1 Conexin a la base de datos :

"AllowedHosts": "\*",

"ConnectionStrings": {

"ConexionMontecastelo": "Data Source=.\\MSSQLSERVER01;Initial Catalog= AmazonMontecastelo;Trusted\_Connection=true;MultipleActiveResultSets=true;TrustServerCertificate=True"

}

Program:

builder.Services.AddSingleton(new Conexion(builder.Configuration.GetConnectionString("ConexionMontecastelo")));

//Registro de interfaz

builder.Services.AddScoped<IProductoRepository, ProductoRepository>();

app.MapControllers();

GlobalInfo:

public static class GlobalInfo

{

public static Usuario UsuarioLogeado { get; set; }

public static bool IsLogged { get { return UsuarioLogeado != null; } }

Se realiza para que siempre este existe los datos en el carrito y venta

public static Carrito MiCarrito { get; set; }

public static Venta MiVenta { get; set; }

public static string LoginView = "/Views/Login/Logueate.cshtml";

}

Conexión:

public class Conexion

{

private readonly string \_connectionstring;

public Conexion(string connection)

{

\_connectionstring = connection;

}

//Método (ObtenerConexion()) que devuelve la conexión.

public SqlConnection ObtenerConexion()

app

{

var conexion = new SqlConnection(\_connectionstring);

conexion.Open();

return conexion;

}

}

public interface IProductoRepository

{

Task<IEnumerable<Productos>> GetAllProducto();

Task<Productos> GetProducto(int ProductoId);

Task CreateProducto(Productos producto);

Task UpdateProducto(Productos producto);

Task DeleteProducto(int? ProductoID);

Task<IEnumerable<Productos>> Filtrar(string? nombre, decimal? precioDesde, decimal? precioHasta);

}

public class ProductoRepository : IProductoRepository

{

private readonly Conexion \_conexion;

public ProductoRepository(Conexion conexion)

{

\_conexion = conexion;

}

public async Task<IEnumerable<Productos>> GetAllProducto()

{

var query = @"SELECT \* from Productos";

using var connection = \_conexion.ObtenerConexion();

return await connection.QueryAsync<Productos>(query);

}

public async Task<Productos> GetProducto(int ProductoId)

{

var query = @"SELECT \*

from Productos Where ProductoID = @ProductoId ";

using var connection = \_conexion.ObtenerConexion();

return await connection.QuerySingleAsync<Productos>(query, new { ProductoId });

}

public async Task CreateProducto(Productos producto)

{

var query = @"Insert into Productos (Precio,Descripcion,Nombre) Values (@Precio,@Descripcion,@Nombre)";

using var connection = \_conexion.ObtenerConexion();

await connection.ExecuteAsync(query, new { producto.Nombre, producto.Descripcion, producto.Precio });

}

public async Task UpdateProducto(Productos producto)

{

var queryProducto = @"UPDATE Productos

SET Descripcion = @Descripcion,

Nombre = @Nombre,

Precio = @Precio

WHERE ProductoID = @ProductoID";

using var connection = \_conexion.ObtenerConexion();

await connection.ExecuteAsync(queryProducto, new { producto.Descripcion,producto.Nombre, producto.Precio,producto.ProductoID });

}

public async Task DeleteProducto(int? ProductoID)

{

var query = @"Delete from Productos WHERE ProductoID = @ProductoID ";

using var connection = \_conexion.ObtenerConexion();

await connection.ExecuteAsync(query, new { ProductoID });

}

public async Task<IEnumerable<Productos>> Filtrar(string? nombre, decimal? precioDesde, decimal? precioHasta)

{

var sqlQuery = "SELECT \* FROM Productos WHERE 1 = 1 ";

using var conexion = \_conexion.ObtenerConexion();

if (nombre is not null)

{

sqlQuery += " AND Nombre LIKE @nombre";

}

if (precioDesde is not null)

{

sqlQuery += " AND Precio >= @precioDesde";

}

if (precioHasta is not null)

{

sqlQuery += " AND Precio <= @precioHasta";

}

return await conexion.QueryAsync<Productos>(sqlQuery,new {nombre,precioDesde,precioHasta});

}

}

Controller

public class AmazonController : Controller

{

private readonly ProductoRepository \_productoRepositorio;

public AmazonController(Conexion context)

{

\_productoRepositorio = new ProductoRepository(context);

}

[Route("/")]

[Route("/Amazon")]

[Route("/Amazon/Productos")]

public async Task<IActionResult> Producto()

{

if (!GlobalInfo.IsLogged)

{

return View(GlobalInfo.LoginView);

}

if (GlobalInfo.UsuarioLogeado.userType != "administrador")

{

TempData["MensajeAccesoDenegado"] = "No tienes permisos para acceder a esta página.";

return RedirectToAction("AccesoDenegado");

}

var productos = await \_productoRepositorio.GetAllProducto();

return View(productos);

}

[Route("/Amazon/agregar")]

public IActionResult Agregar()

{

if (!GlobalInfo.IsLogged)

{

return View(GlobalInfo.LoginView);

}

if (GlobalInfo.UsuarioLogeado.userType != "administrador")

{

TempData["MensajeAccesoDenegado"] = "No tienes permisos para acceder a esta página.";

return RedirectToAction("AccesoDenegado");

}

return View(new Productos());

}

[HttpPost]

[Route("/Amazon/Create")]

public async Task<IActionResult> Create(Productos producto)

{

if (!GlobalInfo.IsLogged)

{

return View(GlobalInfo.LoginView);

}

if (producto.ProductoID > 0)

{

await \_productoRepositorio.UpdateProducto(producto);

}

else

{

await \_productoRepositorio.CreateProducto(producto);

}

if (GlobalInfo.UsuarioLogeado.userType != "administrador")

{

// Redirige o muestra un mensaje de error porque el usuario no es un administrador

TempData["MensajeAccesoDenegado"] = "No tienes permisos para acceder a esta página.";

return RedirectToAction("AccesoDenegado");

}

return RedirectToAction("Producto");

}

[Route("/Amazon/Edit/{id}")]

public async Task<IActionResult> Edit(int id)

{

if (!GlobalInfo.IsLogged)

{

return View(GlobalInfo.LoginView);

}

if (GlobalInfo.UsuarioLogeado.userType != "administrador")

{

// Redirige o muestra un mensaje de error porque el usuario no es un administrador

TempData["MensajeAccesoDenegado"] = "No tienes permisos para acceder a esta página.";

return RedirectToAction("AccesoDenegado");

}

var producto = await \_productoRepositorio.GetProducto(id);

return View("Agregar", producto);

}

[Route("/Amazon/Delete/{id?}")]

public async Task<IActionResult> Delete(int? id)

{

if (!GlobalInfo.IsLogged)

{

return View(GlobalInfo.LoginView);

}

if (GlobalInfo.UsuarioLogeado.userType != "administrador")

{

// Redirige o muestra un mensaje de error porque el usuario no es un administrador

TempData["MensajeAccesoDenegado"] = "No tienes permisos para acceder a esta página.";

return RedirectToAction("AccesoDenegado");

}

await \_productoRepositorio.DeleteProducto(id);

return RedirectToAction("Producto");

}

[Route("/Amazon/ProductoUsuario")]

public IActionResult AccesoDenegado()

{

var mensaje = TempData["MensajeAccesoDenegado"] as string;

ViewBag.MensajeAccesoDenegado = mensaje;

return View();

Vista