Proyecto Final Programado

Nombre del Proyecto: Desarrollo de Plataforma E-commerce para la Tienda de Ropa "Tentación Azul"

Nombre del Estudiante/Desarrollador: José Andrés Cruz Álvarez

Profesor/Curso: Estefanía Boza Villalobos / Programación Avanzada

Fecha: 05 de octubre del 2025

Tabla de Contenido

Parte I: Información General del Proyecto

Como respuesta a la era digital contemporánea en la que ocurren constantes innovaciones, el presente proyecto desarrollará una plataforma web de comercio electrónico para la tienda física de ropa para dama “Tentación Azul”. Esto en la búsqueda

de modernizar sus operaciones para mantener su competitividad, a la vez que aspira a expandir su alcance comercial. Por este medio, se pretende brindar una solución que permita al establecimiento integrarse en la era digital de una manera efectiva, facilitando la gestión de compras de prendas por medio de la plataforma web. Se busca que con esta plataforma los usuarios puedan encontrar el catálogo de prendas disponibles y puedan encargar sus órdenes.

Objetivo General

Desarrollar una plataforma de comercio electrónico para la tienda física de ropa para dama “Tentación Azul”, por medio de la que se modernicen sus operaciones, expanda su alcance comercial y facilite a los clientes la compra de productos en línea.

Objetivos Específicos

1. Elaborar el diseño de la plataforma, creando una interfaz de usuario intuitiva y responsiva que permita a los clientes navegar por el catálogo de productos, agregar productos al carrito de compras y completar el proceso de checkout.
2. Integrar la lógica del backend y la base de datos para gestionar la información de usuarios, productos, inventario y pedidos, garantizando que las transacciones y los datos sean consistentes.
3. Integrar un sistema de administración de contenido para que los administradores de la tienda puedan gestionar el inventario, actualizar precios, visualizar órdenes de compra y dar seguimiento a los pedidos

Justificación del Proyecto

Este proyecto surge como una respuesta estratégica y necesaria ante el contexto actual de transformación digital que domina el sector comercial. Para la tienda "Tentación Azul", la continuidad y el crecimiento en un mercado cada vez más competitivo dependen de su capacidad para adaptarse a los nuevos hábitos de consumo, los cuales demandan canales digitales accesibles y eficientes. La dependencia exclusiva de la venta física limita

significativamente su alcance de mercado y su potencial de crecimiento, generando la necesidad imperante de integrar un canal de ventas en línea que le permita mantener su competitividad.

La implementación de esta plataforma de comercio electrónico resulta fundamental para modernizar las operaciones internas del establecimiento. Actualmente, procesos como la gestión de inventario, la toma de pedidos y el seguimiento al cliente pueden ser susceptibles a demoras y errores cuando se manejan de manera manual o desintegrada. El desarrollo de un sistema centralizado, que incluye un panel de administración intuitivo, optimizará estos flujos de trabajo, reducirá la carga operativa y proporcionará información en tiempo real para una toma de decisiones más ágil y fundamentada. Esta modernización es crucial para mejorar la eficiencia y la escalabilidad del negocio.

Desde la perspectiva del usuario final, el proyecto se centra en mejorar la experiencia del cliente. La plataforma brindará a los consumidores la comodidad de explorar el catálogo completo de productos, verificar la disponibilidad y realizar sus pedidos desde cualquier lugar y en cualquier momento, superando así las barreras de horario y ubicación geográfica inherentes a la tienda física. Al ofrecer un servicio más accesible y conveniente, "Tentación Azul" no solo fortalece la relación con su clientela actual, sino que también se posiciona para captar nuevos segmentos de mercado acostumbrados a las compras en línea.

Finalmente, la implementación de este canal digital representa una evolución natural del modelo de negocio, alineándose con las mejores prácticas del comercio moderno donde la presencia en línea se ha convertido en un estándar del sector. Esta transición no solo responde a una necesidad marcada por la tendencia del mercado, sino que, también proyecta una imagen innovadora y contemporánea de la marca, reforzando su valor en la mente del consumidor.

Alcances Esperados

- El sistema permitirá registrar, consultar y gestionar clientes, productos, categorías, inventario y órdenes de compra a través de la plataforma web.
- Acceso a la información en tiempo real sobre la disponibilidad de productos, el estado del inventario y el historial de pedidos de los clientes.

- Generación de reportes para la toma de decisiones por medio de la información relacionada con las ventas y estado del inventario.

Requerimientos del Proyecto

1. **Requerimientos Funcionales:**

   - **Gestión de Catálogo:** Añadir, editar, desactivar y categorizar productos (nombre, descripción, precio, talla, imágenes, stock).
   - **Gestión de Usuarios:** Registro e inicio de sesión de clientes; panel de administración para gestionar roles y permisos.
   - **Carrito de Compras y Checkout:** Los usuarios podrán agregar productos, revisar su carrito y completar una orden con sus datos personales.

2. **Requerimientos No Funcionales**

   - **Usabilidad:** Interfaz intuitiva y responsiva con una curva de aprendizaje mínima.
   - **Seguridad:** Protección de datos de usuarios y prevención de accesos no autorizados.
   - **Disponibilidad:** La plataforma debe estar operativa al menos el 98% del tiempo.
   - **Escalabilidad:** La arquitectura debe permitir añadir nuevas funcionalidades en el futuro con forme su necesidad vaya surgiendo.

3. **Infraestructura Mínima**

   - **Equipo de Desarrollo:** Computadora con Windows 10 o superior, 8 GB de RAM y 100 GB de espacio libre en disco.
   - **Entorno de Desarrollo:** Visual Studio 2022 con .NET Framework 4.8 o .NET Core 3.1+.
   - **Base de Datos:** SQL Server 2019 o superior (para entorno local).

- **Tecnologías:** HTML, CSS, JavaScript para el frontend; ASP.NET para el backend.

Parte II: Diseño del Sistema (Pre-diseño visual)

Elementos de Diseño

- **Estructura de pantallas:** El presente proyecto cuenta con la sección para el login, luego por medio del control de vistas basado en roles, los administradores podrán tener un panel principal de control de usuarios, órdenes, ítems en stock, actualizar información de los productos, etc; mientras que en el de los clientes se podrá visualizar los ítems a la venta, ver precios, stock disponible por talla, así como su carrito de compra y las órdenes.
- **Colores:**

1. Turquesa Principal - Utilizado en: botones principales, elementos destacados, encabezados de secciones y links de acción
2. Turquesa Claro - Utilizado en: fondos secundarios, estados hover (al pasar el mouse), efectos de onda decorativos
3. Gris Claro - Utilizado en: fondo principal de la aplicación, áreas de contenido general
4. Gris Medio - Utilizado en: bordes de elementos, líneas separadoras, cards secundarios
5. Gris Oscuro - Utilizado en: texto principal, títulos, etiquetas de formularios
6. Blanco - Utilizado en: cards flotantes, fondos de formularios, áreas de contenido destacado

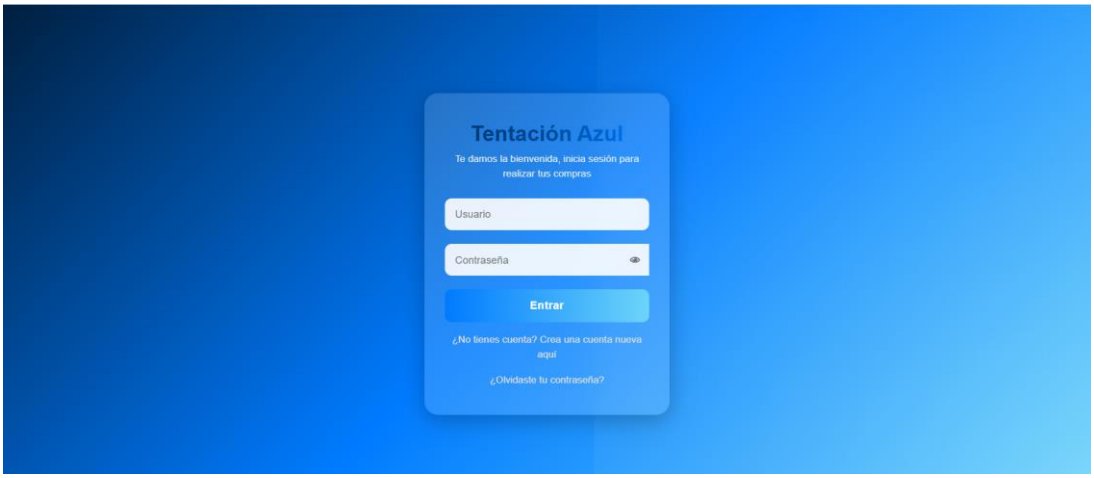
- **Tipografía:**

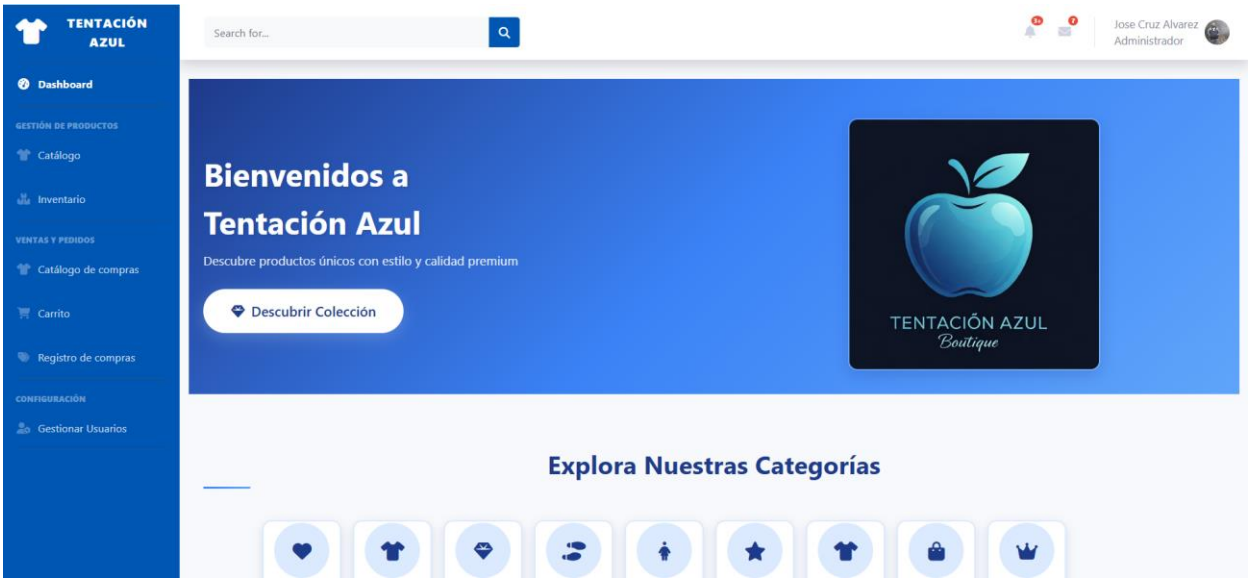
1. **Fuente Principal:** Segoe UI
2. **Fuentes Alternativas:** Roboto, Arial, sans-serif
3. **Tamaños:**
    a. Títulos: 24px - 32px
    b. Subtítulos: 18px - 20px
    c. Texto cuerpo: 14px - 16px
    d. Texto pequeño: 12px - 13px

- **Imágenes:** Iconografía sencilla y representativa para cada módulo.

  Login:

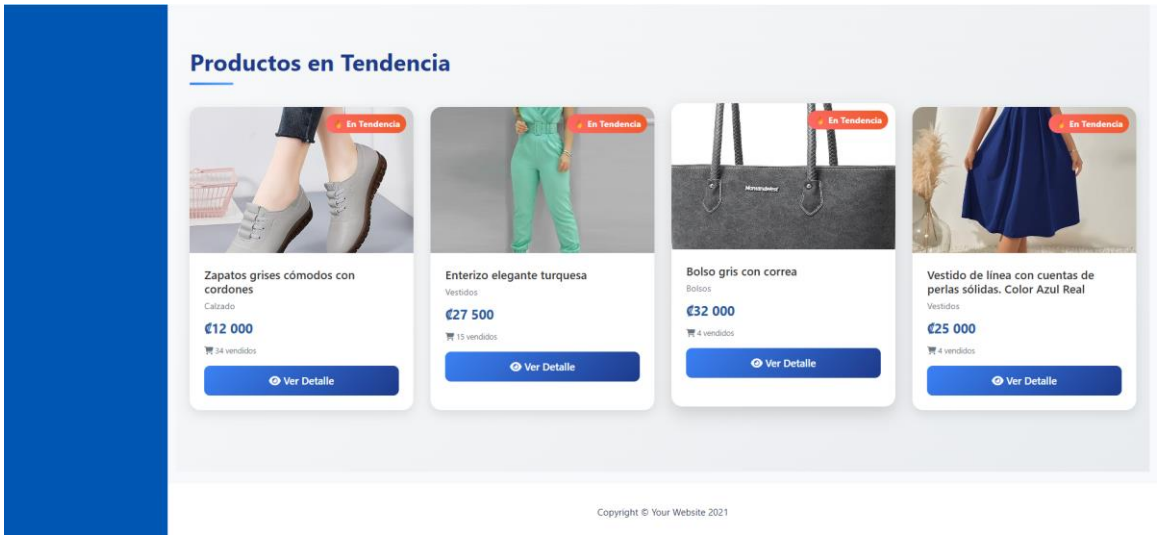
  Breve vista de layout e index

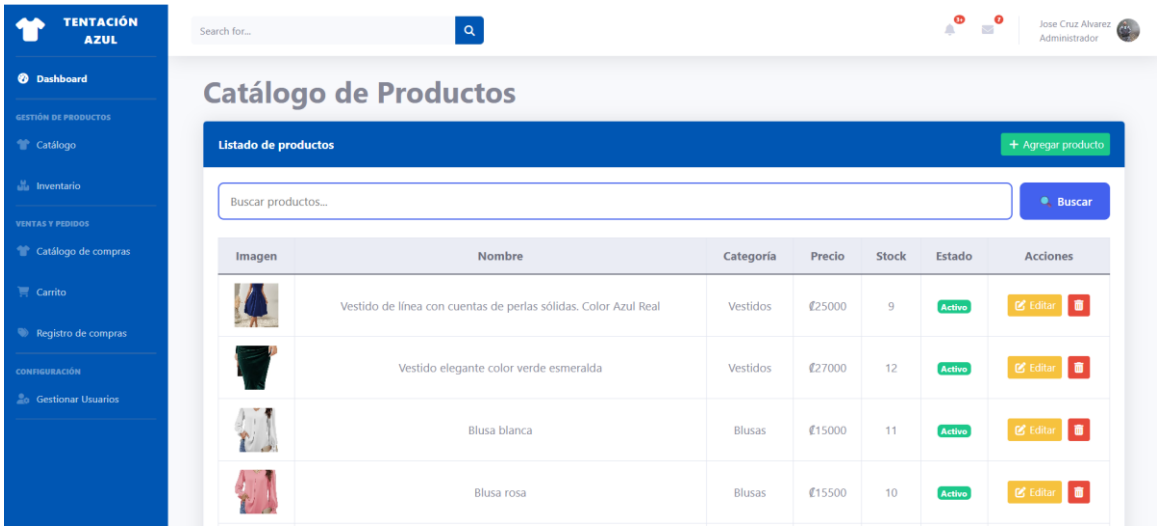


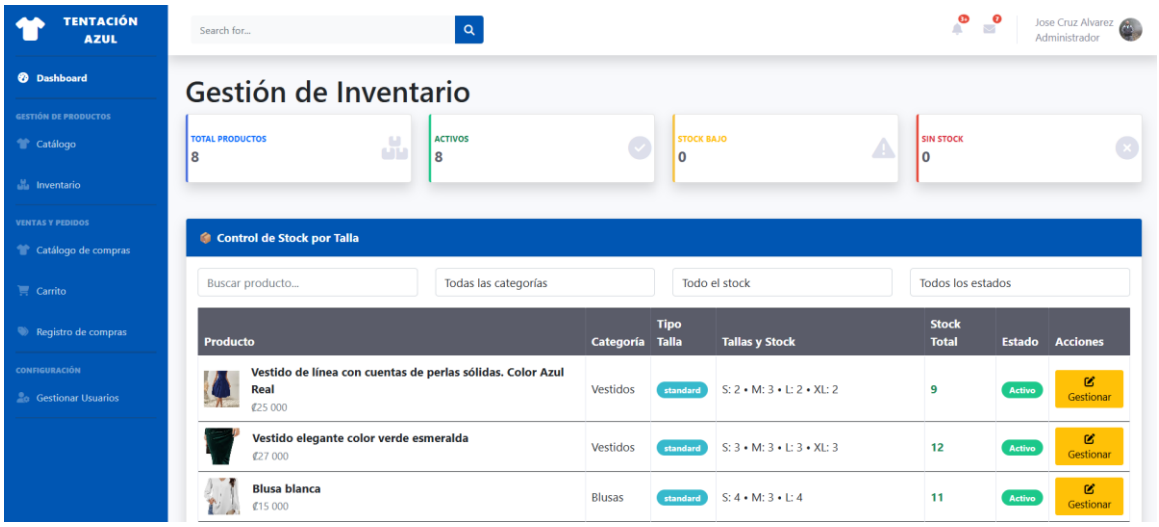


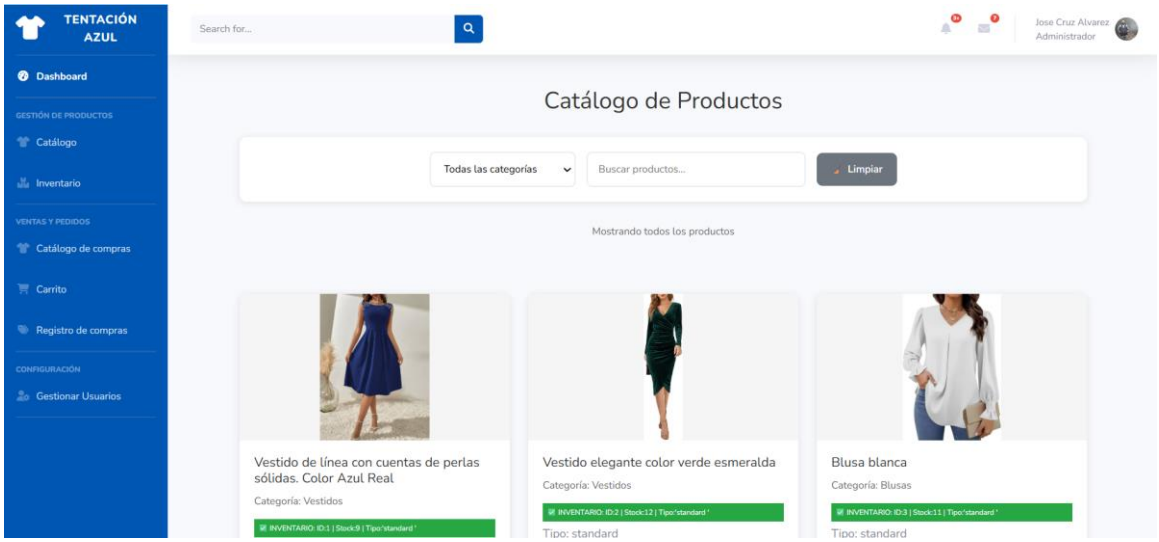
- **Íconos:** Uso de librerías estándar como FontAwesome y Bootstrap.

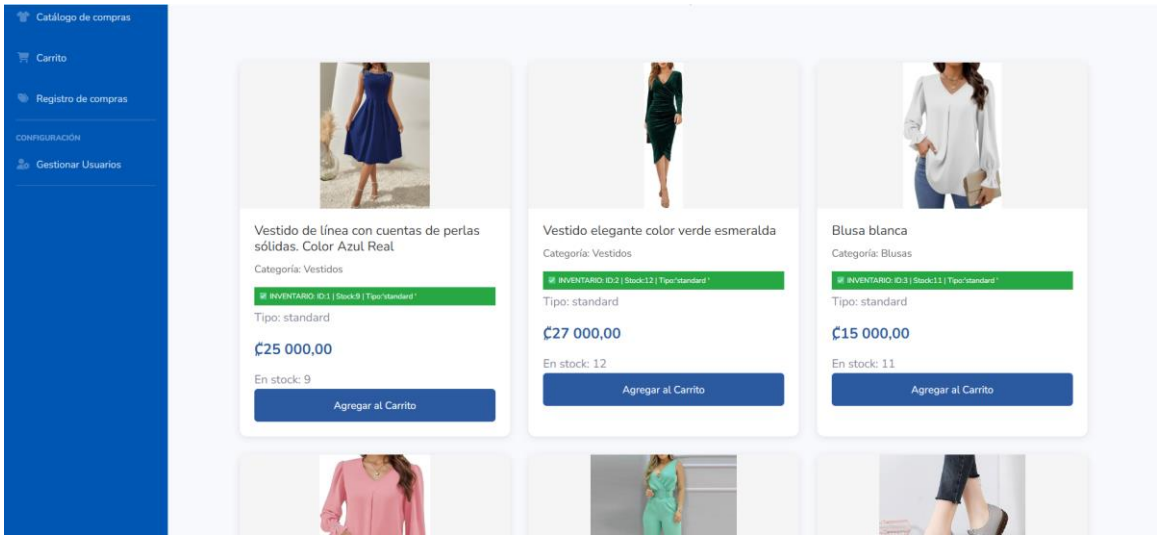

Gestionar Usuarios
Explora Nuestras Categorías
Vestidos
Blusas
Accesorios
Calzado
Pantalones
Faldas
Tops
Bolsos
Bisutería
Productos en Tendencia
En Tendencia
Zapatos grises cómodos con cordones
Enterizo elegante turquesa
Vestidos
Bolso gris con correa
Bolsos
Vestido de línea con cuentas d
perlas sólidas. Color Azul Real

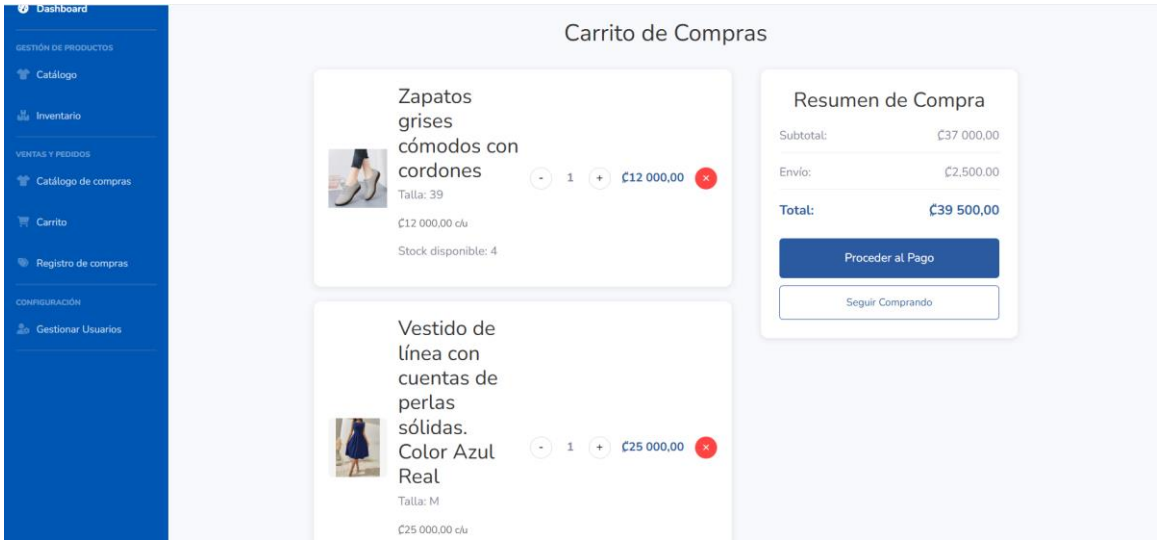

Productos en Tendencia
En Tendencia
Zapatos grises cómodos con cordones
Calzado
₡12 000
24 vendidos
Ver Detalle
Enterizo elegante turquesa
Vestidos
₡27 500
15 vendidos
Ver Detalle
Bolso gris con correa
Bolsos
₡32 000
4 vendidos
Ver Detalle
Vestido de línea con cuentas de perlas sólidas. Color Azul Real
Vestidos
₡25 000
4 vendidos
Ver Detalle
Copyright © Your Website 2021

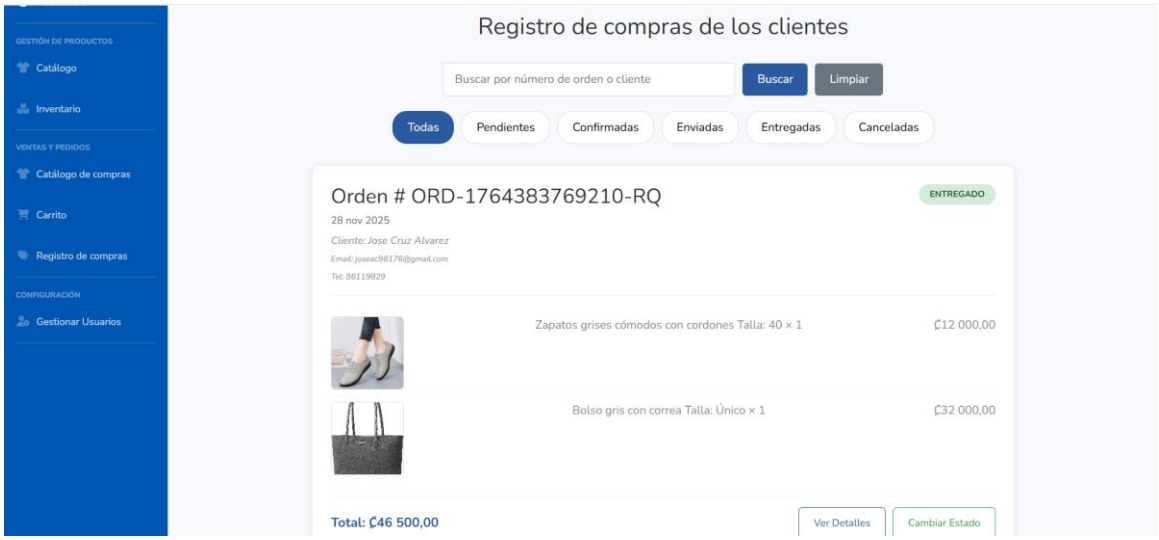

TENTACIÓN AZUL
Search for...
Jose Cruz Alvarez
Administrador
Dashboard
GESTIÓN DE PRODUCTOS
Catálogo
Inventario
VENTAS Y PEDIDOS
Catálogo de compras
Carrito
Registro de compras
CONFIGURACIÓN
Gestionar Usuarios
Catálogo de Productos
Listado de productos
+ Agregar producto
Buscar productos...
Buscar
Imagen
Nombre
Categoría
Precio
Stock
Estado
Acciones
Vestido de línea con cuentas de perlas sólidas. Color Azul Real
Vestidos
₡25000
9
Activo
Editar
Vestido elegante color verde esmeralda
Vestidos
₡27000
12
Activo
Editar
Blusa blanca
Blusas
₡15000
11
Activo
Editar
Blusa rosa
Blusas
₡15500
10
Activo
Editar

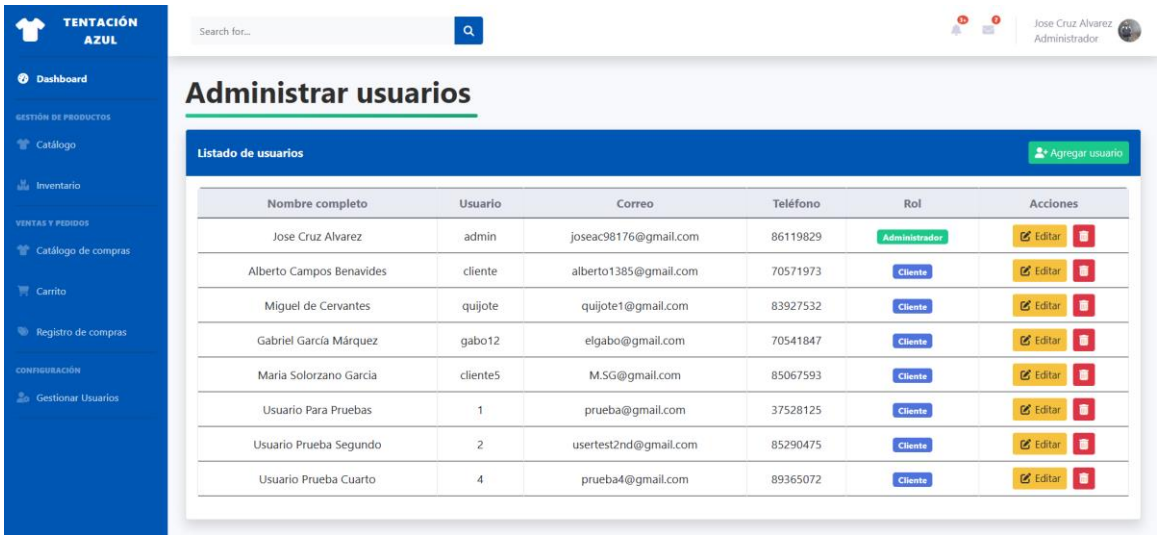

TENTACIÓN AZUL
Search for...
Jose Cruz Alvarez
Administrador
Dashboard
GESTIÓN DE PRODUCTOS
Catálogo
Inventario
VENTAS Y PEDIDOS
Catálogo de compras
Carrito
Registro de compras
CONFIGURACIÓN
Gestionar Usuarios
Gestión de Inventario
TOTAL PRODUCTOS
8
ACTIVOS
8
STOCK BAJO
0
SIN STOCK
0
Control de Stock por Talla
Buscar producto...
Todas las categorías
Todo el stock
Todos los estados
Producto
Categoría
Tipo Talla
Tallas y Stock
Stock Total
Estado
Acciones
Vestido de línea con cuentas de perlas sólidas. Color Azul Real
₡25 000
Vestidos
standard
S: 2 • M: 3 • L: 2 • XL: 2
9
Activo
Gestionar
Vestido elegante color verde esmeralda
₡27 000
Vestidos
standard
S: 3 • M: 3 • L: 3 • XL: 3
12
Activo
Gestionar
Blusa blanca
₡15 000
Blusas
standard
S: 4 • M: 3 • L: 4
11
Activo
Gestionar


TENTACIÓN AZUL
Search for...
Jose Cruz Alvarez
Administrador
Dashboard
GESTIÓN DE PRODUCTOS
Catálogo
Inventario
VENTAS Y PEDIDOS
Catálogo de compras
Carrito
Registro de compras
CONFIGURACIÓN
Gestionar Usuarios
Catálogo de Productos
Todas las categorías
Buscar productos...
Limpiar
Mostrando todos los productos
Vestido de línea con cuentas de perlas sólidas. Color Azul Real
Categoría: Vestidos
Vestido elegante color verde esmeralda
Categoría: Vestidos
Tipo: standard
Blusa blanca
Categoría: Blusas
Tipo: standard


Catálogo de compras
Carrito
Registro de compras
CONFIGURACIÓN
Gestionar Usuarios
Vestido de línea con cuentas de perlas sólidas. Color Azul Real
Categoría: Vestidos
Tipo: standard
₡25 000,00
En stock: 9
Agregar al Carrito
Vestido elegante color verde esmeralda
Categoría: Vestidos
Tipo: standard
₡27 000,00
En stock: 12
Agregar al Carrito
Blusa blanca
Categoría: Blusas
Tipo: standard
₡15 000,00
En stock: 11
Agregar al Carrito


Dashboard
GESTIÓN DE PRODUCTOS
Catálogo
Inventario
VENTAS Y PEDIDOS
Catálogo de compras
Carrito
Registro de compras
CONFIGURACIÓN
Gestionar Usuarios
Carrito de Compras
Zapatos grises cómodos con cordones
Talla: 39
₡12 000,00 c/u
Stock disponible: 4
- 1 + ₡12 000,00
Vestido de línea con cuentas de perlas sólidas. Color Azul Real
Talla: M
₡25 000,00 c/u
- 1 + ₡25 000,00
Resumen de Compra
Subtotal: ₡37 000,00
Envío: ₡2,500.00
Total: ₡39 500,00
Proceder al Pago
Seguir Comprando


GESTIÓN DE PRODUCTOS
Catálogo
Inventario
VENTAS Y PEDIDOS
Catálogo de compras
Carrito
Registro de compras
CONFIGURACIÓN
Gestionar Usuarios
Registro de compras de los clientes
Buscar por número de orden o cliente
Buscar
Limpiar
Todas
Pendientes
Confirmadas
Enviadas
Entregadas
Canceladas
Orden # ORD-1764383769210-RQ
ENTREGADO
28 nov 2025
Cliente: Jose Cruz Alvarez
Email: joseac98176@gmail.com
Tel: 86119829
Zapatos grises cómodos con cordones Talla: 40 × 1
₡12 000,00
Bolso gris con correa Talla: Único × 1
₡32 000,00
Total: ₡46 500,00
Ver Detalles
Cambiar Estado



Explicación

El pre-diseño visual desarrollado para la plataforma e-commerce "Tentación Azul" utiliza tecnologías como ASP.NET Core con arquitectura en capas (DAL, BLL, Controller) para el backend, HTML5, CSS3 y JavaScript/jQuery para el frontend, SQL Server con stored procedures para la base de datos, y librerías como Bootstrap, FontAwesome y SweetAlert2 para la interfaz de usuario. Este diseño establece la estructura de pantallas (login, dashboard, catálogo, carrito), implementa una paleta de colores turquesa y gris, define tipografía moderna (Segoe UI, Roboto) e incorpora iconografía consistente, sirviendo como guía visual preliminar que permite a usuarios y desarrolladores comprender la distribución, apariencia y navegación del sistema antes de su implementación completa, asegurando coherencia entre el diseño y la arquitectura tecnológica del proyecto.

Parte III: Base de Datos

Definición

La base de datos será relacional, diseñada para garantizar integridad y consistencia.

Para el presente proyecto se implementó SQL Server Management Studio (SSMS) como motor de base de datos. Por lo que se trabajó en una base de datos relacional, cuyo propósito es el de gestionar el catálogo, inventario, las órdenes de compra, así como los usuarios de la tienda de e comerse.

Elementos

Con la finalidad de explicar el tipo de datos que se implementaron, así como, las diferentes Primary keys y foreign keys, procedo a adjuntar la información de las diferentes tablas donde se visualizan los diferentes tipos de datos empleados. Así mismo, posteriormente procederé a adjuntar el diagrama de las diferentes tablas para mostrar sus relaciones entre PK y FK.

DESKTOP-QFILD1U.T...pa - dbo.Usuarios

Column Name	Data Type	Allow Nulls
idUsuario	int	☐
nombreUsuario	nvarchar(70)	☐
contrasenaUsuario	nvarchar(70)	☐
fechaRegistro	datetime	☐
idRol	int	☐
idInfoUsuario	int	☐
		☐

Tabla correspondiente a usuarios

DESKTOP-QFILD1U.T...dbo.infoUsuarios

Column Name	Data Type	Allow Nulls
idInfoUsuario	int	☐
nombres	nvarchar(100)	☐
apellido1	nvarchar(50)	☐
apellido2	nvarchar(50)	☐
telefono	nvarchar(20)	☐
correo	nvarchar(80)	☐

Tabla de información personal de los usuarios

DESKTOP-QFILD1U.T..._ropa - dbo.Roles

Column Name	Data Type	Allow Nulls
idRol	int	☐
nombreRol	nvarchar(50)	☐
		☐

Tabla de rol para usuarios

DESKTOP-QFILD1U.T...a - dbo.Productos

Column Name	Data Type	Allow Nulls
idProducto	int	☐
idCategoria	int	☐
nombreProducto	nvarchar(500)	☐
precio	decimal(10, 2)	☐
imagenURL	nvarchar(500)	☐
idEstado	int	☐
		☐

Tabla de productos

DESKTOP-QFILD1U.T....- dbo.Categorias | DESKTOP-QFILD1U.T...a - dbo.Productos

Column Name	Data Type	Allow Nulls
idCategoria	int	☐
nombreCategoria	nvarchar(100)	☐
descripcion	nvarchar(250)	☐
tipoTalla	nchar(10)	☑
		☐

Tabla Categorías

DESKTOP-QFILD1U....opa - dbo.Estados

Column Name	Data Type	Allow Nulls
idEstado	int	☐
nombreEstado	nvarchar(30)	☐
descripcion	nvarchar(100)	☐
		☐

Tabla de Estados

DESKTOP-QFILD1U.T...bo.ProductoTallas

Column Name	Data Type	Allow Nulls
idProductoTalla	int	☐
talla	nvarchar(10)	☐
stock	int	☐
idProducto	int	☐
		☐

Tabla de tallas del producto

DESKTOP-QFILD1U....opa - dbo.Pedidos

Column Name	Data Type	Allow Nulls
idPedido	int	☐
idUsuario	int	☐
fechaPedido	datetime	☐
totalCompra	decimal(10, 2)	☐
estado	nvarchar(20)	☐
direccionEnvio	nvarchar(250)	☐
		☐

Tabla de pedidos

DESKTOP-QFILD1U.T...o.DetallesPedidos

Column Name	Data Type	Allow Nulls
idDetalleOrden	int	☐
idOrden	int	☐
idProducto	int	☐
cantidad	int	☐
precioUnidad	decimal(10, 2)	☐
		☐

Detalles de los pedidos

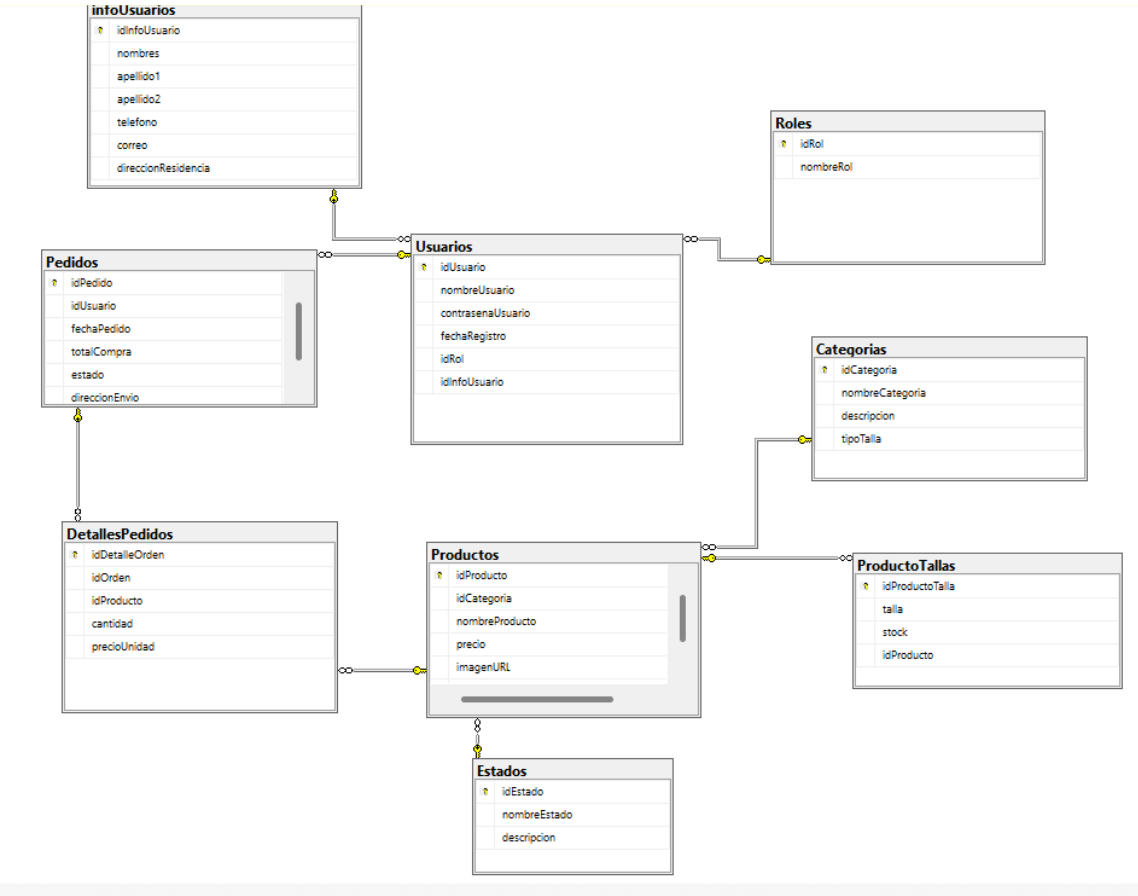
Primeramente, en la parte relacional se muestra a la tabla de usuarios donde se guarda la información de las credenciales del usuario en el sistema. Tiene como PK idUsuario y tiene dos FK que son idRol e idInfoUsuarios, las cuales se dirigen a las PK con esos mismos nombres de la tabla de Rol e info de usuarios. La tabla por control de roles permite el manejo de las vistas, mientras que la tabla de info de los usuarios almacena los datos personales del usuario.

Por otro lado, se presenta a la tabla de productos en la cuál se almacena información pertinente a los diferentes productos que se tienen en existencia para la venta en la tienda. Cuenta con dos FK que son la idCategoria y el idEstado. Estas tablas se dirigen hacia la PK con los mismos nombres en las tablas de Categorías y Estados para referenciarles y conectarles. Categorías sirve para separar qué tipo de prenda se está comprando, así mismo, por medio de un procedimiento almacenado (store procedure), dependiendo la tabla que se ingrese al añadir una nueva prenda, se puede hacer que en la vista dinámicamente se cambie la talla de la ropa dependiendo a la categoría en la que se encuentra cada pieza. Por medio de estado al conectarse con la tabla de producto se puede manejar si la pieza de ropa está en Estado activo porque tiene stock, sin stock cuando no hay existencias o inactivo cuando se saca de la venta una prenda. Además la tabla productos se relaciona con lo que es la de productos por talla por medio de la fk que se puso en esta segunda (idProducto), para que así pueda atribuirse a cada uno de los diferentes bienes el stock que tiene en existencia por talla.

Por último, se visualiza la tabla de pedidos, en la cual se mantiene la información general de cada compra como la fecha en la que se hizo, el monto total, el usuario que la realizó y la dirección del envío. Mientras que por otro lado se tiene una tabla específica para los detalles de ese pedido como los productos que se compraron, la cantidad de los mismos por compra, así como el precio de cada uno de los productos para que por medio de procedimientos almacenados se haga el cálculo total de los mismos y se muestre en la tabla de pedidos como un total por orden.

Esquema y diagrama

Ahora bien, según lo correspondiente al esquema de entidad relación, se presenta el realizado por medio de SQL Server Management Studio:



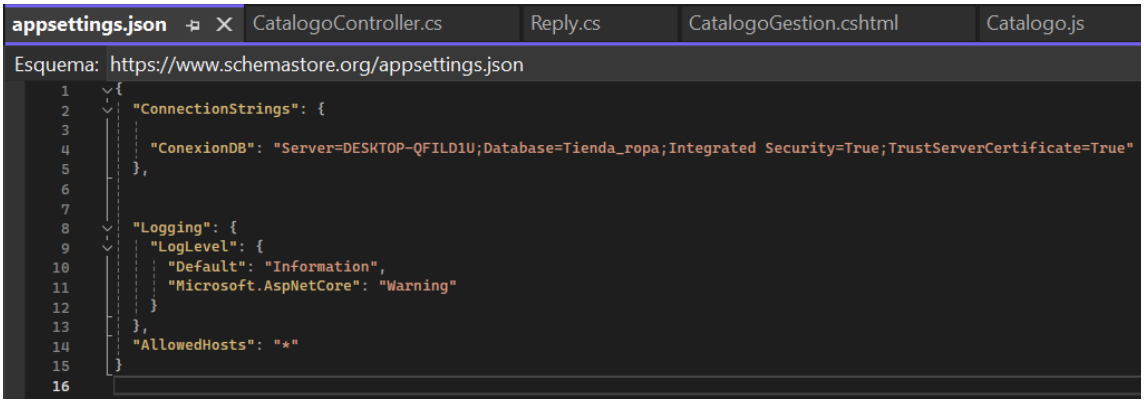
En el diagrama se puede evidenciar que la tabla de Usuarios se conecta por medio de sus fk idRol e idInfoUsuarios a las tablas de los mismos nombres y a las PK que comparten los nombres de esas FK (como fue comentado anteriormente). De ahí se pasa a la tabla de productos que se conecta a la de estados por medio de la fk de idEstado, a la vez que idCategoria se une a la pk con mismo nombre en la tabla de Categorías, mientras que en la tabla de productoTallas, es por medio de una fk de idProducto que se relacionan a la PK de la tabla de productos.

Mientras que los productos se unen a detalles pedidos, cuya tabla recopila la información de los productos por medio de la FK que tienen de idProducto que se refieren a la PK de mismo nombre en la tabla de Productos. A su vez, la FK de idOrden hace referencia a la PK de idPedido en la tabla de Pedidos para que en detalles de la orden se trate la información específica, se contabilice y en la tabla pedido se traten datos como el total de la compra, la fecha y el estado del pedido.

Por último, en la tabla Pedidos se encuentra la fk de idUsuario que se dirige hacia la tabla de Usuarios a su PK con el mismo nombre para que de esta manera se pueda asignar una compra al cliente específico que proceso la orden, de manera que se pueda mantener el registro de las compras por cliente.

Conexión con la aplicación

Para efectos del presente proyecto, es importante destacar se configuró una conexión completamente funcional entre ASP.NET Core y SQL Server usando Entity Framework Core como ORM principal. El método de conexión se realiza a través del apartado de appsetings.json en Visual Studio donde se emplea el siguiente código de conexión:



Parte IV: Desarrollo del Backend

Arquitectura del Backend

Para efectos de este proyecto, en cuanto a la arquitectura de su Backend, se puede destacar la arquitectura por capas por medio del Modelo de Vista-Controlador. El presente se muestra estructurado por capas que van desde el inicio por la Base de Datos, implementada en SQL Server Management, en la que se implementan diversos procedimientos almacenados para cumplir con las diversas funcionalidades de las tablas en función de la lógica que en ellos se implementa.

Posteriormente, dentro del proyecto hay una capa lógica Data Access Layer (DAL), por medio de la cual se obtiene los procedimientos almacenados de la base de datos y en la cuál se emplean los diferentes parámetros que vienen de la base de datos. También se focaliza la implementación de los procedimientos almacenados aquí para brindar una mayor seguridad al proyecto al no exponer en el directamente info de una base de datos ya que para ello se encarga los procedimientos almacenados dentro de la base de datos y en el código este procedimiento simplemente se nombra dentro de esta capa de DAL.

Seguidamente, es importante señalar que este proyecto cuenta con una capa lógica, determinada como Business Logic Layer (BLL). Tal y como se comentó con anterioridad, esta es una capa que permite al desarollador implementar alguna lógica extra en el código que no venga integrada en los procedimientos almacenados y que se necesite adecuar desde el Backend, esto a pesar de que JavaScript puede realizar múltiples funciones que brinden dinamismo al proyecto, puede haber procesos que necesiten implementarse desde la lógica de la capa BLL.

Asimismo, esta capa se conecta directamente con los controladores, los cuales, tienen una gran injerencia al enlazar Backend con el Frontend. Al menos en lo que respecta al presente proyecto, los controladores fungen un papel fundamental ya que reciben la lógica proveniente de la capa BLL a la vez que, son los encargados de que todas aquellas directrices brindadas por medio de JavaScript puedan llegar a buen puerto y ejecutarse en la parte visual para aportarle dinamismo a los diferentes elementos que componen a las vistas de Html del proyecto.

Además, aprovechando el espacio y que fue mencionado, aquí es donde debe abrirse el espacio de poder hablar acerca del JavaScript. Esta ha sido una herramienta medular en el funcionamiento del proyecto, ya que es la principal encargada de hacer al sistema responsivo, brindando un dinamismo importante en el funcionamiento general y en la estética de las transiciones desde un punto hacia otro a través de todo el proyecto. Como se señaló, esta se enlaza en gran medida con los controladores para varios de los procesos que requieren de una reacción de los procesos implementados en el Backend.

Por último, aquí puede destacarse al Html, el cual, es el medio que proporciona de la estética al proyecto, en conjunto con los estilos proporcionados por el CSS. Por lo que, el Html juega un importante papel dentro de todo proyecto porque el qué tan atractivo pueda resultar la estructura de un proyecto, es responsabilidad en gran medida de este sector del proyecto, razón que le vale una importante mención aquí, en conjunto a CSS.

Ahora bien, es importante profundizar en lo correspondiente a la importancia de esta arquitectura por capas, ya que, al realizar esta división en el proyecto, se puede implementar una separación lógica en diversas capas, como ha sido mencionado brevemente en los diferentes sectores que componen este proyecto. Ello aunado a que, la separación de la presentación, la lógica de negocios y la consulta de datos en este proyecto, le brindan un mayor orden, seguridad en el flujo de los datos, así como, de la mantenibilidad.

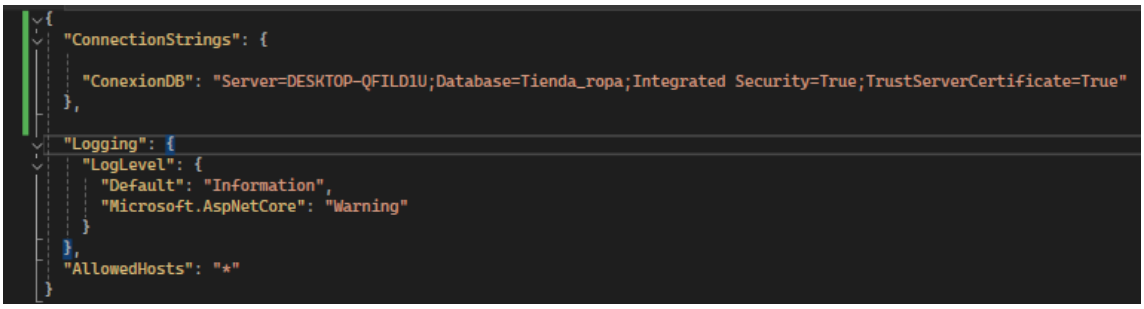
Elementos Clave del Backend

Como primero elemento clave podría mencionarse la seguridad, ya que, al implementarse sessionStorage en el Frontend para manejar datos temporales, o los procedimientos almacenados implementados con parámetros que ayudan a prevenir vulnerabilidades que pueden producirse a partir del SQL injection, lo cual, aporta una importante estabilidad a la información sensible que se podría manejar.

En lo que respecta a la gestión de los errores, puede destacarse que se emplea el manejo de excepciones mediante el elemento Reply. Este se implementa a través de todo el proyecto por medio de las propiedades Ok, Mensaje y Data para brindar una estandarización a las respuestas. Además a través de la aplicación se utiliza el try y catch para facilitar el debugging en el Backend.

También en lo que conviene al uso de servicios externos, puede destacarse que en este proyecto se han utilizado algunos que se encuentran presentes en el Frontend, tal cual, lo puede ser el Bootstrap y algunos CDN como Sweet Alert, JQuery y Font Awesome, los cuales, realizan un aporte importante al hablar de la estética de cualquier proyecto que se programe.

Conexión con la base de datos y el frontend



Conexión con la base de datos

En el código del código del presente proyecto, para hacer la conexión con la base de datos se realizó la implementación de estas líneas en el appsettings.json del Visual Studio, donde por medio de un código pequeño y simple, se puede establecer la conexión sin gran complicación.

Capturas o fragmentos de código:

Para lo correspondiente a este apartado se presentará una serie de fragmentos del código del proyecto que resulta clave para poder comprender de mejor manera su funcionamiento. Aquí se expondrá el código y cómo es que su trayecto desde el BackEnd, comenzando por los procedimientos almacenados en la base de datos, hasta el JavaScript para el dinamismo del FrontEnd.

Primeramente, se expondrá el código correspondiente al login del proyecto. Aquí se mostrará primero el flujo del backend hacia el frontend por medio de imágenes de estos y se explicarán al final los diferentes métodos y cómo es la dinámica por medio de la cual trabajan.

Procedimiento Almacenado

```
ALTER PROCEDURE [dbo].[spIniciarSesion] --Crea el sp y se determina su nombre
@pUsuario AS NVARCHAR (70),      --Determina el parametro Usuario que recibe este sp
@pContrasena AS NVARCHAR (70)    -- Lo mismo de arriba pero con la contraseña

AS

DECLARE @Exito AS BIT  -- Se hace booleano para que se verifique si es verdadero que está obteniendo el usuario y contraseña

BEGIN

    SET NOCOUNT ON; -- Esto es para que no salga el mensaje de "number of rows affected"

    --Ahora se procede a verificar si nuestro usuario existe

    IF EXISTS (

    SELECT 1 FROM Usuarios
    WHERE nombreUsuario = @pUsuario COLLATE Latin1_General_CS_AS  -- Latin1_General_CS_AS sirve para que cuando se haga la veri
    AND contrasenaUsuario = @pContrasena COLLATE Latin1_General_CS_AS --mayúsculas u otros signos y que no dé error automáticam
                                                                      --lo que hay en usuario y contrasena en la base de dato
    )

    BEGIN --A partir de aquí hicimos una modificacioncita para insertar la info de los cookies en el login

    DECLARE @NombreCompleto NVARCHAR (100); -- Almacena el nombre completo del usuario (nombre + apellidos)
    DECLARE @IdUsuario INT;                 -- Guarda el ID del usuario que inició sesión
    DECLARE @IdInfoUsuariosData INT;        -- Guarda el ID relacionado en la tabla infoUsuarios
    DECLARE @IdRol INT;                     -- Guarda el rol del usuario para redirección o validación

    --Obtiene el ide del Usuario autenticado
    SET @IdUsuario = (SELECT idUsuario FROM Usuarios
    WHERE nombreUsuario = @pUsuario COLLATE Latin1_General_CS_AS
    AND contrasenaUsuario = @pContrasena COLLATE Latin1_General_CS_AS)
```

```
    SET @IdInfoUsuariosData = (SELECT idInfoUsuario FROM Usuarios WHERE idUsuario = @IdUsuario)
    SET @NombreCompleto = (SELECT nombres + ' ' + apellido1 + ' ' + apellido2 FROM infoUsuarios WHERE idInfoUsuario = @IdInfoUsuariosData)
    SET @IdRol = (SELECT idRol FROM Usuarios WHERE idUsuario = @IdUsuario)


    -- En caso que sea verdadero el usuario y la contraseña, indica con 1 que el inicio de sesion fue exitoso

        SET @Exito = 1


        --Se devuelven los datos necesarios si fue exitoso, se recibe el nombre completo y el rol
        SELECT @Exito AS Exito, @NombreCompleto AS NombreCompleto, @IdUsuario AS idUsuario, @IdRol AS idRol --Aquí se deja el AS Exito para que

    END

    ELSE

    BEGIN

        SET @Exito = 0
        SELECT @Exito AS Exito --La misma razón de arriba
    END

END;
```

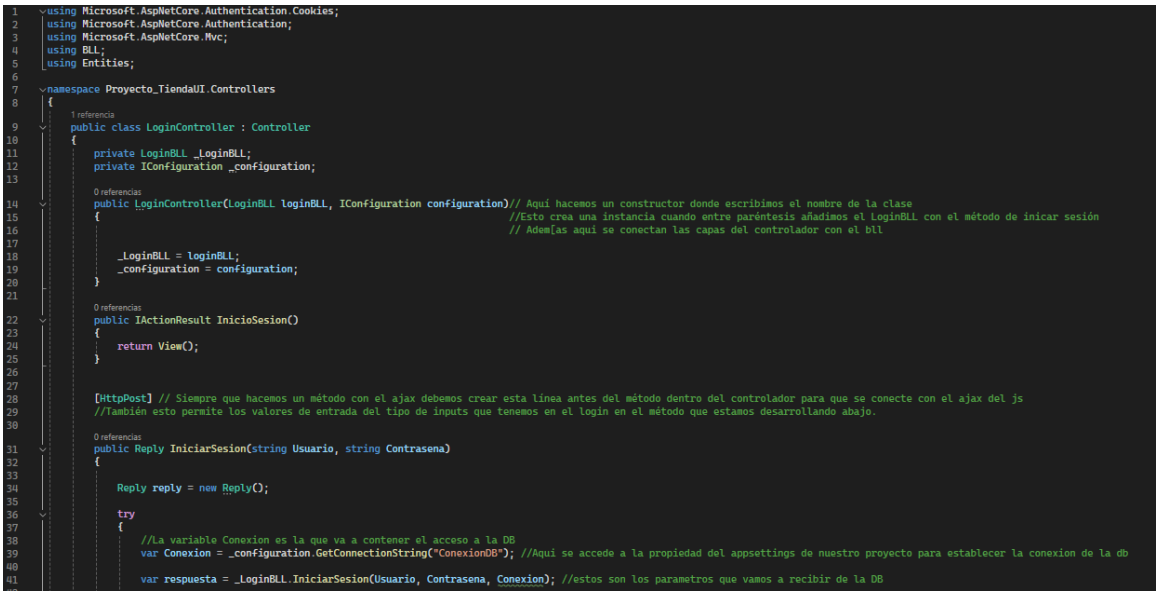
Capa de datos DAL.

```
using Entities;
using System.Data; // esta librer[ia nos permitirá conectar la base de datos
using Microsoft.Data.SqlClient;

namespace DAL
{
    3 referencias
    public class Login
    {
        1 referencia
        public Reply IniciarSesion(string Usuario, string Contrasena, string Conexion) //Al método le estamos brindándole los parámetros que se van a recibir del SP
        {
            List<Usuarios> usuarios = new List<Usuarios>(); // Esta lista con objetos de Usuario va a ser para que nosotros podamos agregar a la lista la info de los cookies
            Reply reply = new Reply();

            try
            {
                //Aqu[i se crea la instancia de conexión

                using (SqlConnection connection = new SqlConnection(Conexion))
                {
                    connection.Open(); //Abrimos la conexión a la base de datos

                    //Ahora llamamos al procedimiento almacenado

                    using (SqlCommand command = new SqlCommand("spIniciarSesion", connection))
                    {
                        //Aqui se le indica al sp que tiene que buscar

                        command.CommandType = CommandType.StoredProcedure;

                        //Vamos a agregarle los parametros que la instancia recibe

                        command.Parameters.Add(new SqlParameter("@pUsuario", SqlDbType.NVarChar, 70) { Value = Usuario });
                        command.Parameters.Add(new SqlParameter("@pContrasena", SqlDbType.NVarChar, 70) { Value = Contrasena });

                        // Aquí se va a la db y recupera si existe el usuario y contraseña
                        using (SqlDataReader reader = command.ExecuteReader())
                        {

                            if (reader.Read())
                            {
                                bool Exito = (bool)reader["Exito"]; //Se pone Exito como booleano porque así esta en la db y si no se hace así da error

                                if (Exito) //Si devuelve que es verdadero, es que el usuario existe
                                {
                                    Usuarios usuario = new Usuarios
                                    {
                                        NombreCompleto = (string)reader["NombreCompleto"],
                                        idRol = (int)reader["idRol"],
                                        idUsuario = (int)reader["idUsuario"]

                                    };

                                    usuarios.Add(usuario); //Esto es en respuesta a la variable almacenada en la lista de objetos de Usuarios
                                    //Traduciendo xD, usuarios es la variable de la lista de Usuarios que interactua con los parametros de la DB
                                    //usuario en singular es la variable donde se va a guardar el usuario especicio que se logea, que con el .Add
                                    //se guardara en la variable usuarios que guarda los objetos que se leen en la lista Usuarios para que se lea nuestra cookie

                                    reply.Ok = true;
                                    reply.Mensaje = "¡Inicio de sesión exitoso!";
                                    reply.Usuarios = usuarios; //Aquí se guarda el usuario en la instancia que se manda a Reply para que se muestre la
                                                               //info y que al dar respuesta entonces en respuesta exista la info guardada en el reply para usuarios y se pueda

                                }
                            }
                            else
                            {
                                // ESTE ES EL CASO QUE FALTABA: Cuando no hay resultados
                                reply.Ok = false;
                                reply.Mensaje = "¡El usuario o la contraseña no son correctos!";
                            }
                        }
                    }
                }
            }
            catch (Exception ex)
            {

                reply.Ok = false;
                reply.Mensaje = ex.Message;

            }
            return reply;
        }
```

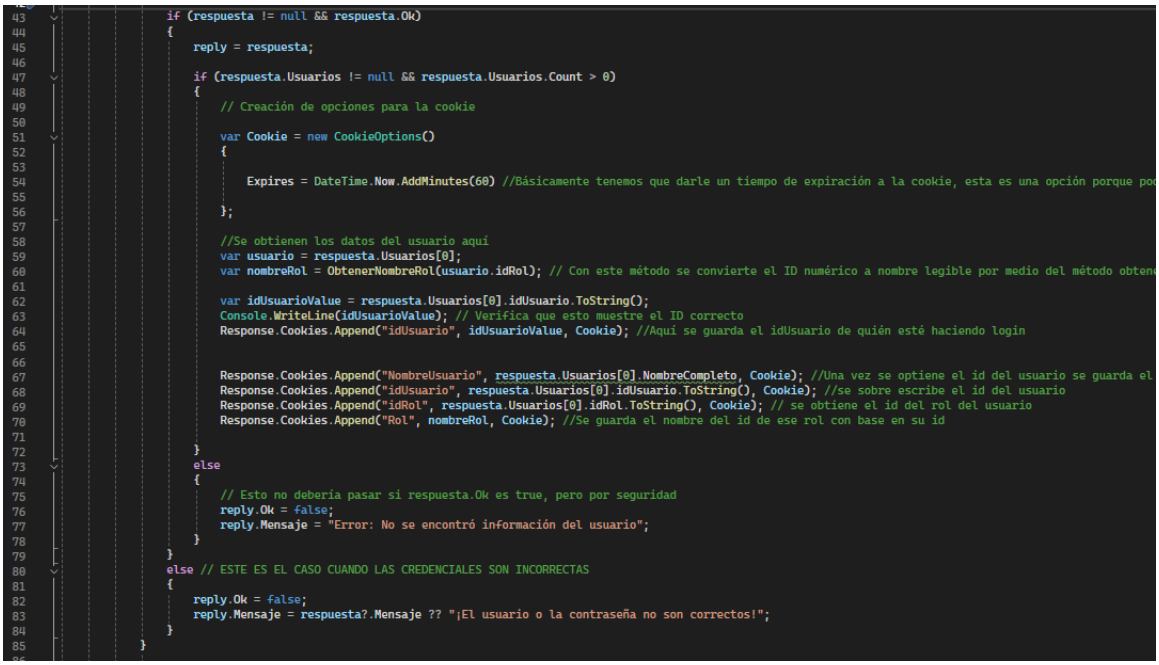
Capa de lógica de negocios BLL

```
using DAL;
using Entities;

namespace BLL
{
    4 referencias
    public class LoginBLL
    {
        private readonly Login _LoginDAL; // Este constructor es una instancia que tenemos que crear para que solamente aquí se tenga acceso a DAL y haya seguridad en el proyecto

        0 referencias
        public LoginBLL(Login loginDAL) //es para que construya todos los métodos que tiene DAL en este BLL
        {
            _LoginDAL = loginDAL;
        }

        1 referencia
        public Reply IniciarSesion(string Usuario, string Contrasena, string Conexion) //hay que hacer el metodo en las diferentes capas con los mismos parametros
        {
            Reply reply = new Reply(); // siempre se crea el reply para que se almacenen los objetos y se tenga la respuesta en el metodo

            try //Dentro del try vamos a hacer que el BLL tenga acceso al DAL, por lo que vamos a llamarlo aqui
            {
                var respuesta = _LoginDAL.IniciarSesion(Usuario, Contrasena, Conexion);

                if (respuesta != null) //Si respuesta no viene en blanco, entonces procede con lo que sigue
                {
                    reply = respuesta; // Este reply devuelve el objeto reply que da los parametros de Usuario, contrasena y conexion

                }

            }
            catch (Exception ex)
            {
                reply.Ok = false;
                reply.Mensaje = ex.Message;

            }
            return reply;
        }
```

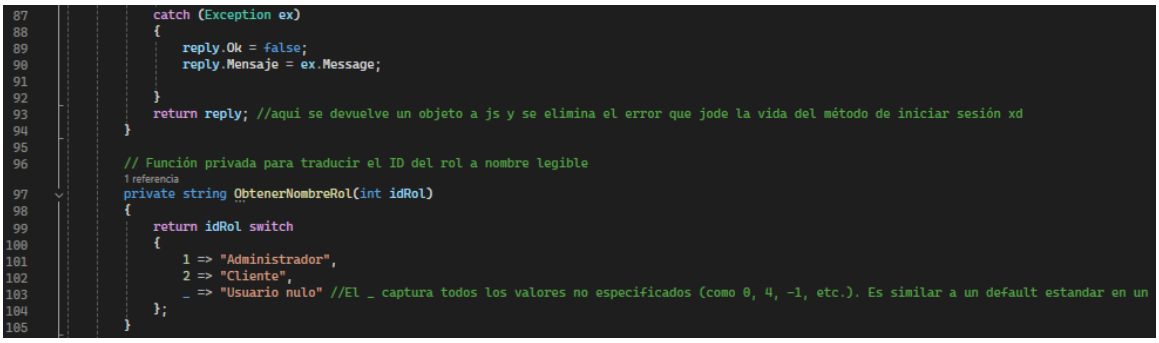
Controlador



```
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Mvc;
using BLL;
using Entities;

namespace Proyecto_TiendaUI.Controllers
{
    1 referencia
    public class LoginController : Controller
    {
        private LoginBLL _LoginBLL;
        private IConfiguration _configuration;

        0 referencias
        public LoginController(LoginBLL loginBLL, IConfiguration configuration)// Aquí hacemos un constructor donde escribimos el nombre de la clase
        {                                                                       //Esto crea una instancia cuando entre paréntesis añadimos el LoginBLL con el método de inicar sesión
                                                                                // Adem[as aquí se conectan las capas del controlador con el bll

            _LoginBLL = loginBLL;
            _configuration = configuration;
        }

        0 referencias
        public IActionResult InicioSesion()
        {
            return View();
        }


        [HttpPost] // Siempre que hacemos un método con el ajax debemos crear esta línea antes del método dentro del controlador para que se conecte con el ajax del js
        //También esto permite los valores de entrada del tipo de inputs que tenemos en el login en el método que estamos desarrollando abajo.

        0 referencias
        public Reply IniciarSesion(string Usuario, string Contrasena)
        {

            Reply reply = new Reply();

            try
            {
                //La variable Conexion es la que va a contener el acceso a la DB
                var Conexion = _configuration.GetConnectionString("ConexionDB"); //Aquí se accede a la propiedad del appsettings de nuestro proyecto para establecer la conexion de la db

                var respuesta = _LoginBLL.IniciarSesion(Usuario, Contrasena, Conexion); //estos son los parametros que vamos a recibir de la DB
```



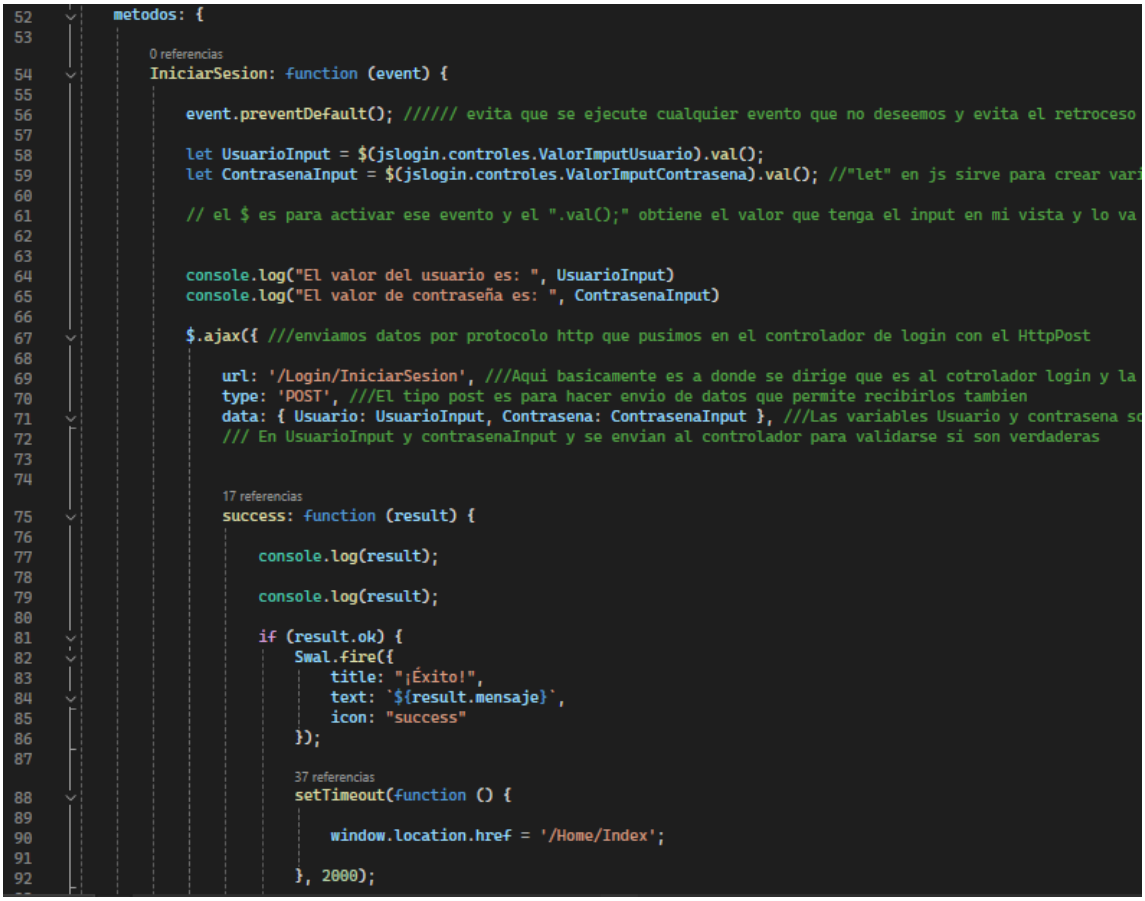
```
            if (respuesta != null && respuesta.Ok)
            {
                reply = respuesta;

                if (respuesta.Usuarios != null && respuesta.Usuarios.Count > 0)
                {
                    // Creación de opciones para la cookie

                    var Cookie = new CookieOptions()
                    {

                        Expires = DateTime.Now.AddMinutes(60) //Básicamente tenemos que darle un tiempo de expiración a la cookie, esta es una opción porque poc

                    };

                    //Se obtienen los datos del usuario aquí
                    var usuario = respuesta.Usuarios[0];
                    var nombreRol = ObtenerNombreRol(usuario.idRol); // Con este método se convierte el ID numérico a nombre legible por medio del método obtene

                    var idUsuarioValue = respuesta.Usuarios[0].idUsuario.ToString();
                    Console.WriteLine(idUsuarioValue); // Verifica que esto muestre el ID correcto
                    Response.Cookies.Append("idUsuario", idUsuarioValue, Cookie); //Aquí se guarda el idUsuario de quién esté haciendo login


                    Response.Cookies.Append("NombreUsuario", respuesta.Usuarios[0].NombreCompleto, Cookie); //Una vez se optiene el id del usuario se guarda el
                    Response.Cookies.Append("idUsuario", respuesta.Usuarios[0].idUsuario.ToString(), Cookie); //se sobre escribe el id del usuario
                    Response.Cookies.Append("idRol", respuesta.Usuarios[0].idRol.ToString(), Cookie); // se obtiene el id del rol del usuario
                    Response.Cookies.Append("Rol", nombreRol, Cookie); //Se guarda el nombre del id de ese rol con base en su id

                }
                else
                {
                    // Esto no debería pasar si respuesta.Ok es true, pero por seguridad
                    reply.Ok = false;
                    reply.Mensaje = "Error: No se encontró información del usuario";
                }
            }
            else // ESTE ES EL CASO CUANDO LAS CREDENCIALES SON INCORRECTAS
            {
                reply.Ok = false;
                reply.Mensaje = respuesta?.Mensaje ?? "¡El usuario o la contraseña no son correctos!";
            }
        }
```



```
        catch (Exception ex)
        {
            reply.Ok = false;
            reply.Mensaje = ex.Message;

        }
        return reply; //aquí se devuelve un objeto a js y se elimina el error que jode la vida del método de iniciar sesión xd
    }

    // Función privada para traducir el ID del rol a nombre legible
    1 referencia
    private string ObtenerNombreRol(int idRol)
    {
        return idRol switch
        {
            1 => "Administrador",
            2 => "Cliente",
            _ => "Usuario nulo" //El _ captura todos los valores no especificados (como 0, 4, -1, etc.). Es similar a un default estandar en un
        };
    }
```

JavaScript

```javascript
jslogin = {  ///NOTA: en mis anotaciones de progra 3 quedé en las notas de vista de login para continuar programando desde allí.

    objetos: {


    },

    controles: {

        ValorImputUsuario: '#inputUsuario', ///Es una variable vacia que va a guardar el nombre del input del usuario
        ValorImputContrasena: '#inputContrasena',
        ValorEmailRecuperar: '#emailRecuperar',
        ModalenPantalla: '#MostrarModalRecuperar',

        //Ids para el modal de crear cuenta
        ModalCrearCuenta: '#MostrarModalCrearCuenta',
        BtnVolverLoginCrearCuenta: '#volverLoginCrearCuenta',

        // Mapeo de los inputs del modal crear cuenta
        inputNuevoUser: '#inputNuevoUser',
        inputContrasenaNuevoUser: '#inputContrasenaNuevoUser',
        inputNombreCompleto: '#inputNombreCompleto',
        inputCorreo: '#inputCorreo',
        inputTelefono: '#inputTelefono',
        dropdownRol: '#dropdownRol',


    },

    botones: {

        ValorButtonLogin: '#btnIniciarSesion', ///Mapeo de nuestro botón de inicio de sesión
        ValorBtnEnviarRecuperacion: '#btnEnviarRecuperacion',
        BtnAMOlvideContra: '#LinkModalOlvideContra',  // Desde ahora AM es Abrir Modal para hacer más breve las variables e igualm
        BtnCMOlvideContra: '#closeModal',
        BtnVolverLogin: '#volverLogin',

        //Botones para creación de nuevo usuario
        BtnAMCrearCuenta: '#LinkCrearNuevaCuenta',
        BtnCMCrearCuentaX: '#closeModalCrearCuenta',
        BtnCMCrearCuentaCancelar: '#btnCancelar',
        BtnVolverLoginCrearCuenta: '#volverLoginCrearCuenta',
        BtnGuardarUsuario: '#btnGuardarUsuario',

```



```
52      metodos: {
53
                0 referencias
54          IniciarSesion: function (event) {
55
56              event.preventDefault(); ////// evita que se ejecute cualquier evento que no deseemos y evita el retroceso
57
58              let UsuarioInput = $(jslogin.controles.ValorImputUsuario).val();
59              let ContrasenaInput = $(jslogin.controles.ValorImputContrasena).val(); //"let" en js sirve para crear vari
60
61              // el $ es para activar ese evento y el ".val();" obtiene el valor que tenga el input en mi vista y lo va
62
63
64              console.log("El valor del usuario es: ", UsuarioInput)
65              console.log("El valor de contraseña es: ", ContrasenaInput)
66
67              $.ajax({ ///enviamos datos por protocolo http que pusimos en el controlador de login con el HttpPost
68
69                  url: '/Login/IniciarSesion', ///Aqui basicamente es a donde se dirige que es al cotrolador login y la
70                  type: 'POST', ///El tipo post es para hacer envio de datos que permite recibirlos tambien
71                  data: { Usuario: UsuarioInput, Contrasena: ContrasenaInput }, ///Las variables Usuario y contrasena so
72                  /// En UsuarioInput y contrasenaInput y se envian al controlador para validarse si son verdaderas
73
74
                    17 referencias
75                  success: function (result) {
76
77                      console.log(result);
78
79                      console.log(result);
80
81                      if (result.ok) {
82                          Swal.fire({
83                              title: "¡Éxito!",
84                              text: `${result.mensaje}`,
85                              icon: "success"
86                          });
87
                            37 referencias
88                          setTimeout(function () {
89
90                              window.location.href = '/Home/Index';
91
92                          }, 2000);
```

```
94                      } else {
95                          Swal.fire({
96                              title: "¡Error!",
97                              text: `${result.mensaje}`,
98                              icon: "warning"
99                          });
100                     }
101
102                 },
103
                    15 referencias
104                 error: function () {
105                     Swal.fire({
106                         title: "¡Error!",
107                         text: `${result.mensaje}`,
108                         icon: "error"
109                     });
110                 }
111             });
112         },
113
```

Ahora, a partir de aquí se explicar el proceso entero del flujo del proyecto en términos generales, al tomar como referencia el funcionamiento de los métodos de inicio de sesión. Primeramente, se parte del procedimiento almacenado en la base de datos. Se inicia delimitando los diferentes parámetros que se utilizarán para el método de inicio de sesión, los cuales incluyen las credenciales del login que puede encontrarse comúnmente, como lo son el nombre del usuario y la contraseña, los cuáles, son recolectados por medio de los inputs del frontend.

Posteriormente, el procedimiento almacenado se conecta con el proyecto por medio de una instancia de conexión en la capa DAL de datos, en esta, se hace un llamado específico por medio del nombre del procedimiento que se está implementando, además, de los parámetros que se utilizarán del mismo, de manera que en el proyecto queda registrada esta información.

Seguidamente, se procede a lo que refiere a la capa de la lógica de negocios con la que cuenta el proyecto, la cual, sería la capa BLL. Para el caso específico del login, solo está haciendo conexión de la capa de datos con el controlador, pero para efectos de otras funciones como cuando se añade un nuevo usuario, en esta capa se integran lo que son el nombre del usuario, con su primer y segundo apellido. Pero como se comentó anteriormente, para efectos de este método en específico, su función es de hacer de puente que une el backend.

Posteriormente, se encuentra lo que sería el controlador, el cual tiene funciones muy importantes, entre las que cabe destacar la filtración de usuarios por rol al hacer inicio de sesión, así de esta manera, se podrá renderizar una vista diferente para el usuario dependiendo el tipo de rol que ejerza. Además, de esto, el controlador, recibirá la información que provenga de los inputs que le proporciona el JavaScript. Con base en ello, puede enviarse esta información hacia la base de datos por medio de las capas anteriormente comentadas, de forma que se verifica si está errónea o si es correcta, de manera que le permita al usuario hacer ingreso o le detenga de iniciar sesión.

Por medio de esta explicación, es que quiere evidenciársele al lector, la manera en que se desenvuelve el flujo de los datos a través de las diferentes capas del proyecto, además, para mostrar cuál es la manera en que funciona el proyecto a partir de su Backend. El resto de métodos, se desenvuelven de la misma manera, solo que pueden

variar dependiendo de los parámetros y la lógica que necesite implementarse para cada uno de ellos.

Conclusión

El proyecto se desarrolló como una propuesta que pudiera brindar solución al hecho de que la tienda “Tentación Azul” no contaba anteriormente con su plataforma de e-comerce, por lo que se buscó solventar esta situación. A partir del presente, puede extraerse que se elaboró una base de datos, que permitirá un mejor registro contable y de inventario para el establecimiento, asimismo, como de una plataforma interactiva tanto para el cliente como para el administrador de la tienda. Así se pudo cumplir con los alcances esperados, ya que, se permite el registro de nuevos usuarios, se permite el ingreso de nueva mercadería, así como, su gestión en lo que es el manejo de stock. Por otro lado también tanto, el cliente como el administrador, pueden realizar seguimiento del historial de pedidos y su estado en tiempo real, por lo que puede destacarse que se cumplió con estas metas trazadas.