Ingeniería de Software

**Proceso de Pruebas**

Entrega 2

Crístofer Canosa Domínguez

Silvia Rodríguez Alcaraz

Orquídea Seijas Salinas

Samuel Soutullo Sobral

FECHA DE ENTREGA: 02/04/2017

**ÍNDICE**

[1 Plan de pruebas 1](#_Toc478864831)

[1.1 Identificador único del documento 1](#_Toc478864832)

[1.2 Introducción 1](#_Toc478864833)

[1.2.1 Documentos relacionados 1](#_Toc478864834)

[1.3 Elementos software a probar 1](#_Toc478864835)

[1.4 Características a probar 1](#_Toc478864836)

[1.5 Características que no se prueban 1](#_Toc478864837)

[1.6 Enfoque general de la prueba 2](#_Toc478864838)

[1.7 Criterios de paso/fallo para cada elemento 2](#_Toc478864839)

[1.8 Criterios de suspensión y requisitos de reanudación 2](#_Toc478864840)

[1.9 Documentos a entregar 2](#_Toc478864841)

[1.10 Actividades de preparación y ejecución de pruebas 2](#_Toc478864842)

[1.10.1 Necesidades de entorno 2](#_Toc478864843)

[1.11 Responsabilidades en la organización y realización de las pruebas 2](#_Toc478864844)

[1.12 Necesidades de personal y de formación 2](#_Toc478864845)

[1.13 Esquema de tiempos 2](#_Toc478864846)

[1.14 Riesgos asumidos por el plan y planes de contingencias para cada riesgo 2](#_Toc478864847)

[1.15 Aprobaciones y firmas con nombre y puesto desempeñado 3](#_Toc478864848)

[2 Diseño de pruebas 3](#_Toc478864849)

[2.1 InterfazImportación 3](#_Toc478864850)

[2.1.1 Identificador único para la especificación y referencia del plan asociado ¡Error! Marcador no definido.](#_Toc478864851)

[2.1.2 Características a probar ¡Error! Marcador no definido.](#_Toc478864852)

[2.1.3 Detalles sobre el plan de pruebas ¡Error! Marcador no definido.](#_Toc478864853)

[2.1.4 Identificador de cada prueba ¡Error! Marcador no definido.](#_Toc478864854)

[2.1.5 Criterios de paso/fallo de la prueba ¡Error! Marcador no definido.](#_Toc478864855)

[2.2 InterfazDAO 6](#_Toc478864856)

[2.2.1 Identificador único para la especificación y referencia del plan asociado ¡Error! Marcador no definido.](#_Toc478864857)

[2.2.2 Características a probar ¡Error! Marcador no definido.](#_Toc478864858)

[2.2.3 Detalles sobre el plan de pruebas ¡Error! Marcador no definido.](#_Toc478864859)

[2.2.4 Identificador de cada prueba ¡Error! Marcador no definido.](#_Toc478864860)

[2.2.5 Criterios de paso/fallo de la prueba ¡Error! Marcador no definido.](#_Toc478864861)

[2.3 InterfazEstadistica 7](#_Toc478864862)

[2.3.1 Identificador único para la especificación y referencia del plan asociado 7](#_Toc478864863)

[2.3.2 Características a probar 7](#_Toc478864864)

[2.3.3 Detalles sobre el plan de pruebas 7](#_Toc478864865)

[2.3.4 Identificador de cada prueba 7](#_Toc478864866)

[2.3.5 Criterios de paso/fallo de la prueba 7](#_Toc478864867)

[2.4 InterfazControlador 8](#_Toc478864868)

[2.4.1 Identificador único para la especificación y referencia del plan asociado 8](#_Toc478864869)

[2.4.2 Características a probar 8](#_Toc478864870)

[2.4.3 Detalles sobre el plan de pruebas 8](#_Toc478864871)

[2.4.4 Identificador de cada prueba 8](#_Toc478864872)

[2.4.5 Criterios de paso/fallo de la prueba 8](#_Toc478864873)

[3 Casos de pruebas (CP) de caja negra 8](#_Toc478864874)

[3.1 InterfazImportación 8](#_Toc478864875)

[3.1.1 Identificador único de la especificación. ¡Error! Marcador no definido.](#_Toc478864876)

[3.1.2 Elementos software ¡Error! Marcador no definido.](#_Toc478864877)

[3.1.3 Especificaciones de cada entrada requerida para ejecutar el caso ¡Error! Marcador no definido.](#_Toc478864878)

[3.1.4 Especificaciones de todas las salidas y las características requeridas (por ejemplo, el tiempo de respuesta) para los elementos que se van a probar. ¡Error! Marcador no definido.](#_Toc478864879)

[3.1.5 Necesidades de entorno ¡Error! Marcador no definido.](#_Toc478864880)

[3.1.6 Requisitos especiales de procedimiento ¡Error! Marcador no definido.](#_Toc478864881)

[3.1.7 Dependencias entre casos ¡Error! Marcador no definido.](#_Toc478864882)

[3.2 InterfazDAO 8](#_Toc478864883)

[3.2.1 Identificador único de la especificación. 8](#_Toc478864884)

[3.2.2 Elementos software 8](#_Toc478864885)

[3.2.3 Especificaciones de cada entrada requerida para ejecutar el caso 8](#_Toc478864886)

[3.2.4 Especificaciones de todas las salidas y las características requeridas (por ejemplo, el tiempo de respuesta) para los elementos que se van a probar. 8](#_Toc478864887)

[3.2.5 Necesidades de entorno 8](#_Toc478864888)

[3.2.6 Requisitos especiales de procedimiento 8](#_Toc478864889)

[3.2.7 Dependencias entre casos 8](#_Toc478864890)

[3.3 InterfazEstadistica 9](#_Toc478864891)

[3.3.1 Identificador único de la especificación. 9](#_Toc478864892)

[3.3.2 Elementos software 9](#_Toc478864893)

[3.3.3 Especificaciones de cada entrada requerida para ejecutar el caso 9](#_Toc478864894)

[3.3.4 Especificaciones de todas las salidas y las características requeridas (por ejemplo, el tiempo de respuesta) para los elementos que se van a probar. 9](#_Toc478864895)

[3.3.5 Necesidades de entorno 9](#_Toc478864896)

[3.3.6 Requisitos especiales de procedimiento 9](#_Toc478864897)

[3.3.7 Dependencias entre casos 9](#_Toc478864898)

[3.4 InterfazControlador 9](#_Toc478864899)

[3.4.1 Identificador único de la especificación. 9](#_Toc478864900)

[3.4.2 Elementos software 9](#_Toc478864901)

[3.4.3 Especificaciones de cada entrada requerida para ejecutar el caso 9](#_Toc478864902)

[3.4.4 Especificaciones de todas las salidas y las características requeridas (por ejemplo, el tiempo de respuesta) para los elementos que se van a probar. 9](#_Toc478864903)

[3.4.5 Necesidades de entorno 9](#_Toc478864904)

[3.4.6 Requisitos especiales de procedimiento 9](#_Toc478864905)

[3.4.7 Dependencias entre casos 9](#_Toc478864906)

[4 Procedimientos de prueba 10](#_Toc478864907)

[4.1 Identificador único de la especificación y referencia a la correspondiente especificación de diseño de prueba. 10](#_Toc478864908)

[4.2 Objetivo del procedimiento y lista de casos que se ejecutan con él. 10](#_Toc478864909)

[4.3 Requisitos especiales para la ejecución (por ejemplo, entorno especial o personal especial). 10](#_Toc478864910)

[4.4 Pasos en el procedimiento 10](#_Toc478864911)

[5 Anexo: plantillas 10](#_Toc478864912)

# 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.

### Documentos relacionados

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

## Elementos software a probar

Como ya se ha mencionado en la introducción, los elementos a probar son los módulos de importación, estadístico, control y DAO a través de pruebas de caja negra sobre sus interfaces.

## Características a probar

Dado que son pruebas funcionales, las características de las interfaces a probar son las entradas pertinentes a sus métodos. Se pretende comprobar, mediante una serie de casos de prueba planteados mediante partición, que para una entrada dada se obtiene la salida esperada.

## Características que no se prueban

Partiendo de que se utilizan pruebas de caja negra, todas las características a nivel estructural del software (caja blanca) no se probarán en esta versión del software. Además, puesto que se ha planteado centrar la búsqueda de errores en las interfaces, no se realizarán pruebas centradas en las funciones incorrectas o ausentes, estructuras de datos o acceso a BBDD, rendimiento o errores de inicialización y terminación.

## Enfoque general de la prueba

Dada la importancia de los formatos para los distintos atributos expuestos en los requisitos, las pruebas se centraron en asegurar su correcta interpretación. Para ello se utilizó principalmente el método de partición para crear casos de prueba.

Como segunda técnica se usó el análisis por valores límite y, con menor frecuencia, la conjetura de errores.

Teniendo en cuenta que las pruebas se centran en generación de clases del modelo, muchos de los métodos se consideraron similares entre sí, asumiendo que si uno de ellos es correcto sería apropiado pensar que el resto también lo serán. Sin embargo, atendiendo al teorema de Pareto, en el caso de que uno de estos métodos falle, será conveniente ampliar los casos de prueba a todos sus métodos similares.

## Criterios de paso/fallo para cada elemento

## Criterios de suspensión y requisitos de reanudación

## Documentos a entregar

## Actividades de preparación y ejecución de pruebas

### Necesidades de entorno

## Responsabilidades en la organización y realización de las pruebas

## Necesidades de personal y de formación

## Esquema de tiempos

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

|  |  |  |
| --- | --- | --- |
| **Riesgo** | **Descripción** | **Contingencia** |
| Fuerte limitación temporal para el planteamiento de las pruebas. | Intervalo de tiempo escaso para diseñar el plan de pruebas. | Generación de pruebas de caja negra sin excesivo nivel de detalle para cumplir las fechas de entrega. |
| Cambios en los requisitos o diseños originales. | Incoherencias y falta de información en el documento de análisis y diseño. | Diálogo intergrupal para solucionar posibles malas interpretaciones. |
| Recursos con escaso tiempo para asignar a las pruebas. | Recursos con poca disponibilidad. | Redistribución de la carga entre  recursos más disponibles. |

## 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

## InterfazImportación

### Prueba I-01

#### Objetivo

Comprobar que un archivo con líneas incompletas, con demasiados campos, o en blanco no provoca un comportamiento inesperado en el programa.

#### 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.

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*).

#### Técnicas de caja blanca

#### 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. Cuando un archivo contenga líneas incorrectas se debe cancelar la importación informando al usuario del error. Por otra parte, un archivo con líneas correctas debe ser correctamente procesado.

#### Resultado de aplicar las técnicas

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

### Prueba I-02

#### Objetivo

Comprobar que un archivo que contenga un campo con un formato o valor incorrecto no provoca un comportamiento inesperado en el programa.

#### Técnicas de caja negra

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

* **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.

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

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.

#### Técnicas de caja blanca

#### Criterios de paso/fallo

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

#### Resultado de aplicar las técnicas

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

### Prueba I-03

#### 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.

#### Técnicas de caja blanca

#### 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.

#### Resultado de aplicar las técnicas

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

## InterfazDAO

### 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.

#### Técnicas de caja blanca

#### 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.

#### Resultado de aplicar las técnicas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Información** | **Tipo de dato** | **Regla** | **Clase válida** | **Clase no válida** |
| insertUser | Usuario | R3 | {1} Id válido. | {2} Id inválido |
|  |  | R5,R1 | {3} Cadena de 6 letras en el Id | {4} Cadena de 5 letras |
|  |  |  |  | {5} Cadena de 7 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. |

### 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.

#### Técnicas de caja blanca

#### 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.

#### Resultado de aplicar las técnicas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 | {7} Id de ítem existente | {6}Id de ítem no existente |

### Prueba D-03

#### Objetivo

Comprobar que la gestión del login de usuario es correcta.

#### Técnicas de caja negra

##### Generación de clases de equivalencia

Solo se proporciona el método isLoged para la gestión del estado de la sesión de usuario.

* **isLoged**
  + **R3**: Id de un usuario existente y un ID de un usuario no existente.
  + **R5**: En base al tipo de ID
    - **R3**: Introducir un ID de un usuario, un ID válido diferente (item o venta) y un ID inválido.

#### Técnicas de caja blanca

#### Criterios de paso/fallo

Cuando se introduzca un usuario correcto debe indicarse adecuadamente su estado de sesión. En caso de introducir dátos erróneos debe notificarse sobre ello.

#### Resultado de aplicar las técnicas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Información** | **Tipo de dato** | **Regla** | **Clase válida** | **Clase no válida** |
| isLoged | String | R3 | {1} Id de usuario existente | {2} Id de usuario no existente |
|  |  | R5,R3 | {3} Id válido | {4} Id inválido |

### Prueba D-04

#### 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.

En base a la definición de la interfaz no tenemos forma de seleccionar el pedido a validar ni el usuario que realiza la validación (para comprobar la gestión de las autorizaciones). Por todo esto solo podremos realizar pruebas en función del estado introducido.

#### Técnicas de caja negra

##### Generación de clases de equivalencia

Solo el metodo validarPedido gestiona esta acción.

* **validarPedido**
  + **R3**: Cadena válida y cadena vacía.
  + **R4**: Marcar pedido como aceptado, rechazado o una cadena inválida.

#### Técnicas de caja blanca

#### 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.

#### Resultado de aplicar las técnicas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Información** | **Tipo de dato** | **Regla** | **Clase válida** | **Clase no válida** |
| validarPedido | String | R3 | {1} Cadena válida | {2} Cadena vacía |
|  |  | R4 | {3} Pedido aceptado | {4} Cadena inválida |
|  |  |  | {5} Pedido rechazado |  |

### Prueba D-05

#### Objetivo

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

#### Técnicas de caja negra

##### Generación de clases de equivalencia

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

* **getHistorialUsuario**
  + **R3**: Id válido de usuario e Id inválido.
  + **R3**: Id de usuario existente e inexistente.

#### Técnicas de caja blanca

#### 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.

#### Resultado de aplicar las técnicas

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

## InterfazEstadistica

### Prueba D-06

#### 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 todos los métodos 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.

* **getValoresBrutos / getMedias / getHistos / getPorcentajes**
  + **R4**: Introducir cada uno de los modos correctos {1,2,3} y uno incorrecto.

#### Técnicas de caja blanca

#### 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.

#### Resultado de aplicar las técnicas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Información** | **Tipo de dato** | **Regla** | **Clase válida** | **Clase no válida** |
| getValoresBrutos / getMedias / getHistos / getPorcentajes | int | R4 | {1} Modo 1 | {2} Nº fuera del rango [1,3] |
|  |  |  | {3} Modo 2 |  |
|  |  |  | {4} Modo 3 |  |

## InterfazControlador

### Prueba C-01

#### Objetivo

Comprobar que la búsqueda de productos funciona correctamente.

#### Técnicas de caja negra

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

* **R3**: introducir una cadena con el identificador de un producto existente y una de uno que no esté registrado pero que cumpla el formato especificado.
* **R5**: los identificadores de productos son de la 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 identificador.
  + **R4**: que N sea {U, I, D} u otro carácter.

#### Técnicas de caja blanca

#### Criterios de paso/fallo

En ninguno de los casos se debe generar una excepción no controlada. Cuando el argumento sea correcto la búsqueda se efectuará apropiadamente.

#### Resultado de aplicar las técnicas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Información** | **Tipo de dato** | **Regla** | **Clase válida** | **Clase no válida** |
| Producto | String | R3 | {1} Identificador existente. | {2} Identificador no existente. |
| R5, R1 | {3} Cadena de 6 letras para la parte alfabética del ID. | {4} Cadena de 5 letras para la parte alfabética del ID |
| {5} Cadena de 7 letras para la parte alfabética del ID |
|  |  | R5, R1 | {6} 3 dígitos para la parte numérica del ID. | {7} 2 dígitos para la parte numérica del ID. |
| {8} 4 dígitos para la parte numérica del ID. |
|  |  | R5, R3 | {9} Todos los campos del ID están presentes. | {10} No todos los campos del ID están presentes. |
|  |  | R4 | {10} N es U | {13} N es un carácter distinto de U, I o D. |
| {11} N es I |
| {12} N es D |

# Casos de pruebas (CP) de caja negra

## InterfazImportacion

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

#### Dependencias

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

1, 4

#### Resultado esperado

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

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

#### Dependencias

Valida la correcta interpretación de una línea que contenga una cantidad de campos inferior a la correcta.

#### Definición

Se llama al método *importarUsuarios()*

*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

#### Dependencias

Valida la correcta interpretación de una línea que contenga una cantidad de campos superior a la correcta.

#### Definición

Se llama al método *importarUsuarios()*

*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

#### Dependencias

Valida la correcta interpretación de una línea en blanco.

#### Definición

Se llama al método *importarUsuarios()*

*(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-02-P-01

#### Dependencias

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

1, 4, 7, 8, 10, 12

#### Resultado esperado

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

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

#### Dependencias

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.

#### Definición

Se llama al método *importarCompra()*.

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

#### Clases que valida

14

#### 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-02-P-03

#### Dependencias

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. 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()*.

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

#### Clases que valida

16

#### 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-02-P-04

#### Dependencias

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 2, 3, 4, 5 y 6.

#### Definición

Se llama al método *importarUsuarios()*

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

#### Clases que valida

2

#### Resultado esperado

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

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

#### Dependencias

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 2, 3, 4, 5 y 6.

#### Definición

Se llama al método *importarUsuarios()*

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

#### Clases que valida

3

#### Resultado esperado

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

### Caso de prueba I-02-P-06

#### Dependencias

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.

#### Definición

Se llama al método *importarUsuarios()*

*U; U-aaaaaa-000; 10-10-2010; Samuel; 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-02-P-07

#### Dependencias

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.

#### Definición

Se llama al método *importarCompra()*

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

#### Clases que valida

11

#### Resultado esperado

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

### Caso de prueba I-02-P-08

#### Dependencias

Valida la correcta interpretación de una línea cuyo identificador ya se encuentra en la base de datos. El usuario *U-aaaaaa-000* se encuentra previamente en la base de datos. En caso de que la prueba falle, se efectuarán también pruebas para cubrir la clase 15.

#### Definición

Se llama al método *importarUsuarios()*

*U; U-aaaaaa-000; 10/10/2010; 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-02-P-09

#### Dependencias

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.

#### Definición

Se llama al método *importarCompra()*.

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

#### Clases que valida

17

#### Resultado esperado

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

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

#### Dependencias

Valida el correcto comportamiento del módulo de importación cuando se le pasa la ruta de un archivo existente. Se presupone que el fichero /*home/usuario/fichero.csv* existe y es un fichero de importación válido. Si la prueba falla, se ejecutará contra todos los métodos de la interfaz.

#### Definición

* 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-03-P-02

#### Dependencias

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

#### Definición

* 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-03-P-03

#### Dependencias

Valida el correcto comportamiento del módulo de importación cuando se le pasa la ruta de un archivo que no existe. Se presupone que el fichero “*/home/usuario/ficheroInexistente.csv*” no existe.

#### Definición

* 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.

## InterfazDAO

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

#### Dependencias

Valida la inserción correcta de un usuario.

#### Definición

Usuario:

| U-abcdef-000 | Manuel | Soutoullo | 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

#### Dependencias

Valida la inserción correcta de un usuario.

#### Definición

Usuario:

| x | Manuel | Soutoullo | 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

#### Dependencias

Valida la inserción correcta de un usuario.

#### Definición

Usuario:

|U-abcde-000 | Manuel | Soutoullo | 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

#### Dependencias

Valida la inserción correcta de un usuario.

#### Definición

Usuario:

|U-abcdef-0000 | Manuel | Soutoullo | 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

#### Dependencias

Valida la inserción correcta de un usuario.

#### Definición

Usuario:

|U-abcdef-| Manuel | Soutoullo | 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

#### Dependencias

Valida la inserción correcta de un usuario.

Se supone que la base de datos solo cuenta con el usuario "U-aaaaaa-000", nombre Limón Novoa, fecha 10-10-2010 y tipo estudiante.

#### Definición

Usuario:

|U-aaaaaa-000 | Manuel | Soutoullo | 10-10-2010 | alumno |

#### Clases que valida

12

#### Resultado esperado

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

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

#### Dependencias

Valida la inserción correcta de un usuario.

Se supone que la base de datos solo cuenta con el usuario "U-aaaaaa-000", nombre Limón Novoa, fecha 10-10-2010 y tipo estudiante.

#### Definición

Usuario:

|U-dddddd-111 | Manuel | Soutoullo | 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

#### Dependencias

Valida la inserción correcta de un item.

#### Definición

Item:

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

#### Clases que valida

1

#### Resultado esperado

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

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

#### Dependencias

Valida la inserción correcta de una venta.

#### Definición

Venta:

|V-abcdef-000 | 10/10/2010 | 20 |

#### Clases que valida

3

#### Resultado esperado

Inserción de la venta con id: V-abcdef-000 en la base de datos.

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

#### Dependencias

Valida la actualización correcta de un item. Suponemos que el único item que existe en la base de datos es |I-abcdef-000 | Robot limpiapiscinas | Limpia piscinas | Exteriores | 50 |10/10/2010|

#### Definición

Item:

|I-abcdef-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-04

#### Dependencias

Valida la inserción correcta de un item.

#### Definición

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-05

#### Dependencias

Valida la inserción correcta de una venta.

#### Definición

Venta:

| x | 10/10/2010 | 20 |

#### 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-06

#### Dependencias

Valida la actualización correcta de un item.

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

#### Definición

Item:

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

#### Clases que valida

5

#### 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

#### Dependencias

Valida si el usuario introducido está logeado en el sistema.

Suponemos que el único usuario existente en la base tiene el id: I-abcdef-000.

#### Definición

Id usuario:

I-abcdef-000

#### Clases que valida

1,3

#### Resultado esperado

Indica el correcto estado de login del usuario.

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

#### Dependencias

Valida si el usuario introducido está logeado en el sistema.

Suponemos que el único usuario existente en la base tiene el id: I-abcdef-000.

#### Definición

Id usuario:

I-abcdef-999

#### Clases que valida

2

#### Resultado esperado

Muestra un error indicando que el usuario no existe. Si simplemente indica que no está logueado se considerará que no pasa la prueba.

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

#### Dependencias

Valida si el usuario introducido está logeado en el sistema.

Suponemos que el único usuario existente en la base tiene el id: I-abcdef-000.

#### Definición

Id usuario:

x

#### Clases que valida

4

#### Resultado esperado

Muestra un error indicando que el usuario no existe. Si simplemente indica que no está logueado se considerará que no pasa la prueba.

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

#### Dependencias

Valida si el usuario introducido está logeado en el sistema.

#### Definición

Estado:

Aceptar

#### Clases que valida

1,3

#### Resultado esperado

El pedido pasa a ser una compra en la base de datos.

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

#### Dependencias

Valida si el usuario introducido está logeado en el sistema.

#### Definición

Estado:

Rechazar

#### Clases que valida

5

#### Resultado esperado

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

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

#### Dependencias

Valida si el usuario introducido está logeado en el sistema.

#### Definición

Estado:

“ “

#### Clases que valida

2

#### Resultado esperado

Se muestra un error indicando que el estado introducido no es correcto.

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

#### Dependencias

Valida si el usuario introducido está logeado en el sistema.

#### Definición

Estado:

abcd

#### Clases que valida

4

#### Resultado esperado

Se muestra un error indicando que el estado introducido no es correcto.

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

#### Dependencias

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

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

#### Definición

Id usuario:

U-abcdef-000

#### Clases que valida

1,3

#### Resultado esperado

Se recupera el historial de compras del usuario.

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

#### Dependencias

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

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

#### Definición

Id usuario:

U-a-000

#### Clases que valida

2

#### Resultado esperado

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

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

#### Dependencias

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

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

#### Definición

Id usuario:

U-abcdef-999

#### Clases que valida

4

#### Resultado esperado

Se muestra un error indicando que el usuario no existe.

### Caso de prueba D-05-P-04

#### Dependencias

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

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

#### Definición

Id usuario:

U-abcdef-123

#### 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.

## InterfazEstadistica

### Nota

Dado que realizando todos los casos de prueba teniendo en cuenta las distintas entradas y métodos de la interfaz serían 16 casos, se ha decido probar 1 entrada por método. La entrada incorrecta se probará en el método que calcula los valores brutos de ventas debido a su sencillez (es una consulta directa a un elemento de la BBDD sin la realización de ningún tipo de operación).

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

#### Dependencias

Valida la consulta de ventas totales diarias.

#### Definición

Modo:

-3

#### Clases que valida

2

#### Resultado esperado

El resultado será un mensaje de error indicando que se ha realizado la consulta de manera incorrecta (se ha introducido un modo erróneo).

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

#### Dependencias

Valida la consulta de la media de ventas semanales.

#### Definición

Modo:

2

#### Clases que valida

3

#### Resultado esperado

El resultado será la media de ventas semanales.

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

#### Dependencias

Valida la consulta del histograma de ventas mensuales.

#### Definición

Modo:

3

#### Clases que valida

4

#### Resultado esperado

El resultado será el histograma de ventas mensuales.

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

#### Dependencias

Valida la consulta de los porcentajes de las ventas diarias.

#### Definición

Modo:

1

#### Clases que valida

4

#### Resultado esperado

El resultado será el porcentaje correspondiente a las ventas diarias.

## InterfazControlador

# Procedimientos de prueba

## Identificador único de la especificación y referencia a la correspondiente especificación de diseño de prueba.

## Objetivo del procedimiento y lista de casos que se ejecutan con él.

## Requisitos especiales para la ejecución (por ejemplo, entorno especial o personal especial).

## Pasos en el procedimiento

# Anexo: plantillas