iSoft Empresa

PROYECTO DE PÁGINA WEB DE VENTA DE INDUMENTARIA DEPORTIVA

Documento de Diseño

Autores: Rebola, Christian

Vazquez, Franco

Versión del documento: 1.0.0

*Historial de revisiones y modificaciones*

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Fecha** | **Cambio** | **Autor** |
| 1.0.0 | 22-Jun-2017 | Versión inicial | Rebola Christian  Vazquez Franco |
|  |  |  |  |
|  |  |  |  |

*Indice de contenidos*

1. Diagramas…………....……………………..……………………………………………..pág 4

Diagrama de clases…………………………………………………………….………pág 4

Diagrama de secuencia………………………………………………………..….…pág 6

1. Patrones utilizados……….…………………………………………….……………...pág 7

Patron Observer…………………………………………………………………………pág 7

Patron Singleton…………………………………………………………………………pág 8

Patron Strategy..…………………………………………………………………………pág 8

1. Pruebas……………………………..….….…………………………………………………pág 9

Pruebas Unitarias y de Integración….…………………………………………pág 9

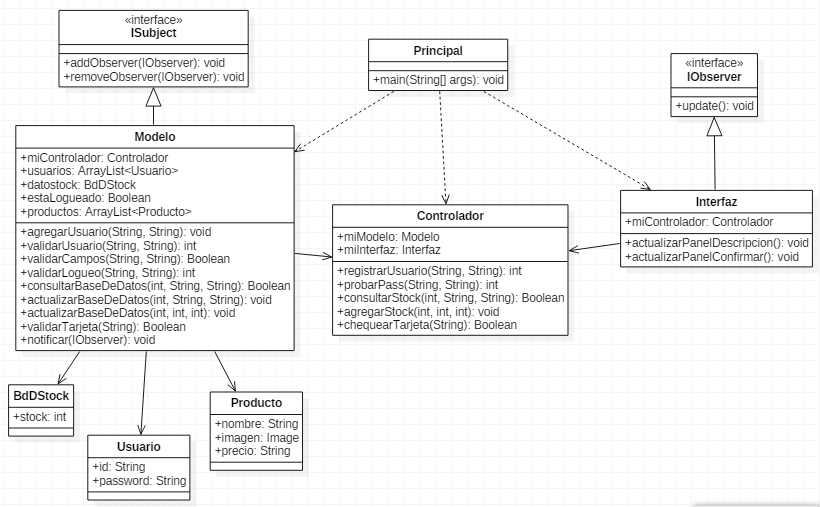
Pruebas de sistema..………..………………………………………………………pág. 11

1. Matriz de trazabilidad…………………………………………..…………………pág 12

**Arquitectura**

**DIAGRAMA DE CLASES**

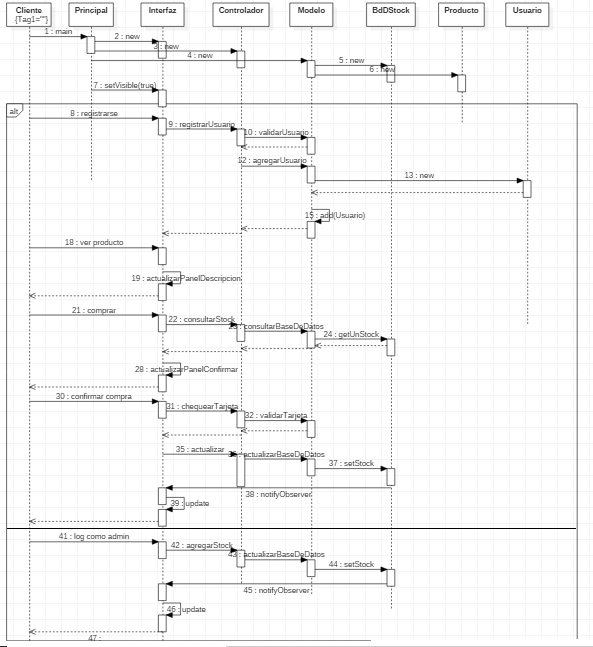
A continuación se enseña el diagrama de clases del sistema completo, en el cual podemos ver la estructura estática del sistema.



Podemos identificar 7 clases concretas, más 2 interfaces:

* Clase Modelo: Es la clase más importante del programa. En ella se alojan todos los datos del sistema, y toda la lógica que este implementa. Esta clase también tiene acceso a la base de datos de productos.
* Clase BdDStock: La base de datos del inventario. En ella se aloja la información de stock para cada producto del catalogo actual.
* Clase Producto: Define un producto del catálogo.
* Clase Usuario: Define un usuario cliente registrado. Cuando se da de alta al sistema, se crea un nuevo usuario y se lo agrega a la lista de usuarios del modelo.
* Clase Interfaz: Aquí se define la parte visual del sistema, se establecen todas las ventanas, paneles o interfaces gráficas de usuario, mediante las cuales representamos todo el modelo permitiendo la interacción entre la aplicación y el cliente.
* Clase Controlador: Esta clase es el puente entre la clase interfaz y modelo. Cuando se ejecuta alguna acción en la interfaz, esta llama a algún método de su controlador adjunto (tanto las clase Interfaz como la clase Modelo cuenta con métodos get y set para indicar cuál es su controlador). Con esto, la interfaz solo se preocupa por lo que se muestra al usuario.
* Clase Principal: Esta clase contiene el método main que ejecuta la aplicación. Este llama a su método interno iniciar(), el cual inicializa el Modelo, la Interfaz y el Controlador y crea las relaciones entre estos. Por último, hace visible la interfaz para que el usuario pueda empezar a interaccionar con el sistema.

**DIAGRAMA DE SECUENCIA**

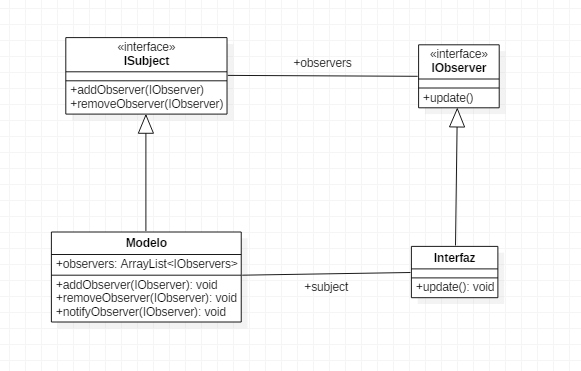


**PATRONES DE DISEÑO UTILIZADOS**

Los patrones de diseño utilizados fueron 2: El patrón Observer y el patrón Singleton.

Patrón Observer

Se ha utilizado Observer en las clases Modelo (subject) e Interfaz (observer) como se muestra a continuación:



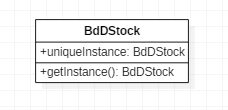
Se ha utilizado este patrón ya que la interfaz toma datos del modelo, que es donde se alojan todos los datos del sistema. Este patrón es fundamental en el patrón de arquitectura con el que estamos trabajando, el MVC, ya que es necesario que la vista sea notificada de manera automática cuando el modelo cambie de estado.

Con este patrón, además de proporcionarnos una manera práctica y entendible de actualizar la Interfaz, también podemos agregar diferentes vistas (por ejemplo, una segunda interfaz separada) sin necesidad de preocuparnos por estar actualizando cada una por separado cada vez que el modelo cambie de estado.

Patrón Singleton

La intención de este patrón consiste en garantizar que una clase sólo tenga una [instancia](https://es.wikipedia.org/wiki/Instancia_(inform%C3%A1tica)) y proporcionar un punto de acceso global a ella.

Nosotros lo usamos en la clase BdDStock:



Esta clase tiene un campo uniqueInstance del mismo tipo de la clase. Su constructor es privado, asi que solo puede llamarse dentro de la misma clase. Por ultimo, implementa el método getInstance para crear la única instancia posible de la clase, o si esta ya esta creada, devolvernos esa instancia.

**private** **static** BdDStock *uniqueInstance* = **null**;

**private** BdDStock(){}

**public** **static** BdDStock getInstance(){

**if** (*uniqueInstance*==**null**){

*uniqueInstance*= **new** BdDStock();

}

**return** *uniqueInstance*;

}

Esta clase soluciona el problema de que, por alguna incongruencia en el código, vayamos a crear más de una base de datos del inventario. Esto podría generar inconsistencias en el programa.

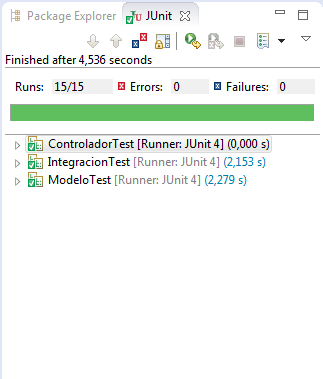
Patrón Strategy

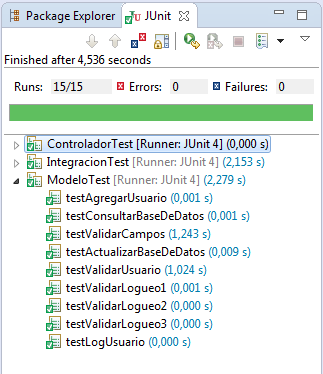
Al implementar solo una clase Interfaz (una sola vist, la cual incluye todos los paneles que se muestran al usuario), vimos innecesaria la utilización de este patrón

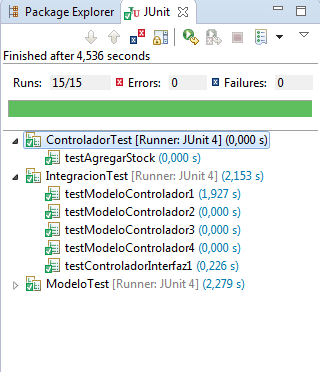
**PRUEBAS UNITARIAS Y DE INTEGRACIÓN**

Se realizaron Unit Test sobre todo el código creado por nosotros. Para ello se utilizó JUnit 4, tanto para los tests unitarios como los test de integración

Para correr los test se necesita tener un compilador de Java y ejecutarlos desde la misma, de esa forma saldrá un mensaje detallando los Test que fallaron y los test que fueron exitosos. En este caso usamos la herramienta de Eclipse creada para este fin:

****

****

****

Como se puede observar, todos los tests tanto unitarios como de integración, pasaron de forma exitosa.

La cobertura abarcada por los Unit Test fue de un 80% del código que implementamos, solo quedaron sin testear los métodos de las clases más simples como Usuario, Producto y BdDStock, además de la clase Interfaz que solo se limita a inicializar y cambiar las vistas.

El **Pass/Fail rate de los Test fue del 100%** por lo que podemos decir que en esta sección no se encontró ningún bug.

**PRUEBAS DE SISTEMA**

Se realizaron las pruebas de sistema con respecto a los casos de prueba definidos anteriormente:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID Caso de prueba | Descripción | Prerrequisitos | Resultado esperado | Resultado obtenido | Estado |
| CP01 | Verificar que la interfaz responda a entradas por mouse | - | OK | OK | Concluido |
| CP02 | Verificar que la interfaz responda a entradas por teclado | - | OK | OK | Concluido |
| CP03 | Acceder al sistema como cliente anónimo y confirmar la compra de un producto | - | Falla | Falla | Concluido |
| CP04 | Acceder al sistema con un usuario ya creado | Dar de alta un nuevo usuario. Desloguearse | OK | OK | Concluido |
| CP05 | Dar de alta un nuevo usuario con un nombre de usuario ya usado | Dar de alta un usuario con el mismo nombre | Falla | Falla | Concluido |
| CP06 | Registrarse sin validar el check box “He leído los términos y condiciones” | - | Falla | Ok | Concluido |
| CP07 | Loguearse con una cuenta de usuario inexistente | - | Falla | Falla | Concluido |
| CP08 | Dar de alta un usuario con una contraseña de menos de 4 caracteres. | - | Falla | Falla | Concluido |
| CP09 | Dar de alta un usuario con un nombre de usuario nulo | - | Falla | Falla | Concluido |
| CP10 | Confirmar la compra de más camisetas de las que hay en stock | Loguearse en el sistema  Setear stock del producto | Falla | Falla | Concluido |
| CP11 | Verificar que la tabla en el panel del admin se condiga con los datos en la base de datos | Modificar la base de datos comprando algunos productos | OK | OK | Concluido |
| CP12 | Introducir un numero de Stock no entero o negativo | Acceder al sistema como administrador | Falla | Falla | Concluido |
| CP13 | Ingresar un Nº de tarjeta de crédito no numérico | Loguearse en el sistema. Comprar un producto | Falla | Falla | Concluido |
| CP14 | Ingresar un Nº de tarjeta de crédito menor a 8 cifras | Loguearse en el sistema. Comprar un producto | Falla | Falla | Concluido |
| CP15 | Dar de alta un usuario con caracteres especiales | - | Falla | OK | Concluido |

Como se observa en la tabla, se han concluido las pruebas para todos los casos de prueba definidos en el documentos de requerimientos.

El **Pass/Fail rate de los Test fue del 87%**, ya que de las 15 pruebas, hubo 2 que fallaron. Estos defectos serán corregidos en la próxima versión del sistema

**MATRIZ DE TRAZABILIDAD**

Para los casos de test unitarios:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | testLogUsuario | testValidarCampos | testValidarUsario | testValidarLogueo | testAgregarUsuario | testActualizarBdD | testConsultarBdD |
| Alta Usuario |  | X | X |  | X |  |  |
| Consulta Catalogo |  |  |  |  |  |  |  |
| Consulta Producto |  |  |  |  |  |  |  |
| Autenticar Usuario | X |  |  | X |  |  |  |
| Confirmar Compra |  |  |  |  |  | X | X |
| Alta Producto |  |  |  |  |  | X |  |
| Baja Producto |  |  |  |  |  | X |  |
| Modificar Producto |  |  |  |  |  | X |  |

Para los casos de prueba:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | CP01 | CP02 | CP03 | CP04 | CP05 | CP06 | CP07 | CP08 | CP09 |
| Alta Usuario | X | X |  |  | X | X |  | X | X |
| Consulta Catalogo | X | X |  |  |  |  |  |  |  |
| Consulta Producto | X | X |  |  |  |  |  |  |  |
| Autenticar Usuario | X | X |  | X |  |  | X |  |  |
| Confirmar Compra | X | X | X |  |  |  |  |  |  |
| Alta Producto | X | X |  |  |  |  |  |  |  |
| Baja Producto | X | X |  |  |  |  |  |  |  |
| Modificar Producto | X | X |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | CP10 | CP11 | CP12 | CP13 | CP14 | CP15 |
| Alta Usuario |  |  |  |  |  | X |
| Consulta Catalogo |  |  |  |  |  |  |
| Consulta Producto |  |  |  |  |  |  |
| Autenticar Usuario |  |  |  |  |  |  |
| Confirmar Compra | X |  |  | X | X |  |
| Alta Producto |  | X |  |  |  |  |
| Baja Producto |  | X |  |  |  |  |
| Modificar Producto |  | X | X |  |  |  |