

# Proyecto BackEnd

## Descripción del Proyecto

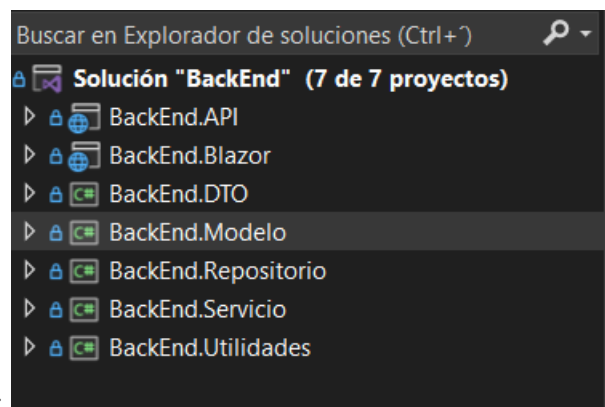
La presente aplicación fue realizada en Visual Studio 2022, utilizando la tecnología .net core 7.0, con los proyectos de Blazor (Aplicación en Webassembly), también tiene su Proyecto API () y otras bibliotecas que ayudan al traspaso de información sin tener que exponer los datos de la Base en el FrontEnd.

En el presente proyecto, se realiza la muestra de una aplicación simulando una tienda de artículos, se tiene los siguientes roles de prueba:

IdUsuario	NombreCompleto	Correo	Clave	Rol
1	Admin	admin@gmail.com	admin	Administrador
5	cliente	cliente@gmail.com	123	Cliente

Se puede hacer el CRUD en todas las tablas, Usuario, Categorías, Productos. Para un correcto funcionamiento de la aplicación se deja el archivo backend.bakbanckend.bak, dentro de la carpeta README. La base de datos backend fue realizada en sqlserver versión 18.11.1.

## Estructura del Proyecto



La estructura de la solución BackEnd es la siguiente:

A continuación explicamos para que sirve cada librería o proyecto:

### 1. API

Se trata de un proyecto, el cual contiene los controladores y la conexión a la Base de Datos y el mismo gracias a Swagger logra dar una vista de la API.

### 2. BLAZOR

Este es el proyecto realizado en Blazor WebAssembly, en el se tiene la estructura del FrontEnd y se alimenta de la API y del Proyecto DTO.

### 3. DTO

Este Proyecto es una biblioteca de clases, es una capa de transferencia de datos, es decir que de una clase

# BackEnd

USUARIO, se crearon las clases para el login, para sesión y para los datos que se necesitan mostrar del usuario y también con las otras tablas se ajustaron algunos datos.

## 4. MODELO

Este proyecto es una biblioteca de clases, en la cual se tienen las clases o entidades originales, que nacieron de la base de datos. (Categoría, Detalle Venta, Usuario, Producto, Venta).

## 5. REPOSITORIO

En esta biblioteca de clases, se tiene el DbContext, que es la conexión a la base de datos original y en él se crean Interfaces y Clases que son de uso genérico.

## 6. SERVICIO

En esta biblioteca de clases, se usa la capa DTO, para crear las interfaces y clases que posteriormente se usan en el proyecto Blazor y muestran los datos en el FrontEnd.

## 7. UTILIDADES

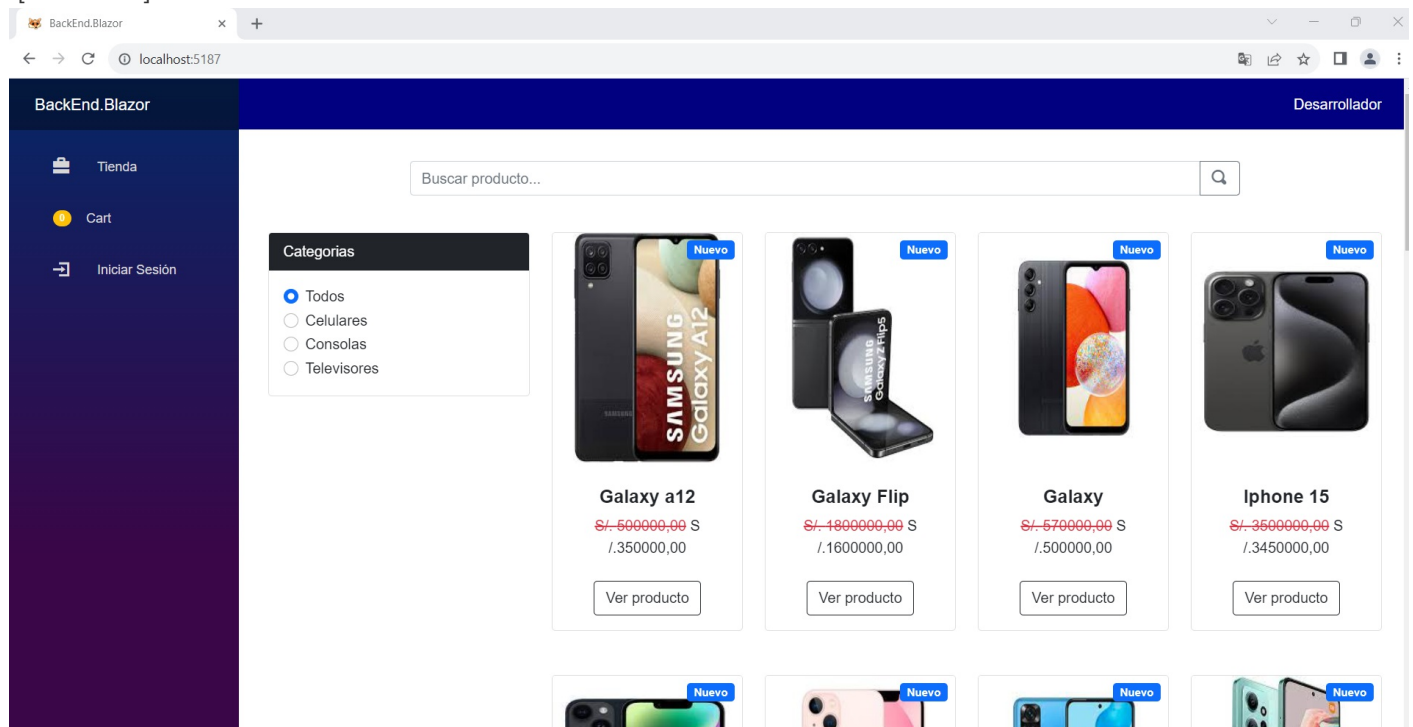
Esta biblioteca de clases, utiliza un nuget llamado **Automapper**, gracias al cual se realiza el mapeo de los datos entre la capa DTO con nuestra Base original, esto para lograr la transferencia de datos correcta.

# Funcionalidades De la aplicación BackEnd

## Pantalla de Inicio

En ella se ingresa sólo al catalogo de productos el cual tiene un filtro para buscar por categorías y un buscador. Para poder comprar un articulo el cliente deberá iniciar sesión.

![Pantalla 1]



## Pantalla Login

En ella se podrá ingresar como administrador o como cliente, además el cliente podrá registrarse, para poder realizar su compra.

### Login

Correo  
 cliente@gmail.com

Contraseña  
 ...

Ingresar

[¿No tienes una cuenta? Regístrate!](#)

## Pantalla Cliente

Como cliente se podrá comprar los artículos del catálogo, se hace una compra ficticia, para simular que un cliente puede ingresar datos de su tarjeta y realizar su compra. En el carro de comprar Cart, podrá eliminar los productos escogidos y aumentar o disminuir la cantidad.


BackEnd.Blazor
Desarrollador

Tienda

cliente@gmail.com

1 Cart

🔌 Salir

Producto	Precio	Cantidad	Total
 Galaxy a12	350000,00	<div style="display: inline-block; border: 1px solid #ccc; padding: 2px 5px;">-</div> <div style="display: inline-block; border: 1px solid #ccc; padding: 2px 5px; margin: 0 5px;">1</div> <div style="display: inline-block; border: 1px solid #ccc; padding: 2px 5px;">+</div>	350000,00

Nombre Titular

cliente

Numero Tarjeta

13124515

Vigencia

CVV

03/27

123

Total a pagar: S/. 350000,00

Procesar pago

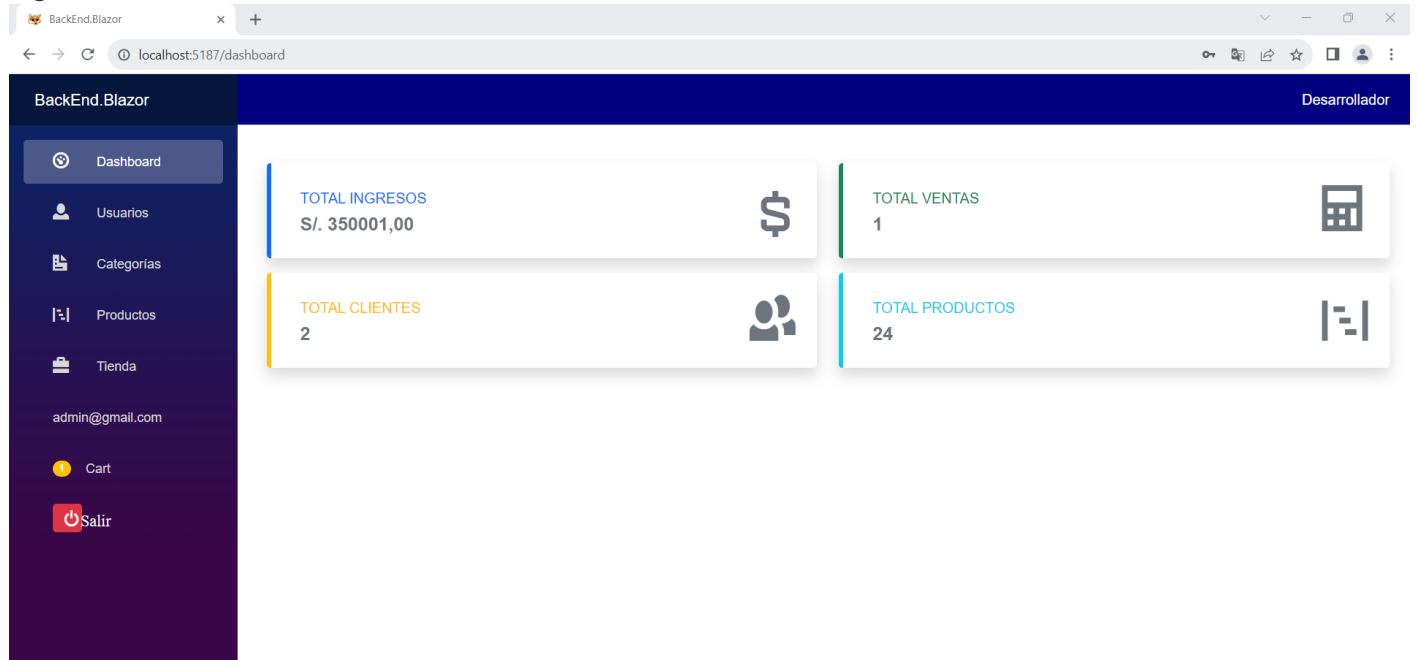
## Pantalla Admin

## BackEnd

En esta pantalla, al ingresar con el usuario **admin@gmail.com** y clave **admin**, el usuario podrá ver varias pantallas:

### Dashboard

En la cual se muestran la cantidad de ventas, el total de productos, total de ingresos y los clientes que se tienen registrados.



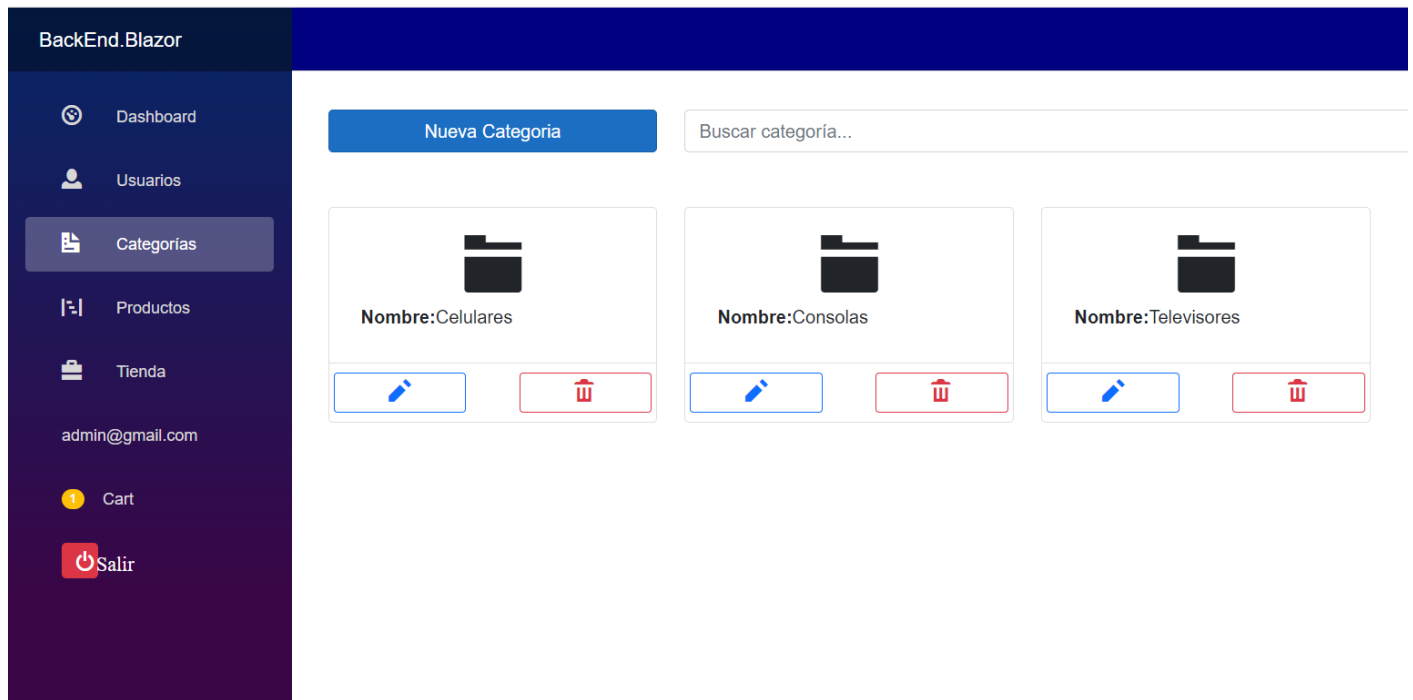
### Usuarios

En esta pantalla se podrán ingresar nuevos usuarios, editarlos y eliminarlos.

### Categorías

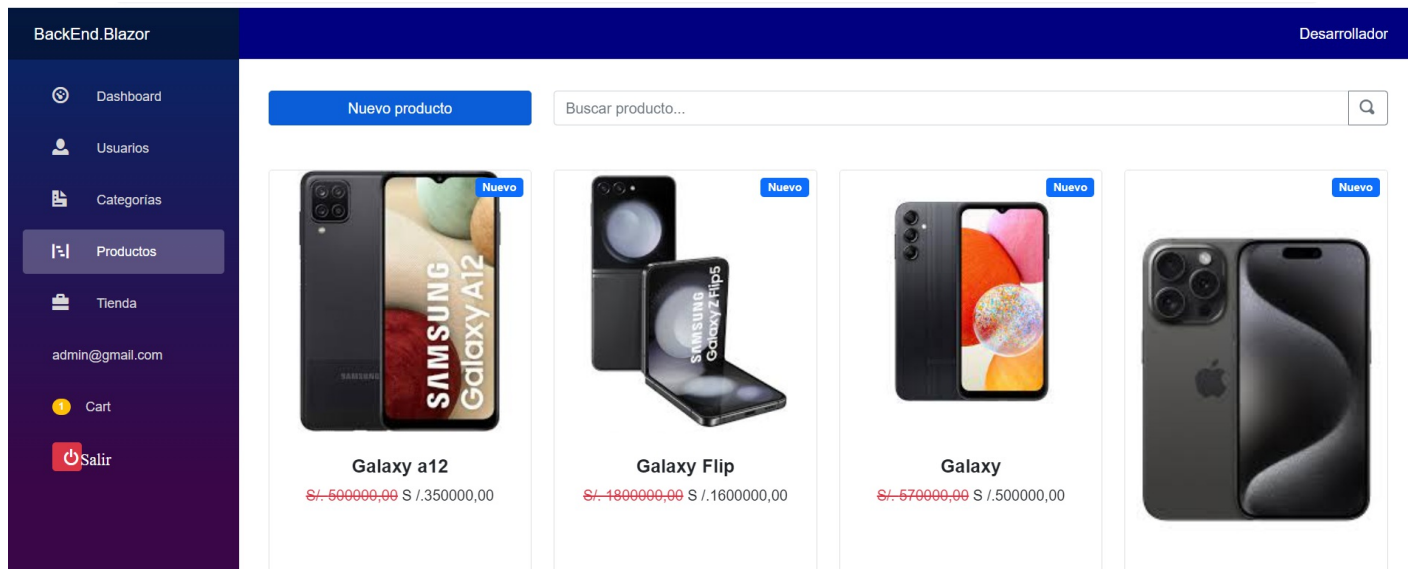
En esta pantalla se muestran las categorías creadas las cuales pueden ser editadas y se pueden crear nuevas categorías.

## BackEnd



## Productos

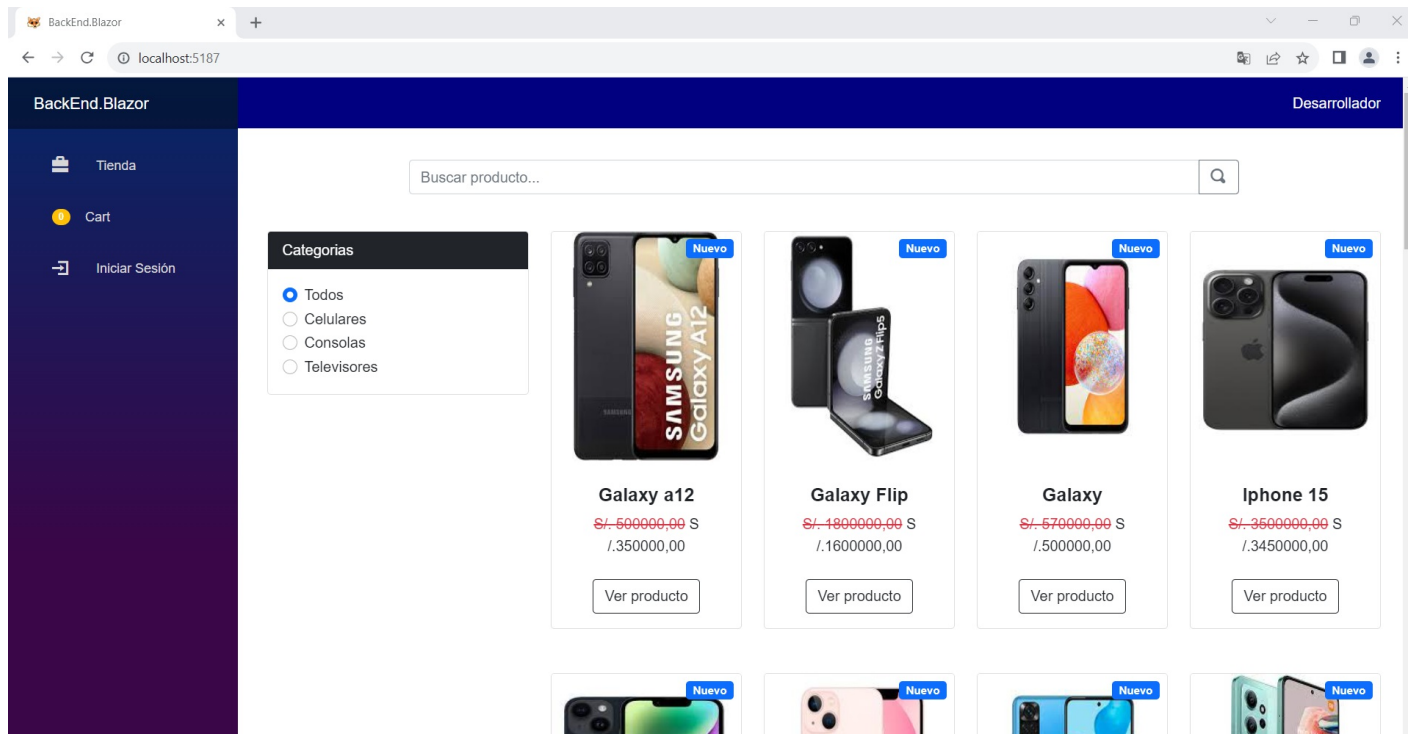
En esta pantalla se pueden agregar, editar o eliminar productos y se ven los productos en su totalidad, con un input para buscarlos, en esta pantalla a diferencia de la pantalla catálogo no se tiene un radio button para seleccionar categorías.



## Catálogo

Esta es la pantalla de inicio de la aplicación, cuando no existe ningún usuario registrado, en ella se pueden ver todos los productos y se puede filtrar o buscar un producto.

# BackEnd

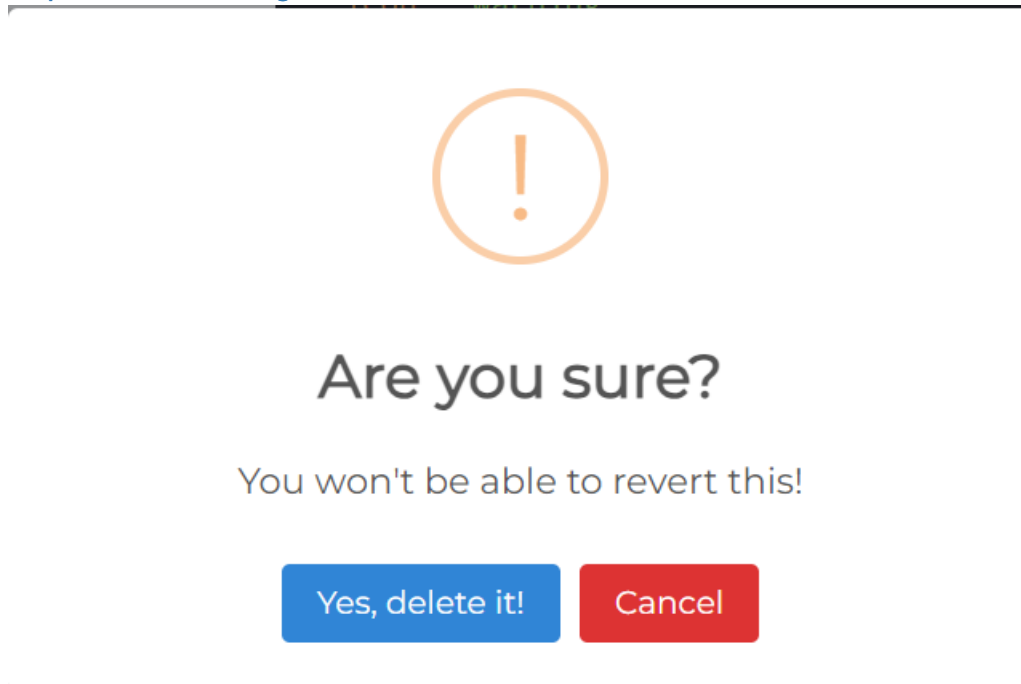


## Otras Pantallas

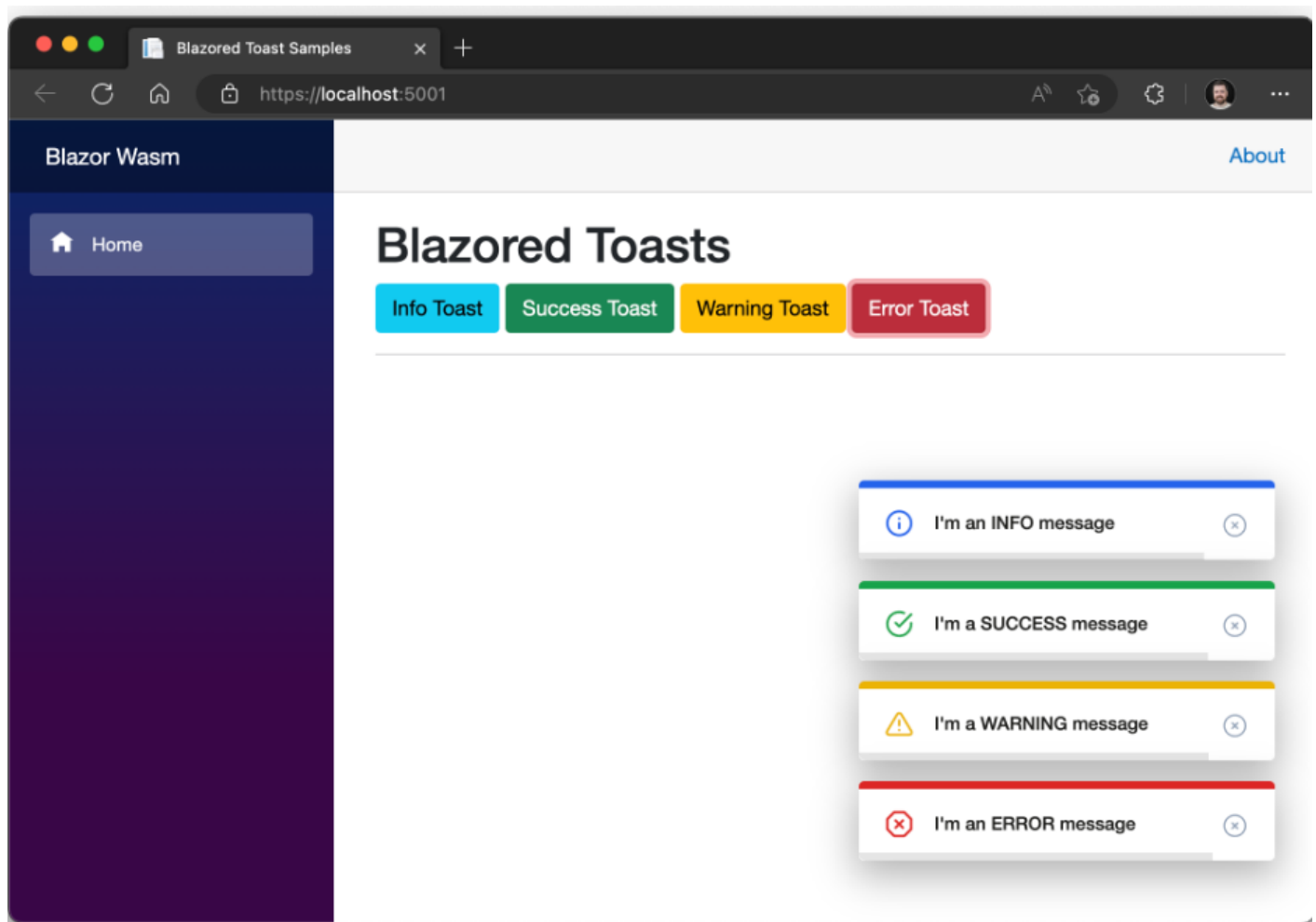
Al hacer click en un producto se mostrará una página con los detalles del producto, imagen, precio y se podrá agregar al carro de compras (cart).

## Otras Funcionalidades

Se muestran mensajes y alertas un poco más amigables con el usuario, usando Nugets como ser Sweetalert2 <https://sweetalert2.github.io/>



También se utiliza Blazored.Toast, para mostrar alertas mas novedosas. <https://github.com/Blazored/Toast>



Por otro lado y no menos importante se tiene el proyecto de api, que se realiza con swagger, esta tecnología está incluida en los proyectos API de Microsoft, por lo que es muy fácil su construcción, teniendo una base de datos o entidades bien definidas. <https://swagger.io/>

Categoria ^	
GET	/api/Categoria/Obtener/{Id} v
GET	/api/Categoria/Lista/{buscar} v
POST	/api/Categoria/Crear v
PUT	/api/Categoria/Editar v
DELETE	/api/Categoria/Eliminar/{Id} v
Dashboard ^	
GET	/api/Dashboard/Resumen v
Producto ^	
GET	/api/Producto/Lista/{buscar} v
GET	/api/Producto/Catalogo/{categoria}/{buscar} v
GET	/api/Producto/Obtener/{Id} v
POST	/api/Producto/Crear v
PUT	/api/Producto/Editar v
DELETE	/api/Producto/Eliminar/{Id} v
Usuario ^	
GET	/api/Usuario/Lista/{rol}/{buscar} v
GET	/api/Usuario/Obtener/{id} v
POST	/api/Usuario/Crear v
POST	/api/Usuario/Autorizacion v
PUT	/api/Usuario/Editar v
DELETE	/api/Usuario/Eliminar/{Id} v
Venta ^	
POST	/api/Venta/Registrar v

## Posible Mejora

Se intentó usar QuickGrid <https://aspnet.github.io/quickgridsamples/>, para una forma de generar tablas de manera mucho más vistosa, pero no fue posible la correcta configuración entonces se decidió dejar pendiente esa mejora.

## Autor del proyecto

Jaime Enrique Dávila Zuazo