fichero *sells.csv*.		

El fichero contiene la siguiente línea: *V; V-AAAAA-000; 10/10/2010; U-AAAAA-000; I-AAAAA-00; 1; 10*		

4.2.10 Camino 10

Prerrequisitos	Entrada	Salida
Existe el fichero *sells.csv*.	*path: "sells.csv"*	El método devuelve *0* y no se realiza ningún cambio en la base de datos.
Se tienen permisos de lectura sobre el fichero *sells.csv*.		
El fichero contiene la siguiente línea: *V; V-AAAAA-000; 10/10/2010; U-AAAAA-00; I-AAAAA-000;; 10*		

4.2.11 Camino 11

Prerrequisitos	Entrada	Salida
Existe el fichero *sells.csv*.	*path: "sells.csv"*	El método devuelve *0* y no se realiza ningún cambio en la base de datos.
Se tienen permisos de lectura sobre el fichero *sells.csv*.		
El fichero contiene la siguiente línea: *V; V-AAAAA-000; 10/10/2010; U-AAAAA-00; I-AAAAA-000;1;*		

4.3 validateOrder

4.3.1 Camino 1

Prerrequisitos	Entrada	Salida

No tiene.	`new Purchase() {`
	`ID_Purchase = "V-AAAAA-000",`
	`order = new Order() {`
	`ID_Order = "O-AAAAA-000",`
	`state = Order.WAITTING,`
	`user = new User() {`
	`ID_User = "U-AAAAA-000",`
	`name = "Usuario",`
	`surname = "Usuario1",`
	`NIF = "12213428H",`
	`date = "24-04-2017",`
	`tipe = User.PID},`
	`validator = "U-EFTGK-234",`
	`lines = new[] {`
	`new Line() {`
	`quantity = 2,`
	`price = 19.99,`
	`item = new Item() {`
	`itemRef = "I-AAAAA-000",`
	`name = "producto",`
	`description = "Descripción del producto",`
	`category = "Cosas",`
	`stock = 50,`
	`availableDate = "01-01-1970"}`
	`}`
	`}`
	`}`
	`date = "04-05-2017",`
	`discount = 0.2`
	`},`
	`decision = true;`
		El método devuelve true e inserta lo anterior en la base, marcando el pedido O-AAAAA-000 como aceptado.

4.3.2 Camino 2

Prerrequisitos	Entrada	Salida

No tiene.	new Purchase() {
	ID_Purchase = "V-AAAAA-000",
	order = new Order() {
	ID_Order = "O-AAAAA-000",
	state = Order.WAITTING,
	user = new User() {
	ID_User = "U-AAAAA-000",
	name = "Usuario",
	surname = "Usuario1",
	NIF = "12213428H",
	date = "24-04-2017",
	tipe = User.PID},
	validator = "U-EFTGK-234",
	lines = new[] {
	new Line() {
	quantity = 2,
	price = 19.99,
	item = new Item() {
	itemRef = "I-AAAAA-000",
	name = "producto",
	description = "Descripción del producto",
	category = "Cosas",
	stock = 50,
	availableDate = "01-01-1970" }
	}
	}
	}
	date = "04-05-2017",
	discount = 0.2
	},
	decision = false;
		El método devuelve false e inserta lo anterior en la base, marcando el pedido O-AAAAA-000 como rechazado.

4.3.3 Camino 3

Prerrequisitos	Entrada	Salida

El servidor de la base de datos está desconectado	new Purchase() {
	ID_Purchase = "V-AAAAA-000",
	order = new Order() {
	ID_Order = "O-AAAAA-000",
	state = Order.WAITTING,
	user = new User() {
	ID_User = "U-AAAAA-000",
	name = "Usuario",
	surname = "Usuario1",
	NIF = "12213428H",
	date = "24-04-2017",
	tipe = User.PID},
	validator = "U-EFTGK-234",
	lines = new[] {
	new Line() {
	quantity = 2,
	price = 19.99,
	item = new Item() {
	itemRef = "I-AAAAA-000",
	name = "producto",
	description = "Descripción del producto",
	category = "Cosas",
	stock = 50,
	availableDate = "01-01-1970"}
	}
	}
	}
	date = "04-05-2017",
	discount = 0.2
	},
	decision = true;
		El método devuelve false sin modificar la base de datos.

4.3.4 Camino 4

Prerrequisitos	Entrada	Salida

La compra V-AAAAA-000 ya existe en la base de datos.	new Purchase() {
	ID_Purchase = "V-AAAAA-000",
	order = new Order() {
	ID_Order = "O-AAAAA-000",
	state = Order.WAITTING,
	user = new User() {
	ID_User = "U-AAAAA-000",
	name = "Usuario",
	surname = "Usuario1",
	NIF = "12213428H",
	date = "24-04-2017",
	tipe = User.PID},
	validator = "U-EFTGK-234",
	lines = new[] {
	new Line() {
	quantity = 2,
	price = 19.99,
	item = new Item() {
	itemRef = "I-AAAAA-000",
	name = "producto",
	description = "Descripción del producto",
	category = "Cosas",
	stock = 50,
	availableDate = "01-01-1970"}
	}
	}
	}
	date = "04-05-2017",
	discount = 0.2
	},
	decision = true
		El método devuelve false sin modificar la base de datos.

4.3.5 Camino 5

Prerrequisitos	Entrada	Salida

El pedido O-AAAAA-000 no existe en la base de datos.	new Purchase() {
	ID_Purchase = "V-AAAAA-000",
	order = new Order() {
	ID_Order = "O-AAAAA-000",
	state = Order.WAITTING,
	user = new User() {
	ID_User = "U-AAAAA-000",
	name = "Usuario",
	surname = "Usuario1",
	NIF = "12213428H",
	date = "24-04-2017",
	tipe = User.PID},
	validator = "U-EFTGK-234",
	lines = new[] {
	new Line() {
	quantity = 2,
	price = 19.99,
	item = new Item() {
	itemRef = "I-AAAAA-000",
	name = "producto",
	description = "Descripción del producto",
	category = "Cosas",
	stock = 50,
	availableDate = "01-01-1970"}
	}
	}
	}
	date = "04-05-2017",
	discount = 0.2
	},
	decision = true
		El método devuelve false sin modificar la base de datos.

4.4 getAllUsers

4.4.1 Camino 1

Prerrequisitos	Entrada	Salida
El usuario U-AAAAA-000 existe en la base de datos.	No tiene.	El método devuelve un ArrayList no vacio.

4.4.2 Camino 2

Prerrequisitos	Entrada	Salida
El servidor de base de datos está desconectado.	No tiene.	El método devuelve un ArrayList vacio.

4.4.3 Camino 3

Prerrequisitos	Entrada	Salida
La tabla Usuario de la base de datos debe estar vacía.	No tiene.	El método devuelve un ArrayList vacio.

4.4.4 Camino 4

Prerrequisitos	Entrada	Salida
El servidor de base de datos está conectado al principio de la ejecución, pero se desconecta antes de cerrar la conexión.	No tiene.	El método devuelve *-1*.

4.5 insertOrder

4.5.1 Camino 1

Prerrequisitos	Entrada	Salida

No tiene.	new Order() {
	ID_Order = "O-AAAAA-000",
	state = Order.WAITTING,
	user = new User() {
	ID_User = "U-AAAAA-000",
	name = "Usuario",
	surname = "Usuario1",
	NIF = "12213428H",
	date = "24-04-2017",
	tipe = User.PID
	},
	validator = "U-EFTGK-234",
	lines = new[] {
	new Line() {
	quantity = 2,
	price = 19.99,
	item = new Item() {
	itemRef = "I-AAAAA-000",
	name = "producto",
	description = "Descripción del producto",
	category = "Cosas",
	stock = 50,
	availableDate = "01-01-1970"
	}
	}
	}
	}
		El método devuelve true e inserta lo anterior en la base.

4.5.2 Camino 2

Prerrequisitos	Entrada	Salida

El usuario U-aaaaa-999 no existe en la base de datos.	new Order() {
	ID_Order = "O-AAAAA-000",
	state = Order.WAITTING,
	user = new User() {
	ID_User = "U-aaaaa-999",
	name = "Usuario",
	surname = "Usuario1",
	NIF = "12213428H",
	date = "24-04-2017",
	tipe = User.PID
	},
	validator = "U-EFTGK-234",
	lines = new[] {
	new Line() {
	quantity = 2,
	price = 19.99,
	item = new Item() {
	itemRef = "I-AAAAA-000",
	name = "producto",
	description = "Descripción del producto",
	category = "Cosas",
	stock = 50,
	availableDate = "01-01-1970"
	}
	}
	}
	}
		El método devuelve false sin modificar la base de datos.

4.5.3 Camino 3

Prerrequisitos	Entrada	Salida

No tiene.	new Order() {
	ID_Order = "O-AAAAA-000",
	state = Order.WAITTING,
	user = new User() {
	ID_User = "U-AAAAA-000",
	name = "Usuario",
	surname = "Usuario1",
	NIF = "12213428H",
	date = "24-04-2017",
	tipe = User.PID
	},
	validator = "U-EFTGK-234",
	lines = new[] { }
	}
		El método devuelve false sin modificar la base de datos.

4.5.4 Camino 4

Prerrequisitos	Entrada	Salida

El producto I-aaaaa-999 no existe en la base de datos.	new Order() {
	ID_Order = "O-AAAAA-000",
	state = Order.WAITTING,
	user = new User() {
	ID_User = "U-AAAAA-000",
	name = "Usuario",
	surname = "Usuario1",
	NIF = "12213428H",
	date = "24-04-2017",
	tipe = User.PID
	},
	validator = "U-EFTGK-234",
	lines = new[] {
	new Line() {
	quantity = 2,
	price = 19.99,
	item = new Item() {
	itemRef = "I-aaaaa-999",
	name = "producto",
	description = "Descripción del producto",
	category = "Cosas",
	stock = 50,
	availableDate = "01-01-1970"
	}
	}
	}
	}
		El método devuelve false sin modificar la base de datos.

4.6 getPorcentajes

4.6.1 Camino 1

Prerrequisitos	Entrada	Salida

Debe existir el usuario
{ID_User = "U-AAAAA-001",
name = "Usuaria",
surname = "Usuario1",
NIF = "12213428H",
date = "24-04-2017",
tipe = User.PID} con
{"V-AAAAA-000", "P-AAAAA-000"{{"P-AAAAA-000","aceptado","U-AAAAA-001","U-EFTGK-234",lines = new[] {
new Line() {
quantity = 2,
price = 19.99,
item = new Item() {
itemRef = "I-AAAAA-000",
name = "producto",
	mode=1	int[30]={0.00, 0.0, 0.0,...,0,100.0}

description = "Descripción del producto", category = "Cosas", stock = 50, availableDate = "01-01-1970" } } }}}, CURDATE(), 0.0}		

4.6.2 Camino 2

Prerrequisitos	Entrada	Salida
Debe existir el usuario {ID_User = "U-AAAAA-001", name = "Usuaria", surname = "Usuario1", NIF = "12213428H", date = "24-04-2017", tipe = User.PID} con la tabla de Compras vacia.	mode=1	int[30]={0.0,0.0,...,0.0}

4.6.3 Camino 3

Prerrequisitos	Entrada	Salida
Debe existir el usuario {ID_User = "U-AAAAA-001", name = "Usuaria", surname = "Usuario1", NIF = "12213428H", date = "24-04-2017", tipe = User.PID} con {"V-AAAAA-000", "P-AAAAA-000"{{"P-AAAAA-000","aceptado","U-AAAAA-001","U-EFTGK-234",lines = new[] { new Line() { quantity = 2, price = 19.99, item = new Item() { itemRef = "I-AAAAA-000", name = "producto", description = "Descripción del producto", category = "Cosas", stock = 50, availableDate = "01-01-1970" } } }}},DATE_ADD (CURDATE(), INTERVAL -1 DAY), 0.0}	mode=2	int[52]={0.0,0.0,...,0.0}

4.6.4 Camino 4

Prerrequisitos	Entrada	Salida
No debe haber compras en la tabla compras ninguna semana en el ultimo año.	mode=2	int[52]={0.0,0.0,...,100.0}

4.6.5 Camino 5

Prerrequisitos	Entrada	Salida
Ninguno.	mode=3	null

4.7 getMedias

4.7.1 Camino 1

Prerrequisitos	Entrada	Salida
Debe existir el usuario { `ID_User = "U-AAAAA-001", name = "Usuaria", surname = "Usuario1", NIF = "12213428H", date = "24-04-2017", tipe = User.PID} con {"V-AAAAA-000", "P-AAAAA-000"{{"P-AAAAA-000","aceptado","U-AAAAA-001","U-EFTGK-234",lines = new[] { new Line() { quantity = 2, price = 19.99, item = new Item() { itemRef = "I-AAAAA-000", name = "producto", description = "Descripción del producto", category = "Cosas", stock = 50, availableDate = "01-01-1970" } } }}}, CURDATE(), 0.0}`	dias=1	El método devuelve un float con valor 1.0.

4.7.2 Camino 2

Prerrequisitos	Entrada	Salida
La tabla Usuario de la base de datos debe estar vacía.	dias=1	El método devuelve un float igual a 0.0.

4.7.3 Camino 3

Prerrequisitos	Entrada	Salida
Debe existir el usuario: `{ID_User = "U-AAAAA-001", name = "Usuaria", surname = "Usuario1", NIF = "12213428H", date = "24-04-2017", tipe = User.PID} con {"V-AAAAA-000", "P-AAAAA-000"{{"P-AAAAA-000","aceptado","U-AAAAA-001","U-EFTGK-234",lines = new[] { new Line() { quantity = 2, price = 19.99, item = new Item() { itemRef = "I-AAAAA-000", name = "producto", description = "Descripción del producto", category = "Cosas", stock = 50, availableDate = "01-01-1970" } } }`	dias=0	El método devuelve un float igual a 0.0.

}}}, CURDATE(), 0.0} dias=0 El método devuelve un float igual a 0.0.		

4.7.4 Camino 4

Prerrequisitos	Entrada	Salida
Debe existir el usuario { ID_User = "U-AAAAA-001", name = "Usuaria", surname = "Usuario1", NIF = "12213428H", date = "24-04-2017", tipe = User.PID} con la tabla de Compras vacia	dias=1	El método devuelve un float igual a 0.0.

4.7.5 Camino 5

Prerrequisitos	Entrada	Salida

Deben existir cinco usuarios
 	
({ ID_User = "U-AAAAA-000",
name = "Usuario",
surname = "Usuario1",
NIF = "12213428H",
date = "24-04-2017",
tipe = User.PID
}, {ID_User = "U-AAAAA-001",
name = "Usuaria",
surname = "Usuario1",
NIF = "12213428H",
date = "24-04-2017",
tipe = User.PID},
{ID_User = "U-AAAAA-002",
name = "Usuarioa",
surname = "Usuario1",
NIF = "12213428H",
date = "24-04-2017",
tipe = User.PID
}, {ID_User = "U-AAAAA-003",
name = "Usuarix",
surname = "Usuario1",
NIF = "12213428H",
date = "24-04-2017",
tipe = User.PID
}, { ID_User = "U-AAAAA-004",
name = "Usuarie",
surname = "Usuario1",
NIF = "12213428H",
date = "24-04-2017",
tipe = User.PID
}) y U-AAAAA-000 debe tener cero compras asociadas.
U-AAAAA-001 debe tener {"V-AAAAA-000", "P-AAAAA-000"{"P-AAAAA-000","aceptado","U-AAAAA-001","U-EFTGK-234"}, DATE_ADD (CURDATE(), INTERVAL -1 DAY), 0.0}
 	
U-AAAAA-002 debe tener {"V-AAAAA-001", P-AAAAA-001{"P-AAAAA-001","aceptado","U-AAAAA-002","U-EFTGK-234",lines = new[] {
new Line() {
quantity = 2,
price = 19.99,
item = new Item() {
itemRef = "I-AAAAA-000",
name = "producto",
description = "Descripción del producto",
	Días=5	El método devuelve un float distinto de 0.8.

category = "Cosas", stock = 50, availableDate = "01-01-1970" } } } }, DATE_ADD (CURDATE(), INTERVAL -1 DAY), 0.0} y {"V-AAAAA-002", P-AAAAA-002{{"P-AAAAA-002","aceptado","U-AAAAA-002","U-EFTGK-234",lines = new[] { new Line() { quantity = 2, price = 19.99, item = new Item() { itemRef = "I-AAAAA-000", name = "producto", description = "Descripción del producto", category = "Cosas", stock = 50, availableDate = "01-01-1970" } } } }}, DATE_ADD (CURDATE(), INTERVAL -1 DAY), 0.0} U-AAAAA-003 debe tener {"V-AAAAA-003","P-AAAAA-003"{"P-AAAAA-003","aceptado","U-AAAAA-003","U-EFTGK-234",lines = new[] { new Line() { quantity = 2, price = 19.99, item = new Item() { itemRef = "I-AAAAA-000", name = "producto", description = "Descripción del producto", category = "Cosas", stock = 50, availableDate = "01-01-1970" } } } }, DATE_ADD (CURDATE(), INTERVAL -3 DAY), 0.0} U-AAAAA-004 debe tener {"V-AAAAA-004", "P-AAAAA-004"{"P-AAAAA-004","aceptado","U-AAAAA-004","U-EFTGK-234"}, DATE_ADD (CURDATE(), INTERVAL -6 DAY), 0.0}		

4.7.6 Camino 6

Prerrequisitos	Entrada	Salida
Debe existir el usuario {ID_User = "U-AAAAA-001", name = "Usuaria", surname = "Usuario1", NIF = "12213428H", date = "24-04-2017",	dias=1	El método devuelve un float igual a 0.0.

tipe = User.PID} con la compra {"V-AAAAA-000", "P-AAAAA-000"{"P-AAAAA-000","aceptado","U-AAAAA-001","U-EFTGK-234"lines = new[] {
new Line() {
quantity = 2,
price = 19.99,
item = new Item() {
itemRef = "I-AAAAA-000",
name = "producto",
description = "Descripción del producto",
category = "Cosas",
stock = 50,
availableDate = "01-01-1970"
}
}
}
}, DATE_ADD (CURDATE(), INTERVAL -2 DAY), 0.0}		

4.8 getValoresBrutos

4.8.1 Camino 1

Prerrequisitos	Entrada	Salida

Debe existir el usuario
{ID_User = "U-AAAAA-001",
name = "Usuaria",
surname = "Usuario1",
NIF = "12213428H",
date = "24-04-2017",
tipe = User.PID} con {"V-AAAAA-000", "P-AAAAA-000"{{"P-AAAAA-000","aceptado","U-AAAAA-001","U-EFTGK-234",lines = new[] {
new Line() {
quantity = 2,
price = 19.99,
item = new Item() {
itemRef = "I-AAAAA-000",
name = "producto",
description = "Descripción del producto",
category = "Cosas",
stock = 50,
availableDate = "01-01-1970"
}
}
}}}, CURDATE(), 0.0}
	dias=1	int[1]={1};

4.8.2 Camino 2

Prerrequisitos	Entrada	Salida
La tabla Usuario de la base de datos debe estar vacía.	dias=1	El método devuelve un int[1]={0}.

4.8.3 Camino 3

Prerrequisitos	Entrada	Salida
Debe existir el usuario {	dias=0	El método devuelve un int[0].

ID_User = "U-AAAAA-001",
name = "Usuaria",
surname = "Usuario1",
NIF = "12213428H",
date = "24-04-2017",
tipe = User.PID} con
{"V-AAAAA-000", "P-AAAAA-000"{{"P-AAAAA-000","aceptado","U-AAAAA-001","U-EFTGK-234",lines = new[] {
new Line() {
quantity = 2,
price = 19.99,
item = new Item() {
itemRef = "I-AAAAA-000",
name = "producto",
description = "Descripción del producto",
category = "Cosas",
stock = 50,
availableDate = "01-01-1970"
}
}
}}}, CURDATE(), 0.0}		

4.8.4 Camino 4

Prerrequisitos	Entrada	Salida
Debe existir el usuario { ID_User = "U-AAAAA-001", name = "Usuaria", surname = "Usuario1", NIF = "12213428H", date = "24-04-2017", tipe = User.PID} con la tabla de Compras vacia	dias=1	int[1]={0};

4.8.5 Camino 5

Prerrequisitos	Entrada	Salida
Deben existir cinco usuarios ({ ID_User = "U-fffff-000", name = "Usuario", surname = "Usuario1", NIF = "12213428H", date = "24-04-2017", tipe = User.PID }, {ID_User = "U-fffff-001", name = "Usuaria", surname = "Usuario1", NIF = "12213428H", date = "24-04-2017", tipe = User.PID}, {ID_User = "U-fffff-002", name = "Usuarioa", surname = "Usuario1", NIF = "12213428H", date = "24-04-2017", tipe = User.PID }, {ID_User = "U-fffff-003", name = "Usuarix", surname = "Usuario1", NIF = "12213428H", date = "24-04-2017", tipe = User.PID }, { ID_User = "U-fffff-004", name = "Usuarie", surname = "Usuario1", NIF = "12213428H", date = "24-04-2017", tipe = User.PID	Días=5	int[5]={0,3,0,1,0}

}) y U-fffff-000 debe tener cero compras asociadas.
U-fffff-001 debe tener {"V-fffff-000", "P-fffff-000"{"P-fffff-000","aceptado","U-f -001","U-EFTGK-234"}, DATE_ADD (CURDATE(), INTERVAL -1 DAY), 0.0}
 	
U-fffff-002 debe tener {"V-fffff-001", P-fffff-001{"P-fffff-001","aceptado","U-fffff-002","U-EFTGK-234",lines = new[] {
new Line() {
quantity = 2,
price = 19.99,
item = new Item() {
itemRef = "I-fffff-000",
name = "producto",
description = "Descripción del producto",
category = "Cosas",
stock = 50,
availableDate = "01-01-1970"
}
}
}
}, DATE_ADD (CURDATE(), INTERVAL -1 DAY), 0.0} y {"V-fffff-002", P-fffff-002{{"P-fffff-002","aceptado","U-fffff-002","U-EFTGK-234",lines = new[] {
new Line() {
quantity = 2,
price = 19.99,
item = new Item() {
itemRef = "I-fffff-000",
name = "producto",
description = "Descripción del producto",
category = "Cosas",
stock = 50,
availableDate = "01-01-1970"
}
}
}
}}, DATE_ADD (CURDATE(), INTERVAL -1 DAY), 0.0}
 	
U-fffff-003 debe tener {"V-fffff-003","P-fffff-003"{"P-fffff-003","aceptado","U-fffff-003","U-EFTGK-234",lines = new[] {
new Line() {
quantity = 2,
price = 19.99,
item = new Item() {
itemRef = "I-fffff-000",
name = "producto",
description = "Descripción del producto",
category = "Cosas",
stock = 50,
availableDate = "01-01-1970"
}
}
}
}, DATE_ADD (CURDATE(), INTERVAL -3 DAY), 0.0}		

U-fffff-004 debe tener {"V-fffff-004", "P-fffff-004"{"P-fffff-004","aceptado","U-fffff-004","U-EFTGK-234"}, DATE_ADD (CURDATE(), INTERVAL -6 DAY), 0.0}		

4.8.6 Camino 6

Prerrequisitos	Entrada	Salida

Debe existir el usuario {
ID_User = "U-AAAAA-001",
name = "Usuaria",
surname = "Usuario1",
NIF = "12213428H",
date = "24-04-2017",
tipe = User.PID} con la compra {"V-AAAAA-000", "P-AAAAA-000"{"P-AAAAA-000","aceptado","U-AAAAA-001","U-EFTGK-234"lines = new[] {
new Line() {
quantity = 2,
price = 19.99,
item = new Item() {
itemRef = "I-AAAAA-000",
name = "producto",
description = "Descripción del producto",
category = "Cosas",
stock = 50,
availableDate = "01-01-1970"
}
}
}
}, DATE_ADD (CURDATE(), INTERVAL -2 DAY), 0.0}
	dias=1	int[1]={0}

5 Procedimientos de prueba

Se creó una Suite de tests con 11 clases de tests distintos para establecer un orden correcto de dependencias entre los 57 tests implementados.

Lo primero en ser probado es el módulo estadístico, separado en 4 bloques de ejecución distintos. En primer lugar se ejecutan aquellos tests que dependen de tener una base de datos vacía o con usuarios pero no compras. En segundo lugar, se inserta una sola compra para el usuario insertado anteriormente. Posteriormente se ejecutan las pruebas que no dependen de un número concreto de datos, pero no funcionan con la base vacía, y por último casos con un mayor número de usuarios y ventas.

Posteriormente se prueban las inserciones del módulo DAO. Se aíslan aquellas pruebas que insertan pedidos, pues todas fallan debido a un error en la lectura del campo "fecha".

La siguiente prueba abarca el módulo de importación, separando las pruebas que dependen de la existencia o no de una entidad concreta.

Por último se prueban las actualizaciones de datos en el DAO y, finalmente, las lecturas que confirman la correcta ejecución de los tests anteriores.

6 Informe de ejecución de pruebas

Las pruebas comenzaron a realizar en el módulo estadístico. En dicho módulo, se encontró un error con respecto a los requisitos exigidos. El *ID* de la clase *Order* (pedido) se codificó como un *Integer*, cuando en los requisitos se exigía que fuese un *String*. A pesar de esto, en la base de datos si utilizan un *String* como *ID*. Esto supuso adaptar la implementación de las pruebas a esta situación, para poder cubrir la funcionalidad del módulo estadístico, así como comprobar si la codificación de las pruebas era correcta. Por lo tanto, el tipo de este *ID* debe ser migrado a *String* cuando se resuelva la incidencia asociada.

A continuación, se realizaron las pruebas sobre el módulo de importación. Nuevamente se incumplió un requisito. En este caso, no se pueden importar archivos con líneas heterogéneas. En otras palabras, el archivo debe contener solo usuarios, o solo ítems o solo ventas. Sin embargo, esto tuvo un impacto muy leve en la realización de las pruebas. Tras realizar las pruebas en este módulo, se observó que la cantidad de errores presentes era mucho mayor de lo esperado. La baja cobertura de este módulo se debe precisamente a esta situación, pues muchas líneas de código no pueden ser ejecutadas debido a excepciones no controladas que se producen durante la ejecución del mismo, sin que esto pueda ser evitado de ninguna manera. Las conclusiones a las que se llega tras ejecutar las pruebas sobre este módulo es que lo más apropiado sería realizar una reimplementación de cero.

Por último, se probó el módulo de acceso a datos (*DAOs*). Se encontraron errores menores con respecto al formato de algunos campos (por ejemplo, es posible insertar usuarios con un ID inválido). No obstante, podría ser válido considerar que esta responsabilidad debería recaer en otros módulos. Lo realmente importante es que la mayoría de casos base funcionan correctamente.

Debido al alto número de errores descubiertos se puede concluir que las pruebas fueron un éxito.

Element	Coverage	Covered Instruct	Missed Instructic
Enso	64,6 %	3.185	1.748
src/main/java	53,8 %	1.684	1.447
(default package)	0,0 %	0	64
Main.java	0,0 %	0	64
control	0,0 %	0	106
ControlModule.java	0,0 %	0	106
DAO	79,8 %	761	193
DAOModule.java	95,7 %	112	5
ItemDAO.java	90,6 %	184	19
PurchaseDAO.java	62,9 %	195	115
SuperDAO.java	100,0 %	57	0
UserDAO.java	79,8 %	213	54
imports	45,9 %	225	265
ImportsModule.java	45,9 %	225	265
model	20,1 %	205	817
Item.java	26,3 %	75	210
Line.java	10,0 %	12	108
Order.java	19,6 %	52	213
Purchase.java	51,9 %	27	25
User.java	13,0 %	39	261
statistics	99,6 %	493	2
StatisticsModule.java	99,6 %	493	2
StatisticsModule	99,6 %	493	2
src/test/java	83,3 %	1.501	301
(default package)	0,0 %	0	3
DAO	63,1 %	260	152
imports	78,7 %	259	70
statistics	92,8 %	982	76

7 Anexo: plantillas

7.1 Plantilla de plan de pruebas

- Identificador único del documento.
- Introducción.
- **Elementos software a probar**: indica los módulos de software que abarcarán las pruebas definidas.
- **Características a probar**: concreta lo anterior para definir los requisitos funcionales sobre los que se plantean las pruebas.
- **Enfoque general de la prueba**: define el procedimiento seguido a la hora de diseñar las pruebas.
- **Criterios de paso/fallo para cada elemento**: define los criterios que se seguirán para considerar una prueba concreta como superada.
- **Criterios de suspensión y requisitos de reanudación**: define situaciones que implicarán la finalización y replantearse el proceso de pruebas.
- **Documentos a entregar**: índice de los documentos que forman el proceso de pruebas.

- **Actividades de preparación y ejecución de pruebas**: define las actividades necesarias para inicializar el entorno sobre el que se ejecutarán las pruebas.
- **Necesidades de entorno**: indica el contexto de datos necesario para evitar los fallos de configuración a la hora de ejecutar las pruebas.
- **Responsabilidades en la organización y realización de las pruebas**: indica los roles de los integrantes del equipo encargado de realizar las pruebas.
- **Necesidades de personal y de formación**: define los conocimientos necesarios por parte de los recursos humanos asociados a la planificación y ejecución de las pruebas.
- **Esquema de tiempos**: diagrama que indica las responsabilidades y tiempos de cada tarea necesaria para la realización del proceso de pruebas.
- Riesgos asumidos por el plan y planes de contingencia para cada riesgo: planificación de los riesgos asociados al proceso de pruebas.
- **Aprobaciones y firmas con nombre y puesto desempeñado**: muestra la aprobación de los integrantes acerca de este documento de especificación.

7.2 Plantilla de diseño de pruebas

- **Objetivo**: indica qué funcionalidad busca probar
- **Técnicas de caja negra**: se indica el desarrollo de aplicación de las técnicas de caja negra, separando los métodos a utilizar (generación de clases de equivalencia, análisis de valores límite, conjetura de errores, etc.).
- **Técnicas de caja blanca**: se indica el desarrollo de aplicación de las técnicas de caja blanca.
- **Criterios de paso/fallo**: indica los criterios a valorar para que la prueba se considere como superada.
- **Resultado de aplicar las técnicas**: incluye los anexos que pueden resultar de la aplicación de las técnicas anteriores, como puede ser la tabla de clases de equivalencia, grafos, etc.

7.3 Plantilla de casos de prueba de caja negra

- **Contexto de ejecución**: se describe el objetivo del caso de prueba y el estado del sistema a la hora de ejecutarse.
- **Definición**: define las entradas y métodos a ejecutar.
- **Clases que valida**: relaciona el caso de prueba con las clases de equivalencia que cubre de la prueba.
- **Resultado esperado**: salida esperada por el programa para que el caso se considere un éxito.

7.4 Plantilla de casos de prueba de caja blanca

- **Prerrequisitos**: elementos que es necesario introducir antes de realizar la prueba para conseguir el resultado deseado.
- **Entrada**: entrada a introducir en el método.
- **Salida**: salida esperada por el programa para que el caso se considere un éxito.

7.5 Documentos relacionados

Nombre	Descripción

02_PlantAnaDisPlan_v1.pdf	Documento de análisis y diseño del software sobre el cual
Std_IEEE_829.pdf	Estándar 829.