

Universidad ORT Uruguay  
Facultad de Ingeniería  
Escuela de Tecnología

## **OBLIGATORIO PROGRAMACIÓN 3**



Carolina Larroque – 254338

Grupo N3G

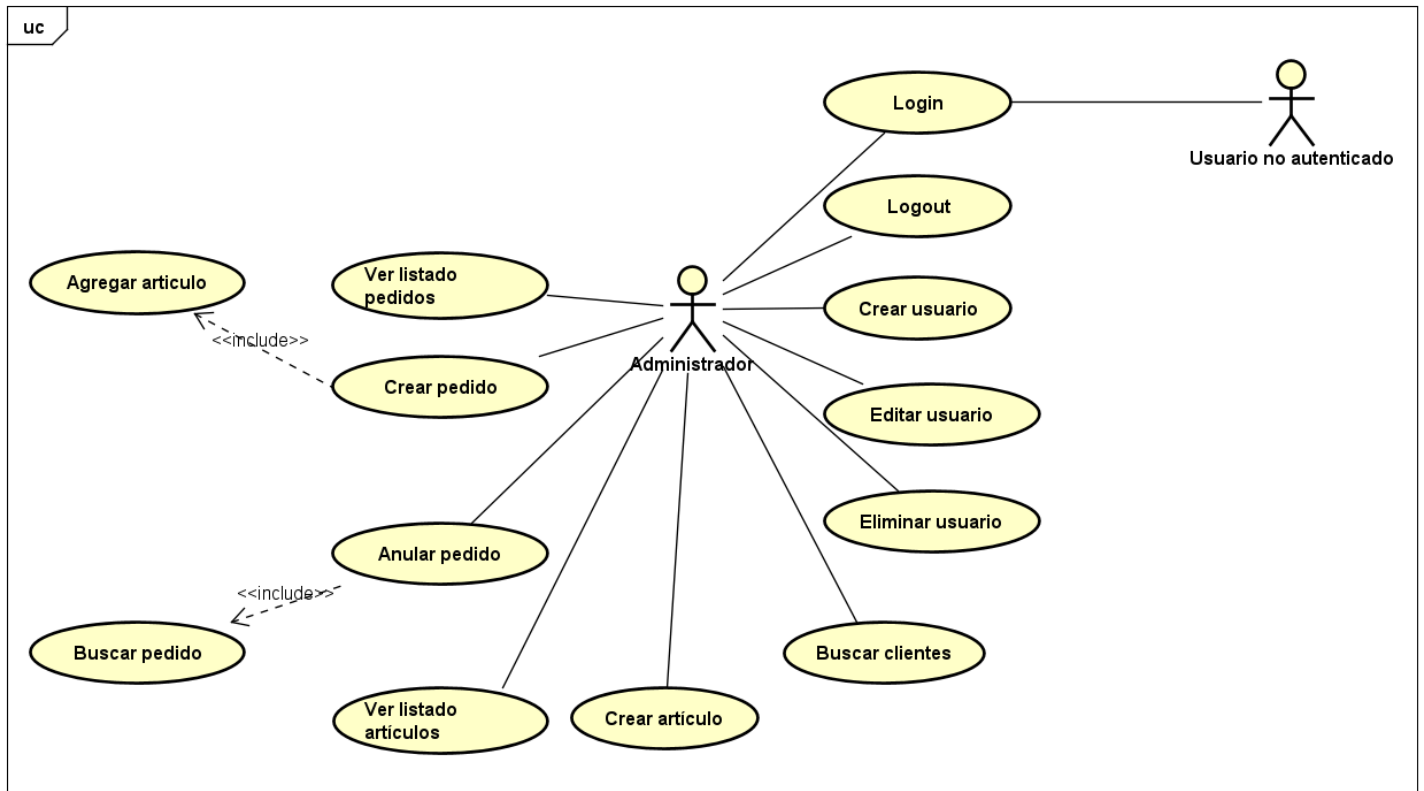
Analista en Tecnologías de la Información

16-05-2024

# Índice

<b>Diagrama de casos de uso</b>	<b>3</b>
<b>Descripción narrativa</b>	<b>4</b>
Alta de Pedido	4
Anulación de Pedido	5
<b>Diagrama de clases</b>	<b>6</b>
<b>Encriptado de Password</b>	<b>7</b>
<b>Web Api</b>	<b>11</b>
Status Codes	11
Artículos	11
Pedidos	12
<b>Código</b>	<b>13</b>

# Diagrama de casos de uso



# Descripción narrativa

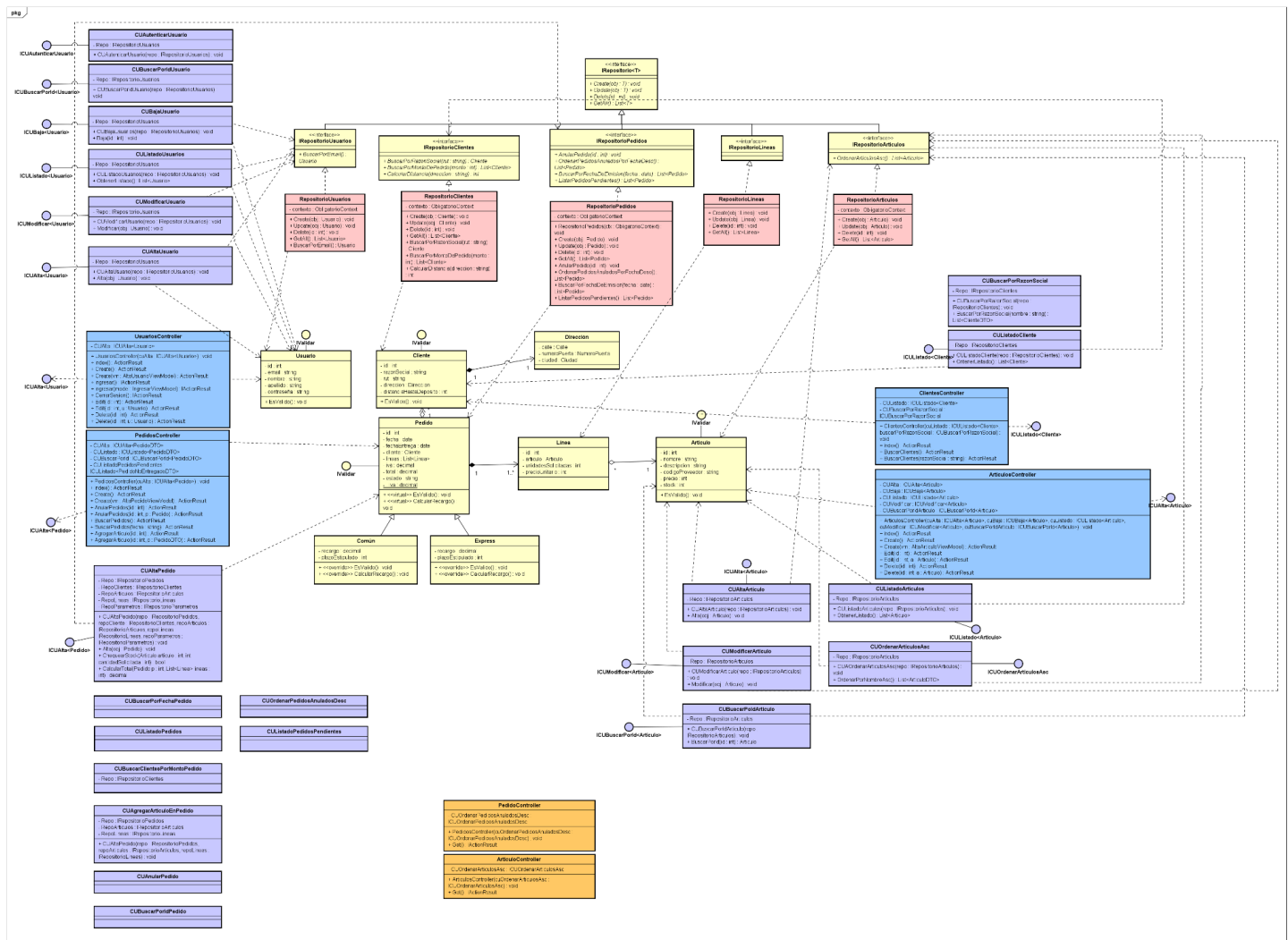
## Alta de Pedido

<b>Identificador:</b> CU-01	<b>Nombre:</b> Alta de pedido
<b>Autor:</b>	Carolina Larroque
<b>Fecha:</b>	11/05/24
<b>Descripción:</b> Permite crear un pedido	
<b>Actor:</b> Usuario autenticado con rol Administrador	
<b>Precondiciones:</b> El usuario debe estar autorizado en el sistema	
<b>Flujo normal:</b> <ol style="list-style-type: none"><li>1. El actor solicita crear un nuevo pedido</li><li>2. El sistema solicita el tipo de pedido, la razon social del cliente, la fecha de entrega, el nombre y la cantidad del artículo</li><li>3. El actor ingresa el tipo de pedido, la razon social del cliente, la fecha de entrega, el nombre, la cantidad del artículo y confirma la creación del pedido.</li><li>4. El sistema comprueba que el cliente seleccionado exista en la base de datos, que al menos un artículo haya sido seleccionado, que el artículo seleccionado exista en la base de datos, que haya stock suficiente para la cantidad seleccionada y que la fecha de entrega corresponda con el tipo de pedido.</li></ol>	
<b>Flujos alternativos:</b> <ol style="list-style-type: none"><li>1. El actor cancela el alta del pedido<ol style="list-style-type: none"><li>a. El pedido no es guardado en la base</li></ol></li><li>2. El actor no selecciona un artículo<ol style="list-style-type: none"><li>a. Se avisa que se requiere al menos un artículo para el alta del pedido</li></ol></li><li>3. El cliente seleccionado no existe en la base de datos<ol style="list-style-type: none"><li>a. Se avisa que el cliente no existe</li></ol></li><li>4. El artículo seleccionado no existe en la base de datos<ol style="list-style-type: none"><li>a. Se avisa que el artículo no existe</li></ol></li><li>5. No existe suficiente stock para la cantidad seleccionada<ol style="list-style-type: none"><li>a. Se avisa que no hay suficiente stock</li></ol></li></ol>	
<b>Flujos excepcionales:</b> <p>El servidor se cae al enviar el alta de pedido</p> <ol style="list-style-type: none"><li>a. Se avisa que la transacción no ha sido exitosa</li></ol>	
<b>Pos condiciones:</b> <p>El pedido y sus líneas son ingresados en el sistema</p>	

## Anulación de Pedido

<b>Identificador:</b> CU-02	<b>Nombre:</b> Anulación de pedido
<b>Autor:</b>	Carolina Larroque
<b>Fecha:</b>	11/05/24
<b>Descripción:</b> Permite anular un pedido	
<b>Actor:</b> Usuario autenticado con rol Administrador	
<b>Precondiciones:</b> El usuario debe estar autorizado en el sistema	
<b>Flujo normal:</b> <ol style="list-style-type: none"><li>1. El actor solicita busca un pedido por su fecha de emisión</li><li>2. El sistema devuelve los pedidos emitidos en la fecha seleccionada</li><li>3. El actor solicita anular el pedido</li><li>4. El sistema devuelve los detalles del pedido a anular</li><li>5. El actor confirma la anulación del pedido</li><li>6. El sistema actualiza en estado del pedido</li></ol>	
<b>Flujos alternativos:</b> <ol style="list-style-type: none"><li>1. El sistema no encuentra pedidos para la fecha solicitada<ol style="list-style-type: none"><li>a. Se avisa que no existen pedidos</li></ol></li></ol>	
<b>Flujos excepcionales:</b> <p>El servidor se cae al enviar la anulación del pedido</p> <ol style="list-style-type: none"><li>b. Se avisa que la transacción no ha sido exitosa</li></ol>	
<b>Pos condiciones:</b> <p>El pedido es actualizado con su nuevo estado</p>	

## Diagrama de clases



# Encriptado de Password

## Input:

You are a C# developer building an MVC app in .Net8 using Entity Framework and a Repository design pattern. You need to create a User and save their password encrypted in the database. At the MVC level, you have a UserController and a Razor View using a "Create" template, where you ask the User to put their name, email and password. How would you encrypt the password and save it in the database?

## Output:

To encrypt the password and save it in the database in an MVC app using Entity Framework and a Repository design pattern, you can follow these steps:

### 1. \*\*Encrypt the Password:\*\*

- First, choose a strong encryption algorithm like AES (Advanced Encryption Standard) or bcrypt for hashing.
- In your C# code, use the chosen encryption algorithm to encrypt the password before saving it to the database. For example, if you're using AES, you might use a library like `System.Security.Cryptography` for encryption.

### 2. \*\*Repository Design Pattern:\*\*

- Implement a UserRepository that handles CRUD operations for User entities.

### 3. \*\*UserController:\*\*

- In the UserController's Create action, receive the user input including the password.
- Encrypt the password using the chosen encryption algorithm.
- Create a new User entity and set its properties, including the encrypted password.
- Use the UserRepository to save the user to the database.

Here's an example of how you might implement these steps:

```
```csharp
// Encryption Utility (using AES as an example)
public class EncryptionUtility
{
    private static readonly byte[] Key = Encoding.UTF8.GetBytes("your_secret_key_here");
    private static readonly byte[] IV =
        Encoding.UTF8.GetBytes("your_initialization_vector_here");

    public static string EncryptString(string plainText)
    {

```

```

        using var aes = Aes.Create();
        aes.Key = Key;
        aes.IV = IV;

        var encryptor = aes.CreateEncryptor(aes.Key, aes.IV);
        using var msEncrypt = new MemoryStream();
        using (var csEncrypt = new CryptoStream(msEncrypt, encryptor,
CryptoStreamMode.Write))
        {
            using (var swEncrypt = new StreamWriter(csEncrypt))
            {
                swEncrypt.Write(plainText);
            }
        }
        return Convert.ToBase64String(msEncrypt.ToArray());
    }
}

```

```

// User Model
public class User
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    public string EncryptedPassword { get; set; }
}

```

```

// UserRepository
public class UserRepository
{
    private readonly YourDbContext _context;

    public UserRepository(YourDbContext context)
    {
        _context = context;
    }

    public void AddUser(User user)
    {
        _context.Users.Add(user);
        _context.SaveChanges();
    }
}

```



```
// UserController
public class UserController : Controller
{
    private readonly UserRepository _userRepository;

    public UserController(UserRepository userRepository)
    {
        _userRepository = userRepository;
    }

    [HttpPost]
    public IActionResult Create(UserViewModel model)
    {
        if (ModelState.IsValid)
        {
            string encryptedPassword = EncryptionUtility.EncryptString(model.Password);

            var user = new User
            {
                Name = model.Name,
                Email = model.Email,
                EncryptedPassword = encryptedPassword
            };

            _userRepository.AddUser(user);
            return RedirectToAction("Index", "Home");
        }

        return View(model);
    }
}
...
```

In this example:

- `EncryptionUtility` provides a method `EncryptString` to encrypt the password using AES.
- `User` is the model class representing a user with an encrypted password field.
- `UserRepository` handles database operations for users.
- `UserController` contains the logic for creating a user, encrypting the password, and saving it using the repository.

**Input:**

how would you generate the Key "your\_secret\_key\_here" and IV  
"your\_initialization\_vector\_here"?

**Output:**

To generate a key and initialization vector (IV) for AES encryption, you can use a secure method to create random bytes for both the key and IV. Here's an updated version of the EncryptionUtility class with methods to generate a random key and IV:

```
```csharp
using System;
using System.IO;
using System.Security.Cryptography;
using System.Text;

public class EncryptionUtility
{
    public static byte[] GenerateRandomKey()
    {
        using var aes = Aes.Create();
        aes.GenerateKey();
        return aes.Key;
    }

    public static byte[] GenerateRandomIV()
    {
        using var aes = Aes.Create();
        aes.GenerateIV();
        return aes.IV;
    }

    public static string EncryptString(string plainText, byte[] key, byte[] iv)
    {
        using var aes = Aes.Create();
        aes.Key = key;
        aes.IV = iv;

        var encryptor = aes.CreateEncryptor(aes.Key, aes.IV);
        using var msEncrypt = new MemoryStream();
        using (var csEncrypt = new CryptoStream(msEncrypt, encryptor,
CryptoStreamMode.Write))
        {
            using (var swEncrypt = new StreamWriter(csEncrypt))
```

```

        {
            swEncrypt.Write(plainText);
        }
    }
    return Convert.ToBase64String(msEncrypt.ToArray());
}
}
...

```

You can then use these methods to generate a key and IV, and encrypt the password:

```

```csharp
// Generate random key and IV
byte[] key = EncryptionUtility.GenerateRandomKey();
byte[] iv = EncryptionUtility.GenerateRandomIV();

// Use the generated key and IV for encryption
string encryptedPassword = EncryptionUtility.EncryptString(model.Password, key, iv);
```

```

This approach ensures that each encryption operation uses a unique key and IV, enhancing the security of your encryption process.

## Web Api

### Status Codes

#### Artículos

| Código | Mensaje                         |
|--------|---------------------------------|
| 200    | Ok                              |
| 404    | No existen artículos            |
| 500    | Ocurrió un error en el servidor |

Pedidos

| Código | Mensaje                         |
|--------|---------------------------------|
| 200    | Ok                              |
| 404    | No existen pedidos              |
| 500    | Ocurrió un error en el servidor |

# Código

Código

```
using LogicaNegocio.Excepciones;
using LogicaNegocio.InterfacesDominio;
using System.ComponentModel.DataAnnotations;
using Microsoft.EntityFrameworkCore;
using System.ComponentModel.DataAnnotations.Schema;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;

namespace LogicaNegocio.Dominio {
    [Index(nameof(Nombre), IsUnique = true)]
    [Index(nameof(CodigoProveedor), IsUnique = true)]
    public class Articulo : IValidar {
        public int Id { get; set; }
        [Required(ErrorMessage = "El nombre es requerido")]
        [StringLength(200, MinimumLength = 10, ErrorMessage = "El nombre debe tener entre 10 y 200
caracteres")]
        public string Nombre { get; set; } // no vacio y unico. Min 10 char max 200 char
        [MinLength(5, ErrorMessage = "La descripción debe tener un mínimo de 5 caracteres")]
        public string Descripcion { get; set; } // largo minimo de 5 caracteres
        [Display(Name = "Código de Proveedor")]
        [Required(ErrorMessage = "El código de proveedor es requerido")]
        [StringLength(13, MinimumLength = 13, ErrorMessage = "El código de proveedor debe tener 13 dígitos")]
        public string CodigoProveedor { get; set; } // 13 digitos, no vacio
        public int Precio { get; set; }
        public int Stock { get; set; }

        public void EsValido() {
            if (Nombre.Length < 10 || Nombre.Length > 200 || string.IsNullOrEmpty(Nombre)) {
                throw new DatosInvalidosException("El nombre debe tener entre 10 y 200 caracteres");
            }

            if (CodigoProveedor.Length != 13 || string.IsNullOrEmpty(CodigoProveedor)) {
                throw new DatosInvalidosException("El código de proveedor es requerido y debe tener 13
dígitos.");
            }

            if (Descripcion.Length < 5) {
                throw new DatosInvalidosException("La descripción debe tener un mínimo de 5 caracteres.");
            }

            if (Precio < 0 || Stock < 0) {
                throw new DatosInvalidosException("Precio y Stock no pueden ser negativos.");
            }
        }
    }
}
```

```

    }
}
using LogicaNegocio.Excepciones;
using LogicaNegocio.InterfacesDominio;
using LogicaNegocio.ValueObjects;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;

namespace LogicaNegocio.Dominio {
    [Index(nameof(Rut), IsUnique = true)]
    public class Cliente : IValidar {
        [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }
        [Display(Name = "Razon Social")]
        public string? RazonSocial { get; set; }
        [StringLength(12, MinimumLength = 12)]
        public string? Rut { get; set; } // 12 digitos
        public Direccion Direccion { get; set; } // calle, numero, ciudad
        public int? DistanciaHastaDeposito { get; set; }
        public void EsValido() {
            if (Rut.Length != 12) {
                throw new DatosInvalidosException("El RUT debe tener 12 dígitos.");
            }
        }
    }
}

using LogicaNegocio.InterfacesDominio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.Dominio {
    public class Linea {
        public int Id { get; set; }
        public Articulo Articulo { get; set; }
        public int UnidadesSolicitadas { get; set; }
        public int PreciodUnitario { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace LogicaNegocio.Dominio {
    public class Parametro {
        public int Id { get; set; }
        public string Nombre { get; set; }
        public decimal Valor { get; set; }
    }
}

using LogicaNegocio.InterfacesDominio;
using LogicaNegocio.Excepciones;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.Dominio {
    public abstract class Pedido : IValidar {
        private DateOnly _fecha = DateOnly.FromDateTime(DateTime.Now);
        public static decimal _iva;

        public int Id { get; set; }
        public DateOnly Fecha { get { return _fecha; } set { _fecha = value; } }
        public DateOnly FechaEntrega { get; set; }
        public Cliente Cliente { get; set; }
        public List<Linea> Lineas { get; set; }
        public decimal Total { get; set; }
        public decimal Iva { get { return _iva; } set { _iva = value; } }
        public decimal Recargo { get; set; }
        public string Estado { get; set; } // Pendiente, Entregado, Anulado

        public virtual void EsValido() {
            if (Lineas.Count == 0) {
                throw new DatosInvalidosException("El pedido debe contener al menos un artículo");
            }
        }

        public abstract void CalcularRecargo(decimal a, decimal b);
    }
}

using LogicaNegocio.InterfacesDominio;
using LogicaNegocio.Excepciones;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.Dominio {
    public class PedidoComun : Pedido, IValidar {
        private int _plazoEstipulado;

        // > 7 dias
        public int PlazoEstipulado {

```

```

        get { return _plazoEstipulado = FechaEntrega.DayNumber - Fecha.DayNumber; }
        set { _plazoEstipulado = value; }
    }

    public override void EsValido() {
        if (PlazoEstipulado < 7) {
            throw new DatosInvalidosException("El plazo de entrega para un pedido común no puede ser
menor a 7 días");
        }
    }

    // si Cliente.DistanciaHastaDeposito > 100 ? 5% : 0
    public override void CalcularRecargo(decimal a, decimal b) {
        Recargo = Cliente != null && Cliente.DistanciaHastaDeposito > 100 ? a : b;
    }
}

}
using LogicaNegocio.InterfacesDominio;
using LogicaNegocio.Excepciones;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.Dominio {
    public class PedidoExpress : Pedido, IValidar {
        private int _plazoEstipulado;

        // <= 5 días
        public int PlazoEstipulado {
            get { return _plazoEstipulado = FechaEntrega.DayNumber - Fecha.DayNumber; }
            set { _plazoEstipulado = value; }
        }

        public override void EsValido() {
            if (PlazoEstipulado > 5) {
                throw new DatosInvalidosException("El plazo de entrega para un pedido express no puede ser
mayor a 5 días");
            }
        }

        // si plazo = 1 ? 15% : 10%
        public override void CalcularRecargo(decimal a, decimal b) {
            Recargo = PlazoEstipulado == 1 ? a : b;
        }
    }
}

using LogicaNegocio.InterfacesDominio;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System;
using System.Collections.Generic;

```



```

using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Text.RegularExpressions;
using Microsoft.EntityFrameworkCore;
using LogicaNegocio.Excepciones;

namespace LogicaNegocio.Dominio {
    [Index(nameof(Email), IsUnique = true)]
    public class Usuario : IValidar {
        [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }
        [EmailAddress]
        public string Email { get; set; } // unico, aplicar formateado
        [RegularExpression(@"^[A-Za-z][A-Za-z' -]*[A-Za-z]$")]
        public string Nombre { get; set; }
        [RegularExpression(@"^[A-Za-z][A-Za-z' -]*[A-Za-z]$")]
        public string Apellido { get; set; }
        [MinLength(6)]
        [RegularExpression(@"^(?=.*[A-Z])(?=.*[a-z])(?=.*[.!\;]).{6,}$")]
        public string Contraseña { get; set; } // min 6 char, mayuscula, minuscula, char especial (punto,
punto y coma, coma, signo de admiración de cierre)
        public string? ContraseñaEncriptada { get; set; }
        [Column(TypeName = "nvarchar(24)")]
        public TipoUsuario Tipo { get; set; }

        public void EsValido() {
            if(!ValidarContraseña(Contraseña)) {
                throw new DatosInvalidosException("La contraseña debe tener un mínimo de 6 caracteres, al
menos una mayuscula y una minuscula y un caracter especial (punto, coma, punto y coma o signo de
admiración)");
            }
            //TODO: VO para nombre y apellido?
        }

        private bool ValidarContraseña(String contraseña) {
            var tieneNumero = new Regex(@"[0-9]+");
            var tieneMayuscula = new Regex(@"[A-Z]+");
            var tieneMinuscula = new Regex(@"[a-z]+");
            var tieneMinSeisChar = new Regex(@".{6,}");
            var tieneSimbolos = new Regex(@"[!;,.\?]+");

            return tieneNumero.IsMatch(contraseña) && tieneMayuscula.IsMatch(contraseña) &&
tieneMinuscula.IsMatch(contraseña) && tieneMinSeisChar.IsMatch(contraseña) &&
tieneSimbolos.IsMatch(contraseña);
        }
    }

    public enum TipoUsuario {
        Administrador,
        Estandar
    }
}

```

```

//
https://learn.microsoft.com/en-us/ef/core/modeling/value-conversions?tabs=data-annotations#pre-defined-conver
sions
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.Excepciones {
    public class DatosInvalidosException : Exception {
        public DatosInvalidosException() {

        }

        public DatosInvalidosException(string mensaje) : base(mensaje) {

        }

        public DatosInvalidosException(string mensaje, Exception e) : base(mensaje, e) {

        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.Excepciones {
    public class DuplicadoException : Exception {
        public DuplicadoException() {

        }

        public DuplicadoException(string mensaje) : base(mensaje) {

        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.Excepciones {
    public class NoStockException : Exception {
        public NoStockException() {

        }
    }
}

```

```

        public NoStockException(string mensaje) : base(mensaje) {

        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.Excepciones {
    public class RegistroNoExisteException : Exception {
        public RegistroNoExisteException() {

        }

        public RegistroNoExisteException(string mensaje) : base(mensaje) {

        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.InterfacesDominio {
    public interface IValidar {
        void EsValido();
    }
}
using LogicaNegocio.Dominio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.InterfacesRepositorios {
    public interface IRepositoryo<T> {
        void Create(T obj);
        void Delete(int id);
        void Update(T obj);
        List<T> GetAll();
        T FindById(int id);
    }
}
using LogicaNegocio.Dominio;
using System;
using System.Collections.Generic;

```

```

using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.InterfacesRepositorios {
    public interface IRepositoryioArticulos : IRepositoryio<Articulo> {
        List<Articulo> OrdenarArticulosAsc();
    }
}
using LogicaNegocio.Dominio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.InterfacesRepositorios {
    public interface IRepositoryioClientes : IRepositoryio<Cliente> {
        List<Cliente> BuscarPorRazonSocial(string nombre);
    }
}
using LogicaNegocio.Dominio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.InterfacesRepositorios {
    public interface IRepositoryioLineas : IRepositoryio<Linea>{
    }
}
using LogicaNegocio.Dominio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.InterfacesRepositorios {
    public interface IRepositoryioParametros {
        public decimal ObtenerIva();
    }
}
using LogicaNegocio.Dominio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.InterfacesRepositorios {
    public interface IRepositoryioPedidos : IRepositoryio<Pedido>{

```

```

        void AnularPedido(int id);
        List<Pedido> BuscarPorFechaDeEmision(DateOnly fecha);
        List<Pedido> OrdenarPedidosAnuladosPorFechaDesc();
        List<Pedido> ListarPedidosPendientes();
        List<Cliente> BuscarClientes(decimal monto);
    }
}
using LogicaNegocio.Dominio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.InterfacesRepositorios {
    public interface IRepositoryioUsuarios : IRepositoryio<Usuario> {
        Usuario BuscarPorEmail(string email);
    }
}
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.ValueObjects {
    ///[Owned]
    public class Calle {
        public int DireccionId { get; set; }
        public string Valor { get; init; }
        private Calle() {

        }

        public Calle(string valor, int direccionId) {
            Valor = valor;
            Validar();
            DireccionId = direccionId;
        }

        private void Validar() {
            if (string.IsNullOrEmpty(Valor))
                throw new Exception("El nombre de la calle es obligatorio");
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace LogicaNegocio.ValueObjects {
    public class Ciudad {
        public int DireccionId { get; set; }
        public string Valor { get; init; }
        private Ciudad() {

        }

        public Ciudad(string valor, int direccionId) {
            Valor = valor;
            Validar();
            DireccionId = direccionId;
        }

        private void Validar() {
            if (string.IsNullOrEmpty(Valor))
                throw new Exception("El nombre de la ciudad es obligatorio");
        }
    }
}

using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.ValueObjects {
    [Owned]
    public class Direccion {
        public int ClienteId { get; set; }
        public Calle Calle { get; set; }
        public NumeroPuerta NumeroPuerta { get; set; }
        public Ciudad Ciudad { get; set; }

        private Direccion() {

        }

        public Direccion(int clienteId, Calle calle, NumeroPuerta nroPuerta, Ciudad ciudad) {
            Calle = calle;
            NumeroPuerta = nroPuerta;
            Ciudad = ciudad;
            ClienteId = clienteId;
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;

```

```

using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.ValueObjects {
    public class NumeroPuerta {
        public int DireccionId { get; set; }
        public string Valor { get; init; }
        private NumeroPuerta() {

        }

        public NumeroPuerta(string valor, int direccionId) {
            Valor = valor;
            Validar();
            DireccionId = direccionId;
        }

        private void Validar() {
            if (string.IsNullOrEmpty(Valor))
                throw new Exception("El numero de puerta es obligatorio");
        }
    }
}

using LogicaNegocio.Dominio;
using LogicaNegocio.ValueObjects;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static System.Runtime.InteropServices.JavaScript.JSType;

namespace LogicaDatos.Repositorios {
    public class ObligatorioContext : DbContext {
        public DbSet<Usuario> Usuarios { get; set; }
        public DbSet<Cliente> Clientes { get; set; }
        public DbSet<Articulo> Articulos { get; set; }
        public DbSet<Pedido> Pedidos { get; set; }
        public DbSet<PedidoExpress> PedidoExpress { get; set; }
        public DbSet<PedidoComun> PedidoComun { get; set; }
        public DbSet<Linea> Lineas { get; set; }
        public DbSet<Parametro> Parametros { get; set; }

        public ObligatorioContext(DbContextOptions options) : base(options) {

        }

        protected override void OnModelCreating(ModelBuilder modelBuilder) {
            base.OnModelCreating(modelBuilder);
            modelBuilder.Entity<Usuario>().ToTable("Usuarios");
            modelBuilder.Entity<Articulo>().ToTable("Articulos");
        }
    }
}

```

```

modelBuilder.Entity<Cliente>().ToTable("Clientes");
modelBuilder.Entity<Pedido>().ToTable("Pedidos");
modelBuilder.Entity<Linea>().ToTable("Lineas");
modelBuilder.Entity<Parametro>().ToTable("Parametros");

//-----
//----- TEST DATA -----
//-----

modelBuilder.Entity<Cliente>(builder => {
builder.OwnsOne(d => d.Direccion, direccion => {
    direccion.OwnsOne(ca => ca.Calle, calle => {
        calle.HasData(
            new { Valor = "Ciudadela", DireccionId = 1 },
            new { Valor = "Reconquista", DireccionId = 2 },
            new { Valor = "Durazno", DireccionId = 3 },
            new { Valor = "San Jose", DireccionId = 4 },
            new { Valor = "Maldonado", DireccionId = 5 },
            new { Valor = "Canelones", DireccionId = 6 }
        );
    });
    direccion.OwnsOne(n => n.NumeroPuerta, numero => {
        numero.HasData(
            new { Valor = "1180", DireccionId = 1 },
            new { Valor = "600", DireccionId = 2 },
            new { Valor = "902", DireccionId = 3 },
            new { Valor = "1060", DireccionId = 4 },
            new { Valor = "2010", DireccionId = 5 },
            new { Valor = "1110", DireccionId = 6 }
        );
    });
    direccion.OwnsOne(ci => ci.Ciudad, ciudad => {
        ciudad.HasData(
            new { Valor = "Montevideo", DireccionId = 1 },
            new { Valor = "Montevideo", DireccionId = 2 },
            new { Valor = "Montevideo", DireccionId = 3 },
            new { Valor = "Montevideo", DireccionId = 4 },
            new { Valor = "Montevideo", DireccionId = 5 },
            new { Valor = "Montevideo", DireccionId = 6 }
        );
    });
    direccion.HasData(
        new { ClienteId = 1 },
        new { ClienteId = 2 },
        new { ClienteId = 3 },
        new { ClienteId = 4 },
        new { ClienteId = 5 },
        new { ClienteId = 6 }
    );
});
builder.HasData(
    new Cliente { Id = 1, RazonSocial = "Fleetium Macs", Rut = "123456789012",
DistanciaHastaDeposito = 98 },

```



```

        new Cliente { Id = 2, RazonSocial = "Pink Folks", Rut = "987654321098",
DistanciaHastaDeposito = 101 },
        new Cliente { Id = 3, RazonSocial = "Thurston & Kim Co.", Rut = "111222333444",
DistanciaHastaDeposito = 96 },
        new Cliente { Id = 4, RazonSocial = "Nueva Helvecia Dolls", Rut = "111222333555",
DistanciaHastaDeposito = 94 },
        new Cliente { Id = 5, RazonSocial = "Fenix", Rut = "111222333666", DistanciaHastaDeposito
= 82 },
        new Cliente { Id = 6, RazonSocial = "AB/CD", Rut = "111222333777", DistanciaHastaDeposito
= 91 }
    );
});

modelBuilder.Entity<Usuario>().HasData(
    new Usuario { Id = 1, Email = "admin@admin.com", Nombre = "Sys", Apellido = "Admin",
Contraseña = "Passw0rd!", Tipo = TipoUsuario.Administrador },
    new Usuario { Id = 2, Email = "rplanta@lz.com", Nombre = "Roberto", Apellido = "Planta",
Contraseña = "Passw0rd!", Tipo = TipoUsuario.Estandar }
);

modelBuilder.Entity<Articulo>().HasData(
    new Articulo { Id = 1, Nombre = "Calculadora Casio", Descripcion = "Calculadora de bolsillo
Casio", CodigoProveedor = "1234567890123", Precio = 300, Stock = 10 },
    new Articulo { Id = 2, Nombre = "Resma A4", Descripcion = "500 hojas para imprimir en formato
A4", CodigoProveedor = "1234567890124", Precio = 250, Stock = 50 },
    new Articulo { Id = 3, Nombre = "Auriculares con Micrófono", Descripcion = "Bluetooth o
conexión USB, cable desmontable", CodigoProveedor = "1234567890125", Precio = 500, Stock = 40 },
    new Articulo { Id = 4, Nombre = "Silla De Escritorio", Descripcion = "Soporte lumbar, altura
ajustable y base giratoria con ruedas", CodigoProveedor = "1234567890126", Precio = 3200, Stock = 20 },
    new Articulo { Id = 5, Nombre = "Notas Adhesivas", Descripcion = "Block de notas
autoadhesivas", CodigoProveedor = "1234597890124", Precio = 50, Stock = 100 },
    new Articulo { Id = 6, Nombre = "Bandas Elásticas", Descripcion = "100 unidades",
CodigoProveedor = "1274597890124", Precio = 150, Stock = 90 },
    new Articulo { Id = 7, Nombre = "Engrampadora", Descripcion = "Hasta 100 hojas",
CodigoProveedor = "1834597890124", Precio = 800, Stock = 10 },
    new Articulo { Id = 8, Nombre = "Cable HDMI", Descripcion = "HD, FullHD y 4K",
CodigoProveedor = "1236597890124", Precio = 200, Stock = 10 }
);

modelBuilder.Entity<Parametro>().HasData(
    new Parametro { Id = 1, Nombre = "IVA", Valor = 0.18M },
    new Parametro { Id = 2, Nombre = "RecargoComun_DistanciaMayor100", Valor = 0.05M },
    new Parametro { Id = 3, Nombre = "RecargoComun_DistanciaMenor100", Valor = 0 },
    new Parametro { Id = 4, Nombre = "RecargoExpress_Plazo1Dia", Valor = 0.15M },
    new Parametro { Id = 5, Nombre = "RecargoExpress_PlazoMayor1Dia", Valor = 0.10M }
);
}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) {
    base.OnConfiguring(optionsBuilder);
    //optionsBuilder.UseSqlServer(

```

```

        // "Data Source=localhost\\SQLEXPRESS; Initial Catalog=ObligatorioProg3; Integrated
Security=SSPI;TrustServerCertificate=True"
        //);
    }
}
// Owned types https://learn.microsoft.com/en-us/ef/core/modeling/owned-entities#configuring-owned-types
// Data Seeding https://learn.microsoft.com/en-us/ef/core/modeling/data-seeding
// Seeding Owned https://github.com/dotnet/efcore/issues/32746
}
using LogicaNegocio.Dominio;
using LogicaNegocio.Excepciones;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaDatos.Repositorios {
    public class RepositorioArticulos : IRepositoryArticulos {
        public ObligatorioContext Contexto { get; set; }

        public RepositorioArticulos(ObligatorioContext ctx) {
            Contexto = ctx;
        }
        public void Create(Articulo obj) {
            obj.EsValido();
            if (!Contexto.Articulos.Any(a => a.Nombre == obj.Nombre) && !Contexto.Articulos.Any(a =>
a.CodigoProveedor == obj.CodigoProveedor)) {
                Contexto.Articulos.Add(obj);
                Contexto.SaveChanges();
            } else {
                throw new DuplicadoException("Ya existe un artículo con ese nombre y código de proveedor.");
            }
        }

        public void Delete(int id) {
            Articulo aBorrar = FindById(id);
            if (aBorrar != null) {
                Contexto.Articulos.Remove(aBorrar);
                Contexto.SaveChanges();
            } else {
                throw new RegistroNoExisteException("El artículo no existe");
            }
        }

        public Articulo FindById(int id) {
            return Contexto.Articulos.Find(id);
        }

        public List<Articulo> GetAll() {
            return Contexto.Articulos.ToList();
        }
    }
}

```

```

    }

    public void Update(Articulo obj) {
        obj.EsValido();
        if (!Contexto.Articulos.Any(a => a.Nombre == obj.Nombre) && !Contexto.Articulos.Any(a =>
a.CodigoProveedor == obj.CodigoProveedor)) {
            Contexto.Articulos.Update(obj);
            Contexto.SaveChanges();
        } else {
            throw new DuplicadoException("Ya existe un artículo con ese nombre y código de proveedor.");
        }
    }

    // Listado con todos los artículos ordenados alfabéticamente en forma ascendente
    public List<Articulo> OrdenarArticulosAsc() {
        return Contexto.Articulos.OrderBy(a => a.Nombre).ToList();
    }
}

using LogicaNegocio.Dominio;
using LogicaNegocio.Excepciones;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaDatos.Repositorios {
    public class RepositorioClientes : IRepositoryClientes {
        public ObligatorioContext Contexto { get; set; }
        public RepositorioClientes(ObligatorioContext ctx) {
            Contexto = ctx;
        }
        public List<Cliente> BuscarPorRazonSocial(string nombre) {
            return Contexto.Clientes.Where(c => c.RazonSocial.Contains(nombre)).ToList();
        }

        public void Create(Cliente obj) {
            obj.EsValido();
            Contexto.Clientes.Add(obj);
            Contexto.SaveChanges();
        }

        public void Delete(int id) {
            Cliente aBorrar = FindById(id);
            if (aBorrar != null) {
                Contexto.Clientes.Remove(aBorrar);
                Contexto.SaveChanges();
            } else {
                throw new RegistroNoExisteException("El cliente no existe");
            }
        }
    }
}

```

```

        public Cliente FindById(int id) {
            return Contexto.Clientes.Find(id);
        }

        public List<Cliente> GetAll() {
            return Contexto.Clientes.ToList();
        }

        public void Update(Cliente obj) {
            obj.EsValido();
            Contexto.Clientes.Update(obj);
            Contexto.SaveChanges();
        }
    }
}

using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaDatos.Repositorios {
    public class RepositorioLineas : IRepositorioLineas {
        public ObligatorioContext Contexto { get; set; }

        public RepositorioLineas(ObligatorioContext ctx) {
            Contexto = ctx;
        }

        public void Create(Linea obj) {
            //obj.EsValido();
            Contexto.Lineas.Add(obj);
            Contexto.SaveChanges();
        }

        public void Delete(int id) {
            throw new NotImplementedException();
        }

        public Linea FindById(int id) {
            throw new NotImplementedException();
        }

        public List<Linea> GetAll() {
            throw new NotImplementedException();
        }

        public void Update(Linea obj) {
            throw new NotImplementedException();
        }
    }
}

```

```

    }
}
using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaDatos.Repositorios {
    public class RepositorioParametros : IRepositorioParametros {
        public ObligatorioContext Contexto { get; set; }
        public RepositorioParametros(ObligatorioContext ctx) {
            Contexto = ctx;
        }

        public decimal ObtenerIva() {
            return Contexto.Parametros.Where(p => p.Nombre == "IVA").OrderByDescending(p =>
p.Id).First().Valor;
        }
    }
}
using LogicaNegocio.Dominio;
using LogicaNegocio.Excepciones;
using LogicaNegocio.InterfacesRepositorios;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static System.Runtime.InteropServices.JavaScript.JSType;

namespace LogicaDatos.Repositorios {
    public class RepositorioPedidos : IRepositorioPedidos {
        public ObligatorioContext Contexto { get; set; }

        public RepositorioPedidos(ObligatorioContext ctx) {
            Contexto = ctx;
        }
        public void Create(Pedido obj) {
            obj.EsValido();
            Contexto.Entry(obj.Cliente).State = EntityState.Unchanged;
            Contexto.Pedidos.Add(obj);
            Contexto.SaveChanges();
        }

        public void Delete(int id) {
            //throw new NotImplementedException();
        }
    }
}

```

```

    public Pedido FindById(int id) {
        return Contexto.Pedidos.Include(p => p.Lineas).FirstOrDefault(p => p.Id == id);
    }

    public List<Pedido> GetAll() {
        return Contexto.Pedidos.ToList();
    }

    public void Update(Pedido obj) {
        obj.EsValido();
        //Contexto.Entry(obj.Cliente).State = EntityState.Unchanged;
        Contexto.Pedidos.Update(obj);
        Contexto.SaveChanges();
    }

    public void AnularPedido(int id) {
        Pedido aAnular = FindById(id);
        if (aAnular != null) {
            aAnular.Estado = "Anulado";
            Contexto.Pedidos.Update(aAnular);
            Contexto.SaveChanges();
        } else {
            throw new RegistroNoExisteException("El pedido no existe");
        }
    }

    public List<Pedido> BuscarPorFechaDeEmision(DateOnly fecha) {
        return Contexto.Pedidos.Where(p => p.Fecha == fecha && p.Estado == "Pendiente").Include(p =>
p.Cliente).ToList();
    }

    public List<Pedido> OrdenarPedidosAnuladosPorFechaDesc() {
        return Contexto.Pedidos.Where(p => p.Estado == "Anulado").Include(p =>
p.Cliente).OrderByDescending(p => p.Fecha).ToList();
    }

    public List<Pedido> ListarPedidosPendientes() {
        return Contexto.Pedidos.Where(p => p.Estado == "Pendiente").Include(p => p.Cliente).ToList();
    }

    public List<Cliente> BuscarClientes(decimal monto) {
        return Contexto.Pedidos.Where(p => p.Total >= monto).Select(p => p.Cliente).Distinct().ToList();
    }
}

//
https://learn.microsoft.com/en-us/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/implementing-inheritance-with-the-entity-framework-in-an-asp-net-mvc-application
using LogicaNegocio.Dominio;
using LogicaNegocio.Excepciones;
using LogicaNegocio.InterfacesRepositorios;
using Microsoft.EntityFrameworkCore;
using System;

```

```

using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaDatos.Repositorios {
    public class RepositorioUsuarios : IRepositoryUsuarios {
        public ObligatorioContext Contexto { get; set; }
        public RepositorioUsuarios(ObligatorioContext ctx) {
            Contexto = ctx;
        }
        public void Create(Usuario obj) {
            obj.EsValido();
            if (!Contexto.Usuarios.Any(u => u.Email.ToLower() == obj.Email.ToLower())) {
                Contexto.Usuarios.Add(obj);
                Contexto.SaveChanges();
            } else {
                throw new DuplicadoException("Ya existe un usuario registrado con ese email.");
            }
        }

        public void Delete(int id) {
            Usuario aBorrar = FindById(id);
            if (aBorrar != null) {
                Contexto.Usuarios.Remove(aBorrar);
                Contexto.SaveChanges();
            } else {
                throw new RegistroNoExisteException("El usuario no existe");
            }
        }

        public void Update(Usuario obj) {
            obj.EsValido();
            if (!Contexto.Usuarios.Any(u => u.Email.ToLower() == obj.Email.ToLower())) {
                Contexto.Usuarios.Update(obj);
                Contexto.SaveChanges();
            } else {
                throw new DuplicadoException("Ya existe un usuario registrado con ese email.");
            }
        }

        public Usuario BuscarPorEmail(string email) {
            return Contexto.Usuarios.Where(u => u.Email.ToLower() == email.ToLower()).First();
        }

        public List<Usuario> GetAll() {
            return Contexto.Usuarios.ToList();
        }

        public Usuario FindById(int id) {
            return Contexto.Usuarios.Find(id);
        }
    }
}

```

```

}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DataTransferObjects {
    public class ArticuloDTO {
        public string Nombre { get; set; }
        public string Descripcion { get; set; }
    }
}
using LogicaNegocio.ValueObjects;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DataTransferObjects {
    public class ClienteDTO {
        public int Id { get; set; }
        public string RazonSocial { get; set; }
        public string Rut { get; set; }
        public string Direccion { get; set; }
        public int? DistanciaHastaDeposito { get; set; }
    }
}
using LogicaNegocio.Dominio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DataTransferObjects {
    public class MapperArticulos {
        public static List<ArticuloDTO> ToListDto(List<Articulo> articulos) {
            return articulos.Select(a => new ArticuloDTO() {
                Nombre = a.Nombre,
                Descripcion = a.Descripcion
            })
                .ToList();
        }
    }
}
using LogicaNegocio.Dominio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

```



```

using System.Threading.Tasks;

namespace DataTransferObjects {
    public class MapperClientes {
        public static List<ClienteDTO> ToListDto(List<Cliente> clientes) {
            return clientes.Select(c => new ClienteDTO() {
                Id = c.Id,
                RazonSocial = c.RazonSocial,
                Rut = c.Rut,
                DistanciaHastaDeposito = c.DistanciaHastaDeposito,
                Direccion = c.Direccion.Calle.Value + " " + c.Direccion.NumeroPuerta.Value + ", " +
c.Direccion.Ciudad.Value
            })
            .ToList();
        }
    }
}

using LogicaNegocio.Dominio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DataTransferObjects {
    public class MapperPedidos {
        public static Pedido CrearEntidad(PedidoDTO pedidoDTO) {
            Pedido p;

            if (pedidoDTO.TipoPedido == "PedidoExpress") {
                p = new PedidoExpress();
            } else {
                p = new PedidoComun();
            }

            p.FechaEntrega = pedidoDTO.FechaEntrega;
            p.Estado = pedidoDTO.Estado;

            return p;
        }

        public static PedidoDTO CrearDTO(Pedido pedido) {
            PedidoDTO pedidoDTO = new PedidoDTO();

            pedidoDTO.Id = pedido.Id;
            pedidoDTO.FechaEntrega = pedido.FechaEntrega;
            pedidoDTO.Estado = pedido.Estado;

            return pedidoDTO;
        }

        public static List<PedidoNoEntregadoDTO> ToListDto(List<Pedido> pedidos) {
            return pedidos.Select(p => new PedidoNoEntregadoDTO() {

```

```

        Id = p.Id,
        FechaEntrega = p.FechaEntrega,
        RazonSocialCliente = p.Cliente.RazonSocial,
        Recargo = p.Recargo,
        Iva = p.Iva,
        Total = p.Total
    })
    .ToList();
}
}
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DataTransferObjects {
    public class PedidoDTO {
        public int Id { get; set; }
        public int IdCliente { get; set; }
        public int IdArticulo { get; set; }
        public int Cantidad { get; set; }
        public DateOnly FechaEntrega { get; set; }
        public string TipoPedido { get; set; }
        public string Estado { get; set; }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DataTransferObjects {
    public class PedidoNoEntregadoDTO {
        // DTO usado para:
        // Listado de pedidos que se pueden anular (Estado == "Pendiente")
        // Listado de pedidos anulados (Estado == "Anulado")
        public int Id { get; set; }
        public DateOnly FechaEntrega { get; set; }
        public string RazonSocialCliente { get; set; }
        public decimal Iva { get; set; }
        public decimal Recargo { get; set; }
        public decimal Total { get; set; }
    }
}
using DataTransferObjects;
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.Excepciones;
using LogicaNegocio.InterfacesRepositorios;
using System;

```

```

using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUAgregarArticuloEnPedido : ICUModificar<PedidoDTO> {
        public IRepositoryPedidos Repo { get; set; }
        public IRepositoryLineas RepoLineas { get; set; }
        public IRepositoryArticulos RepoArticulos { get; set; }
        public CUAgregarArticuloEnPedido(IRepositoryPedidos repo, IRepositoryArticulos
repositorioArticulos, IRepositoryLineas repoLineas) {
            Repo = repo;
            RepoArticulos = repositorioArticulos;
            RepoLineas = repoLineas;
        }
        public void Modificar(PedidoDTO obj) {
            Pedido aModificar = Repo.FindById(obj.Id);
            Articulo articulo = RepoArticulos.FindById(obj.IdArticulo);

            if (articulo != null) {
                // chequear stock
                if (ChequearStock(articulo, obj.Cantidad)) {
                    // crear la Linea
                    Linea nuevaLinea = new Linea();
                    nuevaLinea.Articulo = articulo;
                    nuevaLinea.PreciodUnitario = articulo.Precio;
                    nuevaLinea.UnidadesSolicitadas = obj.Cantidad;
                    RepoLineas.Create(nuevaLinea);

                    // agregarla al pedido
                    aModificar.Lineas.Add(nuevaLinea);
                    aModificar.Total = CalcularTotal(aModificar, aModificar.Lineas);
                } else {
                    throw new NoStockException("No hay suficiente stock del artículo seleccionado");
                }
            } else {
                throw new RegistroNoExisteException("El artículo seleccionado para el pedido no existe");
            }

            Repo.Update(aModificar);
        }

        public static bool ChequearStock(Articulo articulo, int cantidadSolicitada) {
            if (articulo.Stock >= cantidadSolicitada) {
                return true;
            } else {
                return false;
            }
        }

        public static decimal CalcularTotal(Pedido p, List<Linea> lineas) {
            decimal resultado = 0;

```

```

        decimal montoArticulos = 0;
        decimal iva = p.Iva + 1;
        decimal recargo = p.Recargo + 1;

        foreach (Linea linea in lineas) {
            montoArticulos += linea.PreciodUnitario * linea.UnidadesSolicitadas;
        }

        resultado = montoArticulos * iva * recargo;

        return resultado;
    }
}

using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUALtaArticulo : ICUALta<Articulo> {
        public IRepositoryioArticulos Repo { get; set; }
        public CUALtaArticulo(IRepositoryioArticulos repo) {
            Repo = repo;
        }
        public void Alta(Articulo obj) {
            Repo.Create(obj);
        }
    }
}

using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUALtaCliente : ICUALta<Cliente> {
        public IRepositoryioClientes Repo { get; set; }
        public CUALtaCliente(IRepositoryioClientes repo) {
            Repo = repo;
        }
        public void Alta(Cliente obj) {
            Repo.Create(obj);
        }
    }
}

```

```

}
using DataTransferObjects;
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.Excepciones;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUALtaPedido : ICUALta<PedidoDTO> {
        public IRepositoryPedidos Repo { get; set; }
        public IRepositoryClientes RepoClientes { get; set; }
        public IRepositoryArticulos RepoArticulos { get; set; }
        public IRepositoryLineas RepoLineas { get; set; }
        public IRepositoryParametros RepoParametros { get; set; }
        public CUALtaPedido(IRepositoryPedidos repo, IRepositoryClientes repoClientes,
IRepositoryArticulos repoArticulos, IRepositoryLineas repoLineas, IRepositoryParametros repoParametros) {
            Repo = repo;
            RepoClientes = repoClientes;
            RepoArticulos = repoArticulos;
            RepoLineas = repoLineas;
            RepoParametros = repoParametros;
        }
        public void Alta(PedidoDTO obj) {
            Pedido nuevoPedido = MapperPedidos.CrearEntidad(obj);
            Pedido._iva = RepoParametros.ObtenerIva();
            Cliente cliente = RepoClientes.FindById(obj.IdCliente);
            Artículo articulo = RepoArticulos.FindById(obj.IdArticulo);
            if (cliente != null) {
                nuevoPedido.Cliente = cliente;

            } else {
                throw new RegistroNoExisteException("El cliente seleccionado para el pedido no existe");
            }

            if(obj.TipoPedido == "PedidoExpress") {
                nuevoPedido.CalcularRecargo(0.15m,0.1m);

            } else {
                nuevoPedido.CalcularRecargo(0.05m, 0);
            }

            if (articulo != null) {
                // chequear stock
                if(ChequearStock(articulo, obj.Cantidad)) {
                    // crear la Linea
                    Linea nuevaLinea = new Linea();
                    nuevaLinea.Articulo = articulo;
                    nuevaLinea.PreciodUnitario = articulo.Precio;
                }
            }
        }
    }
}

```

```

        nuevaLinea.UnidadesSolicitadas = obj.Cantidad;
        RepoLineas.Create(nuevaLinea);

        // agregarla al pedido
        nuevoPedido.Lineas = new List<Linea> { nuevaLinea };
        nuevoPedido.Total = CalcularTotal(nuevoPedido, nuevoPedido.Lineas);
    } else {
        throw new NoStockException("No hay suficiente stock del artículo seleccionado");
    }
} else {
    throw new RegistroNoExisteException("El artículo seleccionado para el pedido no existe");
}

Repo.Create(nuevoPedido);
}

public static bool ChequearStock(Articulo articulo, int cantidadSolicitada) {
    if(articulo.Stock >= cantidadSolicitada) {
        return true;
    } else {
        return false;
    }
}

public static decimal CalcularTotal(Pedido p, List<Linea> lineas) {
    decimal resultado = 0;
    decimal montoArticulos = 0;
    decimal iva = p.Iva + 1;
    decimal recargo = p.Recargo + 1;

    foreach ( Linea linea in lineas ) {
        montoArticulos += linea.PreciodUnitario * linea.UnidadesSolicitadas;
    }

    resultado = montoArticulos * iva * recargo;

    return resultado;
}
}

using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUALtaUsuario : ICUALta<Usuario>{

```

```

    public IRepositoryUsuarios Repo { get; set; }
    public CUALtaUsuario(IRepositoryUsuarios repo) {
        Repo = repo;
    }
    public void Alta(Usuario obj) {
        byte[] key = EncryptionUtility.GenerateRandomKey();
        byte[] iv = EncryptionUtility.GenerateRandomIV();
        obj.ContraseniaEncriptada = EncryptionUtility.EncryptString(obj.Contraseña, key, iv);
        Repo.Create(obj);
    }
}

public class EncryptionUtility {
    public static byte[] GenerateRandomKey() {
        using var aes = Aes.Create();
        aes.GenerateKey();
        return aes.Key;
    }

    public static byte[] GenerateRandomIV() {
        using var aes = Aes.Create();
        aes.GenerateIV();
        return aes.IV;
    }

    public static string EncryptString(string plainText, byte[] key, byte[] iv) {
        using var aes = Aes.Create();
        aes.Key = key;
        aes.IV = iv;

        var encryptor = aes.CreateEncryptor(aes.Key, aes.IV);
        using var msEncrypt = new MemoryStream();
        using (var csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write)) {
            using (var swEncrypt = new StreamWriter(csEncrypt)) {
                swEncrypt.Write(plainText);
            }
        }
        return Convert.ToBase64String(msEncrypt.ToArray());
    }
}

}
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUAnularPedido : ICUAnularPedido {
        public IRepositoryPedidos Repo { get; set; }
    }
}

```

```

        public CUAnularPedido(IRepositorioPedidos repo) {
            Repo = repo;
        }
        public void Anular(int id) {
            Repo.AnularPedido(id);
        }
    }
}
using LogicaNegocio.Dominio;
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUAutenticarUsuario : ICUAutenticarUsuario {
        public IRepositorioUsuarios Repo { get; set; }

        public CUAutenticarUsuario(IRepositorioUsuarios repo) {
            Repo = repo;
        }

        public bool Autenticar(string email, string contraseña, out Usuario u) {
            u = Repo.BuscarPorEmail(email);
            if (u != null) {
                return u.Contraseña == contraseña;
            }
            return false;
        }
    }
}
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUBajaArticulo : ICUBaja<Articulo> {
        public IRepositorioArticulos Repo { get; set; }
        public CUBajaArticulo(IRepositorioArticulos repo) {
            Repo = repo;
        }
        public void Baja(int id) {
            Repo.Delete(id);
        }
    }
}

```



```

}
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUBajaUsuario : ICUBaja<Usuario> {
        public IRepositoryUsuarios Repo { get; set; }
        public CUBajaUsuario(IRepositoryUsuarios repo) {
            Repo = repo;
        }
        public void Baja(int id) {
            Repo.Delete(id);
        }
    }
}

using DataTransferObjects;
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.Excepciones;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUBuscarClientesPorMontoPedido : ICUBuscarClientesPorMontoPedido {
        public IRepositoryPedidos Repo { get; set; }
        public CUBuscarClientesPorMontoPedido(IRepositoryPedidos repo) {
            Repo = repo;
        }
        public List<ClienteDTO> BuscarPorMontoPedido(decimal monto) {
            List<Cliente> clientesEncontrados = Repo.BuscarClientes(monto);
            if (clientesEncontrados.Count > 0) {
                List<ClienteDTO> clientesDTO = MapperClientes.ToListDto(clientesEncontrados);
                return clientesDTO;
            } else {
                throw new RegistroNoExisteException("No existen clientes que hayan realizado pedidos con un total mayor a ese monto.");
            }
        }
    }
}

using DataTransferObjects;
using LogicaAplicacion.InterfacesCasosUso;

```

```

using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using LogicaNegocio.Excepciones;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static System.Runtime.InteropServices.JavaScript.JSType;

namespace LogicaAplicacion.CasosUso {
    public class CUBuscarPorFechaPedido : ICUBuscarPorFechaPedido {
        public IRepositoryPedidos Repo { get; set; }
        public CUBuscarPorFechaPedido(IRepositoryPedidos repo) {
            Repo = repo;
        }
        public List<PedidoNoEntregadoDTO> BuscarPorFechaPedido(DateOnly fecha) {
            List<PedidoNoEntregadoDTO> dtos = new List<PedidoNoEntregadoDTO>();
            List<Pedido> pedidosEncontrados = Repo.BuscarPorFechaDeEmision(fecha);
            if (pedidosEncontrados.Count > 0) {
                dtos = MapperPedidos.ToListDto(pedidosEncontrados);
            } else {
                throw new RegistroNoExisteException("No existen pedidos para la fecha solicitada");
            }
            return dtos;
        }
    }
}

using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUBuscarPorIdArticulo : ICUBuscarPorId<Articulo> {
        public IRepositoryArticulos Repo { get; set; }
        public CUBuscarPorIdArticulo(IRepositoryArticulos repo) {
            Repo = repo;
        }
        public Articulo BuscarPorId(int id) {
            return Repo.FindById(id);
        }
    }
}

using DataTransferObjects;
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.Excepciones;
using LogicaNegocio.InterfacesRepositorios;

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUBuscarPorIdPedido : ICUBuscarPorId<PedidoDTO> {
        public IRepositoryPedidos Repo { get; set; }

        public CUBuscarPorIdPedido(IRepositoryPedidos repo) {
            Repo = repo;
        }

        public PedidoDTO BuscarPorId(int id) {
            PedidoDTO dto = new PedidoDTO();
            Pedido pedidosEncontrados = Repo.FindById(id);
            if (pedidosEncontrados != null) {
                dto = MapperPedidos.CrearDTO(pedidosEncontrados);
            } else {
                throw new RegistroNoExisteException("El pedido no existe");
            }
            return dto;
        }
    }
}

using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUBuscarPorIdUsuario : ICUBuscarPorId<Usuario> {
        public IRepositoryUsuarios Repo { get; set; }
        public CUBuscarPorIdUsuario(IRepositoryUsuarios repo) {
            Repo = repo;
        }
        public Usuario BuscarPorId(int id) {
            return Repo.FindById(id);
        }
    }
}

using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using LogicaAplicacion.InterfacesCasosUso;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

```

```

using System.Threading.Tasks;
using DataTransferObjects;

namespace LogicaAplicacion.CasosUso {
    public class CUBuscarPorRazonSocial : ICUBuscarPorRazonSocial {
        public IRepositoryClientes Repo { get; set; }
        public CUBuscarPorRazonSocial(IRepositoryClientes repo) {
            Repo = repo;
        }
        public List<ClienteDTO> BuscarPorRazonSocial(string nombre) {
            List<Cliente> clientes = Repo.BuscarPorRazonSocial(nombre);
            List<ClienteDTO> clientesDTO = MapperClientes.ToListDto(clientes);
            return clientesDTO;
        }
    }
}

using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUListadoArticulos : ICUListado<Articulo> {
        public IRepositoryArticulos Repo { get; set; }

        public CUListadoArticulos(IRepositoryArticulos repo) {
            Repo = repo;
        }

        public List<Articulo> ObtenerListado() {
            return Repo.GetAll();
        }
    }
}

using DataTransferObjects;
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUListadoClientes : ICUListado<ClienteDTO> {
        public IRepositoryClientes Repo { get; set; }

        public CUListadoClientes(IRepositoryClientes repo) {

```

```

        Repo = repo;
    }
    public List<ClienteDTO> ObtenerListado() {
        List<Cliente> clientes = Repo.GetAll();
        List<ClienteDTO> clientesDTO = MapperClientes.ToListDto(clientes);
        return clientesDTO;
    }
}

using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUListadoPedidos : ICUListado<Pedido> {
        public IRepositoryPedido Repo { get; set; }

        public CUListadoPedidos(IRepositoryPedido repo) {
            Repo = repo;
        }
        public List<Pedido> ObtenerListado() {
            return Repo.GetAll();
        }
    }
}

using DataTransferObjects;
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUListadoPedidosPendientes : ICUListado<PedidoNoEntregadoDTO> {
        public IRepositoryPedido Repo { get; set; }

        public CUListadoPedidosPendientes(IRepositoryPedido repo) {
            Repo = repo;
        }
        public List<PedidoNoEntregadoDTO> ObtenerListado() {
            List<PedidoNoEntregadoDTO> dtos = new List<PedidoNoEntregadoDTO>();
            List<Pedido> pedidosEncontrados = Repo.ListarPedidosPendientes();
            if (pedidosEncontrados.Count > 0) {
                dtos = MapperPedidos.ToListDto(pedidosEncontrados);
            }
        }
    }
}

```

```

        return dtos;
    }
}
}
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUListadoUsuarios : ICUListado<Usuario> {
        public IRepositoryUsuarios Repo { get; set; }

        public CUListadoUsuarios(IRepositoryUsuarios repo)
        {
            Repo = repo;
        }
        public List<Usuario> ObtenerListado() {
            return Repo.GetAll();
        }
    }
}
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUModificarArticulo : ICUModificar<Articulo> {
        public IRepositoryArticulos Repo { get; set; }

        public CUModificarArticulo(IRepositoryArticulos repo) {
            Repo = repo;
        }
        public void Modificar(Articulo obj) {
            Repo.Update(obj);
        }
    }
}
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;

```

```

using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUModificarUsuario : ICUModificar<Usuario> {
        public IRepositoryUsuarios Repo { get; set; }

        public CUModificarUsuario(IRepositoryUsuarios repo) {
            Repo = repo;
        }
        public void Modificar(Usuario obj) {
            Repo.Update(obj);
        }
    }
}

using DataTransferObjects;
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.CasosUso {
    public class CUOrdenarArticulosAsc : ICUOrdenarArticulosAsc {

        public IRepositoryArticulos Repo { get; set; }
        public CUOrdenarArticulosAsc(IRepositoryArticulos repo) {
            Repo = repo;
        }

        // Listado con todos los artículos ordenados alfabéticamente en forma ascendente
        public List<ArticuloDTO> OrdenarPorNombreAsc() {
            List<ArticuloDTO> dtos = new List<ArticuloDTO>();
            List<Articulo> articulosOrdenados = Repo.OrdenarArticulosAsc();
            if (articulosOrdenados.Count > 0) {
                dtos = MapperArticulos.ToListDto(articulosOrdenados);
            }
            return dtos;
        }
    }
}

using DataTransferObjects;
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace LogicaAplicacion.CasosUso {
    public class CUOrdenarPedidosAnuladosDesc : ICUOrdenarPedidosAnuladosDesc {
        public IRepositoryPedidos Repo { get; set; }
        public CUOrdenarPedidosAnuladosDesc(IRepositoryPedidos repo) {
            Repo = repo;
        }
        public List<PedidoNoEntregadoDTO> OrdenarPorFechaDesc() {
            List<PedidoNoEntregadoDTO> dtos = new List<PedidoNoEntregadoDTO>();
            List<Pedido> pedidosOrdenados = Repo.OrdenarPedidosAnuladosPorFechaDesc();
            if (pedidosOrdenados.Count > 0) {
                dtos = MapperPedidos.ToListDto(pedidosOrdenados);
            }
            return dtos;
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.InterfacesCasosUso {
    public interface ICUAlta<T> {
        void Alta(T obj);
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.InterfacesCasosUso {
    public interface ICUAnularPedido {
        void Anular(int id);
    }
}

using LogicaNegocio.Dominio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.InterfacesCasosUso {
    public interface ICUAutenticarUsuario {
        bool Autenticar(string email, string contraseña, out Usuario u);
    }
}

using System;
using System.Collections.Generic;

```



```

using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.InterfacesCasosUso {
    public interface ICUBaja<T> {
        void Baja(int id);
    }
}

using DataTransferObjects;
using LogicaNegocio.Dominio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.InterfacesCasosUso {
    public interface ICUBuscarClientesPorMontoPedido {
        List<ClienteDTO> BuscarPorMontoPedido(decimal monto);
    }
}

using DataTransferObjects;
using LogicaNegocio.Dominio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.InterfacesCasosUso {
    public interface ICUBuscarPorFechaPedido {
        List<PedidoNoEntregadoDTO> BuscarPorFechaPedido(DateOnly fecha);
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaAplicacion.InterfacesCasosUso {
    public interface ICUBuscarPorId<T> {
        T BuscarPorId(int id);
    }
}

using DataTransferObjects;
using LogicaNegocio.Dominio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace LogicaAplicacion.InterfacesCasosUso {
    public interface ICUBuscarPorRazonSocial {
        List<ClienteDTO> BuscarPorRazonSocial(string nombre);
    }
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace LogicaAplicacion.InterfacesCasosUso {
    public interface ICUListado<T> {
        List<T> ObtenerListado();
    }
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace LogicaAplicacion.InterfacesCasosUso {
    public interface ICUModificar<T> {
        void Modificar(T obj);
    }
}

```

```

using DataTransferObjects;
using LogicaNegocio.Dominio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace LogicaAplicacion.InterfacesCasosUso {
    public interface ICUOrdenarArticulosAsc {
        List<ArticuloDTO> OrdenarPorNombreAsc();
    }
}

```

```

using DataTransferObjects;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace LogicaAplicacion.InterfacesCasosUso {
    public interface ICUOrdenarPedidosAnuladosDesc {
        List<PedidoNoEntregadoDTO> OrdenarPorFechaDesc();
    }
}

```

```

using DataTransferObjects;
using LogicaAplicacion.CasosUso;
using LogicaAplicacion.InterfacesCasosUso;
using LogicaDatos.Repositorios;
using LogicaNegocio.InterfacesRepositorios;
using Microsoft.EntityFrameworkCore;

namespace WebAPI {
    public class Program {
        public static void Main(string[] args) {
            var builder = WebApplication.CreateBuilder(args);

            // Add services to the container.
            builder.Services.AddScoped<ICUOrdenarArticulosAsc, CUOrdenarArticulosAsc>();
            builder.Services.AddScoped<ICUOrdenarPedidosAnuladosDesc, CUOrdenarPedidosAnuladosDesc>();

            builder.Services.AddScoped<IRepositorioArticulos, RepositorioArticulos>();
            builder.Services.AddScoped<IRepositorioPedidos, RepositorioPedidos>();

            string conStr = builder.Configuration.GetConnectionString("Caro-Zenbook");
            builder.Services.AddDbContext<ObligatorioContext>(options => options.UseSqlServer(conStr));

            builder.Services.AddControllers();
            // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
            builder.Services.AddEndpointsApiExplorer();
            builder.Services.AddSwaggerGen();

            var app = builder.Build();

            // Configure the HTTP request pipeline.
            if (app.Environment.IsDevelopment()) {
                app.UseSwagger();
                app.UseSwaggerUI();
            }

            app.UseAuthorization();

            app.MapControllers();

            app.Run();
        }
    }
}

using DataTransferObjects;
using LogicaAplicacion.CasosUso;
using LogicaAplicacion.InterfacesCasosUso;
using Microsoft.AspNetCore.Mvc;

// For more information on enabling Web API for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860

namespace WebAPI.Controllers {

```

```

[Route("api/[controller]")]
[ApiController]
public class PedidosController : ControllerBase {
    public ICUOrdenarPedidosAnuladosDesc CUOrdenarPedidosAnuladosDesc { get; set; }

    public PedidosController(ICUOrdenarPedidosAnuladosDesc cuOrdenarPedidosAnuladosDesc) {
        CUOrdenarPedidosAnuladosDesc = cuOrdenarPedidosAnuladosDesc;
    }

    // GET: api/<PedidosController>
    [HttpGet]
    public IActionResult Get() {
        try {
            List<PedidoNoEntregadoDTO> pedidoDTOS = CUOrdenarPedidosAnuladosDesc.OrdenarPorFechaDesc();
            if(pedidoDTOS.Any()) {
                // 200 ok
                return Ok(pedidoDTOS);
            } else {
                // 404 - Not Found
                return NotFound("No existen pedidos.");
            }
        } catch (Exception ex) {
            // 500 - Internal Server Error
            return StatusCode(500, "Ocurrió un error en el servidor.");
        }
    }

    // GET api/<PedidosController>/5
    [HttpGet("{id}")]
    public string Get(int id) {
        return "value";
    }

    // POST api/<PedidosController>
    [HttpPost]
    public void Post([FromBody] string value) {
    }

    // PUT api/<PedidosController>/5
    [HttpPut("{id}")]
    public void Put(int id, [FromBody] string value) {
    }

    // DELETE api/<PedidosController>/5
    [HttpDelete("{id}")]
    public void Delete(int id) {
    }
}

using DataTransferObjects;
using LogicaAplicacion.InterfacesCasosUso;
using Microsoft.AspNetCore.Mvc;

```

// For more information on enabling Web API for empty projects, visit  
<https://go.microsoft.com/fwlink/?LinkID=397860>

```
namespace WebAPI.Controllers {
    [Route("api/[controller]")]
    [ApiController]
    public class ArticulosController : ControllerBase {
        public ICUOrdenarArticulosAsc CUOrdenarArticulosAsc { get; set; }
        public ArticulosController(ICUOrdenarArticulosAsc cuOrdenarArticulosAsc) {
            CUOrdenarArticulosAsc = cuOrdenarArticulosAsc;
        }

        // GET: api/<ArticulosController>
        [HttpGet]
        public IActionResult Get() {
            try {
                List<ArticuloDTO> articuloDTOS = CUOrdenarArticulosAsc.OrdenarPorNombreAsc();
                if (articuloDTOS.Any()) {
                    // 200 ok
                    return Ok(articuloDTOS);
                } else {
                    // 404 - Not Found
                    return NotFound("No existen artículos.");
                }
            } catch (Exception ex) {
                // 500 - Internal Server Error
                return StatusCode(500, "Ocurrió un error en el servidor.");
            }
        }

        // GET api/<ArticulosController>/5
        [HttpGet("{id}")]
        public string Get(int id) {
            return "value";
        }

        // POST api/<ArticulosController>
        [HttpPost]
        public void Post([FromBody] string value) {
        }

        // PUT api/<ArticulosController>/5
        [HttpPut("{id}")]
        public void Put(int id, [FromBody] string value) {
        }

        // DELETE api/<ArticulosController>/5
        [HttpDelete("{id}")]
        public void Delete(int id) {
        }
    }
}
using DataTransferObjects;
```

```

using LogicaAplicacion.CasosUso;
using LogicaAplicacion.InterfacesCasosUso;
using LogicaDatos.Repositorios;
using LogicaNegocio.Dominio;
using LogicaNegocio.InterfacesRepositorios;
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddSession(options => {
    options.Cookie.HttpOnly = true;
    options.Cookie.IsEssential = true;
});

// Add services to the container.
builder.Services.AddControllersWithViews();

builder.Services.AddScoped<ICUAlta<Articulo>, CUAltaArticulo>();
builder.Services.AddScoped<ICUBaja<Articulo>, CUBajaArticulo>();
builder.Services.AddScoped<ICUListado<Articulo>, CUListadoArticulos>();
builder.Services.AddScoped<ICUModificar<Articulo>, CUModificarArticulo>();
builder.Services.AddScoped<ICUBuscarPorId<Articulo>, CUBuscarPorIdArticulo>();

builder.Services.AddScoped<ICUAlta<Usuario>, CUAltaUsuario>();
builder.Services.AddScoped<ICUBaja<Usuario>, CUBajaUsuario>();
builder.Services.AddScoped<ICUListado<Usuario>, CUListadoUsuarios>();
builder.Services.AddScoped<ICUModificar<Usuario>, CUModificarUsuario>();
builder.Services.AddScoped<ICUBuscarPorId<Usuario>, CUBuscarPorIdUsuario>();
builder.Services.AddScoped<ICUAutenticarUsuario, CUAutenticarUsuario>();

builder.Services.AddScoped<ICUListado<ClienteDTO>, CUListadoClientes>();
builder.Services.AddScoped<ICUBuscarPorRazonSocial, CUBuscarPorRazonSocial>();
builder.Services.AddScoped<ICUBuscarClientesPorMontoPedido, CUBuscarClientesPorMontoPedido>();

builder.Services.AddScoped<ICUAlta<PedidoDTO>, CUAltaPedido>();
builder.Services.AddScoped<ICUBuscarPorFechaPedido, CUBuscarPorFechaPedido>();
builder.Services.AddScoped<ICUAnularPedido, CUAnularPedido>();
builder.Services.AddScoped<ICUBuscarPorId<PedidoDTO>, CUBuscarPorIdPedido>();
builder.Services.AddScoped<ICUModificar<PedidoDTO>, CUAgregarArticuloEnPedido>();
builder.Services.AddScoped<ICUListado<Pedido>, CUListadoPedidos>();
builder.Services.AddScoped<ICUListado<PedidoNoEntregadoDTO>, CUListadoPedidosPendientes>();

builder.Services.AddScoped<IRepositorioUsuarios, RepositorioUsuarios>();
builder.Services.AddScoped<IRepositorioClientes, RepositorioClientes>();
builder.Services.AddScoped<IRepositorioArticulos, RepositorioArticulos>();
builder.Services.AddScoped<IRepositorioPedidos, RepositorioPedidos>();
builder.Services.AddScoped<IRepositorioLineas, RepositorioLineas>();
builder.Services.AddScoped<IRepositorioParametros, RepositorioParametros>();

string conStr = builder.Configuration.GetConnectionString("Caro-Zenbook");
builder.Services.AddDbContext<ObligatorioContext>(options => options.UseSqlServer(conStr));

var app = builder.Build();

```

```

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment()) {
    app.UseExceptionHandler("/Home/Error");
}

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseSession();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
@model Obligatorio.Models.IngresarViewModel
@{
    ViewData["Title"] = "Ingresar";
}
<h1>@ViewData["Title"]</h1>

<div class="row">
    <div class="col-md-4">
        <form asp-action="Ingresar">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Email" class="control-label"></label>
                <input asp-for="Email" class="form-control" />
                <span asp-validation-for="Email" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Password" class="control-label"></label>
                <input asp-for="Password" class="form-control" />
                <span asp-validation-for="Password" class="text-danger"></span>
            </div>

            <div class="form-group">
                <input type="submit" value="Iniciar sesión" class="btn btn-primary mt-2" />
            </div>

            <span class="alert-danger">@ViewBag.Error</span>
        </form>
    </div>
</div>
@model IEnumerable<LogicaNegocio.Dominio.Usuario>

@{
    ViewData["Title"] = "Usuarios";
}

```

```

<div class="text-center">
    <h1 class="display-4">@ViewData["Title"]</h1>
</div>

<div>
    <a asp-action="Create">Crear Usuario</a>
</div>

<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Id)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Nombre)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Email)
            </th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model) {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Id)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Nombre)
                    @Html.DisplayFor(modelItem => item.Apellido)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Email)
                </td>
                <td>
                    @Html.ActionLink("Edit", "Edit", new { id = item.Id }) |
                    @Html.ActionLink("Details", "Details", new { id = item.Id }) |
                    @Html.ActionLink("Delete", "Delete", new { id = item.Id })
                </td>
            </tr>
        }
    </tbody>
</table>
@model LogicaNegocio.Dominio.Usuario

@{
    ViewData["Title"] = "Edit";
}

<h1>Edit</h1>

```



```

<span>@ViewBag.ErrorMessage</span>

<h4>Usuario</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Email" class="control-label"></label>
                <input asp-for="Email" class="form-control" />
                <span asp-validation-for="Email" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Nombre" class="control-label"></label>
                <input asp-for="Nombre" class="form-control" />
                <span asp-validation-for="Nombre" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Apellido" class="control-label"></label>
                <input asp-for="Apellido" class="form-control" />
                <span asp-validation-for="Apellido" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Contraseña" class="control-label"></label>
                <input asp-for="Contraseña" class="form-control" />
                <span asp-validation-for="Contraseña" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Tipo" class="control-label"></label>
                <input asp-for="Tipo" class="form-control" />
                <span asp-validation-for="Tipo" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Save" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
@model LogicaNegocio.Dominio.Usuario

@{
    ViewData["Title"] = "Delete";
}

```

```

<h1>Delete</h1>
<span>@ViewBag.ErrorMessage</span>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Usuario</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Id)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Id)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Email)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Email)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Nombre)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Nombre)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Apellido)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Apellido)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Contraseña)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Contraseña)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Tipo)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Tipo)
        </dd>
    </dl>

    <form asp-action="Delete">
        <input type="submit" value="Delete" class="btn btn-danger" /> |
        <a asp-action="Index">Back to List</a>
    </form>
</div>
@model LogicaNegocio.Dominio.Usuario

```

```

@{
    ViewData["Title"] = "Create";
}

<h1>Create</h1>
<span>@ViewBag.ErrorMessage</span>

<h4>Usuario</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Email" class="control-label"></label>
                <input asp-for="Email" class="form-control" />
                <span asp-validation-for="Email" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Nombre" class="control-label"></label>
                <input asp-for="Nombre" class="form-control" />
                <span asp-validation-for="Nombre" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Apellido" class="control-label"></label>
                <input asp-for="Apellido" class="form-control" />
                <span asp-validation-for="Apellido" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Contraseña" class="control-label"></label>
                <input asp-for="Contraseña" class="form-control" />
                <span asp-validation-for="Contraseña" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Tipo" class="control-label"></label>
                <input asp-for="Tipo" class="form-control" />
                <span asp-validation-for="Tipo" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
<!DOCTYPE html>

```

```

<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Obligatorio</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <link rel="stylesheet" href="~/Obligatorio.styles.css" asp-append-version="true" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom
box-shadow mb-3">
            <partial name="_Menu.cshtml" />
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted">
        <div class="container">
            &copy; 2024 - Obligatorio
        </div>
    </footer>

    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>

    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
@using Microsoft.AspNetCore.Http;

<div class="container">
    <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index"> Obligatorio</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse"
aria-controls="navbarSupportedContent"
        aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="navbar-collapse collapse d-sm-inline-flex">
        <ul class="navbar-nav flex-grow-1">
            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
            </li>

            <!-- Menú de Administrador -->
            @if (Context.Session.GetString("TIPOUSUARIO") == "Administrador") {
                <li class="nav-item">

```

```

        <a class="nav-link" asp-controller="Pedidos" asp-action="Index">Pedidos</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" asp-controller="Articulos" asp-action="Index">Articulos</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" asp-controller="Clientes" asp-action="Index">Clientes</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" asp-controller="Usuarios" asp-action="Index">Usuarios</a>
    </li>
}
</ul>

<!-- Menú de Sesión -->
<ul class="navbar-nav flex-grow-1 justify-content-end">
    @if (Context.Session.GetString("TIPOUSUARIO") == null) {
        <li class="nav-item">
            <a class="nav-link text-dark" asp-controller="Usuarios"
asp-action="Ingresar">Ingresar</a>
        </li>
    } else {
        <li class="nav-item">
            <a class="nav-link" asp-controller="Usuarios" asp-action="CerrarSesion">Cerrar Sesión</a>
        </li>
    }
</ul>
</div>
</div>
@model IEnumerable<DataTransferObjects.PedidoNoEntregadoDTO>

@{
    ViewData["Title"] = "Index";
}

<div class="text-center">
    <h1 class="display-4">@ViewData["Title"]</h1>
</div>

<div>
    <a asp-action="Create">Crear Pedido</a>
</div>
<div>
    <a asp-action="BuscarPedidos">Buscar Pedidos</a>
</div>

<span>@ViewBag.ErrorMessage</span>

<h3>Pedidos pendientes</h3>

<table class="table">
    <thead>
        <tr>

```

```

        <th>
            @Html.DisplayNameFor(model => model.FechaEntrega)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.RazonSocialCliente)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Recargo)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Iva)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Total)
        </th>
    </tr>
</thead>
<tbody>
    @foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.FechaEntrega)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.RazonSocialCliente)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Recargo)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Iva)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Total)
            </td>
            <td>
                @Html.ActionLink("Agregar Artículo", "AgregarArticulo", new { id = item.Id })
            </td>
        </tr>
    }
</tbody>
</table>

```

```
@model AltaPedidoViewModel
```

```

@{
    ViewData["Title"] = "Create";
}

```

```

<h1>Create</h1>
<span>@ViewBag.ErrorMessage</span>

```

```

<h4>Pedido</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <legend class="col-form-label">Tipo de Pedido</legend>
                <div class="form-check form-check-inline">
                    <label asp-for="TipoPedido" class="form-check-label">Pedido Común</label>
                    <input asp-for="TipoPedido" class="form-check-input" type="radio" value="PedidoComun">
                </div>
                <div class="form-check form-check-inline">
                    <label asp-for="TipoPedido" class="form-check-label">Pedido Express</label>
                    <input asp-for="TipoPedido" class="form-check-input" type="radio" value="PedidoExpress">
                </div>
            </div>
            <div class="form-group mt-3">
                <label asp-for="IdCliente">Cliente</label>
                <select asp-for="IdCliente" class="form-select">
                    @foreach (var cliente in Model.Clientes) {
                        <option value="@cliente.Id">@cliente.RazonSocial</option>
                    }
                </select>
            </div>
            <div class="form-group mt-3">
                <label asp-for="FechaEntrega" class="control-label"></label>
                <input asp-for="FechaEntrega" type="date" class="form-control"
value="@DateOnly.FromDateTime(DateTime.Today).ToString("yyyy-MM-dd")" />
                <span asp-validation-for="FechaEntrega" class="text-danger"></span>
            </div>
            <div class="form-group mt-3">
                <div class="row">
                    <div class="col-md-9">
                        <label asp-for="IdArticulo">Artículo</label>
                        <select asp-for="IdArticulo" class="form-select">
                            @foreach (var articulo in Model.Articulos) {
                                <option value="@articulo.Id">@articulo.Nombre</option>
                            }
                        </select>
                    </div>
                    <div class="col-md-3">
                        <label asp-for="Cantidad" class="control-label"></label>
                        <input asp-for="Cantidad" class="form-control" type="number" />
                        <span asp-validation-for="Cantidad" class="text-danger"></span>
                    </div>
                </div>
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-primary mt-3" />
            </div>
        </form>
    </div>

```

```

        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
@model IEnumerable<DataTransferObjects.PedidoNoEntregadoDTO>

@{
    ViewData["Title"] = "Anular Pedidos";
}

<div class="text-center">
    <h1 class="display-4">@ViewData["Title"]</h1>
</div>

<div class="row justify-content-center">
    <div class="col-md-4">
        <form asp-action="BuscarPedidos">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label class="control-label">Fecha de emisión:</label>
                <input type="date" name="fecha" class="form-control"
value="@DateOnly.FromDateTime(DateTime.Today).ToString("yyyy-MM-dd")" />
            </div>
            <div class="form-group">
                <input type="submit" value="Buscar" class="btn btn-primary mt-2" />
            </div>
        </form>
    </div>
</div>

<br>

@if(Model != null) {
    @if (Model.Count() > 0) {
        <table class="table">
            <thead>
                <tr>
                    <th>
                        @Html.DisplayNameFor(model => model.FechaEntrega)
                    </th>
                    <th>
                        @Html.DisplayNameFor(model => model.RazonSocialCliente)
                    </th>
                    <th>
                        @Html.DisplayNameFor(model => model.Total)
                    </th>
                </tr>
            </thead>
        </table>
    }
}

```



```

        <th></th>
    </tr>
</thead>
<tbody>
    @foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.FechaEntrega)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.RazonSocialCliente)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Total)
            </td>
            <td>
                @Html.ActionLink("Anular Pedido", "AnularPedidos", new { id = item.Id })
            </td>
        </tr>
    }
</tbody>
</table>
} else {
    <span>@ViewBag.ErrorMessage</span>
}
}
@model DataTransferObjects.PedidoDTO

@{
    ViewData["Title"] = "Anular Pedido";
}

```

```

<div class="text-center">
    <h1 class="display-4">@ViewData["Title"]</h1>
</div>

<h3>Seguro que quieres anular este pedido?</h3>
<div>
    <h4>Pedido</h4>
    <hr />
    <dl class="row">
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Id)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Id)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.FechaEntrega)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.FechaEntrega)
        </dd>
    </dl>

```

```

        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Estado)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Estado)
        </dd>
    </dl>

    <form asp-action="AnularPedidos">
        <input type="submit" value="Anular" class="btn btn-danger" /> |
        <a asp-action="Index">Back to List</a>
    </form>
</div>
@model DataTransferObjects.PedidoDTO

@{
    ViewData["Title"] = "Agregar artículos";
}

<div class="text-center">
    <h1 class="display-4">@ViewData["Title"]</h1>
</div>
<span>@ViewBag.ErrorMessage</span>

<div class="row">
    <div class="col-md-4">
        <form asp-action="AgregarArticulo">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group mt-3">
                <dl class="row">
                    <dt class="col-sm-2">
                        @Html.DisplayNameFor(model => model.Id)
                    </dt>
                    <dd class="col-sm-10">
                        @Html.DisplayFor(model => model.Id)
                    </dd>
                </dl>
                <div class="row">
                    <div class="col-md-9">
                        <label asp-for="IdArticulo">Artículo</label>
                        <select asp-for="IdArticulo" class="form-control">
                            @foreach (var articulo in ViewBag.Articulos) {
                                <option value="@articulo.Id">@articulo.Nombre</option>
                            }
                        </select>
                    </div>
                    <div class="col-md-3">
                        <label asp-for="Cantidad" class="control-label"></label>
                        <input asp-for="Cantidad" class="form-control" type="number" />
                        <span asp-validation-for="Cantidad" class="text-danger"></span>
                    </div>
                </div>
            </div>
        </form>
    </div>

```

```

        </div>

        <div class="form-group">
            <input type="submit" value="Agregar" class="btn btn-primary mt-3" />
        </div>
    </form>
</div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>

@section Scripts {
    @{
        await Html.RenderPartialAsync("_ValidationScriptsPartial");
    }
}
@{
    ViewData["Title"] = "Home page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://learn.microsoft.com/aspnet/core">building Web apps with ASP.NET Core</a>.</p>
</div>
@model IEnumerable<DataTransferObjects.ClienteDTO>

@{
    ViewData["Title"] = "Clientes";
}

<div class="text-center">
    <h1 class="display-4">@ViewData["Title"]</h1>
</div>

<div>
    <a asp-action="BuscarClientes">Buscar Por Razon Social</a>
</div>
<div>
    <a asp-action="BuscarPorMonto">Buscar Por Monto de Pedido</a>
</div>
<span>@ViewBag.ErrorMessage</span>

<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Id)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.RazonSocial)
            </th>
        </tr>
    </thead>
    <tbody>

```

```

        </th>
        <th>
            @Html.DisplayNameFor(model => model.Rut)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Direccion)
        </th>
    </tr>
</thead>
<tbody>
    @foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Id)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.RazonSocial)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Rut)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Direccion)
            </td>
        </tr>
    }
</tbody>
</table>
@model IEnumerable<DataTransferObjects.ClienteDTO>

@{
    ViewData["Title"] = "Buscar Por Monto de Pedido";
}

<div class="text-center">
    <h1 class="display-4">@ViewData["Title"]</h1>
</div>

<div class="row">
    <div class="col-md-4">
        <form asp-action="BuscarPorMonto">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>

            <div class="form-group">
                <label class="control-label">Monto:</label>
                <input type="number" name="monto" class="form-control" />
            </div>
            <div class="form-group">
                <input type="submit" value="Buscar" class="btn btn-primary mt-2" />
            </div>
        </form>
    </div>
</div>
</div>

```

```

@if (Model.Count() > 0) {
    <table class="table">
        <thead>
            <tr>
                <th>
                    @Html.DisplayNameFor(model => model.Id)
                </th>
                <th>
                    @Html.DisplayNameFor(model => model.RazonSocial)
                </th>
                <th>
                    @Html.DisplayNameFor(model => model.Rut)
                </th>
                <th>
                    @Html.DisplayNameFor(model => model.DistanciaHastaDeposito)
                </th>
            </tr>
        </thead>
        <tbody>
            @foreach (var item in Model) {
                <tr>
                    <td>
                        @Html.DisplayFor(modelItem => item.Id)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.RazonSocial)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.Rut)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.DistanciaHastaDeposito)
                    </td>
                </tr>
            }
        </tbody>
    </table>
} else {
    <span>@ViewBag.ErrorMessage</span>
}
@model IEnumerable<DataTransferObjects.ClienteDTO>

@{
    ViewData["Title"] = "Buscar Clientes";
}

<div class="text-center">
    <h1 class="display-4">@ViewData["Title"]</h1>
</div>

```

```

<div class="row">
  <div class="col-md-4">
    <form asp-action="BuscarPorMonto">
      <div asp-validation-summary="ModelOnly" class="text-danger"></div>

      <div class="form-group">
        <label class="control-label">Razon Social:</label>
        <input type="text" name="razonSocial" class="form-control" />
      </div>
      <div class="form-group">
        <input type="submit" value="Buscar" class="btn btn-primary mt-2" />
      </div>
    </form>
  </div>
</div>

```

```

@if (Model.Count() > 0) {
  <table class="table">
    <thead>
      <tr>
        <th>
          @Html.DisplayNameFor(model => model.Id)
        </th>
        <th>
          @Html.DisplayNameFor(model => model.RazonSocial)
        </th>
        <th>
          @Html.DisplayNameFor(model => model.Rut)
        </th>
        <th>
          @Html.DisplayNameFor(model => model.Direccion)
        </th>
      </tr>
    </thead>
    <tbody>

      @foreach (var item in Model) {
        <tr>
          <td>
            @Html.DisplayFor(modelItem => item.Id)
          </td>
          <td>
            @Html.DisplayFor(modelItem => item.RazonSocial)
          </td>
          <td>
            @Html.DisplayFor(modelItem => item.Rut)
          </td>
          <td>
            @Html.DisplayFor(modelItem => item.Direccion)
          </td>
        </tr>
      }
    </tbody>
  </table>
}

```

```

        </tbody>
    </table>
} else {
    <span>@ViewBag.ErrorMessage</span>
}
@model IEnumerable<LogicaNegocio.Dominio.Articulo>

@{
    ViewData["Title"] = "Artículos";
}

<div class="text-center">
    <h1 class="display-4">@ViewData["Title"]</h1>
</div>

<div>
    <a asp-action="Create">Crear Artículo</a>
</div>

<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Id)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Nombre)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Descripcion)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Precio)
            </th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model) {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Id)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Nombre)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Descripcion)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Precio)
                </td>
            </tr>
        }
    </tbody>
</table>

```

```

                @Html.ActionLink("Edit", "Edit", new { id = item.Id }) |
                @Html.ActionLink("Details", "Details", new { id = item.Id }) |
                @Html.ActionLink("Delete", "Delete", new { id = item.Id })
            </td>
        </tr>
    }
</tbody>
</table>
@model LogicaNegocio.Dominio.Articulo

@{
    ViewData["Title"] = "Edit";
}

<h1>Edit</h1>
<span>@ViewBag.ErrorMessage</span>

<h4>Articulo</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Nombre" class="control-label"></label>
                <input asp-for="Nombre" class="form-control" />
                <span asp-validation-for="Nombre" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Descripcion" class="control-label"></label>
                <input asp-for="Descripcion" class="form-control" />
                <span asp-validation-for="Descripcion" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="CodigoProveedor" class="control-label"></label>
                <input asp-for="CodigoProveedor" class="form-control" />
                <span asp-validation-for="CodigoProveedor" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Precio" class="control-label"></label>
                <input asp-for="Precio" class="form-control" />
                <span asp-validation-for="Precio" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Stock" class="control-label"></label>
                <input asp-for="Stock" class="form-control" />
                <span asp-validation-for="Stock" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Save" class="btn btn-primary mt-2" />
            </div>
        </form>
    </div>

```



</div>

<div>

</div>

@section Scripts {

    @{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

@model LogicaNegocio.Dominio.Articulo

@{

    ViewData["Title"] = "Delete";

}

<h1>Delete</h1>

<span>@ViewBag.ErrorMessage</span>

<h3>Are you sure you want to delete this?</h3>

<div>

    <h4>Articulo</h4>

    <hr />

    <dl class="row">

        <dt class = "col-sm-2">

            @Html.DisplayNameFor(model => model.Id)

        </dt>

        <dd class = "col-sm-10">

            @Html.DisplayFor(model => model.Id)

        </dd>

        <dt class = "col-sm-2">

            @Html.DisplayNameFor(model => model.Nombre)

        </dt>

        <dd class = "col-sm-10">

            @Html.DisplayFor(model => model.Nombre)

        </dd>

        <dt class = "col-sm-2">

            @Html.DisplayNameFor(model => model.Descripcion)

        </dt>

        <dd class = "col-sm-10">

            @Html.DisplayFor(model => model.Descripcion)

        </dd>

        <dt class = "col-sm-2">

            @Html.DisplayNameFor(model => model.CodigoProveedor)

        </dt>

        <dd class = "col-sm-10">

            @Html.DisplayFor(model => model.CodigoProveedor)

        </dd>

        <dt class = "col-sm-2">

            @Html.DisplayNameFor(model => model.Precio)

        </dt>

        <dd class = "col-sm-10">

            @Html.DisplayFor(model => model.Precio)

        </dd>

```

        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Stock)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Stock)
        </dd>
    </dl>

    <form asp-action="Delete">
        <input type="submit" value="Delete" class="btn btn-danger" /> |
        <a asp-action="Index">Back to List</a>
    </form>
</div>
@model LogicaNegocio.Dominio.Articulo

@{
    ViewData["Title"] = "Create";
}

<h1>Create</h1>
<span>@ViewBag.ErrorMessage</span>

<h4>Articulo</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Nombre" class="control-label"></label>
                <input asp-for="Nombre" class="form-control" />
                <span asp-validation-for="Nombre" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Descripcion" class="control-label"></label>
                <input asp-for="Descripcion" class="form-control" />
                <span asp-validation-for="Descripcion" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="CodigoProveedor" class="control-label"></label>
                <input asp-for="CodigoProveedor" class="form-control" />
                <span asp-validation-for="CodigoProveedor" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Precio" class="control-label"></label>
                <input asp-for="Precio" class="form-control" />
                <span asp-validation-for="Precio" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Stock" class="control-label"></label>
                <input asp-for="Stock" class="form-control" />
                <span asp-validation-for="Stock" class="text-danger"></span>
            </div>
        </form>
    </div>

```

```

        <div class="form-group">
            <input type="submit" value="Create" class="btn btn-primary mt-2" />
        </div>
    </form>
</div>
</div>

```

```

<div>
    <a asp-action="Index">Back to List</a>
</div>

```

```

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}

```

```
using System.ComponentModel.DataAnnotations;
```

```

namespace Obligatorio.Models {
    public class IngresarViewModel {
        [Required]
        [EmailAddress]
        public string Email { get; set; }
        [Required]
        public string Password { get; set; }
    }
}

```

```

namespace Obligatorio.Models {
    public class ErrorViewModel {
        public string RequestId { get; set; }
        public string Mensaje { get; set; }

        public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);
    }
}

```

```

using DataTransferObjects;
using LogicaNegocio.Dominio;

```

```

namespace Obligatorio.Models {
    public class AltaPedidoViewModel {
        public List<ClienteDTO>? Clientes { get; set; }
        public int IdCliente { get; set; }
        public List<Articulo>? Articulos { get; set; }
        public int IdArticulo { get; set; }
        public DateOnly FechaEntrega { get; set; }
        public string TipoPedido { get; set; }
        public int Cantidad { get; set; }
    }
}

```

```

}
using System;
using System.Linq;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Filters;

```

```

namespace Obligatorio.Filters {
    public class PrivadoAttribute : Attribute, IActionFilter {
        public string TipoUsuarios { get; set; }

        public void OnActionExecuted(ActionExecutedContext context) {
        }

        public void OnActionExecuting(ActionExecutingContext context) {
            bool usuarioAutenticado =
!string.IsNullOrEmpty(context.HttpContext.Session.GetString("TIPOUSUARIO"));
            if (!usuarioAutenticado) {
                context.Result = new RedirectResult("/home/ingresar");
            } else {
                if (!string.IsNullOrEmpty(TipoUsuarios)) {
                    if
(!TipoUsuarios.Split(",").Contains(context.HttpContext.Session.GetString("TIPOUSUARIO")))
                        context.Result = new RedirectResult("/home/error");
                }
            }
        }
    }
}

using LogicaAplicacion.CasosUso;
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Http;
using Obligatorio.Models;
using LogicaNegocio.Excepciones;
using Obligatorio.Filters;

namespace Obligatorio.Controllers {
    public class UsuariosController : Controller {
        public ICUAlta<Usuario> CUAlta { get; set; }
        public ICUBaja<Usuario> CUBaja { get; set; }
        public ICUListado<Usuario> CUListado { get; set; }
        public ICUModificar<Usuario> CUModificar { get; set; }
        public ICUAutenticarUsuario CUAutenticarUsuario { get; set; }
        public ICUBuscarPorId<Usuario> CUBuscarPorIdUsuario { get; set; }

        public UsuariosController(ICUAutenticarUsuario cuAutenticarUsuario, ICUAlta<Usuario> cuAlta,
ICUListado<Usuario> cuListado, ICUModificar<Usuario> cuModificar, ICUBuscarPorId<Usuario>
cuBuscarPorIdUsuario, ICUBaja<Usuario> cuBaja) {
            CUAutenticarUsuario = cuAutenticarUsuario;
            CUAlta = cuAlta;
            CUListado = cuListado;
            CUModificar = cuModificar;
            CUBuscarPorIdUsuario = cuBuscarPorIdUsuario;
            CUBaja = cuBaja;
        }
    }
}

```

```

[Privado(TipoUsuarios = "Administrador")]
public IActionResult Index() {
    return View(CUListado.ObtenerListado());
}

//-----
//----- INGRESAR -----
//-----
public IActionResult Ingresar() {
    return View();
}

[HttpPost]
public IActionResult Ingresar(IngresarViewModel model) {
    if (!ModelState.IsValid) {
        ViewBag.Error = "Email o Contraseña no pueden ser vacíos";
        return View();
    }

    if (CUAutenticarUsuario.Autenticar(model.Email, model.Password, out Usuario u)) {
        HttpContext.Session.SetString("EMAIL", model.Email);
        HttpContext.Session.SetString("TIPOUSUARIO", u.Tipo.ToString());
        return RedirectToAction("Index", "Home");
    }

    HttpContext.Session.Clear();

    ViewBag.Error = "Las credenciales no son correctas.";
    return View();
}

public IActionResult CerrarSesion() {
    HttpContext.Session.Clear();
    return RedirectToAction("Index", "Home");
}

//-----
//----- CREATE -----
//-----
[Privado(TipoUsuarios = "Administrador")]
public ActionResult Create() {
    return View();
}

[Privado(TipoUsuarios = "Administrador")]
[HttpPost]
public ActionResult Create(Usuario nuevo) {
    try {
        CUALta.Alta(nuevo);
        return RedirectToAction("Index", "Usuarios");
    } catch (DatosInvalidosException e) {
        ViewBag.ErrorMessage = e.Message;
    } catch (DuplicadoException e) {

```

```

        ViewBag.ErrorMessage = e.Message;
    } catch (Exception e) {
        ViewBag.ErrorMessage = e.Message;
    }

    return View();
}

//-----
//----- UPDATE -----
//-----
[Privado(TipoUsuarios = "Administrador")]
public ActionResult Edit(int id) {
    Usuario u = CUBuscarPorIdUsuario.BuscarPorId(id);
    return View(u);
}

[Privado(TipoUsuarios = "Administrador")]
[HttpPost]
public ActionResult Edit(int id, Usuario u) {
    try {
        CUModificar.Modificar(u);
        return RedirectToAction("Index", "Usuarios");
    } catch (DatosInvalidosException e) {
        ViewBag.ErrorMessage = e.Message;
    } catch (DuplicadoException e) {
        ViewBag.ErrorMessage = e.Message;
    } catch (Exception e) {
        ViewBag.ErrorMessage = e.Message;
    }

    return View();
}

//-----
//----- DELETE -----
//-----
[Privado(TipoUsuarios = "Administrador")]
public ActionResult Delete(int id) {
    Usuario u = CUBuscarPorIdUsuario.BuscarPorId(id);
    return View(u);
}

[Privado(TipoUsuarios = "Administrador")]
[HttpPost]
public ActionResult Delete(int id, Usuario u) {
    try {
        CUBaja.Baja(id);
        return RedirectToAction("Index", "Usuarios");
    } catch (RegistroNoExisteException e) {
        ViewBag.ErrorMessage = e.Message;
    } catch (Exception e) {
        ViewBag.ErrorMessage = e.Message;
    }
}

```

```

        }
        return View();
    }
}

using DataTransferObjects;
using LogicaAplicacion.CasosUso;
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.Excepciones;
using Microsoft.AspNetCore.Mvc;
using Obligatorio.Filters;
using Obligatorio.Models;

namespace Obligatorio.Controllers {
    public class PedidosController : Controller {
        public ICUListado<Articulo> CUListadoArticulos { get; set; }
        public ICUListado<ClienteDTO> CUListadoClientes { get; set; }
        public ICUALta<PedidoDTO> CUALta { get; set; }
        public ICUBuscarPorFechaPedido CUBuscarPorFechaPedido { get; set; }
        public ICUANularPedido CUAnularPedido { get; set; }
        public ICUBuscarPorId<PedidoDTO> CUBuscarPorIdPedido { get; set; }
        public ICUListado<PedidoNoEntregadoDTO> CUListadoPedidosPendientes { get; set; }
        public ICUModificar<PedidoDTO> CUAgregarArticulo { get; set; }

        public PedidosController(
            ICUListado<Articulo> cuListadoArticulos,
            ICUListado<ClienteDTO> cuListadoClientes,
            ICUALta<PedidoDTO> cuAlta,
            ICUBuscarPorFechaPedido cuBuscarPorFechaPedido,
            ICUANularPedido cuAnularPedido,
            ICUBuscarPorId<PedidoDTO> cuBuscarPorIdPedido,
            ICUListado<PedidoNoEntregadoDTO> cuListadoPedidosPendientes,
            ICUModificar<PedidoDTO> cuAgregarArticulo
        ) {
            CUListadoArticulos = cuListadoArticulos;
            CUListadoClientes = cuListadoClientes;
            CUALta = cuAlta;
            CUBuscarPorFechaPedido = cuBuscarPorFechaPedido;
            CUAnularPedido = cuAnularPedido;
            CUBuscarPorIdPedido = cuBuscarPorIdPedido;
            CUListadoPedidosPendientes = cuListadoPedidosPendientes;
            CUAgregarArticulo = cuAgregarArticulo;
        }

        [Privado(TipoUsuarios = "Administrador")]
        public IActionResult Index() {
            return View(CUListadoPedidosPendientes.ObtenerListado());
        }
    }
}

//-----

```

```

//----- CREATE -----
//-----
[Privado(TipoUsuarios = "Administrador")]
public ActionResult Create() {
    AltaPedidoViewModel vm = new AltaPedidoViewModel() {
        Articulos = CUListadoArticulos.ObtenerListado(),
        Clientes = CUListadoClientes.ObtenerListado()
    };
    return View(vm);
}

[Privado(TipoUsuarios = "Administrador")]
[HttpPost]
public ActionResult Create(AltaPedidoViewModel vm) {
    try {
        PedidoDTO dto = new PedidoDTO();
        dto.IdCliente = vm.IdCliente;
        dto.TipoPedido = vm.TipoPedido;
        dto.FechaEntrega = vm.FechaEntrega;
        dto.IdArticulo = vm.IdArticulo;
        dto.Cantidad = vm.Cantidad;
        dto.Estado = "Pendiente";

        CUALta.Alta(dto);

        return RedirectToAction("Index", "Pedidos");
    } catch (DatosInvalidosException e) {
        ViewBag.ErrorMessage = e.Message;
    } catch (RegistroNoExisteException e) {
        ViewBag.ErrorMessage = e.Message;
    } catch (NoStockException e) {
        ViewBag.ErrorMessage = e.Message;
    } catch (Exception e) {
        ViewBag.ErrorMessage = e.Message;
    }

    vm.Articulos = CUListadoArticulos.ObtenerListado();
    vm.Clientes = CUListadoClientes.ObtenerListado();
    return View(vm);
}

//----- ANULAR -----
//-----
[Privado(TipoUsuarios = "Administrador")]
public ActionResult AnularPedidos(int id) {
    PedidoDTO p = CUBuscarPorIdPedido.BuscarPorId(id);
    return View(p);
}

[Privado(TipoUsuarios = "Administrador")]
[HttpPost]
public ActionResult AnularPedidos(int id, Pedido p) {

```



```

        try {
            CUAnularPedido.Anular(id);
            return RedirectToAction("Index", "Pedidos");
        } catch (Exception e) {
            ViewBag.ErrorMsg = e.Message;
        }
        return View();
    }

//-----
//----- BUSCAR -----
//-----
[Privado(TipoUsuarios = "Administrador")]
public ActionResult BuscarPedidos() {
    return View();
}

[Privado(TipoUsuarios = "Administrador")]
[HttpPost]
public ActionResult BuscarPedidos(string fecha) {
    try {
        DateOnly fechaABuscar = DateOnly.Parse(fecha);
        List<PedidoNoEntregadoDTO> pedidos =
CUBuscarPorFechaPedido.BuscarPorFechaPedido(fechaABuscar);
        return View(pedidos);
    } catch (RegistroNoExisteException e) {
        ViewBag.ErrorMsg = e.Message;
    } catch (Exception e) {
        ViewBag.ErrorMsg = e.Message;
    }
    return View();
}

//-----
//----- AGREGAR ARTICULO -----
//-----
[Privado(TipoUsuarios = "Administrador")]
public ActionResult AgregarArticulo(int id) {
    PedidoDTO p = CUBuscarPorIdPedido.BuscarPorId(id);
    ViewBag.Articulos = CUListadoArticulos.ObtenerListado();
    return View(p);
}

[Privado(TipoUsuarios = "Administrador")]
[HttpPost]
public ActionResult AgregarArticulo(int id, PedidoDTO p) {
    try {
        CUAgregarArticulo.Modificar(p);
        return RedirectToAction("Index", "Pedidos");
    } catch (RegistroNoExisteException e) {
        ViewBag.ErrorMsg = e.Message;
    } catch (NoStockException e) {
        ViewBag.ErrorMsg = e.Message;
    }
}

```

```

        } catch (Exception e) {
            ViewBag.ErrorMsg = e.ToString();
        }

        ViewBag.Articulos = CUListadoArticulos.ObtenerListado();
        return View(p);
    }
}

using Obligatorio.Models;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using System;
using System.Collections.Generic;
using System.Diagnostics;

namespace Obligatorio.Controllers {
    public class HomeController : Controller {

        public IActionResult Index() {
            return View();
        }

    }
}

using Microsoft.AspNetCore.Mvc;
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.Excepciones;
using DataTransferObjects;

namespace Obligatorio.Controllers {
    public class ClientesController : Controller {
        public ICUListado<ClienteDTO> CUListado { get; set; }
        public ICUBuscarPorRazonSocial CUBuscarPorRazonSocial { get; set; }
        public ICUBuscarClientesPorMontoPedido CUBuscarClientesPorMontoPedido { get; set; }

        public ClientesController(ICUListado<ClienteDTO> cuListado, ICUBuscarPorRazonSocial
cuBuscarPorRazonSocial, ICUBuscarClientesPorMontoPedido cuBuscarClientesPorMontoPedido)
        {
            CUListado = cuListado;
            CUBuscarPorRazonSocial = cuBuscarPorRazonSocial;
            CUBuscarClientesPorMontoPedido = cuBuscarClientesPorMontoPedido;
        }

        public IActionResult Index() {
            return View(CUListado.ObtenerListado());
        }

        public ActionResult BuscarClientes() {
            return View(CUListado.ObtenerListado());
        }
    }
}

```

```

    }

    [HttpPost]
    public ActionResult BuscarClientes(string razonSocial) {
        List<ClienteDTO> clientes = CUBuscarPorRazonSocial.BuscarPorRazonSocial(razonSocial);
        if (clientes.Count == 0) ViewBag.ErrorMsg = "No existen registros";
        return View(clientes);
    }

    public ActionResult BuscarPorMonto() {
        return View(CUListado.ObtenerListado());
    }

    [HttpPost]
    public ActionResult BuscarPorMonto(decimal monto) {
        try {
            List<ClienteDTO> clientes = CUBuscarClientesPorMontoPedido.BuscarPorMontoPedido(monto);
            return View(clientes);
        } catch (RegistroNoExisteException e) {
            ViewBag.ErrorMsg = e.Message;
        } catch (Exception e) {
            ViewBag.ErrorMsg = e.Message;
        }
        return View();
    }
}

}

using LogicaAplicacion.CasosUso;
using LogicaAplicacion.InterfacesCasosUso;
using LogicaNegocio.Dominio;
using LogicaNegocio.Excepciones;
using Microsoft.AspNetCore.Mvc;

namespace Obligatorio.Controllers {
    public class ArticulosController : Controller {
        public ICUAlta<Articulo> CUAlta { get; set; }
        public ICUBaja<Articulo> CUBaja { get; set; }
        public ICUListado<Articulo> CUListado { get; set; }
        public ICUModificar<Articulo> CUModificar { get; set; }
        public ICUBuscarPorId<Articulo> CUBuscarPorIdArticulo { get; set; }

        public ArticulosController(ICUAlta<Articulo> cuAlta, ICUListado<Articulo> cuListado,
            ICUBuscarPorId<Articulo> cuBuscarPorIdArticulo, ICUModificar<Articulo> cuModificar, ICUBaja<Articulo> cuBaja)
        {
            CUAlta = cuAlta;
            CUBaja = cuBaja;
            CUListado = cuListado;
            CUModificar = cuModificar;
            CUBuscarPorIdArticulo = cuBuscarPorIdArticulo;
        }

        public IActionResult Index() {
            return View(CUListado.ObtenerListado());
        }
    }
}

```

```

}

//-----
//----- CREATE -----
//-----

public ActionResult Create() {
    return View();
}

[HttpPost]
public ActionResult Create(Articulo nuevo) {
    try {
        CUALta.Alta(nuevo);
        return RedirectToAction("Index", "Articulos");
    } catch (DatosInvalidosException e) {
        ViewBag.ErrorMsg = e.Message;
    } catch (DuplicadoException e) {
        ViewBag.ErrorMsg = e.Message;
    } catch (Exception e) {
        ViewBag.ErrorMsg = e.Message;
    }

    return View();
}

//-----
//----- UPDATE -----
//-----

public ActionResult Edit(int id) {
    Articulo a = CUBuscarPorIdArticulo.BuscarPorId(id);
    return View(a);
}

[HttpPost]
public ActionResult Edit(int id, Articulo a) {
    try {
        CUModificar.Modificar(a);
        return RedirectToAction("Index", "Articulos");
    } catch (DatosInvalidosException e) {
        ViewBag.ErrorMsg = e.Message;
    } catch (DuplicadoException e) {
        ViewBag.ErrorMsg = e.Message;
    } catch (Exception e) {
        ViewBag.ErrorMsg = e.Message;
    }

    return View();
}

//-----
//----- DELETE -----
//-----

public ActionResult Delete(int id) {

```

```

        Artículo a = CUBuscarPorIdArticulo.BuscarPorId(id);
        return View(a);
    }

    [HttpPost]
    public ActionResult Delete(int id, Artículo a) {
        try {
            CUBaja.Baja(id);
            return RedirectToAction("Index", "Articulos");
        } catch (RegistroNoExisteException e) {
            ViewBag.ErrorMsg = e.Message;
        } catch (Exception e) {
            ViewBag.ErrorMsg = e.Message;
        }
        return View();
    }
}

}

use ObligatorioProg3;
select * from Usuarios
select * from Clientes
select * from Articulos
select * from Lineas
select * from Pedidos

insert into Usuarios(email, nombre, apellido, contraseña, tipo) values ('admin@admin.com', 'Sys',
'Admin','Passw0rd!','Administrador');
insert into Usuarios(email, nombre, apellido, contraseña, tipo) values ('rplanta@lz.com', 'Roberto',
'Planta','Passw0rd!','Estandar');

insert into Clientes(razonSocial, rut, Direccion_Calle_Valor, Direccion_NumeroPuerta_Valor,
Direccion_Ciudad_Valor, DistanciaHastaDeposito) values ('Fleetium Macs', '123456789012','Ciudadela','1180',
'Montevideo', 98);
insert into Clientes(razonSocial, rut, Direccion_Calle_Valor, Direccion_NumeroPuerta_Valor,
Direccion_Ciudad_Valor, DistanciaHastaDeposito) values ('Pink Folks', '987654321098', 'Reconquista', '600',
'Montevideo', 101);
insert into Clientes(razonSocial, rut, Direccion_Calle_Valor, Direccion_NumeroPuerta_Valor,
Direccion_Ciudad_Valor, DistanciaHastaDeposito) values ('Thurston & Kim Co.', '111222333444', 'Durazno',
'902', 'Montevideo', 96);
insert into Clientes(razonSocial, rut, Direccion_Calle_Valor, Direccion_NumeroPuerta_Valor,
Direccion_Ciudad_Valor, DistanciaHastaDeposito) values ('Nueva Helvecia Dolls', '111222333555', 'San Jose',
'1060', 'Montevideo', 94);
insert into Clientes(razonSocial, rut, Direccion_Calle_Valor, Direccion_NumeroPuerta_Valor,
Direccion_Ciudad_Valor, DistanciaHastaDeposito) values ('Fenix', '111222333666', 'Maldonado', '2010',
'Montevideo', 82);
insert into Clientes(razonSocial, rut, Direccion_Calle_Valor, Direccion_NumeroPuerta_Valor,
Direccion_Ciudad_Valor, DistanciaHastaDeposito) values ('AB/CD', '111222333777', 'Canelones', '1110',
'Montevideo', 91);

insert into Articulos(Nombre, Descripcion,CodigoProveedor, Precio, Stock) values ('Calculadora Casio',
'Calculadora de bolsillo Casio', '1234567890123', 300, 10);
insert into Articulos(Nombre, Descripcion,CodigoProveedor, Precio, Stock) values ('Resma A4', '500 hojas
para imprimir en formato A4', '1234567890124', 250, 50);

```

```
insert into Articulos(Nombre, Descripcion,CodigoProveedor, Precio, Stock) values ('Auriculares con
Micrófono', 'Bluetooth o conexión USB, cable desmontable', '1234567890125', 3200, 20);
insert into Articulos(Nombre, Descripcion, CodigoProveedor, Precio, Stock) values ('Silla De Escritorio',
'Soporte lumbar, altura ajustable y base giratoria n ruedas', '1234567890126', 3200, 20);
insert into Articulos(Nombre, Descripcion, CodigoProveedor, Precio, Stock) values ('Notas Adhesivas', 'Block
de notas autoadhesivas', '1234597890124', 50, 100);
insert into Articulos(Nombre, Descripcion, CodigoProveedor, Precio, Stock) values ('Bandas Elásticas', '100
unidades', '1274597890124', 150, 90);
insert into Articulos(Nombre, Descripcion, CodigoProveedor, Precio, Stock) values ('Engrampadora', 'Hasta 100
hojas', '1834597890124', 800, 10);
insert into Articulos(Nombre, Descripcion, CodigoProveedor, Precio, Stock) values ('Cable HDMI', 'HD, FullHD
y 4K,', '1236597890124', 200, 10);
```

```
insert into Parametros(Nombre, Valor) values ('IVA',0.18);
insert into Parametros(Nombre, Valor) values ('RecargoComun_DistanciaMayor100',0.05);
insert into Parametros(Nombre, Valor) values ('RecargoComun_DistanciaMenor100',0);
insert into Parametros(Nombre, Valor) values ('RecargoExpress_Plazo1Dia',0.15);
insert into Parametros(Nombre, Valor) values ('RecargoExpress_PlazoMayor1Dia',0.10);
```

```
select * from __EFMigrationsHistory
DELETE FROM __EFMigrationsHistory WHERE MigrationId='20240516201117_init'
```