

Tienda Sport Shop



IMARKETPLACE

Índice

- Introducción
- Descripción general
- Mockups
- Patrones de diseño web
- Diagrama de clases
- Metodología y planificación
- Descripción de la implementación
- Problemas encontrados
- Posibles mejoras y ampliaciones

INTRODUCCIÓN:

En este proyecto hemos realizado un proyecto web de una tienda de ropa deportiva, en la que tenemos diferentes páginas y rutas. El modelo a seguir sería el de cualquier tienda de ropa online, con su página principal con ofertas y productos destacados, catálogo de productos con información de los productos, carrito de compras, etc. Además contaremos con todas las posibles gestiones de un posible administrador, con sus páginas adaptadas para modificar cualquier aspecto que este quiera cambiar, es decir los CRUD de cada elemento importante a cambiar en la base de datos (producto, categoría, usuario, ...). Más adelante explicaremos la metodología que hemos utilizado y las tecnologías empleadas para llevar a cabo este proyecto.

DESCRIPCIÓN GENERAL:

- Las tecnologías empleadas para este proyecto son las siguientes:

Editor de código: El editor que hemos usado es IntelliJ IDEA. El entorno de desarrollo IntelliJ IDEA se selecciona para facilitar el desarrollo y la depuración del código. Su conjunto de herramientas avanzadas y su interfaz de usuario intuitiva optimizan el proceso de desarrollo, mejorando la productividad del equipo. Además de esto, es ideal para trabajar con el lenguaje escogido y los frameworks o bases de datos escogidas.

Base de datos: Hemos usado una base de datos relacional con tablas y relaciones entre las tablas. Hemos empleado migraciones para portar la información de la base de datos al proyecto, como las tablas, atributos y las relaciones. Para poblar esas tablas hemos hecho uso de unos seeders, que son como unos datos ficticios de prueba para tener información con la que poder trabajar en la base de datos.

Motor de base de datos:

PostgreSQL: Como sistema de gestión de bases de datos relacional, PostgreSQL se utiliza para almacenar y gestionar la información clave de la tienda, incluyendo detalles de productos, pedidos y usuarios. Proporciona un entorno robusto y seguro para la persistencia de datos.

PGAdmin: Es una herramienta de administración y desarrollo diseñada para trabajar con bases de datos PostgreSQL. En el contexto de nuestro proyecto de tienda de ropa, PGAdmin se utiliza como una interfaz gráfica para gestionar eficientemente la base de datos PostgreSQL.

Backend:

Java: El lenguaje de programación Java es fundamental para el desarrollo del backend de la aplicación. La plataforma Java ofrece robustez y portabilidad, garantizando un rendimiento óptimo y una fácil integración con otras tecnologías. Hemos creado un proyecto con el

framework de java Spring Boot, que proporciona herramientas para crear rutas y apis en el backend, además de clases que gestionan el acceso y administración de la base de datos.

Framework:

Spring Boot: La aplicación se basa en el framework Spring Boot, que simplifica el desarrollo de aplicaciones Java ofreciendo una configuración mínima y una rápida puesta en marcha. Spring Boot facilita la creación de servicios web eficientes y escalables. Este framework nos ha servido tanto para el backend como para el frontend. Todo se gestiona en el mismo proyecto, siguiendo el modelo “vista-controlador-servicio” (modelo-vista-controlador) además del uso de clases que gestionan nuestro proyecto con un paradigma de programación orientada a objetos.

Frontend:

Thymeleaf: Para la creación de las vistas HTML, se emplea Thymeleaf, un motor de plantillas que simplifica la integración de datos dinámicos en las páginas web. Esto permite una presentación atractiva y dinámica de la información para los usuarios finales.

La arquitectura del proyecto es la siguiente:

Estructura del Backend:

Controladores: Clases que manejan las solicitudes HTTP y sirven como punto de entrada al sistema. En estas clases se definen todas las rutas utilizadas tanto internamente como para usar desde el navegador. Estas rutas sirven para la comunicación entre el cliente y el servidor, es decir el frontend y el backend, para poder realizar las operaciones CRUD. También hacen uso de llamadas a servicios y a las vistas con los datos necesarios.

Servicios: Contienen la lógica de negocio y gestionan la interacción entre los controladores y la capa de acceso a datos. Reciben los datos de los controladores y los procesan, gestionan el manejo de estos datos con la base de datos y devuelven el contenido necesario y que se necesita en el controlador.

Modelos (Clases): Representan las estructuras de datos utilizadas en la aplicación, mapeando objetos Java a entidades de la base de datos. Son necesarios para poder pasar objetos a las vistas y mostrarlas en la interfaz de usuario. También se usan para poder representar los datos de la base de datos en forma de objetos con los que podemos trabajar y manipular.

Acceso a Datos y PostgreSQL:

Se utiliza Java Persistence API (JPA) para mapear objetos Java a entidades de base de datos. Las migraciones se emplean para crear y gestionar la estructura de la base de datos. Los seeders proporcionan datos ficticios para poblar las tablas y facilitar el desarrollo.

Frontend:

Thymeleaf: Thymeleaf se utiliza como motor de plantillas para la creación de vistas HTML dinámicas. En estas plantillas se pueden recibir objetos del backend (que representan una entidad de alguna clase con los datos respectivos) y mostrarlos en la interfaz de usuario. De

esta forma podemos hacer uso de algunos elementos de HTML estáticos y algunos elementos dinámicos que son los datos que van cambiando según el objeto recibido.

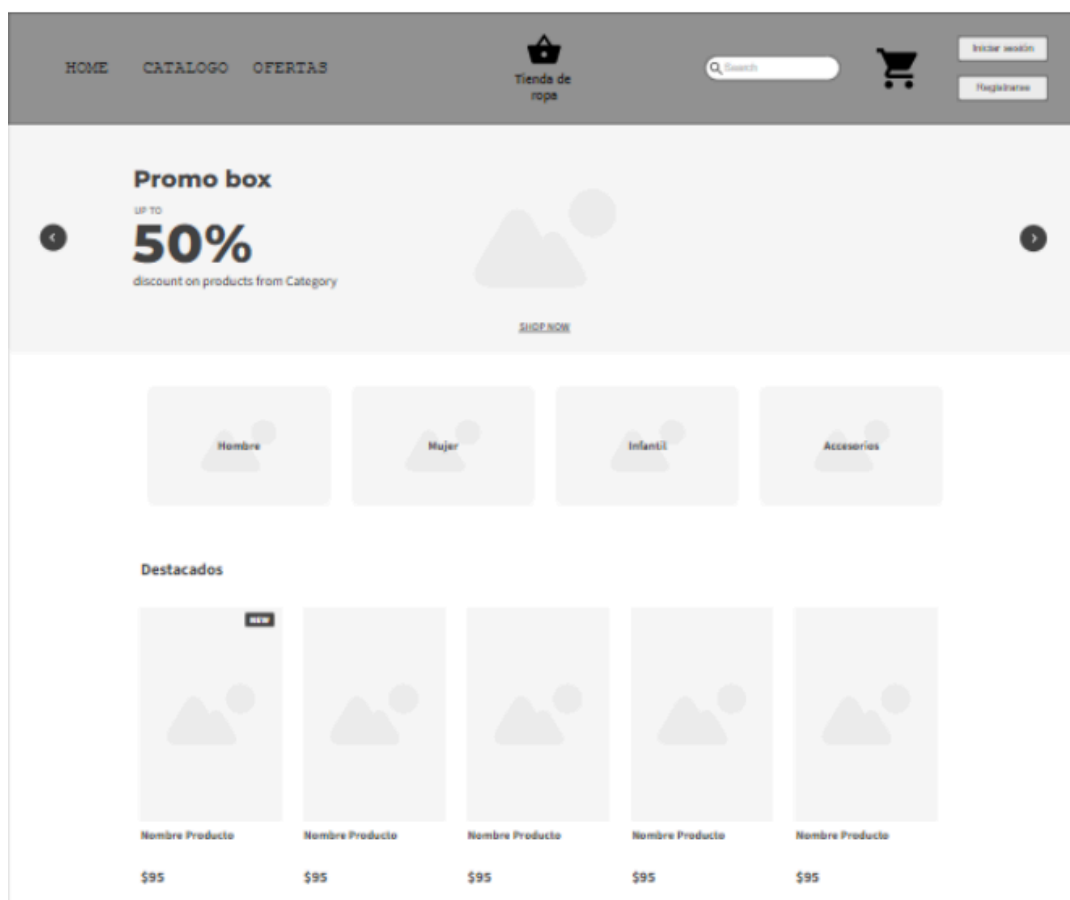
Bootstrap y CSS: Se hace uso del framework de css Bootstrap para un diseño responsivo y estilizado, complementado con CSS puro personalizado para adaptar la interfaz a la identidad visual de la tienda.

JavaScript: Se incorpora muy pocos fragmentos de JavaScript para mejorar la interactividad en ciertos elementos de la interfaz de usuario, proporcionando funcionalidades dinámicas sin depender de bibliotecas externas (este no es el lenguaje principal por ello solo hemos hecho uso en algunos momentos muy puntuales y solo para la interfaz de usuario).

MOCKUPS

Al comienzo del desarrollo del proyecto diseñamos una serie de mockups en los que reflejamos un diseño provisional del aspecto de la página web:

HOME



ABOUT US





CATÁLOGO



DETALLES DEL PRODUCTO

[HOME](#) [CATALOGO](#) [OFERTAS](#)


Tienda de
ropa



[Iniciar sesión](#)
[Registrarse](#)

"nombreProducto"



89.99 €

112 en stock

Referencia: 775-ACV


Talla: M


Color: Rojo

[Añadir al carrito](#)

LOGIN

[HOME](#) [CATALOGO](#) [OFERTAS](#)


Tienda de
ropa



[Iniciar sesión](#)
[Registrarse](#)

Logo de la tienda

INICIO SESION

Nombre de usuario

Contraseña

[He olvidado mi contraseña](#)

[Iniciar Sesión](#)

[¿Aún no tienes cuenta? Regístrate](#)

REGISTRO

[HOME](#) [CATALOGO](#) [OFERTAS](#)


Tienda de
ropa



[Iniciar sesión](#)
[Registrarse](#)

REGISTRO


<input type="text" value="Nombre"/>	<input type="text" value="Apellidos"/>
<input type="text" value="Correo electrónico"/>	<input data-bbox="1114 683 1141 705" type="text" value="Fecha de nacimiento"/>
<input type="text" value="Número de teléfono"/>	<input type="text" value="DNI/NIF"/>
<input type="text" value="Dirección"/>	<input type="text" value="Número de puerta"/> <input type="text" value="Código Postal"/>
<input type="text" value="Repetir contraseña"/>	<input type="text" value="Contraseña"/>


☒ [Aceptar los términos y condiciones](#)


[Registrarse](#)

PAGINA DE USUARIOS

[HOME](#) [CATALOGO](#) [OFERTAS](#)


Tienda de
ropa







Nombre Apellido1 Apellido2
Edad

Cambiar contraseña

<input type="text" value="Contraseña"/>
<input type="text" value="Repetir contraseña"/>

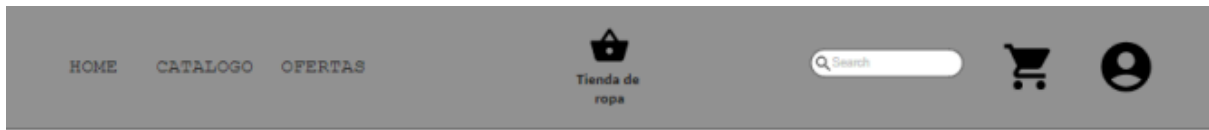
[Cambiar](#)

[Ver direcciones](#)


[Ver Tarjetas](#)

[Ver Pedidos](#)

CARRITO



Carrito

	Nombre producto Descripción, color, talla	
1		\$95 \$70 You save 25%
	Nombre producto Descripción, color, talla	
2		\$54

Resumen

Producto 1	\$149.00
Descuentos	\$-25.00

Total \$124.00














CHECKOUT

*Los gastos de envío se calcularán en el checkout

CHECKOUT





Checkout


<p>Email johndoe@gmail.com</p> <p>Mobile phone +1 0*****09</p>	<p>Address 123*****32 ****50 IL*****</p>	<p><input checked="" type="radio"/> Pick-up point \$16</p> <p>Ship to:</p> <p>Shipping: 2-4 weeks Chadon</p> <p><input type="radio"/> Home delivery \$32</p> <p>Groceries Kiosk, 35 Illinois St, Toronto</p> <p>Shipping: 3-5 weeks</p>	<p><input checked="" type="radio"/> Debit / Cred</p> <p>1234 5678 9012 3456</p> <p>MM / YY CVC 3 digits</p> <p>Name of the card holder</p> <p></p> <p>All transaction are secure and encrypted with SecuritySystem</p> <p><input type="radio"/> Pay</p> <p><input type="radio"/> Apple</p>												
<table><tbody><tr><td></td><td>Nombre Descripción, color, talla</td><td></td></tr><tr><td>1</td><td></td><td>\$95 \$70 You save 25%</td></tr><tr><td></td><td>Nombre Descripción, color, talla</td><td></td></tr><tr><td>2</td><td></td><td>\$54</td></tr></tbody></table>				Nombre Descripción, color, talla		1		\$95 \$70 You save 25%		Nombre Descripción, color, talla		2		\$54	<p>COMPLETE PURCHASE</p>
	Nombre Descripción, color, talla														
1		\$95 \$70 You save 25%													
	Nombre Descripción, color, talla														
2		\$54													

VER PEDIDOS

[HOME](#) [CATALOGO](#) [OFERTAS](#)








PEDIDOS


Pedido1 Fecha: 10/2/2001



Productos:
Camiseta nike, Pantalon adidas

Direccion: Groceries Kiosk, 35 Illinois St, Toronto
Tarjeta

Pedido2 Fecha: 10/2/2001





Productos:
Camiseta nike, Pantalon adidas


Direccion: Groceries Kiosk, 35 Illinois St, Toronto
Tarjeta

VER DIRECCIONES

[HOME](#) [CATALOGO](#) [OFERTAS](#)







Mis direcciones

Direccion 1


Direccion: Groceries Kiosk, 35 Illinois St, Toronto
Localidad: Alicante
Provincia/Estado: Alicante
Código 03008


Direccion 2


Direccion: Groceries Kiosk, 35 Illinois St, Toronto
Localidad: Alicante
Provincia/Estad Alicante
Código 03008

VER TARJETAS

[HOME](#) [CATALOGO](#) [OFERTAS](#)


Tienda de ropa





Mis Tarjetas

Tarjeta 1



Banco: BANCO SABADELL

Numero: **** *54

Pin: ****

Código *14

Tarjeta 1



Banco: BANCO SABADELL

Numero: **** *54

Pin: ****

Código *14

PATRONES DE DISEÑO WEB

Patrones de identificación de usuarios

Tenemos un patrón de registro para almacenar la información de los usuarios:

[Home](#) [About](#) [Catalogo](#)

Buscar por nombre de prod [Login](#) [Registro](#)

Registro

Correo electrónico

Nombre

Contraseña

Apellidos

Teléfono

Código Postal

Pais

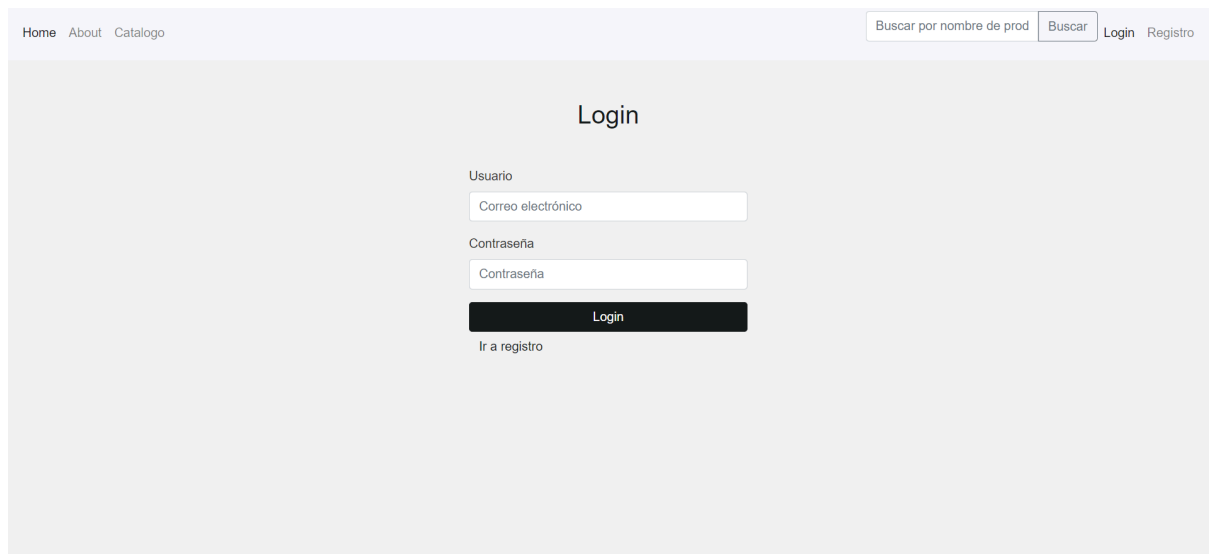
Población

Dirección

Registro

[Ir a login](#)

Después los usuarios tendrán que pasar un formulario con sus datos (patrón login)

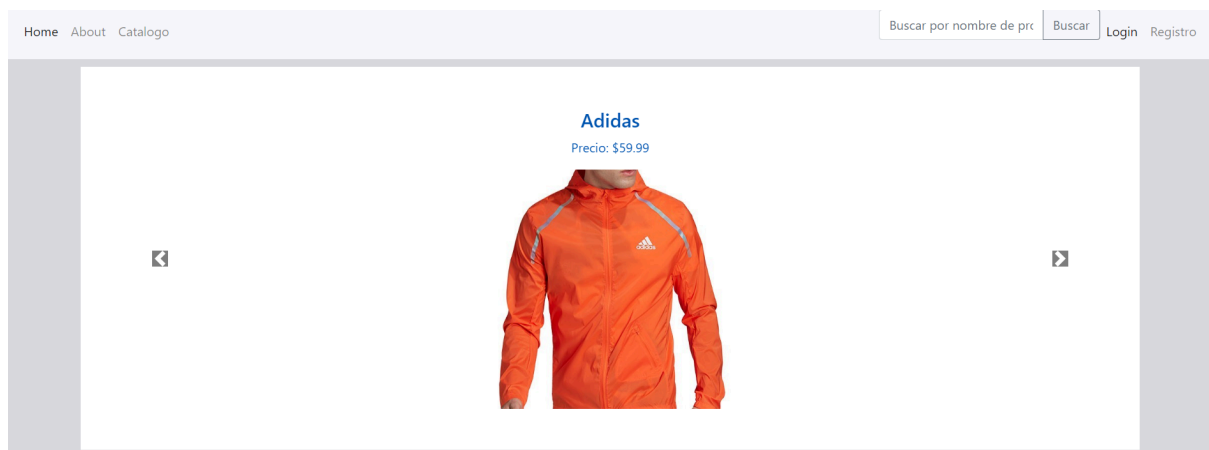


The screenshot shows a web application's login page. At the top, there is a navigation bar with links for 'Home', 'About', and 'Catalogo'. On the right side of the navigation bar, there is a search bar with the placeholder text 'Buscar por nombre de prod', a 'Buscar' button, and links for 'Login' and 'Registro'. The main content area is titled 'Login' and contains a form with two input fields: 'Usuario' (with the placeholder 'Correo electrónico') and 'Contraseña' (with the placeholder 'Contraseña'). Below these fields is a black 'Login' button. At the bottom of the form, there is a link that says 'Ir a registro'.

Finalmente tendrán la opción de desloguearse. Cuando salgan de la aplicación se desplegará automáticamente.

Patrones de página principal

Esto es la vista que encontrará el usuario cuando entre en la aplicación, la página consta de una bandeja de entrada que mostrará artículos destacados y un panel de control que permite al usuario acceder a todas las funcionalidades posibles.



Productos destacados

Patrones de formulario

Para acceder a las funcionalidades de usuario registrado el cliente debe pasar los formularios de registro y login. Cabe destacar que se han elaborado los formularios de manera breve para facilitar la mayor comodidad al usuario.

The screenshot shows a web registration form titled "Registro". At the top, there is a navigation bar with links: "Home", "About", "Catalogo", "Buscar por nombre de prod", "Buscar", "Login", and "Registro". The form itself is centered and contains the following fields:

- Correo electrónico:** A text input field.
- Nombre:** A text input field.
- Contraseña:** A text input field.
- Apellidos:** A text input field.
- Teléfono:** A text input field.
- Código Postal:** A text input field.
- Pais:** A text input field.
- Población:** A text input field.
- Dirección:** A text input field.

Below the input fields is a large black button labeled "Registro". At the bottom of the form, there is a link that says "Ir a login".

Patrones de navegación

La navbar está presente en todas las vistas, en ella hay un menú de primer nivel y otras herramientas de navegación como la barra de búsqueda.

This screenshot shows the navigation bar (navbar) from the previous image. It contains the following elements:

- Links: "Home", "About", "Catalogo", "Buscar por nombre de prod", "Buscar", "Login", and "Registro".

Patrones de listados

En el menú de administrador hay acceso a listas de modelos de datos que se manejan en la aplicación, estas se presentan en un patrón de lista tabular combinado con patrones de acciones en un listado para acceder a los CRUD.

Home About Administración Catalogo Carrito

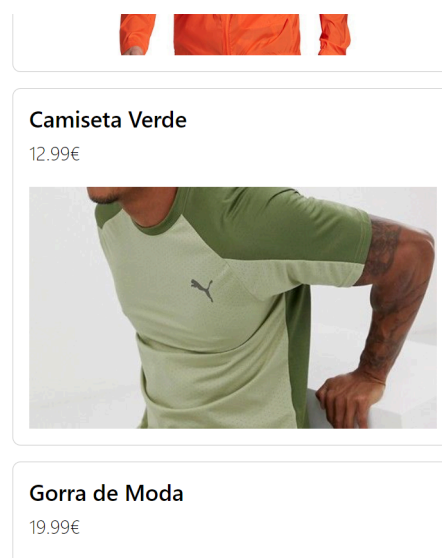
Buscar por nombre de prc admin

[Crear Usuario](#)

Lista de Usuarios

Id	Correo Electrónico	Nombre	Apellidos	Password	Teléfono	Código Postal	País	Población	Dirección		
1	usuario1@example.com	Usuario1	Apellidos1	contraseña1	123456789	12345	País1	Población1	Dirección1	Editar	Eliminar
2	usuario2@example.com	Usuario2	Apellidos2	contraseña2	987654321	54321	País2	Población2	Dirección2	Editar	Eliminar
3	usuario3@example.com	Usuario3	Apellidos3	contraseña3	456789012	67890	País3	Población3	Dirección3	Editar	Eliminar
4	usuario4@example.com	Usuario4	Apellidos4	contraseña4	321098765	54321	País4	Población4	Dirección4	Editar	Eliminar
5	usuario5@example.com	Usuario5	Apellidos5	contraseña5	876543210	98765	País5	Población5	Dirección5	Editar	Eliminar
6	admin@admin.com	admin	admin	123456	123456789	98785	País6	Población6	Dirección6	Editar	Eliminar

En el catálogo hay un patrón de grid de imágenes que muestra varios datos de los productos disponibles.



Patrones de búsqueda en el sitio

En cuanto a los patrones de búsqueda hay de varios tipos como la búsqueda simple anteriormente explicada o el filtrado de productos por categorías.

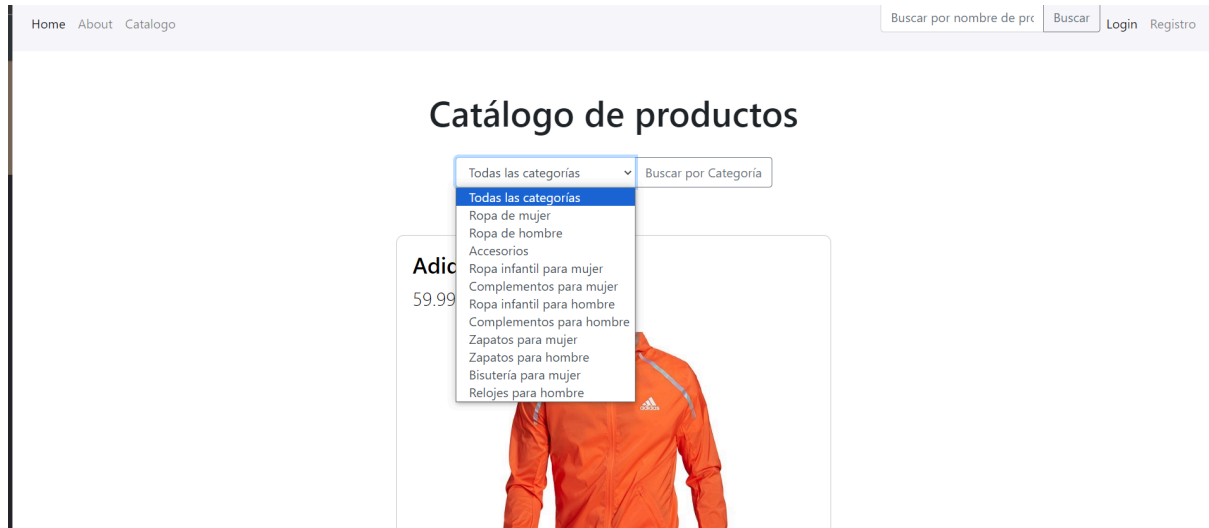
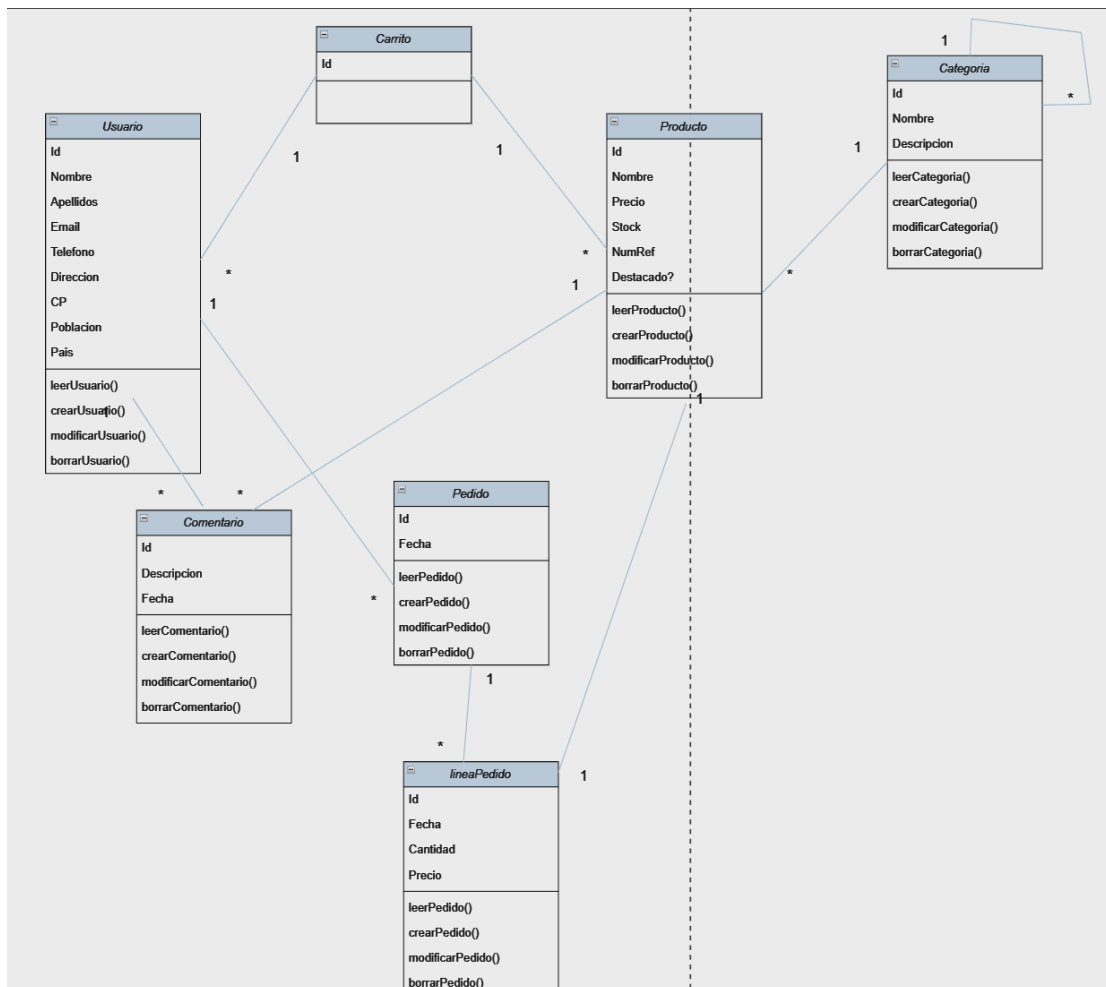


DIAGRAMA DE CLASES:



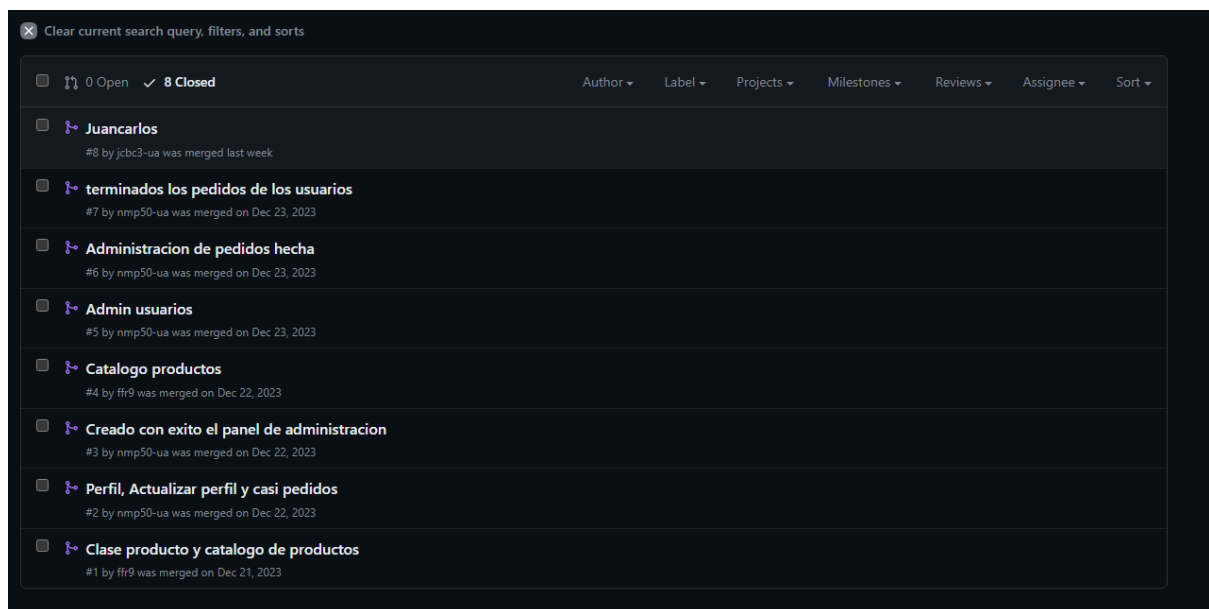
En la base de datos almacenamos los datos de los usuarios, los cuales tienen varias funciones y pueden interactuar con todas las demás clases que hemos registrado. El usuario podrá crear comentarios relacionados con los productos. Cada usuario tendrá asignado un carrito, el que almacenará los productos que el usuario desee. El catálogo de productos muestra el listado de productos registrados en la base de datos, con la posibilidad de poder asignar a los productos una categoría. También se podrán crear subcategorías. Una vez finalizado el proceso de compra, los usuarios tendrán acceso a una lista de pedidos.

Metodología y planificación

Hemos dividido el proyecto en 2 fases: fase previa y fase de desarrollo:

- Durante la fase previa, los cuatro integrantes hicimos un "brainstorming" para fijar las funcionalidades necesarias y la estructura de datos. Una vez definidos todos las bases del proyecto nos dividimos las siguientes tareas: definición de funcionalidades, diseño de diagrama de clases, creación de mockups y listado de endpoints.
- La fase de desarrollo la dividimos en partes también, aunque primero una persona se encargó de crear el proyecto inicial. Respecto a la distribución del trabajo discutimos sobre la dificultad de cada tarea y llegamos a la conclusión de dividirlo tal que así:
 - Noel: Registro, login, perfil de usuario, home, administración de usuarios, administración de pedidos y panel de administrador.
 - Paco: Catálogo de productos, navbar, administración de productos y catálogo de búsqueda.^o
 - Juan Carlos: Catálogo con filtro de categorías, administración de categorías y detalles de producto con comentarios.
 - Adri: Checkout, carrito, about us y catálogo de destacados.

También organizamos las tareas en github creando pull requests:



Descripción de la implementación

La implementación de usuarios consiste en un formulario de registro previo en el que se hace una llamada POST al controlador, que primero comprueba si los datos introducidos son correctos y después crea una nueva fila la bd de usuario. A continuación el cliente deberá introducir los datos en un formulario login que comprueba que coincidan los datos. El usuario tiene su propia página de usuario en la que puede modificar sus datos personales.

```
<div class="container" style="...">
  <div class="row justify-content-center mt-5">
    <div class="col-md-6">
      <h2 style="...">Crear Usuario</h2>
      <form method="post" th:action="@{/admin/usuarios/crear}" th:object="${registroData}" class="needs-validation" novalidate>
        <div class="form-row">
          <div class="form-group col-md-6">
            <label for="eMail" style="...">Correo electrónico</label>
            <input id="eMail" class="form-control" name="eMail" placeholder="Correo electrónico"
              type="email" th:field="*{email}" required/>
            <div class="invalid-feedback">
              Por favor, introduce un correo electrónico válido.
            </div>
          </div>
          <div class="form-group col-md-6">
            <label for="nombre" style="...">Nombre</label>
            <input id="nombre" class="form-control" name="nombre" placeholder="Nombre" type="text"
              th:field="*{nombre}" />
          </div>
        </div>
        <div class="form-group">
          <label for="password" style="...">Contraseña</label>
          <input id="password" class="form-control" name="password" placeholder="Contraseña"
            type="password" th:field="*{password}" required/>
          <div class="invalid-feedback">
            La contraseña es obligatoria.
          </div>
        </div>
      </form>
    </div>
  </div>
</div>
```

Se ha creado una excepción que comprueba si el usuario está logueado para los métodos de funcionalidades de usuario registrado.

```
if (carrito.getUsuario() == null) {
    throw new UsuarioNoLogeadoException();
}
```

Para la implementación de productos se ha diseñado de tal forma que es el administrador quien tiene acceso a todas las funciones CRUD de la clase producto. Aquí se muestra un ejemplo de cómo se crea un producto en la aplicación. Primero el administrador rellena un formulario HTML introduciendo los datos del nuevo producto. Una vez le de al botón de submit, los datos se enviarán al controlador que comprobará la sesión del usuario y si es correcta enviará los datos a un servicio de productos.

```
@PostMapping("/admin/tiendaropa/productos/nuevo")
public String nuevoProducto(@ModelAttribute ProductoData productoData, Model model, RedirectAttributes flash, HttpSession session){
    if(!comprobarUsuarioLogeado()){
        throw new UsuarioNoLogeadoException();
    }
    UsuarioData usuario = usuarioService.findById(managerUserSession.usuarioLogeado());
    model.addAttribute("usuario", usuario);
    productoService.crearProducto(productoData.getNombre(), productoData.getPrecio(), productoData.getStock(), productoData.getNumref(), productoData.getDe
    flash.addFlashAttribute("mensaje", "Producto creado correctamente");
    return "redirect:/admin/tiendaropa/catalogo";
}
```

En el servicio se enviarán los datos al repositorio de productos donde se interactúa con la base de datos de forma directa.

```
@Transactional
public ProductoData crearProducto(String nombre, float precio, Integer stock, String numref, boolean destacado, Long categoriaid){
    logger.debug("Creando producto " + nombre);
    Producto producto = new Producto(nombre, precio, stock, numref, destacado, categoriaid);
    productoRepository.save(producto);
    return modelMapper.map(producto, ProductoData.class);
}
```

La implementación del carrito fue bastante tediosa debido a la complejidad del concepto planteado. Cada usuario tiene asignado un carrito único y es en la página de detalles de producto donde empieza el proceso de añadir un producto al carrito. A continuación el controlador comprueba 3 cosas: la sesión del usuario, si existe el producto y el stock. Si pasa por todos los filtros se comprueba si existe la línea en el carrito del usuario, si existe se actualizará la cantidad de producto en el carrito y sino se crea una nueva línea de carrito. Finalmente se actualiza el stock del producto añadido.

```
public void anadirProductos(Carrito carrito, Long productoId, int cantidad)
    throws ProductoNotFoundException, UsuarioNoLogeadoException {
    // Verificar si el usuario está logeado
    if (carrito.getUsuario() == null) {
        throw new UsuarioNoLogeadoException();
    }

    // Obtener el producto
    Producto producto = productoRepository.findById(productoId).orElse(other: null);
    if (producto == null) {
        throw new ProductoNotFoundException();
    }

    // Verificar si hay suficiente stock
    if (producto.getStock() < cantidad) {
        throw new SinStockException();
    }

    // Buscar si ya existe una línea de carrito para este producto en el carrito actual
    LineaCarrito lineaExistente = carrito.getLineascarrito().stream() Stream<LineaCarrito>
        .filter(linea -> linea.getProducto().getNombre().equals(producto.getNombre()))
        .findFirst() Optional<LineaCarrito>
        .orElse(other: null);

    if (lineaExistente != null) {
        // Si ya existe, actualizar la cantidad
        lineaExistente.setCantidad(lineaExistente.getCantidad() + cantidad);
        logger.debug("Línea existente dentro: " + lineaExistente.getCantidad());

        lineaCarritoRepository.save(lineaExistente);
    } else {
        // Si no existe, crear una nueva línea de carrito
        LineaCarrito nuevaLinea = new LineaCarrito();
        nuevaLinea.setProducto(producto);
        nuevaLinea.setCantidad(cantidad);
        carrito.addLineascarrito(nuevaLinea);
    }
}
```

En catálogo existe la posibilidad de filtrar por categoría y se ha implementado de la siguiente manera. Al comienzo del catálogo hay un menú desplegable con un listado de las categorías existentes, el usuario selecciona la que desee y se llama a un controlador específico para generar un grid con los productos de la categoría seleccionada.

```
<h2>Catálogo de productos</h2>
<div style="...">
  <form method="post" th:action="@{/tiendaropa/productos/buscarPorCategoria}">
    <div class="input-group mb-3">
      <select class="form-control" id="categoriaId" name="categoriaId">
        <option value="">Todas las categorías</option>
        <option th:each="cat : ${categorias}" th:value="${cat.id}" th:text="${cat.nombre}"></option>
      </select>
      <div class="input-group-append">
        <button class="btn btn-outline-secondary" type="submit">Buscar por Categoría</button>
      </div>
    </div>
  </form>
</div>
```

Problemas encontrados

Subcategorías: al comienzo de la fase previa del proyecto planteamos un sistema de subcategorías que consistía en que cada categoría podía tener diferentes subcategorías. Esto implicaba una modificación en el diagrama de datos que a la hora de implementarlo nos dió muchos problemas. Finalmente se arregló y funciona perfectamente.

Definición de modelos de datos: A la hora de crear los modelos de datos en intellij nos dió muchos problemas ya que debido a la alta conectividad de la base de datos, los modelos de datos tenían muchas dependencias entre sí y no nos pudimos dividir el trabajo de la manera que queríamos. Finalmente se encargó un miembro del equipo en hacer todos los modelos a la vez.

Mejoras y ampliaciones

Interfaz llamativa: llegamos a la conclusión de que con más tiempo se podría mejorar el aspecto de la interfaz para que sea más atractiva a nuevos clientes.

Tallas: en un futuro sería posible ampliar el modelo de datos de producto para que se puedan trabajar con tallas.

Tablas de análisis de datos: también se podría desarrollar un apartado de análisis de datos comerciales para poder analizar las ventas u otros datos.

Subcategorías: se podría implementar el sistema de subcategorías anteriormente explicado.