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

[Ingeniería de Software i](#_Toc482824029)

[Proceso de Pruebas i](#_Toc482824030)

[Entrega 3 i](#_Toc482824031)

[1 FECHA DE ENTREGA: 19/05/2017 i](#_Toc482824032)

[1.1 ÍNDICE i](#_Toc482824033)

[2 Plan de pruebas 1](#_Toc482824034)

[2.1 Identificador único del documento 1](#_Toc482824035)

[2.2 Introducción 1](#_Toc482824036)

[2.3 Clases software a probar 1](#_Toc482824037)

[2.4 Funcionalidades a probar 1](#_Toc482824038)

[2.5 Funcionalidades que no se prueban 1](#_Toc482824039)

[2.6 Enfoque general de la prueba 1](#_Toc482824040)

[2.7 Criterios de paso/fallo para cada elemento 1](#_Toc482824041)

[2.8 Criterios de suspensión y requisitos de reanudación 1](#_Toc482824042)

[2.9 Documentos a entregar 2](#_Toc482824043)

[2.10 Actividades de preparación y ejecución de pruebas 2](#_Toc482824044)

[2.11 Necesidades de entorno 2](#_Toc482824045)

[2.12 Responsabilidades en la organización y realización de las pruebas 2](#_Toc482824046)

[2.13 Necesidades de personal y de formación 2](#_Toc482824047)

[2.14 Esquema de tiempos 3](#_Toc482824048)

[2.15 Riesgos asumidos por el plan y planes de contingencias para cada riesgo 3](#_Toc482824049)

[2.16 Aprobaciones y firmas con nombre y puesto desempeñado 3](#_Toc482824050)

[3 Diseño de pruebas 4](#_Toc482824051)

[3.1 Pruebas importación 4](#_Toc482824052)

[3.1.1 Prueba I-01 4](#_Toc482824053)

[3.1.2 Prueba I-02 8](#_Toc482824054)

[3.2 Pruebas de inserción y lectura sobre la base de datos 9](#_Toc482824055)

[3.2.1 Prueba D-01 9](#_Toc482824056)

[3.2.2 Prueba D-02 10](#_Toc482824057)

[3.2.3 Prueba D-03 10](#_Toc482824058)

[3.2.4 Prueba D-04 12](#_Toc482824059)

[3.2.5 Prueba D-05 13](#_Toc482824060)

[3.2.6 Prueba D-06 13](#_Toc482824061)

[3.3 InterfazEstadistica 14](#_Toc482824062)

[3.3.1 Prueba E-01 14](#_Toc482824063)

[3.3.2 Prueba E-02 15](#_Toc482824064)

[3.4 Pruebas sobre el controlador. 15](#_Toc482824065)

[3.4.1 Nota 15](#_Toc482824066)

[4 Casos de prueba de caja negra 16](#_Toc482824067)

[4.1 InterfazImportacion 16](#_Toc482824068)

[4.1.1 Caso de prueba I-01-P-01 16](#_Toc482824069)

[4.1.2 Caso de prueba I-01-P-02 16](#_Toc482824070)

[4.1.3 Caso de prueba I-01-P-03 16](#_Toc482824071)

[4.1.4 Caso de prueba I-01-P-04 17](#_Toc482824072)

[4.1.5 Caso de prueba I-01-P-04 17](#_Toc482824073)

[4.1.6 Caso de prueba I-01-P-05 17](#_Toc482824074)

[4.1.7 Caso de prueba I-01-P-056 18](#_Toc482824075)

[4.1.8 Caso de prueba I-01-P-07 18](#_Toc482824076)

[4.1.9 Caso de prueba I-0-P-08 19](#_Toc482824077)

[4.1.10 Caso de prueba I-01-P-09 19](#_Toc482824078)

[4.1.11 Caso de prueba I-01-P-10 19](#_Toc482824079)

[4.1.12 Caso de prueba I-01-P-11 20](#_Toc482824080)

[4.1.13 Caso de prueba I-01-P-12 20](#_Toc482824081)

[4.1.14 Caso de prueba I-01-P-13 20](#_Toc482824082)

[4.1.15 Caso de prueba I-02-P-01 21](#_Toc482824083)

[4.1.16 Caso de prueba I-02-P-02 21](#_Toc482824084)

[4.1.17 Caso de prueba I-02-P-03 22](#_Toc482824085)

[4.2 Lectura e inserción 22](#_Toc482824086)

[4.2.1 Caso de prueba D-01-P-01 22](#_Toc482824087)

[4.2.2 Caso de prueba D-01-P-02 22](#_Toc482824088)

[4.2.3 Caso de prueba D-01-P-03 23](#_Toc482824089)

[4.2.4 Caso de prueba D-01-P-04 23](#_Toc482824090)

[4.2.5 Caso de prueba D-01-P-05 23](#_Toc482824091)

[4.2.6 Caso de prueba D-01-P-06 24](#_Toc482824092)

[4.2.7 Caso de prueba D-01-P-07 24](#_Toc482824093)

[4.2.8 Caso de prueba D-02-P-01 24](#_Toc482824094)

[4.2.9 Caso de prueba D-02-P-02 25](#_Toc482824095)

[4.2.10 Caso de prueba D-02-P-03 25](#_Toc482824096)

[4.2.11 Caso de prueba D-02-P-04 25](#_Toc482824097)

[4.2.12 Caso de prueba D-02-P-05 26](#_Toc482824098)

[4.2.13 Caso de prueba D-02-P-06 26](#_Toc482824099)

[4.2.14 Caso de prueba D-03-P-01 27](#_Toc482824100)

[4.2.15 Caso de prueba D-03-P-02 28](#_Toc482824101)

[4.2.16 Caso de prueba D-04-P-01 29](#_Toc482824102)

[4.2.17 Caso de prueba D-04-P-02 29](#_Toc482824103)

[4.2.18 Caso de prueba D-04-P-03 29](#_Toc482824104)

[4.2.19 Caso de prueba D-04-P-04 30](#_Toc482824105)

[4.2.20 Caso de prueba D-05-P-01 30](#_Toc482824106)

[4.2.21 Caso de prueba D-05-P-02 31](#_Toc482824107)

[4.2.22 Caso de prueba D-05-P-03 31](#_Toc482824108)

[4.3 InterfazEstadistica 31](#_Toc482824109)

[4.3.1 Nota 31](#_Toc482824110)

[4.3.2 Caso de prueba E-01-P-01 32](#_Toc482824111)

[4.3.3 Caso de prueba E-01-P-02 32](#_Toc482824112)

[4.3.4 Caso de prueba E-01-P-03 32](#_Toc482824113)

[4.3.5 Caso de prueba E-01-P-04 33](#_Toc482824114)

[4.3.6 Caso de prueba E-01-P-05 33](#_Toc482824115)

[4.3.7 Caso de prueba E-01-P-06 33](#_Toc482824116)

[5 Casos de prueba de caja blanca 34](#_Toc482824117)

[5.1 importarUsuarios 34](#_Toc482824118)

[5.1.1 Camino 1 34](#_Toc482824119)

[5.1.2 Camino 2 34](#_Toc482824120)

[5.1.3 Camino 3 34](#_Toc482824121)

[5.1.4 Camino 4 34](#_Toc482824122)

[5.1.5 Camino 6 34](#_Toc482824123)

[5.1.6 Camino 6 34](#_Toc482824124)

[5.1.7 Camino 7 34](#_Toc482824125)

[5.1.8 Camino 8 35](#_Toc482824126)

[5.1.9 Camino 9 35](#_Toc482824127)

[5.1.10 Camino 10 35](#_Toc482824128)

[5.2 importaCompra 35](#_Toc482824129)

[5.2.1 Camino 1 35](#_Toc482824130)

[5.2.2 Camino 2 35](#_Toc482824131)

[5.2.3 Camino 3 35](#_Toc482824132)

[5.2.4 Camino 4 36](#_Toc482824133)

[5.2.5 Camino 5 36](#_Toc482824134)

[5.2.6 Camino 6 36](#_Toc482824135)

[5.2.7 Camino 7 36](#_Toc482824136)

[5.2.8 Camino 8 36](#_Toc482824137)

[5.2.9 Camino 9 36](#_Toc482824138)

[5.2.10 Camino 10 37](#_Toc482824139)

[5.2.11 Camino 11 37](#_Toc482824140)

[5.3 validateOrder 38](#_Toc482824141)

[5.3.1 Camino 1 38](#_Toc482824142)

[5.3.2 Camino 2 38](#_Toc482824143)

[5.3.3 Camino 3 39](#_Toc482824144)

[5.3.4 Camino 4 39](#_Toc482824145)

[5.3.5 Camino 5 40](#_Toc482824146)

[5.4 getAllUsers 40](#_Toc482824147)

[5.4.1 Camino 1 40](#_Toc482824148)

[5.4.2 Camino 2 40](#_Toc482824149)

[5.4.3 Camino 3 40](#_Toc482824150)

[5.4.4 Camino 4 40](#_Toc482824151)

[5.5 insertOrder 41](#_Toc482824152)

[5.5.1 Camino 1 41](#_Toc482824153)

[5.5.2 Camino 2 41](#_Toc482824154)

[5.5.3 Camino 3 41](#_Toc482824155)

[5.5.4 Camino 4 43](#_Toc482824156)

[6 Procedimientos de prueba 43](#_Toc482824157)

[7 Anexo: plantillas 43](#_Toc482824158)

[7.1 Plantilla de plan de pruebas 43](#_Toc482824159)

[7.2 Plantilla de diseño de pruebas 44](#_Toc482824160)

[7.3 Plantilla de casos de prueba 44](#_Toc482824161)

[7.4 Documentos relacionados 44](#_Toc482824162)

# Plan de pruebas

## Identificador único del documento

PDP\_v1

## 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.

## Clases software a probar

//TODO

## Funcionalidades a probar

//TODO

## Funcionalidades que no se prueban

//TODO

## 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.

## 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.

## 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.

## Documentos a entregar

* Plan de pruebas
* Informe de ejecución

## 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.

## 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.

## 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. |

## Necesidades de personal y de formación

E 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).

## Esquema de tiempos



## 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. |

## 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.

# Diseño de pruebas

## Pruebas importación

### Prueba I-01

#### Objetivo

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

#### 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*).

##### 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 |
| R3 | {20} Primer campo con el valor adecuado para el método llamado | {21} Primer campo con un valor incorrecto para el método llamado |
| VRef |
| IdxUser | String | R3 | {22} Identificador referenciado previamente en base de datos. | {23} Identificador referenciado no previamente en base de datos. |
| ItemRef |

#### Técnicas de caja blanca

##### importarUsuarios

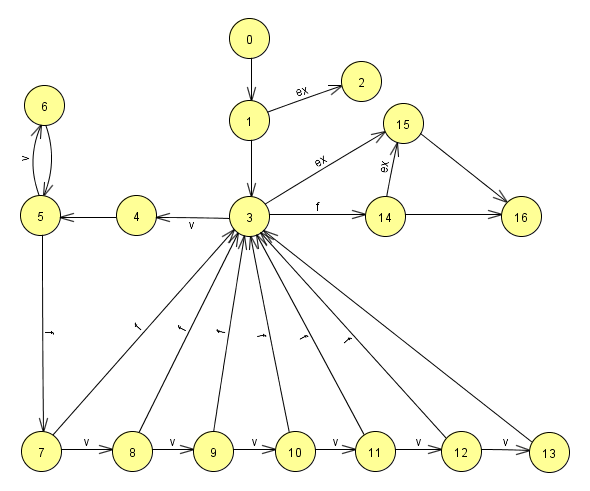


Figura 1. Grafo del método importarUsuario

###### 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;

###### Complejidad ciclomática

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

###### 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.

##### ImportarCompra

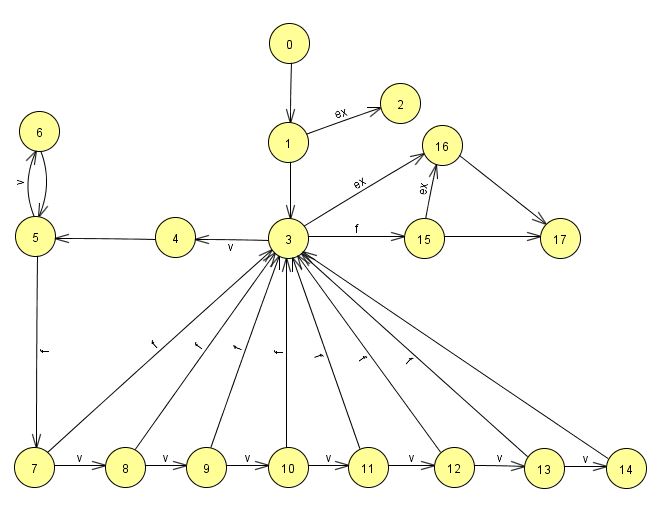


Figura 2. Grafo del método importarCompra

###### 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;‘

###### Complejidad ciclomática

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

###### 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.

#### 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.

### Prueba I-02

#### Objetivo

Comprobar que la importación se realiza correctamente cuando se pasan argumentos válidos y que no se genera ninguna excepción sin controlar cuando se pasan argumentos inválidos.

#### 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.

##### 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. |

#### 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.

## Pruebas de inserción y lectura sobre la base de datos

### Prueba D-01

#### Objetivo

Comprobar que las inserciones o modificaciones de usuarios se reflejan correctamente en la base de datos. Además, se busca que, en caso de error, este sea correctamente gestionado.

#### Técnicas de caja negra

##### 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.

##### 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. |

#### 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.

### Prueba D-02

#### Objetivo

Comprobar que las inserciones o modificaciones de items y ventas se reflejan correctamente en la base de datos.

Además, se busca que, en caso de error, este sea correctamente gestionado.

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.

#### Técnicas de caja negra

##### 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.

##### 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 |

#### 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.

### Prueba D-03

#### 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.

#### Técnicas de caja negra

##### Generación de clases de equivalencia

Solo el metodo validateOrder gestiona esta acción.

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

##### 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 |

#### Técnicas de caja blanca

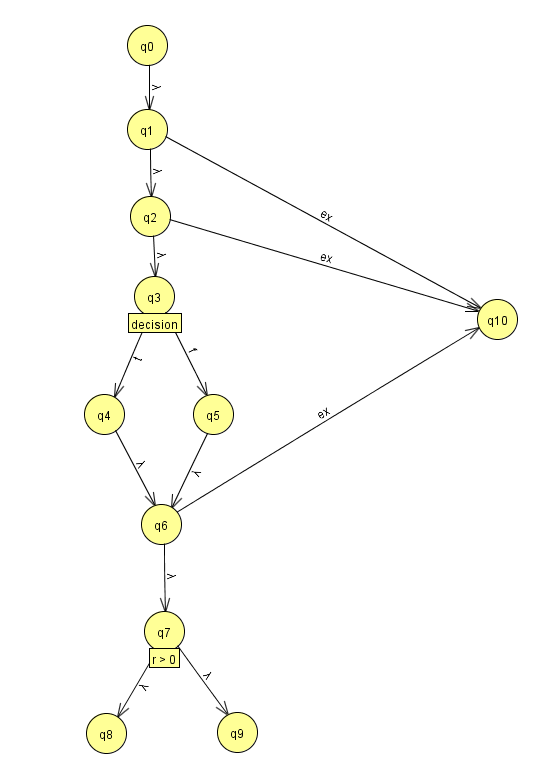


Figura 3. Grafo del método validateOrder

##### 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.

##### Complejidad ciclomática

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

##### 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.

#### 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.

### Prueba D-04

#### 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.*

#### Técnicas de caja negra

##### 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.

##### 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 |

#### Técnicas de caja blanca

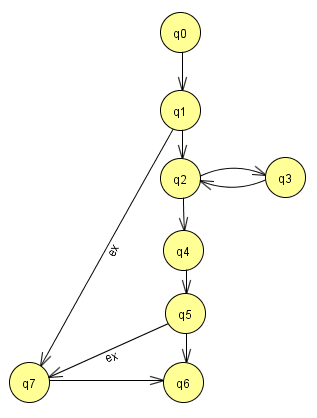


Figura 4. Grafo del método getAllUsers

##### 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.

##### Complejidad ciclomática

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

##### 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

#### 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.

### Prueba D-05

#### Objetivo

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

#### Técnicas de caja negra

##### 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.

##### 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 |

#### 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.

### Prueba D-06

#### Objetivo

Comprobar si se puede insertar un pedido correctamente.

#### Técnicas de caja blanca

##### insertOrder

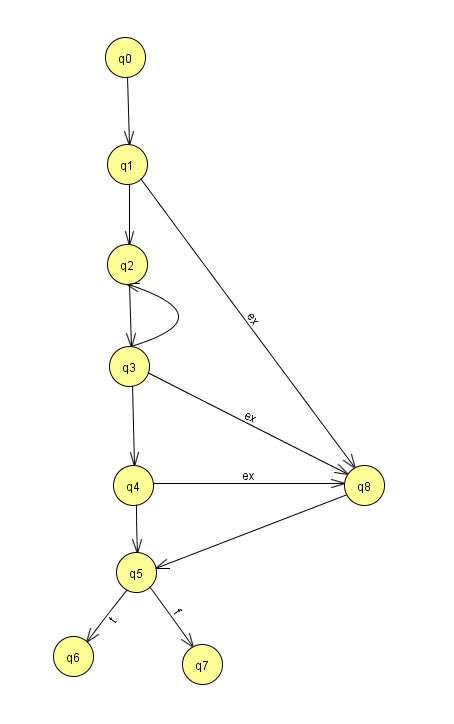


Figura 5. Grafo del método insertOrder

##### 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.

##### Complejidad ciclomática

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

## Pruebas sobre el módulo Estadistico

### Prueba E-01

#### Objetivo

Comprobar que para entradas incorrectas en los métodos de la interfaz no se producen situaciones no controladas o inesperadas.

#### Técnicas de caja negra

##### 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.

##### 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 |  |

#### Técnicas de caja blanca

##### getPorcentajes

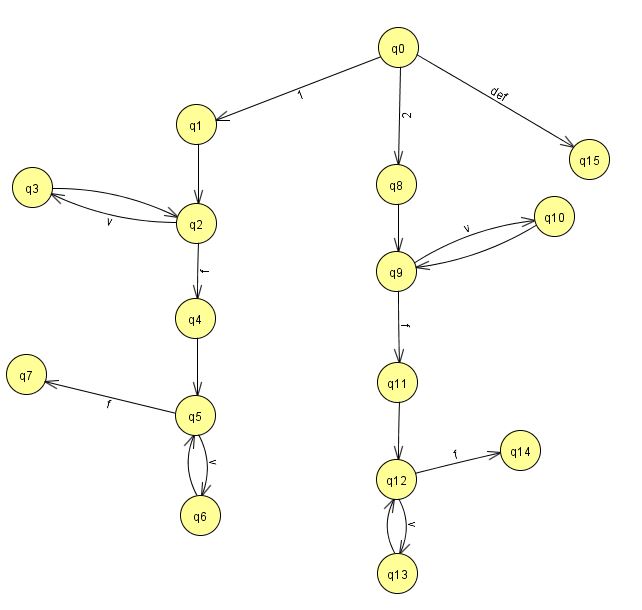


Figura 6. Grafo del método getPorcentajes

###### 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).

###### Complejidad ciclomática

V(G) = r = 5

###### 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

#### 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.

### Prueba E-02

#### Objetivo

Comprobar que para entradas incorrectas en los métodos de la interfaz no se producen situaciones no controladas o inesperadas.

#### Técnicas de caja negra

##### 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.

#### *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 |

#### Técnicas de caja blanca

##### GetMedias

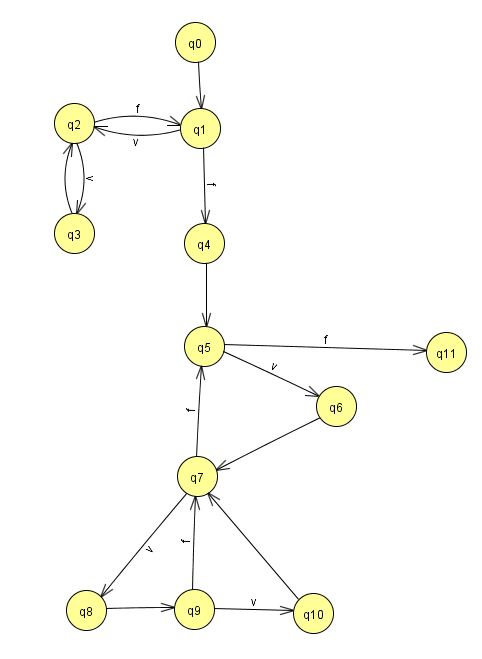


Figura 7. Grafo del método getMedias

###### 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.

###### Complejidad ciclomática

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

###### 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.

##### GetValoresBrutos

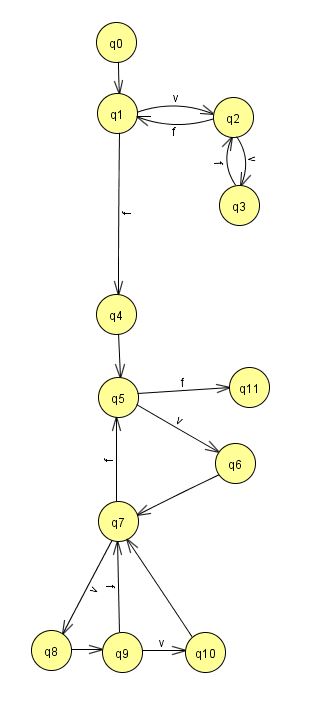


Figura 8. Grafo del método getValoresBrutos

###### 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.

###### Complejidad ciclomática

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

###### 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.

#### 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.

## Pruebas sobre el controlador.

### 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.

# Casos de prueba de caja negra

## Pruebas de importación de datos

### Caso de prueba I-01-P-01

#### Contexto de ejecución

En la base de datos no debe existir un usuario con el identificador *U-aaaaaa-000*.

#### Definición

Se llama al método importarUsuarios(). En este caso se valida la correcta interpretación y procesamiento de una línea que contenga todos los campos necesarios con valores correctos.

U; U-aaaaaa-000; 10/10/2010; Samuel; Soutullo Sobral; 77013889E

#### Clases que valida

1, 4

#### Resultado esperado

Importación del usuario con ID *U-aaaaaa-000*.

### Caso de prueba I-01-P-02

#### Contexto de ejecución

En la base de datos no debe existir un usuario con el identificador *U-aaaaaa-000*.

#### Definición

Se llama al método importarUsuarios(). En este caso se valida la correcta interpretación de una línea que contenga una cantidad de campos inferior a la correcta.

U; U-aaaaaa-000; 10/10/2010; Samuel; Soutullo Sobral

#### Clases que valida

2

#### Resultado esperado

Notificación de error por parte de la aplicación, abortando toda la importación. No se modifica la base de datos.

### Caso de prueba I-01-P-03

#### Contexto de ejecución

En la base de datos no debe existir un usuario con el identificador *U-aaaaaa-000*.

#### Definición

Se llama al método importarUsuarios(). En este caso se valida la correcta interpretación de una línea que contenga una cantidad de campos superior a la correcta.

U; U-aaaaaa-000; 10/10/2010; Samuel; Soutullo Sobral; 77013889E; asdfg

#### Clases que valida

3

#### Resultado esperado

Notificación de error por parte de la aplicación, abortando toda la importación. No se modifica la base de datos.

### Caso de prueba I-01-P-04

#### Contexto de ejecución

En la base de datos no debe existir un usuario con el identificador *U-aaaaaa-000*.

#### Definición

Se llama al método importarUsuarios(). En este caso se valida la correcta interpretación de una línea en blanco.

(Línea en blanco)

#### Clases que valida

5

#### Resultado esperado

No se notifica de ningún error, simplemente se salta la línea en blanco.

### Caso de prueba I-01-P-05

#### Contexto de ejecución

Valida la correcta interpretación y procesamiento de una línea que contenga todos los campos necesarios con valores correctos. En la base de datos no debe existir un usuario con el identificador *U-aaaaaa-000*.

#### Definición

Se llama al método importarUsuarios()

U; U-aaaaaa-000; 10/10/2010; Samuel; Soutullo Sobral; 77013889E

#### Clases que valida

6, 9, 12, 13, 15, 17

#### Resultado esperado

Importación del usuario con ID *U-aaaaaa-000*.

### Caso de prueba I-01-P-06

#### Contexto de ejecución

En la base de datos debe existir una compra con el identificador 77013889E.

//TODO: comprobar este contexto

#### Definición

Se llama al método *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.

U; U-aaaaaa-000; 10/10/2010; Samuel; Soutullo Sobral; 77013889E

#### Clases que valida

19

#### Resultado esperado

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.

### Caso de prueba I-01-P-07

#### Contexto de ejecución

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*.

#### Definición

Se llama al método *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.

V; V-aaaaaa-000; 10/10/2010; U-aaaaaa-000; I-aaaaaa-000; 1; 1.53

#### Clases que valida

21

#### Resultado esperado

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.

### Caso de prueba I-01-P-08

#### Contexto de ejecución

Debe existir un usuario en la base de datos con identificador U-aaaaaa-000.

#### Definición

Se llama al método 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.

U; U-aaaaaa-000; 10/10/2010; ; Soutullo Sobral; 77013889E

#### Clases que valida

7

#### Resultado esperado

La aplicación notificará del error correspondiente. La base de datos no se verá modificada.

### Caso de prueba I-0-P-09

#### Contexto de ejecución

Debe existir un usuario en la base de datos con identificador U-aaaaaa-000.

#### Definición

Se llama al método 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.

U; U-aaaaaa-000; 10/10/2010; aaaaa[…(x255)…]aaaa ; Soutullo Sobral; 77013889E

#### Clases que valida

8

#### Resultado esperado

La aplicación notificará del error correspondiente. La base de datos no se verá modificada.

### Caso de prueba I-01-P-10

#### Contexto de ejecución

Debe existir un usuario en la base de datos con identificador U-aaaaaa-000.

#### Definición

Se llama al método 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.

U; U-aaaaaa-000; July 4, 2010; Samuel; Soutullo Sobral; 77013889E

#### Clases que valida

13

#### Resultado esperado

La aplicación notificará del error correspondiente. La base de datos no se verá modificada.

### Caso de prueba I-01-P-11

#### Contexto de ejecución

Debe existir un usuario con identificador U-aaaaaa-000 y un *ítem* con identificador I-aaaaaa-000 en la base de datos.

#### Definición

Se llama al método *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.

V; V-aaaaaa-000; 10/10/2010; U-aaaaaa-000; I-aaaaaa-000; 1; 125ª

#### Clases que valida

16

#### Resultado esperado

La aplicación notificará del error correspondiente. La base de datos no se verá modificada.

### Caso de prueba I-01-P-12

#### Contexto de ejecución

El usuario *U-aaaaaa-000* se encuentra previamente en la base de datos.

#### Definición

Se llama al método 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.

U; U-aaaaaa-000; 10/10/2010; Samuel; Soutullo Sobral; 77013889E

#### Clases que valida

18

#### Resultado esperado

La aplicación notificará del error correspondiente. La base de datos no se verá modificada.

### Caso de prueba I-01-P-13

#### Contexto de ejecución

Los identificadores de usuario (; U-aaaaaa-000) e *ítem* (I-aaaaaa-000) no se encuentran en la base de datos.

#### Definición

Se llama al método *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.

V; V-aaaaaa-000; 10/10/2010; U-aaaaaa-000; I-aaaaaa-000; 1; 1.53

#### Clases que valida

22

#### Resultado esperado

La aplicación notificará del error correspondiente. La base de datos no se verá modificada.

### Caso de prueba I-01-P-14

#### Contexto de ejecución

La base de datos contiene el usuario *U-aaaaaa-000* y los productos *I-aaaaaa-000* y *I-aaaaaa-001*.

#### Definición

Se llama al método *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*.

V; V-aaaaaa-000; 10/10/2010; U-aaaaaa-000; I-aaaaaa-000; 1; 1.53

V; V-aaaaaa-000; 10/10/2010; U-aaaaaa-000; I-aaaaaa-001; 1; 2.64

#### Clases que valida

Este caso de prueba se ha obtenido en base a conjetura de errores.

#### Resultado esperado

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*.

### Caso de prueba I-02-P-01

#### Contexto de ejecución

El fichero /*home/usuario/fichero.csv* existe y es un fichero de importación válido.

#### Definición

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.

* Se llama al método *importarUsuario()*
  + **Path**: “/home/usuario/fichero.csv”

#### Clases que valida

1, 3

#### Resultado esperado

La importación se realizará correctamente.

### Caso de prueba I-02-P-02

#### Contexto de ejecución

El fichero /*home/usuario/fichero.csv* existe pero es un fichero de importación inválido.

#### Definición

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

* Se llama al método *importarUsuario()*
  + **Path**: “!·$%&/()=?¿”

#### Clases que valida

2

#### Resultado esperado

Se notificará al usuario del error. No se generará ninguna excepción sin controlar.

### Caso de prueba I-02-P-03

#### Contexto de ejecución

El fichero “*/home/usuario/ficheroInexistente.csv*” no existe.

#### Definición

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

* Se llama al método *importarUsuario()*
  + **Path**: “/home/usuario/ficheroInexistente.csv”

#### Clases que valida

4

#### Resultado esperado

Se notificará al usuario del error. No se generará ninguna excepción sin controlar.

## Pruebas de lectura e inserción en la base de datos

### Caso de prueba D-01-P-01

#### Contexto de ejecución

No debe existir un usuario con identificador U-abcde-000 en la BBDD.

#### Definición

Se valida la inserción correcta de un usuario.

Usuario:

| U-abcde-000 | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |

#### Clases que valida

1,3,6,9,10.

#### Resultado esperado

Inserción del usuario con id: U-abcdef-000 en la base de datos.

### Caso de prueba D-01-P-02

#### Contexto de ejecución

No debe existir un usuario con igual identificador en la BBDD.

#### Definición

Se valida la inserción correcta de un usuario.

Usuario:

| x | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |

#### Clases que valida

2

#### Resultado esperado

Notificación de error por parte de la aplicación. Sin consecuencias en la base de datos.

### Caso de prueba D-01-P-03

#### Contexto de ejecución

No debe existir un usuario con igual identificador en la BBDD.

#### Definición

Valida la inserción correcta de un usuario.

Usuario:

|U-abcd-000 | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |

#### Clases que valida

4

#### Resultado esperado

Notificación de error por parte de la aplicación. Sin consecuencias en la base de datos.

### Caso de prueba D-01-P-04

#### Contexto de ejecución

No debe existir un usuario con igual identificador en la BBDD.

#### Definición

Valida la inserción correcta de un usuario.

Usuario:

|U-abcde-0000 | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |

#### Clases que valida

8

#### Resultado esperado

Notificación de error por parte de la aplicación. Sin consecuencias en la base de datos.

### Caso de prueba D-01-P-05

#### Contexto de ejecución

No debe existir un usuario con igual identificador en la BBDD.

#### Definición

Valida la inserción correcta de un usuario.

Usuario:

|U-abcdef-| Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |

#### Clases que valida

10

#### Resultado esperado

Notificación de error por parte de la aplicación. Sin consecuencias en la base de datos.

### Caso de prueba D-01-P-06

#### Contexto de ejecución

La base de datos solo cuenta con el usuario “U-aaaaa-000”, nombre Limón Novoa, fecha 10-10-2010 y tipo estudiante.

#### Definición

Valida la modificación correcta de un usuario.

Usuario:

|U-abcde-000 | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |

#### Clases que valida

12

#### Resultado esperado

En la base de datos el usuario cuyo id es U-aaaaa-000 pasa a tener el nombre Manuel Soutoullo.

### Caso de prueba D-01-P-07

#### Contexto de ejecución

La base de datos solo cuenta con el usuario “U-aaaaa-000”, nombre Limón Novoa, fecha 10-10-2010 y tipo estudiante.

#### Definición

Valida la modificación correcta de un usuario.

Usuario:

|U-ddddd-111 | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |

#### Clases que valida

13

#### Resultado esperado

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.

### Caso de prueba D-02-P-01

#### Contexto de ejecución

No debe haber un *ítem* con el mismo identificador (I-abcde-000) en la base de datos.

#### Definición

Valida la inserción correcta de un item.

Item:

|I-abcde-000 | Robot limpiapiscinas | Limpia piscinas | Exteriores | 50 |10/10/2010|

#### Clases que valida

1

#### Resultado esperado

Inserción del item con id: I-abcde-000 en la base de datos.

### Caso de prueba D-02-P-02

#### Contexto de ejecución

El único *item* que existe en la base de datos es |I-abcde-000 | Robot limpiapiscinas | Limpia piscinas | Exteriores | 50 |10/10/2010|

#### Definición

Valida la actualización correcta de un *item*.

*Item*:

|I-abcde-000 | Robot limpiapiscinas | Limpia piscinas de forma eficiente | Exteriores | 50 |10-10-2010|

#### Clases que valida

5

#### Resultado esperado

Actualización correcta del *item* anterior con la descripción actualizada.

### Caso de prueba D-02-P-03

#### Contexto de ejecución

No debe existir un *ítem* con el mismo identificador en la base de datos.

#### Definición

Valida la inserción correcta de un *item*.

Item:

| x | Robot limpiapiscinas | Limpia piscinas de forma eficiente | Exteriores | 50 |10-10-2010|

#### Clases que valida

2

#### Resultado esperado

Muestra un error y no se pierde la consistencia de la base de datos.

### Caso de prueba D-02-P-04

#### Contexto de ejecución

El usuario “U-AAAAA-000” y el producto “I-AAAAA-000” existen en la base de datos.

#### Definición

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-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”

}

}

}

#### Clases que valida

4

#### Resultado esperado

Muestra un error y no se pierde la consistencia de la base de datos.

### Caso de prueba D-02-P-05

#### Contexto de ejecución

El único item que existe en la base de datos es |I-abcde-000 | Robot limpiapiscinas | Limpia piscinas | Exteriores | 50 |10/10/2010|

#### Definición

Valida la actualización correcta de un item.

Item:

| I-abcde-999 | Maletín de cuero | Maletín de portátil | Accesorios | 50 |10-10-2010 |

#### Clases que valida

6

#### Resultado esperado

Muestra un error indicando que el elemento a editar no existe o, alternativamente, inserta en la base el elemento nuevo.

### Caso de prueba D-03-P-01

#### Contexto de ejecución

Existencia del pedido en la base de datos.

#### Definición

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;

#### Clases que valida

1

#### Resultado esperado

El pedido pasa a ser una compra en la base de datos y se marca como aceptado.

### Caso de prueba D-03-P-02

#### Contexto de ejecución

Existencia del pedido en la base de datos.

#### Definición

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;

#### Clases que valida

2

#### Resultado esperado

El pedido se marca como rechazado o bien se elimina de la base de datos.

### Caso de prueba D-04-P-01

#### Contexto de ejecución

El usuario U-abcde-000 existe y tiene compras en su historial.

#### Definición

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

Id usuario:

|U-abcde-000 | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |

#### Clases que valida

1,3

#### Resultado esperado

Se recupera el historial de compras del usuario.

### Caso de prueba D-04-P-02

#### Contexto de ejecución

El usuario U-abcde-000 existe y tiene la compra introducida en el caso de prueba “caja blanca 4.5.1”

#### Definición

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

Id usuario:

| U-a-000 | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |

#### Clases que valida

2

#### Resultado esperado

Se muestra un error indicando que el id proporcionado no es válido.

### Caso de prueba D-04-P-03

#### Contexto de ejecución

El usuario U-abcde-000 existe y tiene la compra introducida en el caso de prueba “caja blanca 4.5.1”.

#### Definición

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

Id usuario:

| U-abcde-999 | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |

#### Clases que valida

4

#### Resultado esperado

Se muestra un error indicando que el usuario no existe.

### Caso de prueba D-04-P-04

#### Contexto de ejecución

El usuario U-abcde-123 existe y no tiene compras en su historial.

#### Definición

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

Id usuario:

| U-abcde-123 | Manuel | Soutoullo | 77013889E | 10-10-2010 | alumno |

#### Clases que valida

Este caso se infiere del estudio por valores límite.

#### Resultado esperado

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.

### Caso de prueba D-05-P-01

#### Contexto de ejecución

El único ítem existente en la base es: |I-abcde-000 | Robot limpiapiscinas | Limpia piscinas | Exteriores | 50 |10/10/2010|.

#### Definición

Comprueba la recuperación de un Item.

Id:

I-abcde-000

#### Clases que valida

1,3

#### Resultado esperado

Se devuelve |I-abcde-000 | Robot limpiapiscinas | Limpia piscinas | Exteriores | 50 |10/10/2010|.

### Caso de prueba D-05-P-02

#### Contexto de ejecución

El único ítem existente en la base es: |I-abcde-000 | Robot limpiapiscinas | Limpia piscinas | Exteriores | 50 |10/10/2010|.

#### Definición

Comprueba la recuperación de un Item.

Id:

I-s-000

#### Clases que valida

2

#### Resultado esperado

Se indica que el id es incorrecto.

### Caso de prueba D-05-P-03

#### Contexto de ejecución

El único ítem existente en la base es: |I-abcde-000 | Robot limpiapiscinas | Limpia piscinas | Exteriores | 50 |10/10/2010|.

#### Definición

Comprueba la recuperación de un Item.

Id:

I-abcde-999

#### Clases que valida

4

#### Resultado esperado

Se indica que el ítem no existe.

## Pruebas sobre el módulo Estadistico

### 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.

### Caso de prueba E-01-P-01

#### Contexto de ejecución

Debe existir alguna venta en la base de datos para los días consultados con el fin de hacer más consistentes las pruebas.

#### Definición

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

Días:

* 3

#### Clases que valida

2

#### Resultado esperado

El resultado esperado será una excepción.

### Caso de prueba E-01-P-02

#### Contexto de ejecución

Debe existir alguna venta en la base de datos para los días consultados con el fin de hacer más consistentes las pruebas.

#### Definición

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

Días:

5

#### Clases que valida

1

#### Resultado esperado

El resultado será la media de ventas en el conjunto de días dado.

### Caso de prueba E-01-P-03

#### Contexto de ejecución

Debe existir alguna venta en la base de datos para el año consultado con el fin de hacer más consistentes las pruebas.

#### Definición

Valida la consulta del histograma de ventas anuales.

Modo:

3

#### Clases que valida

4

#### Resultado esperado

El resultado será el histograma de ventas anuales.

### Caso de prueba E-01-P-04

#### Contexto de ejecución

Debe existir alguna venta en la base de datos para el intervalo de tiempo consultado con el fin de hacer más consistentes las pruebas.

#### Definición

Valida la consulta del histograma de ventas semanales.

Modo:

1

#### Clases que valida

1

#### Resultado esperado

El resultado será el histograma de ventas semanales.

### Caso de prueba E-01-P-05

#### Contexto de ejecución

Debe existir alguna venta en la base de datos para el intervalo de tiempo consultado con el fin de hacer más consistentes las pruebas.

#### Definición

Valida la consulta de los porcentajes de las ventas mensuales.

Modo:

2

#### Clases que valida

3

#### Resultado esperado

El resultado será el porcentaje correspondiente a las ventas mensuales.

### Caso de prueba E-01-P-06

#### Contexto de ejecución

Debe existir alguna venta en la base de datos para el intervalo de tiempo consultado con el fin de hacer más consistentes las pruebas.

#### Definición

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

Modo:

0

#### Clases que valida

2

#### Resultado esperado

El resultado esperado será una excepción.

# Casos de prueba de caja blanca

## importarUsuarios

### Camino 1

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **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* |

### Camino 2

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| No existe el fichero *users.csv* | *path:”users.csv”* | El método devuelve *-1*. |

### Camino 3

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **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* |

### Camino 4

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **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();* |

### Camino 6

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **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* |

### Camino 6

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **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* |

### Camino 7

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **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* |

### Camino 8

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **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* |

### Camino 9

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **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* |

### Camino 10

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **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* |

## importaCompra

### Camino 1

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **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* |

### Camino 2

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| No existe el fichero *sells.csv*. | *path: “sells.csv”* | El método devuelve *-1.* |

### Camino 3

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **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*. |

### Camino 4

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **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();* |

### Camino 5

|  |  |  |
| --- | --- | --- |
| **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: *A; V-AAAAA-000; 10/10/2010; U-AAAAA-000; I-AAAAA-000; 1; 10* |

### Camino 6

|  |  |  |
| --- | --- | --- |
| **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-00; 10/10/2010; U-AAAAA-000; I-AAAAA-000; 1; 10* |

### Camino 7

|  |  |  |
| --- | --- | --- |
| **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; 3/10/2010; U-AAAAA-000; I-AAAAA-000; 1; 10* |

### Camino 8

|  |  |  |
| --- | --- | --- |
| **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; 10* |

### 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* |

### 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* |

### 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;* |

## validateOrder

### 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. |

### 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. |

### 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. |

### 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. |

### 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. |

## getAllUsers

### 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. |

### Camino 2

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

### Camino 3

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

### 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*. |

## insertOrder

### 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. |

### Camino 2

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| El usuario U-AAAAA-000 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-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. |

### 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. |

### Camino 4

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| El producto I-AAAAA-000 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-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. |

## getPorcentajes

### Camino 1

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| La compra V-AAAAA-000 tiene que estar en la base de datos con fecha del día ¡a la prueba. | mode=1 | Porcentajes correspondientes a cada día en forma de array de floats no vacío y con algún valor distinto de 0.0 (mínimo 1). |

### Camino 2

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| No debe haber compras ningún día. | mode=1 | Array vacío. |

### Camino 3

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| La compra V-AAAAA-000 tiene que estar en la base de datos con fecha del día previo a la prueba. | mode=2 | Porcentajes correspondientes a cada semana en forma de array de floats no vacío y con algún valor distinto de 0.0 (mínimo 1). |

### Camino 4

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| No debe haber compras ninguna semana. | mode=2 | Array vacio. |

### Camino 5

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| Ninguno. | mode=3 | null |

## getMedias

### Camino 1

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| Debe existir el usuario U-AAAAA-000 en la tabla Usuario de la base de datos y la compra V-AAAAA-000 debe estar asociada a este en la tabla Compra, la fecha de compra tiene que ser del mismo día de la prueba. | dias=1 | El método devuelve un float distinto de 0. |

### 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. |

### Camino 3

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| Debe existir el usuario U-AAAAA-000 en la tabla Usuario de la base de datos y la compra V-AAAAA-000 debe estar asociada a este en la tabla Compra, la fecha de compra tiene que ser del mismo día de la prueba. | dias=0 | El método devuelve un float igual a 0. |

### Camino 4

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| Debe existir el usuario U-AAAAA-000 en la tabla Usuario de la base de datos y la tabla compras debe estar vacía. | dias=1 | El método devuelve un float igual a 0. |

### Camino 5

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| Deben existir cinco usuarios (U-AAAAA-000, U-AAAAA-001, U-AAAAA-002, U-AAAAA-003, U-AAAAA-004) y U-AAAAA-000 debe tener cero compras asociadas, U-AAAAA-001 debe tener V-AAAAA-000 asociada con fecha de 1 día antes de la prueba, U-AAAAA-002 debe tener V-AAAAA-001 y V-AAAAA-002 asociadas con fecha de 1 día antes de la prueba, U-AAAAA-003 debe tener V-AAAAA-003 con fecha de 3 días antes de la prueba y U-AAAAA-004 debe tener V-AAAAA-004 asociada con fecha de 6 días antes de la prueba. | Días=5 | El método devuelve un float distinto de 0. |

### Camino 6

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| Debe existir el usuario U-AAAAA-000 en la tabla Usuario de la base de datos y la compra V-AAAAA-000 debe estar asociada a este en la tabla Compra, la fecha de compra tiene que ser de 2 días antes de la prueba. | dias=1 | El método devuelve un float igual a 0. |

## getValoresBrutos

### Camino 1

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| Debe existir el usuario U-AAAAA-000 en la tabla Usuario de la base de datos y la compra V-AAAAA-000 debe estar asociada a este en la tabla Compra, la fecha de compra tiene que ser del mismo día de la prueba. | dias=1 | El método devuelve un int[] no vacio. |

### Camino 2

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| La tabla Usuario de la base de datos debe estar vacía. | dias=1 | El método devuelve un int[] vacio. |

### Camino 3

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| Debe existir el usuario U-AAAAA-000 en la tabla Usuario de la base de datos y la compra V-AAAAA-000 debe estar asociada a este en la tabla Compra, la fecha de compra tiene que ser del mismo día de la prueba. | dias=0 | El método devuelve un int[] vacio. |

### Camino 4

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| Debe existir el usuario U-AAAAA-000 en la tabla Usuario de la base de datos y la tabla compras debe estar vacía. | dias=1 | El método devuelve un int[] vacio. |

### Camino 5

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| Deben existir cinco usuarios (U-AAAAA-000, U-AAAAA-001, U-AAAAA-002, U-AAAAA-003, U-AAAAA-004) y U-AAAAA-000 debe tener cero compras asociadas, U-AAAAA-001 debe tener V-AAAAA-000 asociada con fecha de 1 día antes de la prueba, U-AAAAA-002 debe tener V-AAAAA-001 y V-AAAAA-002 asociadas con fecha de 1 día antes de la prueba, U-AAAAA-003 debe tener V-AAAAA-003 con fecha de 3 días antes de la prueba y U-AAAAA-004 debe tener V-AAAAA-004 asociada con fecha de 6 días antes de la prueba. | Días=5 | El método devuelve un int[] no vacio con valores que pueden ser iguales o distintos a cero. |

### Camino 6

|  |  |  |
| --- | --- | --- |
| **Prerrequisitos** | **Entrada** | **Salida** |
| Debe existir un usuario en la tabla Usuario de la base de datos y una compra asociada a este en la tabla Compra, pero esta compra debe haber sido realizada previamente al día en que se realice la prueba. | dias=1 | El método devuelve un int[] no vacio con todos los valores iguales a 0. |

# Procedimientos de prueba

No es necesario seguir ningún orden concreto al ejecutar los casos de prueba, pues todos ellos son idempotentes.

# Anexo: plantillas

## 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.

## 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.

## 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.

## 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.

## Documentos relacionados

|  |  |
| --- | --- |
| Nombre | Descripción |
| 02\_PlantAnaDisPlan\_v1.pdf | Documento de análisis y diseño del software sobre el cual |
| Std\_IEEE\_829.pdf | Estándar 829. |