APLICACIÓN JAVA EE para SUPERMERCADO ONLINE

REALIZADO POR	FIRMA	FECHA	REVISADO POR	FIRMA	FECHA
Pablo Almohalla Gómez **Pedro Monje Onandía**					

HISTORIA DE LAS MODIFICACIONES

REVISIÓN	FECHA REVISIÓN	FECHA APROBACIÓN	MOTIVO REVISIÓN
1	10/01/2022		Se han añadido la arquitectura y diseño detallado de la aplicación
2	10/01/2022		Se ha añadido el plan de pruebas
3	11/01/2022		Se ha modificado el plan de pruebas
4	14/01/2022		Se ha modificado el diagrama de arquitectura
5	17/01/2022		Versión Final

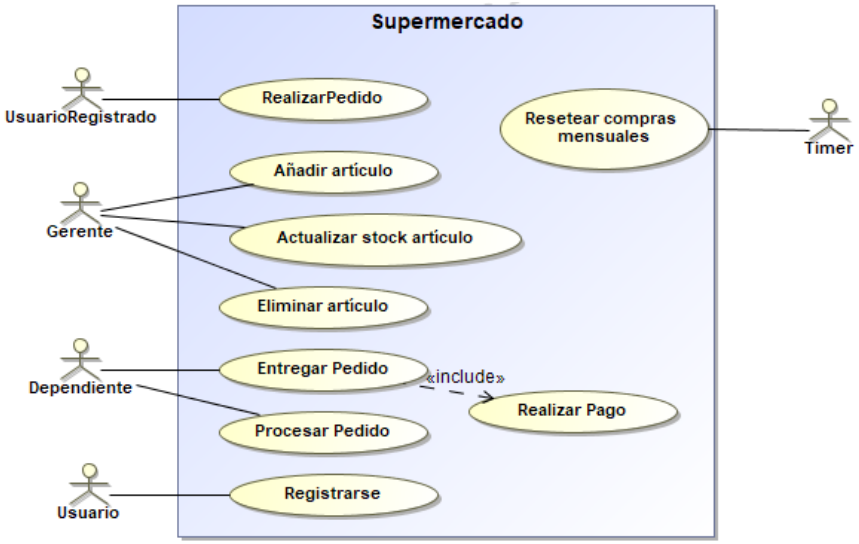
ÍNDICE

1. DESCRIPCIÓN DEL SISTEMA

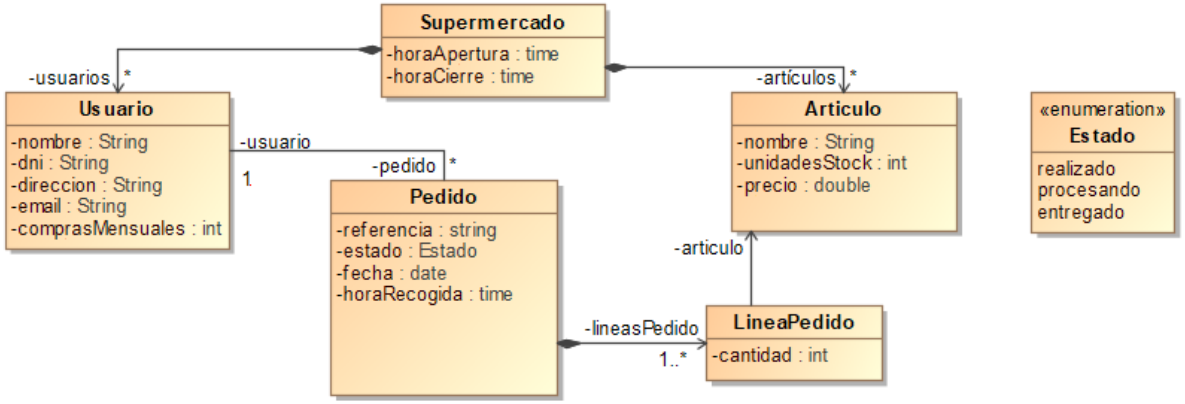
Se quiere desarrollar una aplicación para la realización de compras en un pequeño supermercado de barrio que ha decidido facilitar el encargo de pedidos.

El objetivo de la aplicación es que los usuarios puedan hacer sus pedidos a través de la web, cuya funcionalidad principal es gestionar la realización y entrega de pedidos de los clientes y gestionar el inventario de los artículos disponibles.

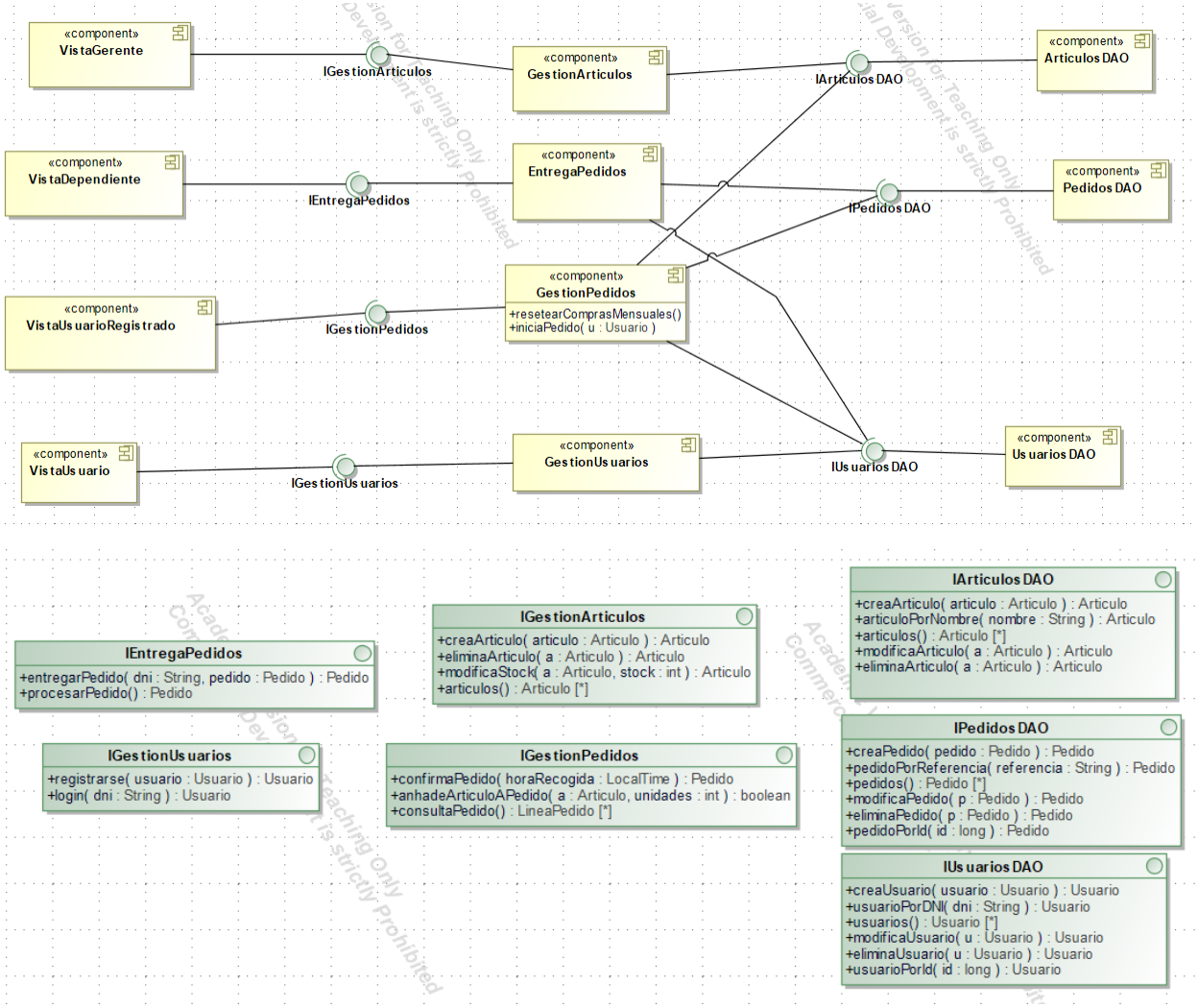
2. DIAGRAMA CASOS DE USO



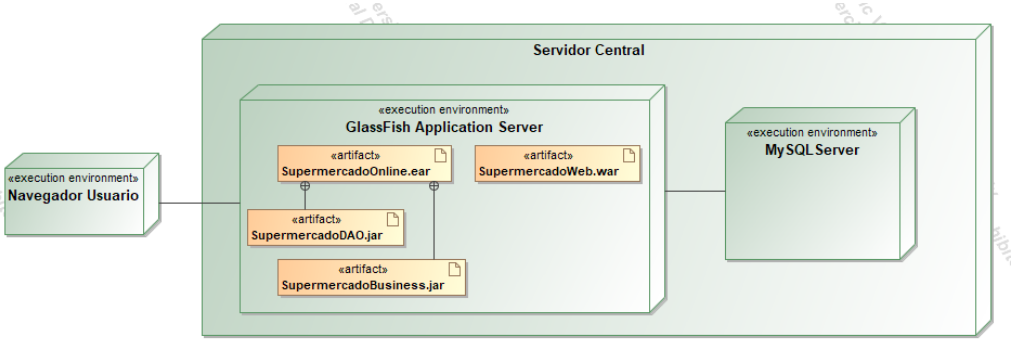
3. MODELO DE DOMINIO



4. ARQUITECTURA DE LA APLICACIÓN



5. DIAGRAMA DE DESPLIEGUE



6. PLAN DE PRUEBAS

Los niveles de prueba que se van a aplicar son los siguientes:

- **Pruebas de aceptación.** Las pruebas de aceptación se definirán siguiendo una estrategia basada en casos de uso y se ejecutarán de forma manual para comprobar el correcto funcionamiento de la aplicación, de manera complementaria se utilizará Selenium para automatizar uno de los casos de prueba.
- **Pruebas de integración.** La estrategia para la definición del orden de las pruebas de integración será jerárquica. Se probará:
    - La integración entre la capa de negocio y la de persistencia. En este caso, para la definición de los casos de prueba se utilizarán técnica de métodos y caja negra y se utilizará JUnit y el EJB Container.
    - La integración entre las tres capas. En este caso, para la definición de los casos de prueba se utilizarán técnica de casos de uso y se desplegarán las capas en Glassfish con el fin de comprobar su correcta integración y funcionamiento.
- **Pruebas unitarias.** Se utilizará la técnica de prueba de métodos, usando técnicas de caja negra para la definición de los casos de prueba de cada método de cada clase o componente. Será necesaria la utilización de JUnit y Mockito.

A continuación, se muestra una especificación detallada de los casos de prueba a aplicar en cada nivel mencionado anteriormente.

6.1. PRUEBAS DE ACEPTACIÓN

En base a los casos de uso se identifican los siguientes escenarios:

A1. CU: Realizar pedido

a. Pedido válido (usuario existente, stock suficiente, horario de recogida válido)

b. Pedido válido (usuario existente, stock suficiente, horario de recogida válido, con descuento por más de 10 compras)

c. Pedido no válido (usuario existente, añadir un artículo del pedido con stock insuficiente, horario de recogida válido)

d. Pedido no válido (usuario inexistente)

e. Pedido no válido (horario de recogida inválido)

Los casos de prueba definidos para cada uno de estos escenarios, suponiendo que el estado inicial del sistema es el siguiente:

Usuario registrado:

Jose Manuel García Sánchez, 27516415G, josemanuel@gmail.com, Calle Castilla 36

El sistema también contiene los siguientes elementos:

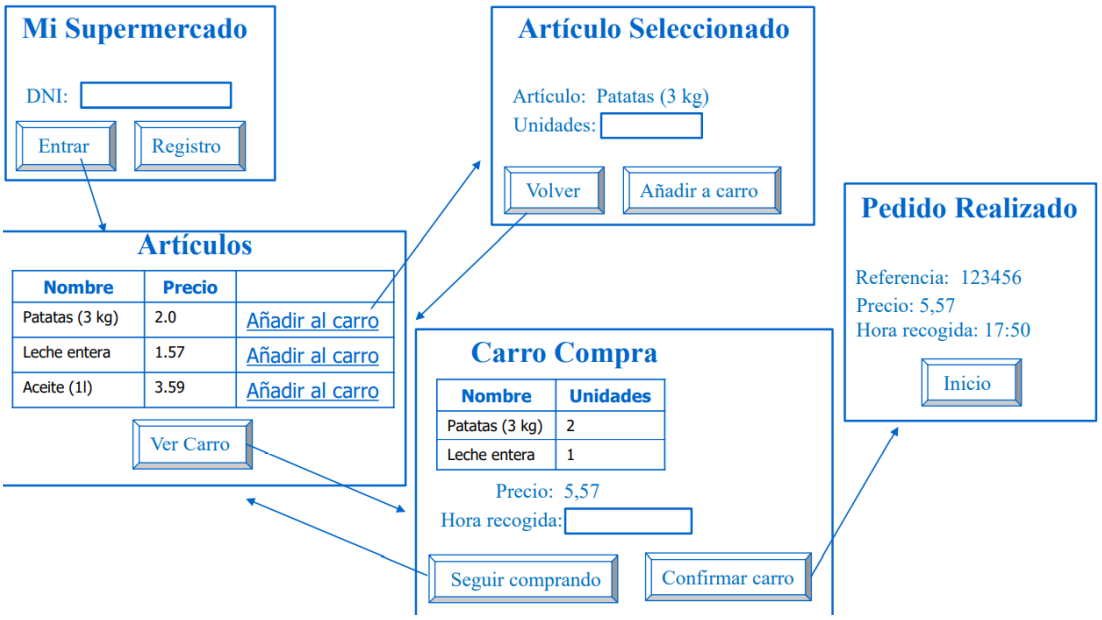


Tabla 1. Casos de prueba de aceptación (ejecutados en este orden)

Identificador	Entrada	Resultado
A1.a	27516415G, 1 ud patatas, 17:50h	Pedido válido
A1.b	27516415G (con 11 compras), 1 ud patatas, 17:50h	Pedido válido
A1.c	27516415G, 5000 ud patatas, 17:50h	Stock insuficiente
A1.d	11111111A, 1 ud patatas, 17:50h	Usuario inexistente
A1.e	27516415G, 1 ud patatas, 23:00h	Fecha inválida

6.2. PRUEBAS DE INTEGRACIÓN

El orden las pruebas y los casos de prueba a realizar sería el siguiente:

1. *Pruebas de integración de la capa DAO*. Los casos de prueba son los mismos que los de las pruebas unitarias nombradas UID.x pero se parte de una situación inicial con una base de datos en la que se tiene un artículo (1 kg de patatas, 30 unidades de stock y 5 euros de precio).
2. *Pruebas de integración de la capa de negocio GestionPedidos con IArticulosDAO, IUsuariosDAO e IPedidosDAO*. Los casos de prueba son los mismos que los de las pruebas unitarias nombradas UGP.x pero se parte de una situación inicial con una base de datos en la que se tiene los datos declarados en el apartado anterior referente a los casos de uso.

6.3. PRUEBAS UNITARIAS

- **Pruebas unitarias de la capa DAO ArticulosDao**

  Se aplica prueba de métodos, siendo los casos de prueba definidos para cada método los siguientes y partiendo de una situación inicial en la que se tiene un artículo (1 kg de patatas, 30 unidades de stock y 5 euros de precio):

Tabla 2. Método Articulo creaArticulo(Articulo a)

Identificador	Entrada	Valor esperado
UID.1a	Pan,30 unidades,0.3 euros	Devuelve el Artículo Pan ya creado
UID.1b	1 kg de patatas,30 unidades de stock y 5 euros	null

Tabla 3. Método Articulo articuloPorNombre(String nombre)

Identificador	Entrada	Valor esperado
UID.2a	1 kg de patatas	Devuelve el Artículo 1 kg de patatas
UID.2b	Solomillo	null

Tabla 4. Método List<Articulo> articulos()

Identificador	Entrada	Valor esperado
UID.3a	Situación inicial, tienes el Artículo 1 kg de patatas	Lista con 1 elemento (el artículo 1 kg de patatas)

UID.3b	No tienes ningún Artículo creado	Lista Vacía

Tabla 5. Método Articulo modificaArticulo(Articulo a)

Identificador	**Entrada**	**Valor esperado**
UID.4a	Artículo 1 kg de patatas	Devuelve el Artículo 1 kg de patatas.
UID.4b	Artículo Vino	Null, debido a que el artículo no existe en la BBDD.

Tabla 6. Método Articulo eliminaArticulo(Articulo a)

Identificador	**Entrada**	**Valor esperado**
UID.5a	Artículo 1 kg de patatas	Devuelve el Artículo 1 kg de patatas.
UID.5b	Artículo Vino	Null, debido a que el artículo no existe en la BBDD.

Tabla 7. Método Articulo articuloPorId(long id)

Identificador	**Entrada**	**Valor esperado**
UID.6a	Id del Artículo 1 kg de patatas	Devuelve el Artículo con esa id (1 kg de patatas).
UID.6b	Id de un Artículo que no está en la BBDD	null

- **Pruebas unitarias de la capa de negocio GestionPedidos**

Para poder llevar a cabo estas pruebas, será necesario el uso de mocks para las interfaces IArticulosDAO, IUsuariosDAO e IPedidosDAO. Se aplica prueba de métodos, siendo los casos de prueba definidos para cada método los siguientes:

Tabla 8. Método confirmaPedido(LocalTime horaRecogida)

Identificador	**Entrada**	**Valor esperado**
UGP.1a	27516415G, 1 ud patatas, 17:50h	Pedido con 1 ud de patatas
UGP.1b	11111111A, 1 ud patatas, 17:50h	null (Usuario no existente)
UGP.1c	27516415G, 1 ud patatas, 23:00h	null (hora inválida)

Tabla 9. Método anhadeArticuloAPedido(Articulo a, int unidades)

Identificador	Entrada	Valor esperado
UGP.2a	(null, 1)	false
UGP.2b	(Patatas, 2)	true
UGP.2c	(Patatas, 5000)	false
UGP.2d	(Patatas, -3)	false

Tabla 10. Método consultaPedido()

Identificador	Entrada	Valor esperado
UGP.3a	Sin líneas de pedido	null
UGP.3b	Con líneas de pedido	Líneas de pedido

6.4. PRUEBAS IMPLEMENTADAS

Para comprobar el correcto funcionamiento de la **capa de negocio**, se han implementado las **pruebas unitarias** relativas al método **confirmaPedido(LocalTime HoraRecogida)** de **GestionPedidos**. Durante la realización de estas pruebas no se ha encontrado ningún problema.

Para comprobar el correcto funcionamiento de la **capa de persistencia**, se ha implementado una **prueba unitaria** del método **creaArticulo(Articulo a)** de **ArticulosDAO**. Al realizar estas pruebas hemos encontrado dificultades a la hora de mockear el método **persist** del EntityManager, ya que este método es void, pero finalmente se ha conseguido solucionar.

Para corroborar que el funcionamiento conjunto de las **capas de negocio y persistencia** es el esperado, se programa una **prueba de integración** utilizando el contenedor embebido (**EJB Container**). Gracias a esto se verifica el funcionamiento del método **confirmaPedido(LocalTime HoraRecogida)** en un entorno real.
A la hora de realizar esta prueba se han encontrado problemas para utilizar el **EJBContainer** por las instrucciones **lookup** y el sistema **JNDI**.

Por último, las **pruebas de aceptación** se han ejecutado manualmente, y tras comprobar el correcto funcionamiento de la aplicación se ha procedido a automatizar la primera prueba de aceptación (**caso válido**) con **Selenium**.

Pedro Monje Onandía
Pablo Almohalla Gómez