

Patrones software

Gestión de almacenes



María Rodrigo González

Alejandro Niso Sánchez

Índice

[Descripción del proyecto 2](#_Toc125304948)

[Requisitos del proyecto 3](#_Toc125304949)

[Requisitos funcionales 3](#_Toc125304950)

[Requisitos no funcionales 7](#_Toc125304951)

[Manual de usuario 9](#_Toc125304952)

[Patrones empleados 14](#_Toc125304953)

[Patrón de creación – Singleton 14](#_Toc125304954)

[Patrón de creación – Builder 15](#_Toc125304955)

[Patrón de creación – Factory Method 16](#_Toc125304956)

[Patrón estructural – Facade 17](#_Toc125304957)

[Patrón estructural – Proxy 18](#_Toc125304958)

[Patrón de comportamiento – Strategy 19](#_Toc125304959)

[Patrón de comportamiento – Template Method 20](#_Toc125304960)

[Patrón de comportamiento – Memento 21](#_Toc125304961)

# Descripción del proyecto

Se ha creado un proyecto encargado de realizar la gestión de un almacén. Este proyecto facilita a los empleados del dicho almacén llevar un registro de entradas de productos y, también, de los pedidos realizados por determinados clientes.

A la vez, se puede consultar la información de los clientes, los proveedores y los productos, al igual que se les puede dar de alta o de baja, e incluso se puede ordenar y buscar ya sea por un id o por un determinado nombre.

Está aplicación está pensada para un entorno local, una conexión local, que, únicamente sirva para facilitar la actividad de trabajo de los almacenes. Cuenta, en este caso, con una base de datos que contiene las siguientes tablas con los siguientes atributos: la tabla **Clientes**, que es la encargada de almacenar la información de aquellos que realizarán una compra de productos, cuyos atributos son su id, nombre, correo, su dirección y teléfono. Bastante parecida es la tabla **Proveedor** que tiene los mismos atributos, y representa a los que suministran productos al almacén. **Producto** cuenta con un id, el nombre del producto, nombre y una descripción, pues los productos, depende del tipo que sean pueden ser inflamables, normales, pesados o frágiles.

Las otras tablas ya reflejan la propia actividad laboral del establecimiento. Se muestra una tabla de **Entrada** donde se mostrarán las entradas de productos en el almacén, contando así con el id del proveedor, el id del producto, la cantidad y la fecha. Por el otro lado, tenemos la salida de productos en forma de **Pedido**, aunque, esta vez cuenta con el id del cliente que realiza dicho pedido. Para finalizar, contamos con una tabla de las **Existencias** que se tienen.

El sistema cuenta tanto con un inicio de sesión para aquellos empleados que ya estén dados de alta, como con un registro de usuario, para aquellos que no. Una vez se ha podido acceder, se muestra un menú principal que muestra diferentes opciones.

La primera es un menú de gestión para clientes, donde se puede dar de alta a uno de ellos, rellenando un formulario con sus datos. Se puede dar de baja a un cliente también por su id, realizando una búsqueda en la base de datos y eliminándolo. También se pueden ordenar y buscar los clientes que se encuentran en el sistema, ya sea por su id o su nombre.

Los menús de gestión de proveedores y productos son prácticamente iguales que el anterior, puesto que lo que se pretende realizar es bastante similar a la gestión de clientes. Lo que cambian son las consultas realizadas a la base de datos.

En cuanto a los menús de entrada y pedido, en ambos se puede registrar una nueva entrada o un nuevo pedido, pero en el de pedido se puede terminar de completar uno ya existente.

El proyecto se ha realizado en java, y se ha empleado Java Swing para crear las interfaces de la aplicación.

Para la creación de la base de datos, se ha empleado PostgreSQL que, con la herramienta pgAdmin4, se han terminado de modelar las tablas y las relaciones.

# Requisitos del proyecto

## Requisitos funcionales

|  |  |
| --- | --- |
| Id\_requisito | RF01 |
| Nombre | Iniciar sesión |
| Descripción | Un usuario podrá acceder al sistema usando su nombre de usuario y contraseña, una vez ya registrado. |

|  |  |
| --- | --- |
| Id\_requisito | RF02 |
| Nombre | Registro de usuario |
| Descripción | El usuario podrá registrarse en el sistema eligiendo su nombre de usuario y contraseña, además de usar un correo. |

|  |  |
| --- | --- |
| Id\_requisito | RF03 |
| Nombre | Dar de alta cliente |
| Descripción | Registrar los datos del cliente en la base de datos. |

|  |  |
| --- | --- |
| Id\_requisito | RF04 |
| Nombre | Dar de alta usuario proveedor |
| Descripción | Registrar los datos del proveedor en la base de datos. |

|  |  |
| --- | --- |
| Id\_requisito | RF05 |
| Nombre | Dar de alta producto |
| Descripción | Registrar los datos del producto. |

|  |  |
| --- | --- |
| Id\_requisito | RF06 |
| Nombre | Dar de baja cliente |
| Descripción | Eliminar los datos del cliente de la base de datos. |

|  |  |
| --- | --- |
| Id\_requisito | RF07 |
| Nombre | Dar de baja proveedor |
| Descripción | Eliminar los datos del proveedor en la base de datos. |

|  |  |
| --- | --- |
| Id\_requisito | RF08 |
| Nombre | Dar de baja producto |
| Descripción | Eliminar el producto de la base de datos. |

|  |  |
| --- | --- |
| Id\_requisito | RF09 |
| Nombre | Búsqueda de cliente |
| Descripción | Realizar una búsqueda de un cliente en la base de datos según ciertos filtros. |

|  |  |
| --- | --- |
| Id\_requisito | RF10 |
| Nombre | Búsqueda de proveedor |
| Descripción | Realizar una búsqueda de un proveedor en la base de datos según ciertos filtros |

|  |  |
| --- | --- |
| Id\_requisito | RF11 |
| Nombre | Búsqueda de producto |
| Descripción | Realizar una búsqueda de un producto en la base de datos según ciertos filtros |

|  |  |
| --- | --- |
| Id\_requisito | RF12 |
| Nombre | Registro de producto (entrada) |
| Descripción | Se realizará una inserción en la base de datos del pedido hecho al proveedor en concreto. |

|  |  |
| --- | --- |
| Id\_requisito | RF13 |
| Nombre | Registro de pedido (salida) |
| Descripción | Se realiza una inserción en la base de datos del pedido realizado por el cliente. (Habrá un campo que indique si ha sido o no completado) |

|  |  |
| --- | --- |
| Id\_requisito | RF14 |
| Nombre | Histórico de pedidos de entrada (de los proveedores) |
| Descripción | Se registrarán en un documento todos los productos que hemos comprado a los proveedores |

|  |  |
| --- | --- |
| Id\_requisito | RF15 |
| Nombre | Histórico de pedidos de salida (de los clientes) |
| Descripción | Se registrarán en un documento todos los productos que hemos vendido a los clientes |

|  |  |
| --- | --- |
| Id\_requisito | RF16 |
| Nombre | Ordenar Clientes |
| Descripción | Se mostrarán por pantalla todos los clientes existentes en el sistema ordenados por diferentes campos a elección del usuario |

|  |  |
| --- | --- |
| Id\_requisito | RF17 |
| Nombre | Ordenar Proveedores |
| Descripción | Se mostrarán por pantalla todos los proveedores existentes en el sistema ordenados por diferentes campos a elección del usuario |

|  |  |
| --- | --- |
| Id\_requisito | RF18 |
| Nombre | Ordenar Productos |
| Descripción | Se mostrarán por pantalla todos los productos existentes en el sistema ordenados por diferentes campos a elección del usuario. |

## Requisitos no funcionales

|  |  |
| --- | --- |
| Id\_requisito | RNF01 |
| Nombre | Disponibilidad de la aplicación |
| Descripción | La aplicación estará en uso para cualquier empleado mientras que el almacén esté abierto. |

|  |  |
| --- | --- |
| Id\_requisito | RNF02 |
| Nombre | Disponibilidad multiplataforma |
| Descripción | El sistema se podrá ejecutar en los sistemas operativos principales. |

|  |  |
| --- | --- |
| Id\_requisito | RNF03 |
| Nombre | Bases de datos |
| Descripción | El sistema deberá funcionar sobre una base de datos PostgreSQL. |

|  |  |
| --- | --- |
| Id\_requisito | RNF04 |
| Nombre | Lenguaje de programación |
| Descripción | El lenguaje de programación a utilizar será Java. |

|  |  |
| --- | --- |
| Id\_requisito | RNF05 |
| Nombre | Hardware |
| Descripción | La aplicación no requiere más que unos requisitos mínimos de memoria, como 1 Gb de RAM, y unos 500 Mb de almacenamiento. |

|  |  |
| --- | --- |
| Id\_requisito | RNF06 |
| Nombre | Ataques |
| Descripción | Al tratarse de una aplicación en remoto, no será necesario el uso de wifi, por lo que la mayoría de los ataques están descartados. En caso de detectarse alguno, se detendrá el sistema hasta ser evaluado por un profesional del área de informática. |

|  |  |
| --- | --- |
| Id\_requisito | RNF07 |
| Nombre | Protección de datos |
| Descripción | Almacene en el sistema estará amparado por la Ley Orgánica de Protección de Datos española. |

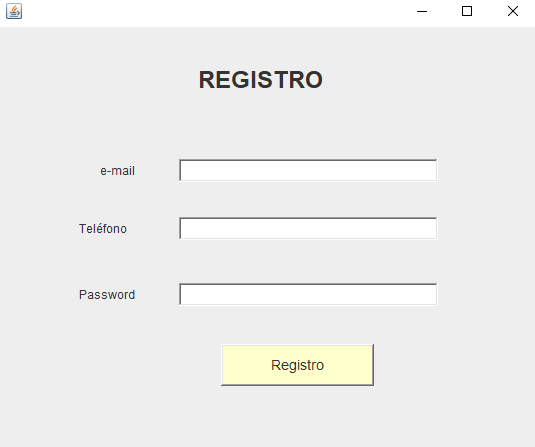
|  |  |
| --- | --- |
| Id\_requisito | RNF08 |
| Nombre | Backup de históricos |
| Descripción | Ante la posibilidad de una posible pérdida de datos o algún ataque “manual”, se llevará un backup de las entradas y de los pedidos en un txt que será utilizado para recuperar el sistema frente a pérdidas |

# Manual de usuario

El usuario/empleado que desee acceder y realizar uso de la aplicación, desde un primer momento, deberá o bien registrarse si es un nuevo empleado del almacén, o bien iniciar sesión, lo cuál significará que ya ha sido dado de alta previamente y no es un nuevo empleado.



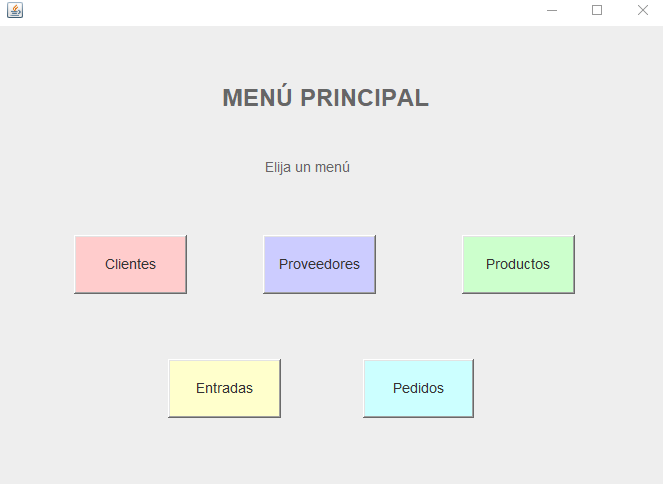
En la página de registro, se deberá introducir el correo electrónico, el teléfono y la contraseña que elija el usuario.



En cuanto al inicio de sesión, bastará con el correo y la contraseña.



Una vez ya metidos en el sistema, accederemos a un menú principal. Este menú principal a su vez cuenta con otros 5 pequeños menú o submenú.



Empezaremos por el de clientes.



Este menú, al igual que los otros dos menús siguientes, cuenta con distintas funcionalidades, entre ellas, dar de alta a un nuevo cliente, también se puede dar de baja a un cliente por su id, mostrar de forma ordenada a los clientes que tiene el almacén, ya sea por su id o por su nombre, y finalmente también se podrá buscar a uno o varios clientes, también pudiendo ser por su id y/o nombre.

Menú de proveedores:



Menú de productos:



Para dar de alta la entrada de un determinado producto en el almacén, mediante el menú de entradas, podremos rellenar el formulario que se nos muestra siempre indicando el id del producto que previamente se ha debido registrar en el sistema, junto con el id del proveedor que lo suministra y la cantidad indicada.



Finalmente, contamos con el menú de pedidos, con el cual podemos dar de alta un determinado pedido, o, incluso, poder completar un pedido ya existente.

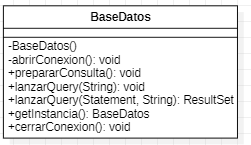


Todas las páginas de los menús cuentan con un botón para acceder al menú principal tantas veces como se quiera.

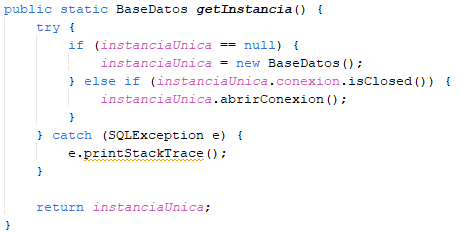
# Patrones empleados

## Patrón de creación – Singleton

Se ha usado el patrón de creación Singleton para la creación e instanciación constante de la base de datos como si fuese un objeto. Al fin y al cabo, se ha creado la clase que, a su vez, formaliza y crea la base de datos como única instancia para ser llamada por cualquiera de las clases de la aplicación.



El código que asegura que la instancia sea única en toda la aplicación es el siguiente:

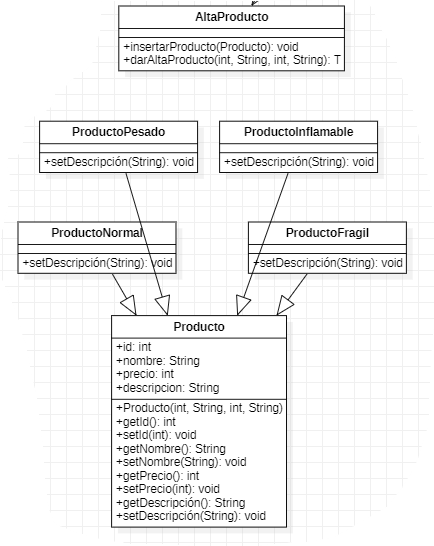


## Patrón de creación – Builder

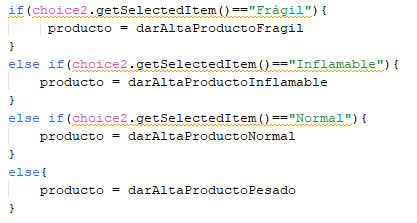
Para este apartado, hemos empleado el patrón Builder para la creación de productos. En concreto 4 productos, los cuales heredan de una clase padre Producto.

A la hora de introducir un producto en el sistema, según sea su descripción, siendo las disponibles “producto frágil”, “producto pesado”, “producto normal” o “producto inflamable”, se dará de alta un producto de cualquiera de los 4 posibles y se procederá a insertarlo en la base de datos.

Todos los productos tienen descripciones diferentes, pero cuentan con numerosos atributos compartidos que son iguales entre sí. Es por ello que hemos decidido hacer uso de este patrón.



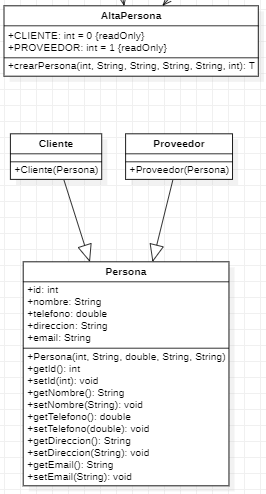
El código que decide qué tipo de producto es creado en función de la elección del usuario en el menú desplegable es:



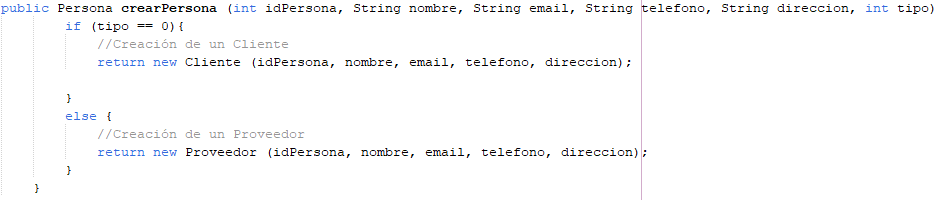
## Patrón de creación – Factory Method

Se emplea el patrón factory method para la creación de personas. En función del tipo de persona, siendo 0 un cliente y 1 un proveedor, se creará un elemento de tipo Cliente o Proveedor.

Además, tanto clientes como proveedores tienen numerosos atributos comunes, por lo que se ha decidido que deben heredar de la clase Persona para poder reutilizar código.

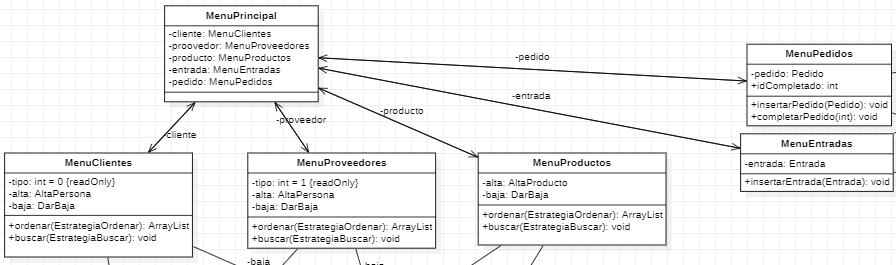


El código que implementa esta funcionalidad es:



## Patrón estructural – Facade

Se ha empleado el patrón facade o fachada a la hora de crear un menú principal. Este menú aparece tras haber iniciado sesión apropiadamente en el sistema y se puede ver que toda la actividad de la aplicación está centralizada en dicho menú, es decir, en esta página principal se puede mostrar los diferentes menús de gestión con los que cuenta la aplicación, pudiendo acceder a cualquiera de ellos y realizar las diferentes acciones permitidas.



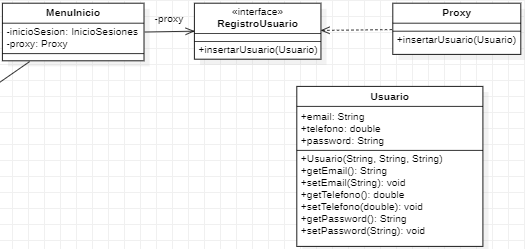
En cada uno de los botones de la interfaz del menú principal se tiene como acción el menú correspondiente a la elección hecha por el usuario, como se puede apreciar en el código:



## Patrón estructural – Proxy

El patrón proxy se ha utilizado como Proxy de protección. Se ha utilizado una interfaz de Registro usuario que implementa el método **insertarUsuario** y este a su vez utiliza la clase Usuario, que es la que está protegida.

Tras esto, contamos con la clase Proxy que implementa la interfaz previamente mencionada e incorpora y desarrolla el método citado.

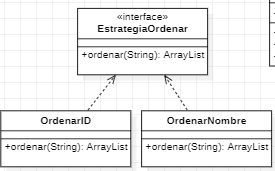


## Patrón de comportamiento – Strategy

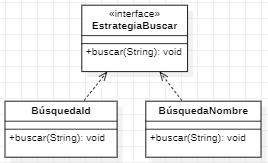
Para el uso de este patrón hemos realizado dos tipos de estrategias:

* Una estrategia para ordenar.
* Una estrategia para buscar.

En cuanto a la estrategia de ordenar, contamos con su correspondiente interfaz (EstrategiaOrdenar) que posteriormente implementamos en las clases OrdenaId y OrdenaNombre.



Y, por otro lado, sucede lo mismo con la estrategia de buscar, pues la interfaz (EstrategiaBuscar) se implementa en las clases de BusquedaId y BusquedaNombre.



Las clases que implementan estos métodos (MenuClientes, MenuProveedores y MenuProductos), contendrán las operaciones para realizarlo:

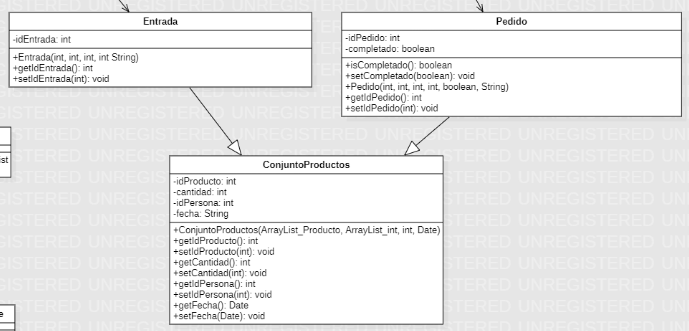


## Patrón de comportamiento – Template Method

Para la implementación de este patrón, contamos con una clase que actúa a modo de plantilla, la cuál es **ConjuntoProductos**, que contiene la mayoría de los atributos comunes de las clases que heredan de la misma.

La clase **Entrada** hereda de la clase padre previamente mencionada, pero esta contiene su id de entrada propio.

La clase **Pedido** también hereda de **ConjuntoProductos**, pero esta difiere de **Entrada** en que tiene un id de pedido y el booleano que indica si un pedido está completado o no.



El código que evidencia el uso de este patrón es el siguiente:

* ConjuntoProductos contiene todos los atributos comunes:

Texto

Descripción generada automáticamente

* Entrada extiende de ConjuntoProductos e incorpora el atributo particular idEntrada:

Imagen que contiene Texto

Descripción generada automáticamente

* Pedido extiende de ConjuntoProductos e incorpora los dos atributos particulares idPedido y completado:

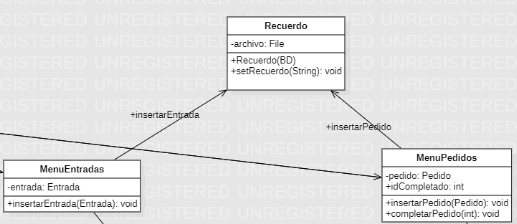
Texto

Descripción generada automáticamente

## Patrón de comportamiento – Memento

Para emplear el patrón Memento, hemos usado la clase de **MenuPedidos** para crear un recuerdo, objeto de la propia clase **Recuerdo**. Una vez se inserte un pedido en la base de datos, se registrará dicho pedido y se guardará en un archivo de texto.

Ese archivo de texto lo declaramos en el constructor del objeto **Recuerdo**.



Como se puede apreciar, también registraremos en el recuerdo las entradas de productos en el almacén.