



Proyecto final LUNIQ

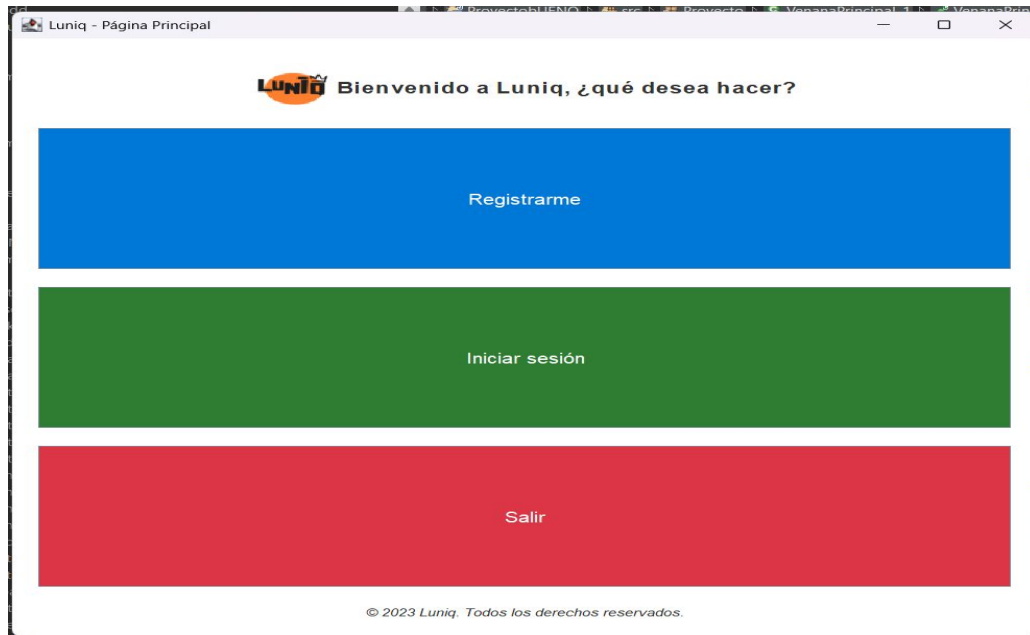
Diego Capellán Fernández y
Alejandro Hernández Murga

Explicación inicial

Nuestro proyecto trata sobre una página para ver ropa desarrollada en Java con interfaz gráfica Swing, la cual hemos llamado LUNIQ.

Permite a los usuarios registrarse, iniciar sesión, ver productos con imágenes y detalles, y gestionar su perfil. Está conectada a una base de datos MySQL donde se guardan los usuarios y productos.






En primer lugar vamos a mostrar la ventana de inicio que cuenta con una interfaz bonita que nos da a elegir entre tres opciones a realizar.

A continuación, si no pulsamos en salir, lo lógico sería registrarnos en caso de que no tengamos cuenta, al pulsar en el botón llegaríamos a la ventana de inicio de sesión, una vez ahí podemos ver una interfaz con los que el usuario podrá escribir su correo, nombre de usuario y contraseña. Una vez escritos, estos datos se guardarán en nuestra base de datos, en nuestra tabla de usuarios.

Registro - LUNIQ



Registrarse

En un periquete

Correo electrónico *

Contraseña *

Ⓢ Tu contraseña debe tener al menos 8 caracteres

Nombre de usuario *

REGISTRARSE

Al crear una cuenta, aceptas nuestros [Términos de uso](#).
Consulta nuestra [Política de privacidad](#).

Una vez nos registremos, se nos abrirá automáticamente la ventana de inicio de sesión, esta tiene una interfaz muy parecida a la de registro.

En esta interfaz deberemos insertar el nombre usuario y la contraseña con la que nos hemos registrado anteriormente. Al darle al botón de iniciar sesión, el programa comprobará si ese nombre de usuario y esa contraseña están en la BBDD y si lo están se nos abrirá la ventana principal de la tienda:



Iniciar sesión

Bienvenido a Luniq, a continuación inicie sesión:

Nombre de usuario

Contraseña

INICIAR SESIÓN

Al crear una cuenta, aceptas nuestros [Términos de uso](#).
Consulta nuestra [Política de privacidad](#).

**Al iniciar sesión entraremos en la ventana principal de la tienda.
Dentro de este creamos otra serie de paneles más pequeños.**

Dentro de estos paneles más pequeños, metemos la imagen del producto, llamada desde la base de datos y metemos el nombre del producto (también traído de la BBDD en un botón para que al pulsarlo nos lleve a otra ventana con información de cada producto

La interfaz de la ventana de la tienda también cuenta con un botón para entrar a la ventana del perfil del usuario y con un logo de nuestra marca diseñado por nosotros



Nuestros Productos



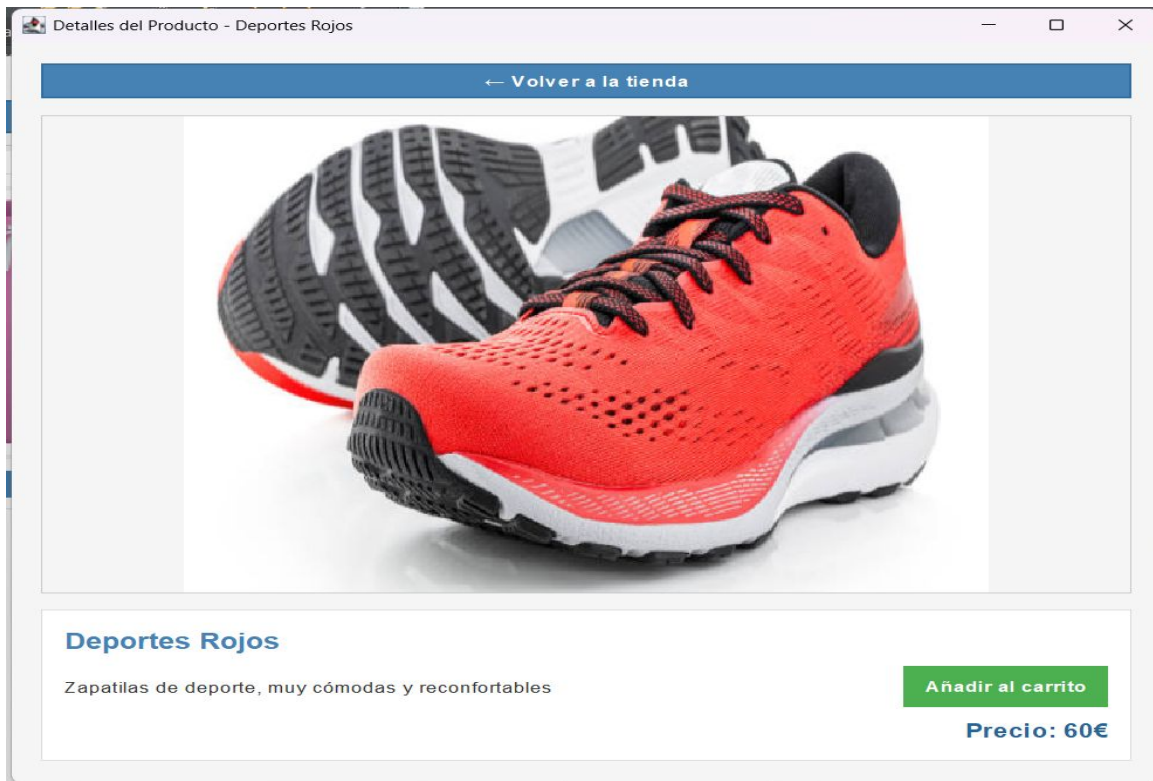
Camiseta Under Armour



Camiseta de Solo Leveling



Mi Perfil



Llegados a la ventana de la tienda, si nos gusta algún producto podremos entrar a la ventana de detalles del producto en la cual se llama a la base de datos para traer la información del producto junto con su foto.

Si en lugar de ver los productos el usuario quiere ver la información de su perfil solo tiene que dale al botón de perfil desde su ventana principal ahí podremos ver una interfaz bonita en los que aparecerá la información introducida al iniciar sesión.

También aparecerán otros JTextField vacíos en los que al escribir, se guardará la información faltante en la tabla de usuarios:



Esta es la ventana de perfil, introduce tus datos:

ID de usuario:	<input type="text" value="27"/>
Nombre:	<input type="text"/>
Apellido:	<input type="text"/>
E-mail:	<input type="text" value="User@gmail.com"/>
Nombre usuario:	<input type="text" value="User"/>
Contraseña:	<input type="text" value="User"/>
País:	<input type="text"/>
Fecha registro:	<input type="text" value="2025-05-26 00:48:58"/>

Aplicar cambios

Volver atrás

A continuación vamos a ver un poco lo que hemos hecho en lo referente a la asignatura de Entorno de desarrollo.

En primer lugar Tenemos los Test de Junit 5 que le hemos realizado a nuestras clases para comprobar el correcto funcionamiento del programa y testear los posibles fallos:

Para ello hemos insertado la librería de Junit5 en nuestro proyecto para que los test pudieran funcionar correctamente, en la siguiente diapositiva veremos algún ejemplo

eclipse-workspace - ProyectoBUENO/src/ProyectoTests/ClaseConexionTest.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Ex... JUnit x VenanaPrinci... Ventana_Perf... Ventana_Deta... Ventana_Inic... ProductoTes... ClaseConexion... UsuarioTe

ProjectobUENO src ProyectoTests ClaseConexionTest testConexionExitosa() : void

Finished after 1,282 seconds

Runs: 2/2 x Errors: 0 x Failures: 0

> ClaseConexionTest [Runner: JUnit

```
8
9 import static org.junit.jupiter.api.Assertions.*;
10
11 class ClaseConexionTest {
12
13     @Test
14     void testConexionExitosa() {
15         ClaseConexion conexion = new ClaseConexion("localhost", "3306", "root", "", "proyecto_bbdd");
16
17         // Simulamos que la conexión fue exitosa
18         assertTrue(true);
19
20
21     @Test
22     void testLoginUsuario() {
23         ClaseConexion conexion = new ClaseConexion("localhost", "3306", "root", "", "proyecto_bbdd");
24
25         // Suponiendo que existe este usuario en la BD de prueba
26         boolean resultado = conexion.login("d", "d");
27
28         assertTrue(resultado);
29     }
30
31 }
```

En segundo lugar, al terminar el proyecto hemos refactorizar el código para conseguir que quede lo más limpio y entendible posible.

Para ello hemos cambiado el nombre de algunos elementos ya que no se entendía muy bien a qué se referían.

También hemos eliminado fragmentos de código que no tenían utilidad como los syso que íbamos colocando para comprobar donde había errores.

Por último también hemos ordenado las ventanas para que los botones, cajas de texto etc estén ordenados también en el código.

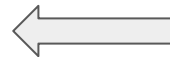

```
// Botón con nombre del producto
JButton btn = new JButton(producto.getNombre());
btn.setFont(new Font("Arial", Font.BOLD, 14));
btn.setBackground(Color.WHITE);
btn.setBorder(new LineBorder(new Color(200, 200, 200), 1));
btn.setFocusPainted(false);

// Añadir ActionListener al botón
btn.addActionListener(e -> {
    Ventana_DetalleProducto detalle = new Ventana_DetalleProducto(
        producto, this.conexion, this.nombreUsuarioActual);
    System.out.println("dnddn");
});
```

```
System.out.println("dnddn");
```

```
// Botón con nombre del producto
JButton btnNombreProducto = new JButton(producto.getNombre());
btnNombreProducto.setFont(new Font("Arial", Font.BOLD, 14));
btnNombreProducto.setBackground(Color.WHITE);
btnNombreProducto.setBorder(new LineBorder(new Color(200, 200, 200), 1));
btnNombreProducto.setFocusPainted(false);

// Añadir ActionListener al botón
btnNombreProducto.addActionListener(e -> {
    Ventana_DetalleProducto detalle = new Ventana_DetalleProducto(
        producto, this.conexion, this.nombreUsuarioActual);
    detalle.setVisible(true);
});
```

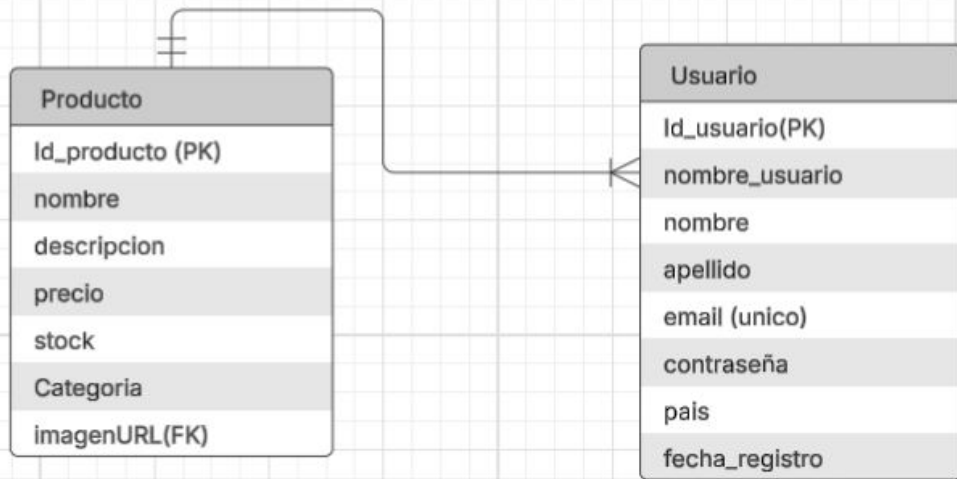


Eliminado

En cuanto a base de datos hemos implementado en nuestro trabajo un diagrama de entidad relación y un trigger que se activa antes de insertar un usuario y verifica que al insertar un nuevo usuario, cumpla con la nomenclatura indicada:

```
1 -- Trigger para validar email antes de INSERT en usuarios
2 DELIMITER //
3 CREATE TRIGGER `before_usuarios_insert`
4 BEFORE INSERT ON `usuarios`
5 FOR EACH ROW
6 BEGIN
7     IF NEW.email NOT LIKE '%@%.%' THEN
8         SIGNAL SQLSTATE '45000'
9         SET MESSAGE_TEXT = 'El email debe contener "@" y un dominio válido';
10    END IF;
11 END //
12 DELIMITER ;
```

Nombre		Tiempo	Evento			
<input type="checkbox"/>	before_usuarios_insert	BEFORE	INSERT	 Editar	 Exportar	 Eliminar



Un usuario puede tener **muchos productos** (asociados a él) y **Muchos productos** pertenece a **un solo usuario**



TECNOLOGÍAS UTILIZADAS

Lenguaje: Java SE 17

Interfaz Gráfica: Java Swing

Base de Datos: XAMP + PHPMyAdmin

Persistencia: JDBC

Gestión de Código: Git + GitHub

Pruebas: JUnit 5

IDE Utilizado: Eclipse

