



Entrega Obligatoria
Programación III
Prof.: Lucas López



Mateo Martinatto – 311047



Kevin González – 290552

Mateo Martinatto

Kevin González

Fecha de Entrega: 16/05/2024

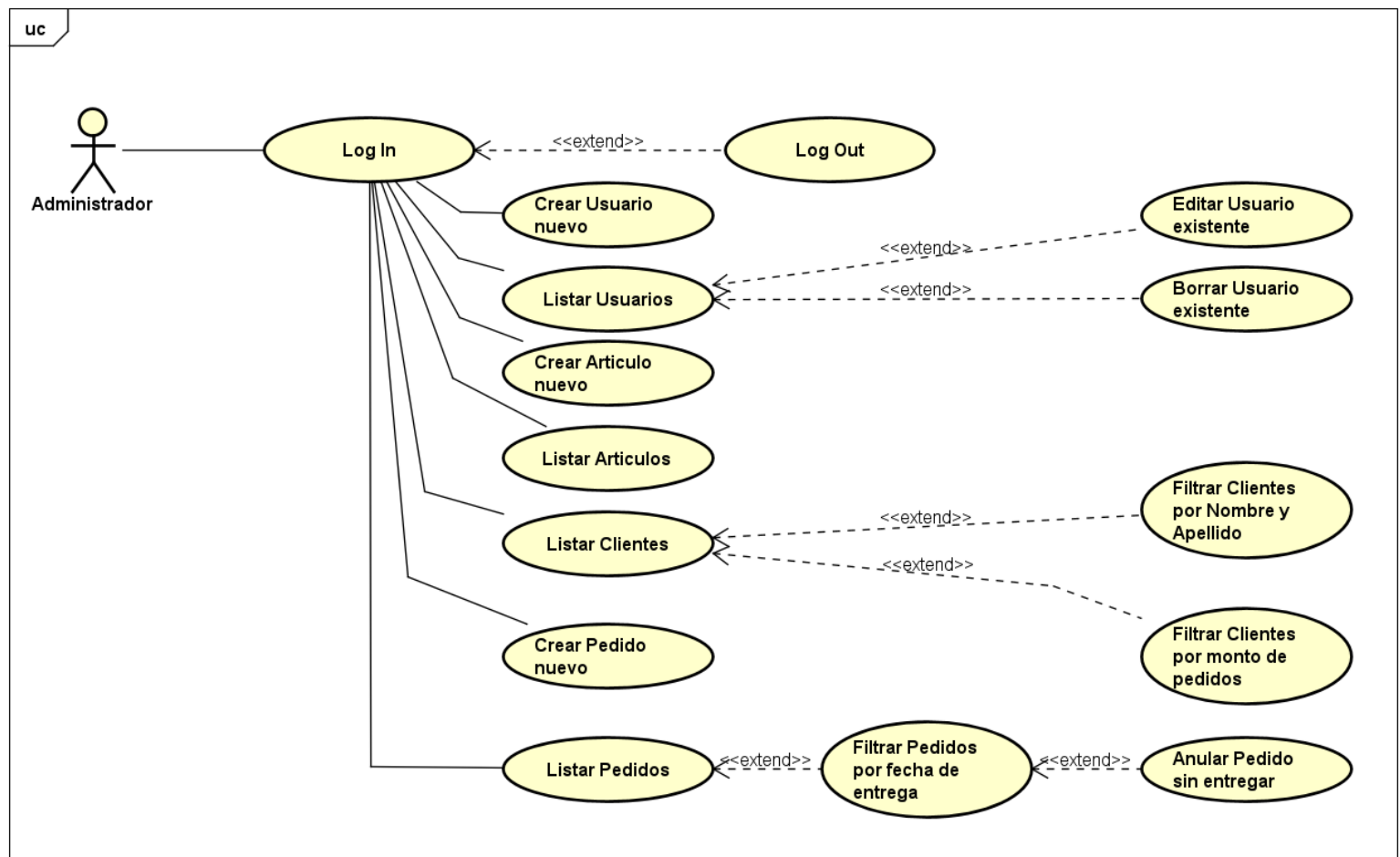
Contenido

Casos de Uso	3
Diagrama de Casos de Uso	3
Casos de Uso Narrativos.....	4
Diagrama de Clases	5
Lógica de Negocio	8
Acceso a Datos y WebApi.....	8
Lógica Aplicación.....	9
WebApp.....	9
Código Fuente.....	10
Acceso a Datos	10
PapeleriaContext	10
RepositorioAdministradorEF	10
RepositorioArticuloEF.....	12
RepositorioClienteEF	13
RepositorioConfiguracionEF	15
RepositorioPedidoEF	16
ApiRest.....	17
ArticulosController	17
PedidosController	18
Program.....	19
LogicaAplicacion.....	20
CasosDeUso.....	20
DTOs	32
Encriptado.....	35
InterfacesCasosDeUso.....	36
Mappers	40
LogicaNegocio	44
Entidades	44
Excepciones	51
InterfacesEntidades	55
InterfacesRepositorio.....	55
ValueObjects.....	57
WebApp.....	58
AdminController.....	58
ArticulosController	60
ClientesController	61
HomeController	63

LoginController	63
PedidosController	64
Program.....	67

Casos de Uso

Diagrama de Casos de Uso



Casos de Uso Narrativos

Identificador: CU-01	Nombre: Alta de Pedido
Autores:	Mateo Martinatto, Kevin González
Fecha:	16/05/2024
Descripción: Permite registrar un nuevo pedido en el sistema.	
Actor: Usuario administrador.	
Precondiciones: <ul style="list-style-type: none">- El usuario debe haber iniciado sesion correctamente en el sistema.- Debe haber clientes registrados en el sistema.- Debe haber artículos registrados en el sistema.- El articulo debe tener stock suficiente.	
Postcondiciones: <ul style="list-style-type: none">- Se agrega el pedido con sus respectivas líneas en la base de datos.	
Flujo normal: <ol style="list-style-type: none">1. El usuario ingresa al sistema2. El usuario hace clic en Agregar Pedido3. El usuario selecciona el cliente al cual el pedido estará relacionado4. El usuario selecciona la fecha en la cual se entregará el pedido5. El usuario chequea si el pedido es exprés o común.6. El usuario selecciona el articulo el cual será pedido7. El usuario ingresa la cantidad de unidades que se están pidiendo.8. El usuario hace clic en “Agregar Linea”9. El sistema hace las correspondientes verificaciones para asegurar que el articulo tiene suficiente stock para la/las unidades pedidas.10. El usuario hace clic en “Crear Pedido”11. El sistema verifica que el pedido esté correcto. Se le asigna un espacio en la base de datos y se muestra en la lista.	
Flujos Alternativos: <ol style="list-style-type: none">1. El usuario intenta entrar sin estar ingresado:<ol style="list-style-type: none">a. El usuario es redirigido a la vista de log in.4. El usuario ingresa una fecha anterior a la fecha actual:<ol style="list-style-type: none">a. El pedido no se guarda.4. El usuario no ingresa una fecha:<ol style="list-style-type: none">b. El pedido no se guarda.7. El usuario no ingresa la cantidad:<ol style="list-style-type: none">a. El pedido no se guarda.	
Flujos excepcionales: <ul style="list-style-type: none">- La comunicación con el servidor se corta durante el registro. El pedido es descartado en su totalidad	

Identificador: CU-02	Nombre: Anulación de Pedido
Autores:	Mateo Martinatto, Kevin González
Fecha:	16/05/2024
Descripción: Permite anular un pedido que no haya sido entregado en el sistema.	
Actor: Usuario administrador.	
Precondiciones: <ul style="list-style-type: none"> - El usuario debe haber iniciado sesión correctamente en el sistema. - Debe haber pedidos registrados en el sistema. 	
Postcondiciones: <ul style="list-style-type: none"> - Se anula el pedido que no fue entregado y no se muestra en el listado de pedidos. 	
Flujo normal: <ol style="list-style-type: none"> 1. El usuario ingresa al sistema. 2. El usuario hace clic en Listar pedidos. 3. El usuario filtra los pedidos según una fecha mayor a la fecha actual. 4. El usuario selecciona un pedido para anularlo 5. El sistema verifica que ese pedido sea válido y que la fecha sea correcta. 6. El sistema anula el pedido y no lo muestra en la lista. 	
Flujos Alternativos: <ol style="list-style-type: none"> 1. El usuario intenta entrar sin estar ingresado: <ol style="list-style-type: none"> a. El usuario es redirigido a la vista de log in. 2. El usuario intenta ingresar la fecha actual o anterior a la actual: <ol style="list-style-type: none"> b. El usuario es redirigido al listado con el mensaje: "Fecha inválida." 	
Flujos excepcionales: -	

Método de Hasheo

Se nos pidió utilizar inteligencia artificial generativa para encontrar un método para encriptar la contraseña del usuario a la hora de registrarlo. Para esto, utilizamos Copilot (Inteligencia Artificial Generativa de Microsoft):

Pregunta: "hi. how do I hash a string in visual studio 2022 with C#?" > "Hola. ¿Cómo encripto un string en Visual Studio 2022 en C#?"

Respuesta:

Certainly! In Visual Studio 2022 using C#, you can hash a string using various algorithms. One common choice is the SHA-256 algorithm. Let me provide you with a simple example of how to hash a string using SHA-256:

```
using System;
```

```
using System.Security.Cryptography;
```

```
using System.Text;
```

```
public class Hash
```

```
{
```

```
    public static string GetHashSha256(string text)
```

```

{
    byte[] bytes = Encoding.UTF8.GetBytes(text); // Convert the string to bytes
    using (SHA256Managed hashString = new SHA256Managed())
    {
        byte[] hash = hashString.ComputeHash(bytes); // Compute the hash
        string hashString = string.Empty;
        foreach (byte x in hash)
        {
            hashString += String.Format("{0:x2}", x); // Convert each byte to hexadecimal
        }
        return hashString;
    }
}
}

```

// Example usage:

```

string input = "Hello, world!";
string hashedResult = Hash.GetHashSha256(input);
Console.WriteLine($"Hashed result: {hashedResult}");

```

In this example:

- We convert the input string to bytes using `Encoding.UTF8.GetBytes`.
- We compute the SHA-256 hash using `SHA256Managed`.
- The resulting hash is represented as a hexadecimal string.

Al utilizar este método, nos dimos cuenta que Visual Studio nos informó que este método estaba obsoleto, por lo que decidimos consultarle de vuelta a Copilot con el objetivo de ver si había otra forma de resolverlo:

The IDE said this: SYSLIB0021 'SHA256Managed' is obsolete: 'Derived cryptographic types are obsolete. Use the Create method on the base type instead.'

Respuesta:

I apologize for the oversight. Thank you for pointing that out! Let's update the code to use the recommended approach. Instead of directly using `SHA256Managed`, we'll use the `SHA256.Create()` method. Here's the revised code snippet:

```

using System;
using System.Security.Cryptography;
using System.Text;

public class Hash
{
    public static string GetHashSha256(string text)
    {
        byte[] bytes = Encoding.UTF8.GetBytes(text); // Convert the string to bytes
        using (SHA256 hashAlgorithm = SHA256.Create()) // Use SHA256.Create() instead
        {
            byte[] hash = hashAlgorithm.ComputeHash(bytes); // Compute the hash
            string hashString = string.Empty;
            foreach (byte x in hash)
            {
                hashString += String.Format("{0:x2}", x); // Convert each byte to hexadecimal
            }
            return hashString;
        }
    }
}

```

```

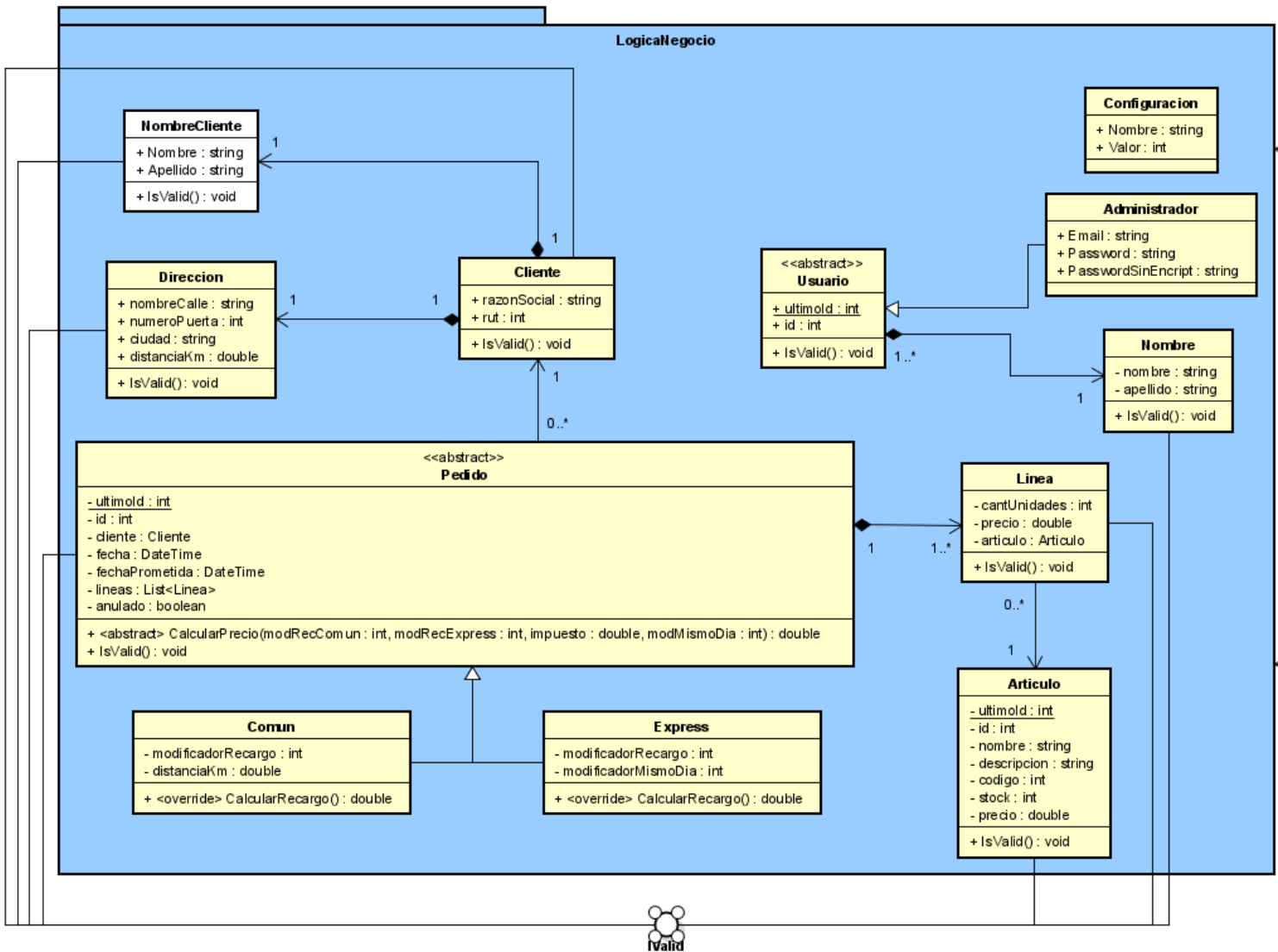
// Example usage:
string input = "Hello, world!";
string hashedResult = Hash.GetHashSha256(input);
Console.WriteLine($"Hashed result: {hashedResult}");

```

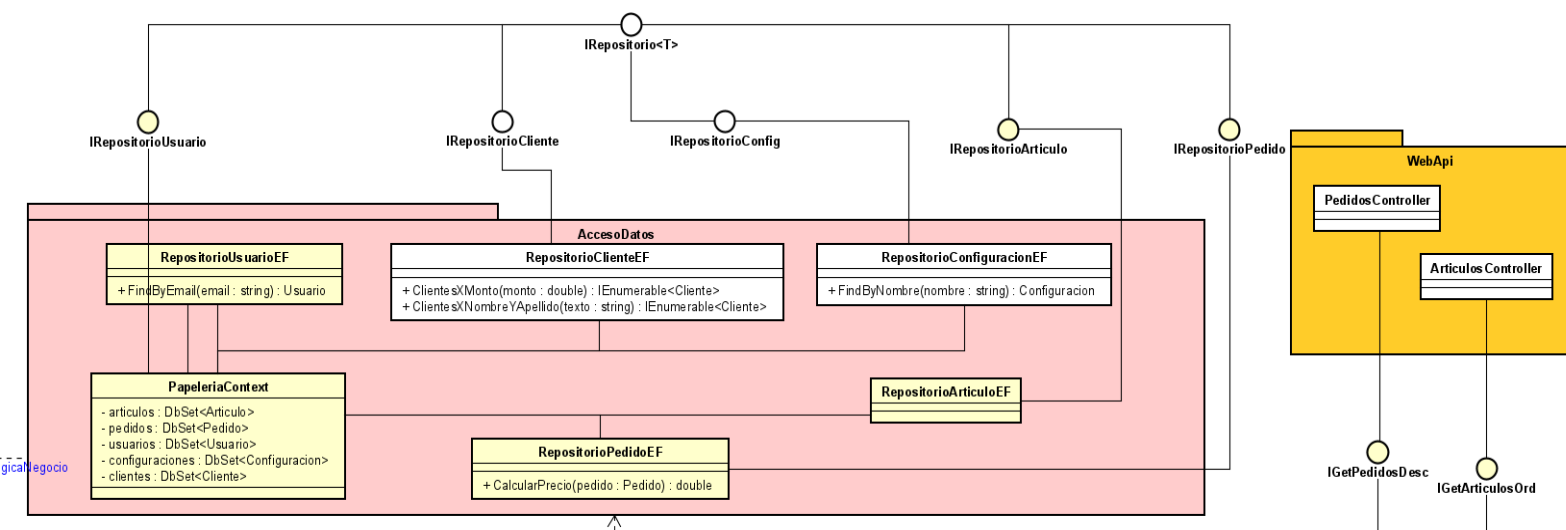
Lo que hicimos con este código fue colocarlo en una clase dentro de la lógica de aplicación (Hash.cs). De esta forma cada vez que necesitemos encriptar los datos, se abrirá una instancia de esa clase y se encriptaran los datos.

Diagrama de Clases

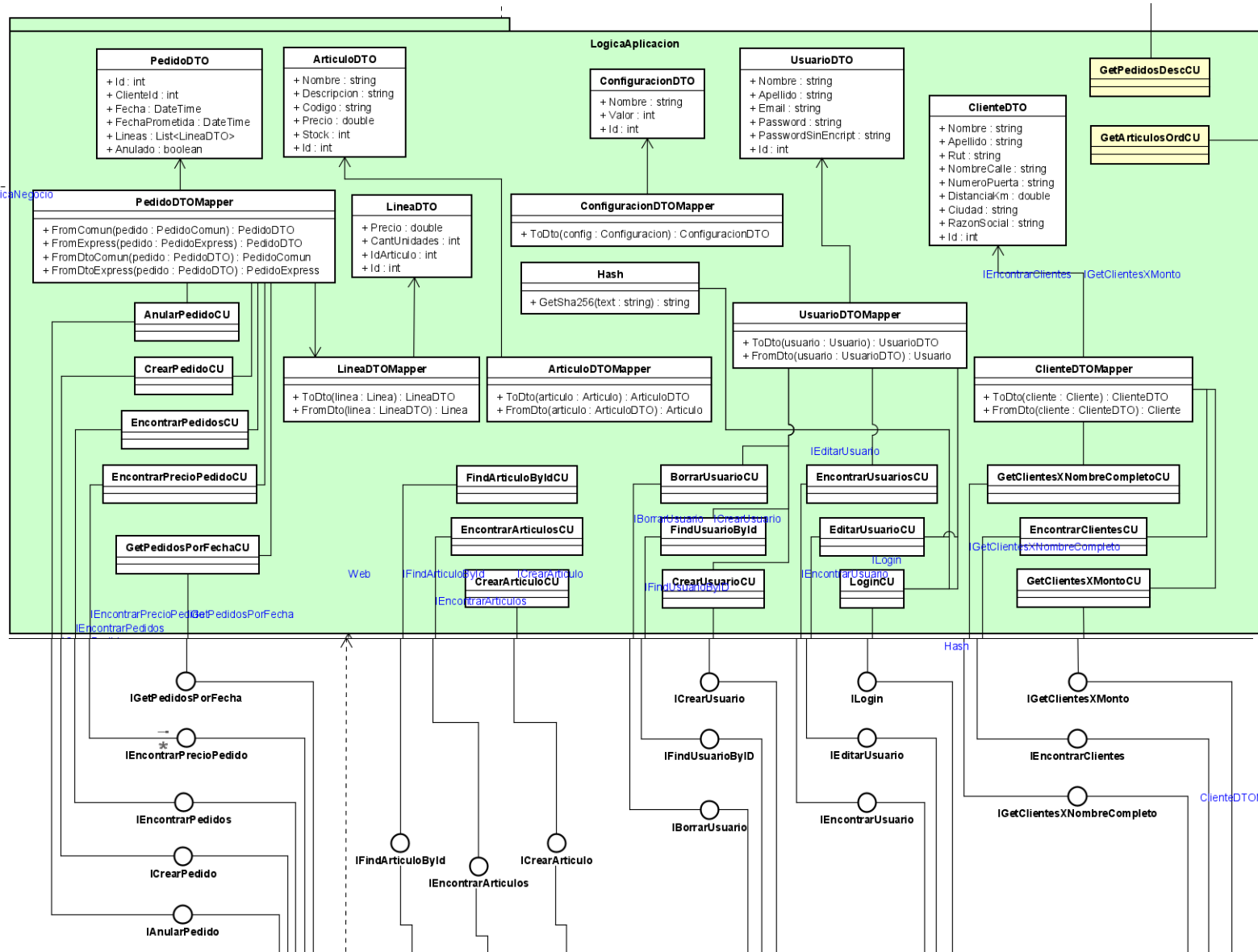
Lógica de Negocio



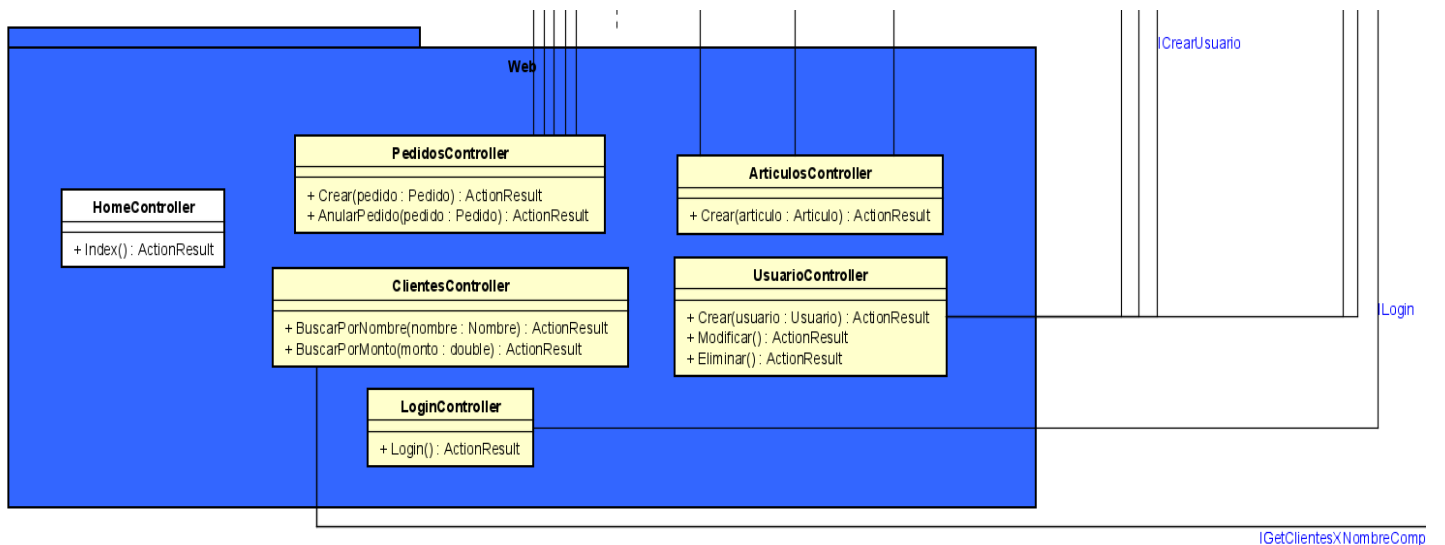
Acceso a Datos y WebApi



Lógica Aplicación



WebApp



Código Fuente

Acceso a Datos

PapeleriaContext

```
using Papeleria.LogicaNegocio;
using LogicaNegocio.Entidades;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection.Emit;
using System.Text;
using System.Threading.Tasks;
using Azure.Core.GeoJson;
using System.Security.Cryptography;

namespace AccesoDatos.EntityFramework
{
    public class PapeleriaContext : DbContext
    {
        public DbSet<Usuario> Usuarios;
        public DbSet<Cliente> Clientes { get; set; }
        public DbSet<Administrador> Admins { get; set; }
        public DbSet<Pedido> Pedidos { get; set; }
        public DbSet<Articulo> Articulos { get; set; }
        public DbSet<Configuracion> Configuraciones { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            optionsBuilder.UseSqlServer(@"SERVER=(localdb)\MsSqlLocalDb;DATABASE=Papeler
iaObligatorio;Integrated Security=true;");
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<PedidoComun>();
            modelBuilder.Entity<PedidoExpress>();
        }
    }
}
```

RepositorioAdministradorEF

```
using LogicaNegocio.Entidades;
using LogicaNegocio.InterfacesRepositorio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection.Metadata.Ecma335;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;

namespace AccesoDatos.EntityFramework
{
    public class RepositorioAdministradorEF : IRepositorioAdministrador
    {
        private PapeleriaContext _context;
        public RepositorioAdministradorEF()
        {
        }
    }
}
```

```

{
    this._context = new PapeleriaContext();
}
public bool Add(Administrador aAgregar)
{
    try
    {
        this._context.Admins.Add(aAgregar);
        this._context.SaveChanges();
        return true;
    }
    catch (Exception ex)
    {
        throw ex;
    }
}

public IEnumerable<Administrador> FindAll()
{
    return this._context.Admins;
}

public Administrador FindByEmail(string email)
{
    try
    {
        return this._context.Admins.Where(admin => admin.Email ==
email).FirstOrDefault();
    }
    catch (Exception ex) { throw ex; }
}

public Administrador FindByID(int id)
{
    return this._context.Admins.Where(admin => admin.Id ==
id).FirstOrDefault();
}

public bool Remove(int id)
{
    try
    {
        Administrador aBorrar = FindByID(id);
        this._context.Admins.Remove(aBorrar);
        this._context.SaveChanges();
        return true;
    }
    catch (Exception ex)
    {
        throw ex;
    }
}

public bool Update(Administrador aModificar)
{
    try
    {
        this._context.Admins.Update(aModificar);
        this._context.SaveChanges();
        return true;
    }
    catch (Exception ex)
    {

```

```

        throw ex;
    }
}
}
}

```

RepositorioArticuloEF

```

using LogicaNegocio.Entidades;
using LogicaNegocio.InterfacesRepositorio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

namespace AccesoDatos.EntityFramework.Repositorios

```

{
    public class RepositorioArticuloEF : IRepositoryArticulo
    {
        private PapeleriaContext _context;
        public RepositorioArticuloEF()
        {
            this._context = new PapeleriaContext();
        }
        public bool Add(Articulo aAgregar)
        {
            try
            {
                this._context.Articulos.Add(aAgregar);
                this._context.SaveChanges();
                return true;
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        public IEnumerable<Articulo> FindAll()
        {
            return this._context.Articulos;
        }

        public Articulo FindByID(int id)
        {
            return this._context.Articulos.Where(articulo => articulo.Id ==
id).FirstOrDefault();
        }

        public bool Remove(int id)
        {
            try
            {
                Articulo aBorrar = FindByID(id);
                this._context.Articulos.Remove(aBorrar);
                this._context.SaveChanges();
                return true;
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }
    }
}

```

```

        }
    }

    public bool Update(Articulo aModificar)
    {
        try
        {
            this._context.Articulos.Update(aModificar);
            this._context.SaveChanges();
            return true;
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }
}

```

RepositorioClienteEF

```

using AccesoDatos.EntityFramework;
using Microsoft.EntityFrameworkCore;
using LogicaNegocio.Entidades;
using Papeleria.LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaNegocio.ValueObjects;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.AccesoDatos.EntityFramework.Repositorios
{
    public class RepositorioClienteEF : IRepositoryCliente
    {
        private PapeleriaContext _context;
        public RepositorioClienteEF()
        {
            this._context = new PapeleriaContext();
        }
        public bool Add(Cliente aAgregar)
        {
            try
            {
                this._context.Clientes.Add(aAgregar);
                this._context.SaveChanges();
                return true;
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        public IEnumerable<Cliente> ClientesXmonto(double monto)
        {
            List<Cliente> clientes = new List<Cliente>();
            int iva = this._context.Configuraciones
                .Where(config => config.Nombre == "IVA")
                .FirstOrDefault().Valor;
            int modRecargoCom = this._context.Configuraciones

```

```

        .Where(config => config.Nombre ==
"ModRecComun")
        .FirstOrDefault().Valor;
        int modRecargoExp = this._context.Configuraciones
        .Where(config => config.Nombre ==
"ModRecExpress")
        .FirstOrDefault().Valor;
        int modMismoDia = this._context.Configuraciones
        .Where(config => config.Nombre ==
"ModMismoDia")
        .FirstOrDefault().Valor;
        IEnumerable<Pedido> pedidos = _context.Pedidos
        .Include(pedido => pedido.Lineas)
        .AsEnumerable()
        .Where(pedido => pedido.CalcularPrecio(iva,
modRecargoCom, modRecargoExp, modMismoDia) >= monto)
        .ToList();

        foreach (Pedido pedido in pedidos)
        {
            if (!clientes.Contains(pedido.ClienteObj))
clientes.Add(pedido.ClienteObj);
        }
        return clientes;
    }

    public IEnumerable<Cliente> ClientesXnombreYapellido(string
especifica)
    {
        return this._context.Clientes.Where(cliente =>
cliente.NombreCliente.Nombre.Contains(especifica)
||
cliente.NombreCliente.Apellido.Contains(especifica)).ToList();
    }

    public IEnumerable<Cliente> FindAll()
    {
        return this._context.Clientes;
    }

    public Cliente FindByID(int id)
    {
        return this._context.Clientes.Where(cliente => cliente.Id ==
id).FirstOrDefault();
    }

    public bool Remove(int id)
    {
        try
        {
            Cliente aBorrar = FindByID(id);
            this._context.Clientes.Remove(aBorrar);
            this._context.SaveChanges();
            return true;
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }

    public bool Update(Cliente aModificar)
    {

```

```

        try
        {
            this._context.Clientes.Update(aModificar);
            this._context.SaveChanges();
            return true;
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }
}
}
}

```

RepositorioConfiguracionEF

```

using AccesoDatos.EntityFramework;
using LogicaNegocio.Entidades;
using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaNegocio.InterfacesRepositorio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.AccesoDatos.EntityFramework.Repositorios
{
    public class RepositorioConfiguracionEF : IRepositoryConfig
    {
        PapeleriaContext _context;
        public RepositorioConfiguracionEF()
        {
            this._context = new PapeleriaContext();
        }

        public bool Add(Configuracion aAgregar)
        {
            throw new NotImplementedException();
        }

        public IEnumerable<Configuracion> FindAll()
        {
            throw new NotImplementedException();
        }

        public Configuracion FindByID(int id)
        {
            throw new NotImplementedException();
        }

        public Configuracion FindByNombre(string nombre)
        {
            return this._context.Configuraciones.Where(config =>
config.Nombre == nombre).FirstOrDefault();
        }

        public bool Remove(int id)
        {
            throw new NotImplementedException();
        }

        public bool Update(Configuracion aModificar)

```

```

        {
            throw new NotImplementedException();
        }
    }
}

```

RepositorioPedidoEF

```

using LogicaNegocio.Entidades;
using LogicaNegocio.InterfacesRepositorio;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AccesoDatos.EntityFramework.Repositorios
{
    public class RepositorioPedidoEF : IRepositoryPedido
    {
        PapeleriaContext _context;
        public RepositorioPedidoEF()
        {
            this._context = new PapeleriaContext();
        }
        public bool Add(Pedido aAgregar)
        {
            try
            {
                Articulo elArticulo;
                foreach (Linea linea in aAgregar.Lineas)
                {
                    elArticulo = this._context.Articulos.Where(articulo =>
articulo.Id == linea.ArticuloId).FirstOrDefault();
                    elArticulo.Stock = elArticulo.Stock -
linea.CantUnidades;
                }
                this._context.Pedidos.Add(aAgregar);
                this._context.SaveChanges();
                return true;
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        public IEnumerable<Pedido> FindAll()
        {
            return this._context.Pedidos.Include(pedido => pedido.Lineas);
        }

        public Pedido FindByID(int id)
        {
            return this._context.Pedidos.Where(pedido => pedido.Id ==
id).FirstOrDefault();
        }

        public bool Remove(int id)
        {
            try

```



```

        {
            Pedido aBorrar = FindByID(id);
            this._context.Pedidos.Remove(aBorrar);
            this._context.SaveChanges();
            return true;
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }

    public bool Update(Pedido aModificar)
    {
        try
        {
            this._context.Pedidos.Update(aModificar);
            this._context.SaveChanges();
            return true;
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }

    public double CalcularPrecio(Pedido pedido)
    {
        return pedido.CalcularPrecio((this._context.Configuraciones.
            Where(config => config.Nombre == "IVA").FirstOrDefault()).Valor,
            (this._context.Configuraciones.
            Where(config => config.Nombre ==
"ModRecComun").FirstOrDefault()).Valor,
            (this._context.Configuraciones.
            Where(config => config.Nombre ==
"ModRecExpress").FirstOrDefault()).Valor,
            (this._context.Configuraciones.
            Where(config => config.Nombre ==
"ModMismoDia").FirstOrDefault()).Valor);
    }
}
}

```

ApiRest

ArticulosController

```

using LogicaNegocio.InterfacesRepositorio;
using Microsoft.AspNetCore.Mvc;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Articulo;
using System.Net;

```

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

```

namespace ApiRest.Controllers
{
    [Route("api/Articulos")]
    [ApiController]
    public class ArticulosController : ControllerBase
    {

```

```

        private IEncontrarArticulosOrd _encontrarArticulosOrd;

        public ArticulosController(IEncontrarArticulosOrd
encontrarArticulosOrd, IGetPedidosDesc getPedidosDesc)
        {
            _encontrarArticulosOrd = encontrarArticulosOrd;
        }

        // GET api/<ArticulosController>
        [HttpGet(Name = "GetAllArticulos")]
        public ActionResult<IEnumerable<ArticuloDTO>> Get()
        {
            try
            {
                IEnumerable<ArticuloDTO> articulos =
_encontrarArticulosOrd.GetArticulosOrd();
                if (articulos == null) return NoContent();
                return Ok(articulos);
            }
            catch (Exception ex)
            {
                return BadRequest(ex.Message);
            }
        }
    }
}

```

PedidosController

```

using Microsoft.AspNetCore.Mvc;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Articulo;

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

namespace ApiRest.Controllers
{
    [Route("api/Pedidos")]
    [ApiController]
    public class PedidosController : ControllerBase
    {
        private IGetPedidosDesc _getPedidosDesc;
        public PedidosController(IGetPedidosDesc getPedidosDesc)
        {
            _getPedidosDesc = getPedidosDesc;
        }

        // GET: api/<PedidosController>
        [HttpGet(Name = "GetAllPedidos")]
        public ActionResult<PedidoDTO> Get()
        {
            try
            {
                IEnumerable<PedidoDTO> pedidos =
_getPedidosDesc.GetPedidosDesc();
                if (pedidos == null) return NoContent();
                return Ok(pedidos);
            }
            catch (Exception ex)
            {

```

```

        return BadRequest(ex.Message);
    }
}
}
}

```

Program

```

using AccesoDatos.EntityFramework.Repositorios;
using LogicaNegocio.InterfacesRepositorio;
using Papeleria.AccesoDatos;
using Papeleria.AccesoDatos.EntityFramework.Repositorios;
using Papeleria.LogicaAplicacion.CasosDeUso.Articulos;
using Papeleria.LogicaAplicacion.CasosDeUso.Pedidos;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Articulo;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Cliente;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Pedido;
using Papeleria.LogicaNegocio.InterfacesRepositorio;
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

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

builder.Services.AddScoped<IRepositorioArticulo, RepositorioArticuloEF>();
builder.Services.AddScoped<IRepositorioPedido, RepositorioPedidoEF>();

builder.Services.AddScoped<IEncontrarArticulosOrd,
EncontrarArticulosOrdCU>();
builder.Services.AddScoped<IGetPedidosDesc, GetPedidosDescCU>();

var app = builder.Build();

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

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

```

LogicaAplicacion

CasosDeUso

BorrarAdminCU

```
using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Administrador;
using Papeleria.LogicaAplicacion.Mappers;
using Papeleria.LogicaNegocio.Excepciones;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Administradores
{
    public class BorrarAdminCU : IBorrarAdmin
    {
        private IRepositoryoAdministrador _repositorioAdministrador;
        public BorrarAdminCU(IRepositoryoAdministrador
repositorioAdministrador)
        {
            this._repositorioAdministrador = repositorioAdministrador;
        }

        public void BorrarAdmin(int id)
        {
            this._repositorioAdministrador.Remove(id);
        }
    }
}
```

CrearAdminCU

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using LogicaNegocio.Entidades;
using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.Mappers;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Administrador;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Administradores
{
    public class CrearAdminCU : ICrearAdmin
    {
        private IRepositoryoAdministrador _repositorioAdmin;
        public CrearAdminCU(IRepositoryoAdministrador repositorioAdmin)
        {
            _repositorioAdmin = repositorioAdmin;
        }
        public void CrearAdmin(AdministradorDTO adminDto)
        {
            Administrador admin = AdministradorDTOMapper.FromDto(adminDto);
            Hash hash = new Hash();
            try
            {
                admin.IsValid();
            }
        }
    }
}
```

```

        admin.PasswordSinEncript = admin.Password;
        admin.Password = hash.GetHashSha256(adminDto.Password);
        this._repositorioAdmin.Add(admin);
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
}
}

```

EditarAdminCU

```

using LogicaNegocio.Entidades;
using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Administrador;
using Papeleria.LogicaAplicacion.Mappers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Administradores
{
    public class EditarAdminCU : IEditarAdmin
    {
        private IRepositoryioAdministrador _repositorioAdmin;
        public EditarAdminCU(IRepositoryioAdministrador repositorioAdmin)
        {
            _repositorioAdmin = repositorioAdmin;
        }
        public bool EditarAdmin(AdministradorDTO aModificar)
        {
            Administrador admin =
            AdministradorDTOMapper.FromDto(aModificar);
            Hash hash = new Hash();
            try
            {
                admin.Password = aModificar.PasswordSinEncript;
                admin.PasswordSinEncript = aModificar.PasswordSinEncript;
                admin.IsValid();
                admin.Password =
                hash.GetHashSha256(aModificar.PasswordSinEncript);
                return this._repositorioAdmin.Update(admin);
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }
    }
}
}

```

EncontrarAdminsCU

```
using LogicaNegocio.Entidades;
using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Administrador;
using Papeleria.LogicaAplicacion.Mappers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Administradores
{
    public class EncontrarAdminsCU : IEncontrarAdmins
    {
        private IRepositoryioAdministrador _repositorioAdministrador;
        public EncontrarAdminsCU(IRepositoryioAdministrador
repositorioAdministrador)
        {
            this._repositorioAdministrador = repositorioAdministrador;
        }
        public IEnumerable<AdministradorDTO> FindAllAdmins()
        {
            IEnumerable<Administrador> administradores =
this._repositorioAdministrador.FindAll();
            List<AdministradorDTO> administradoresDTO = new
List<AdministradorDTO>();
            foreach(Administrador admin in administradores)
            {
                administradoresDTO.Add(AdministradorDTOMapper.ToDto(admin));
            }
            return administradoresDTO;
        }
    }
}
```

FindAdminByIDCU

```
using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Administrador;
using Papeleria.LogicaAplicacion.Mappers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Administradores
{
    public class FindAdminByIdCU : IFindAdminById
    {
        private IRepositoryioAdministrador _repositorioAdmin;
        public FindAdminByIdCU(IRepositoryioAdministrador repositorioAdmin)
        {
            _repositorioAdmin = repositorioAdmin;
        }
        public AdministradorDTO FindAdminById(int id)
        {
            return
AdministradorDTOMapper.ToDto(this._repositorioAdmin.FindByID(id));
        }
    }
}
```

```

    }
}

```

LoginCU

```

using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Administrador;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
using System.Reflection.Metadata;
using Papeleria.LogicaNegocio.Excepciones;
using LogicaNegocio.Entidades;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Administradores
{
    public class LoginCU : ILogin
    {
        private IRepositorioAdministrador _repositorioAdmin;
        public LoginCU(IRepositorioAdministrador repositorioAdmin)
        {
            _repositorioAdmin = repositorioAdmin;
        }
        public bool Login(string email, string password)
        {
            Administrador admin = this._repositorioAdmin.FindByEmail(email);

            if (admin != null)
            {
                Hash hash = new Hash();
                if(hash.GetHashSha256(password) == admin.Password) return
true;

                //Hashea la contraseña.
                //Verifica si es la misma que ya está en la BD
                //Si es así, devuelve true.
                else return false;
            }
            else return false;
        }
    }
}

```

CrearArticuloCU

```

using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Administrador;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
using System.Reflection.Metadata;
using Papeleria.LogicaNegocio.Excepciones;
using LogicaNegocio.Entidades;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Administradores

```

```

{
    public class LoginCU : ILogin
    {
        private IRepositorioAdministrador _repositorioAdmin;
        public LoginCU(IRepositorioAdministrador repositorioAdmin)
        {
            _repositorioAdmin = repositorioAdmin;
        }
        public bool Login(string email, string password)
        {
            Administrador admin = this._repositorioAdmin.FindByEmail(email);

            if (admin != null)
            {
                Hash hash = new Hash();
                if(hash.GetHashSha256(password) == admin.Password) return
true;

                //Hashea la contraseña.
                //Verifica si es la misma que ya está en la BD
                //Si es así, devuelve true.
                else return false;
            }
            else return false;
        }
    }
}

EncontrarArticuloCU
using LogicaNegocio.Entidades;
using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Articulo;
using Papeleria.LogicaAplicacion.Mappers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Articulos
{
    public class EncontrarArticulosCU : IEncontrarArticulos
    {
        private IRepositorioArticulo _repositorioArticulo;
        public EncontrarArticulosCU(IRepositorioArticulo
repositorioArticulo)
        {
            _repositorioArticulo = repositorioArticulo;
        }

        public IEnumerable<ArticuloDTO> EncontrarArticulos()
        {
            IEnumerable<Articulo> articulos =
this._repositorioArticulo.FindAll();
            List<ArticuloDTO> articulosDTO = new List<ArticuloDTO>();
            foreach (Articulo articulo in articulos)
            {
                articulosDTO.Add(ArticuloDTOMapper.ToDto(articulo));
            }
            return articulosDTO;
        }
    }
}

```


EncontrarArticulosOrdCU

```
using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Articulo;
using Papeleria.LogicaAplicacion.Mappers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Articulos
{
    public class EncontrarArticulosOrdCU : IEncontrarArticulosOrd
    {
        private IRepositoryArticulo _repositorioArticulo;
        public EncontrarArticulosOrdCU(IRepositoryArticulo
repositorioArticulo)
        {
            _repositorioArticulo = repositorioArticulo;
        }

        public IEnumerable<ArticuloDTO> GetArticulosOrd()
        {
            return this._repositorioArticulo.FindAll().OrderBy(articulo =>
articulo.Nombre).Select(articulo => ArticuloDTOMapper.ToDto(articulo));
        }
    }
}
```

FindArticuloByIDCU

```
using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Articulo;
using Papeleria.LogicaAplicacion.Mappers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Articulos
{
    public class FindArticuloByIdCU : IEncontrarXIdArticulo
    {
        private IRepositoryArticulo _repositorioArticulo;
        public FindArticuloByIdCU(IRepositoryArticulo repositorioArticulo)
        {
            _repositorioArticulo = repositorioArticulo;
        }

        public ArticuloDTO FindById(int id)
        {
            return
ArticuloDTOMapper.ToDto(_repositorioArticulo.FindById(id));
        }
    }
}
```

EncontrarClientesCU

```
using LogicaNegocio.Entidades;
using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Cliente;
using Papeleria.LogicaAplicacion.Mappers;
using Papeleria.LogicaNegocio.InterfacesRepositorio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Clientes
{
    public class EncontrarClientesCU : IEncontrarClientes
    {
        private IRepositoryCliente _repositorioCliente;
        public EncontrarClientesCU(IRepositoryCliente repositorioCliente)
        {
            this._repositorioCliente = repositorioCliente;
        }
        public IEnumerable<ClienteDTO> FindAllClientes()
        {
            IEnumerable<Cliente> clientes=
this._repositorioCliente.FindAll();
            List<ClienteDTO> clientesDTO= new List<ClienteDTO>();
            foreach (Cliente cliente in clientes)
            {
                clientesDTO.Add(ClienteDTOMapper.ToDto(cliente));
            }
            return clientesDTO;
        }
    }
}
```

GetClientesXmontoCU

```
using LogicaNegocio.Entidades;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Cliente;
using Papeleria.LogicaAplicacion.Mappers;
using Papeleria.LogicaNegocio.InterfacesRepositorio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Clientes
{
    public class GetClientesXmontoCU : IGetClientesXmonto
    {
        private IRepositoryCliente _repositorioCliente;
        public GetClientesXmontoCU(IRepositoryCliente repositorioCliente)
        {
            _repositorioCliente = repositorioCliente;
        }

        public IEnumerable<ClienteDTO> GetClientesXmonto(double monto)
        {
            IEnumerable<Cliente> clientes =
            _repositorioCliente.ClientesXmonto(monto);
            List<ClienteDTO> clientesDTO = new List<ClienteDTO>();
        }
    }
}
```

```

        foreach (Cliente cliente in clientes)
        {
            clientesDTO.Add(ClienteDTOMapper.ToDto(cliente));
        }
        return clientesDTO;
    }
}

GetClientesXNombreYapellidoCU
using LogicaNegocio.Entidades;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Cliente;
using Papeleria.LogicaAplicacion.Mappers;
using Papeleria.LogicaNegocio.InterfacesRepositorio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Clientes
{
    public class GetClientesXnombreYapellidoCU :
    IGetClientesXnombreYapellido
    {
        private IRepositoryCliente _repositorioCliente;
        public GetClientesXnombreYapellidoCU(IRepositoryCliente
repositorioCliente)
        {
            _repositorioCliente = repositorioCliente;
        }
        public IEnumerable<ClienteDTO> GetClientesXnombreYapellido(string
especifica)
        {
            IEnumerable<Cliente> clientes =
_repositorioCliente.ClientesXnombreYapellido(especifica);
            List<ClienteDTO> clientesDTO = new List<ClienteDTO>();
            foreach (Cliente cliente in clientes)
            {
                clientesDTO.Add(ClienteDTOMapper.ToDto(cliente));
            }
            return clientesDTO;
        }
    }
}

AnularPedidoCU
using LogicaNegocio.Entidades;
using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Pedido;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Pedidos
{
    public class AnularPedidoCU : IAnularPedido
    {
        private IRepositoryPedido _repositorioPedido;
        public AnularPedidoCU(IRepositoryPedido repositorioPedido)
        {

```

```

        _repositorioPedido = repositorioPedido;
    }
    public void AnularPedido(int id)
    {
        try
        {
            Pedido elPedido = _repositorioPedido.FindByID(id);
            elPedido.Anulado = true;
            _repositorioPedido.Update(elPedido);
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }
}

}

CrearPedidoCU
using LogicaNegocio.Entidades;
using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Pedido;
using Papeleria.LogicaAplicacion.Mappers;
using Papeleria.LogicaNegocio.Excepciones;
using Papeleria.LogicaNegocio.InterfacesRepositorio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Pedidos
{
    public class CrearPedidoCU : ICrearPedido
    {
        private IRepositoryPedido _repositorioPedido;
        private IRepositoryConfig _repositorioConfig;
        public CrearPedidoCU(IRepositoryPedido repositorioPedido,
            IRepositoryConfig repositorioConfig)
        {
            _repositorioPedido = repositorioPedido;
            _repositorioConfig = repositorioConfig;
        }

        public void CrearPedido(PedidoDTO pedido, Boolean esExpress)
        {
            if (esExpress)
            {
                try
                {
                    Pedido elPedido =
                        PedidoDTOMapper.FromDtoExpress(pedido);
                    elPedido.Fecha = DateTime.Today;

                    elPedido.IsValid(this._repositorioConfig.FindByNombre("PlazoEstipulado").Valor);
                    this._repositorioPedido.Add(elPedido);
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }
        }
    }
}

```

```

        }
        else
        {
            try
            {
                Pedido elPedido = PedidoDTOMapper.FromDtoComun(pedido);
                elPedido.Fecha = DateTime.Today;

                elPedido.IsValid(this._repositorioConfig.FindByNombre("PlazoEstipuladoCom").
                Valor);

                this._repositorioPedido.Add(elPedido);
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }
    }
}

```

EncontrarPedidosCU

```

using LogicaNegocio.Entidades;
using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Pedido;
using Papeleria.LogicaAplicacion.Mappers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Pedidos
{
    public class EncontrarPedidosCU : IEncontrarPedidos
    {
        private IRepositoryPedido _repositorioPedido;
        public EncontrarPedidosCU(IRepositoryPedido repositorioPedido)
        {
            _repositorioPedido = repositorioPedido;
        }

        public IEnumerable<PedidoDTO> EncontrarPedidos()
        {
            IEnumerable<Pedido> pedidos = this._repositorioPedido.FindAll();
            List<PedidoDTO> pedidosDto = new List<PedidoDTO>();
            foreach (Pedido pedido in pedidos)
            {
                if (pedido is PedidoComun)
                pedidosDto.Add(PedidoDTOMapper.FromComun(pedido as PedidoComun));
                else pedidosDto.Add(PedidoDTOMapper.FromExpress(pedido as
                PedidoExpress));
            }
            return pedidosDto;
        }
    }
}

```

EncontrarPrecioPedidoCU

```

using LogicaNegocio.Entidades;
using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Pedido;

```

```

using Papeleria.LogicaAplicacion.Mappers;
using Papeleria.LogicaNegocio.InterfacesRepositorio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Pedidos
{
    public class EncontrarPrecioPedidoCU : IEncontrarPrecioPedido
    {
        private IRepositoryPedido _repositorioPedido;
        private IRepositoryCliente _repositorioCliente;
        public EncontrarPrecioPedidoCU(IRepositoryPedido repositorioPedido,
        IRepositoryCliente repositorioCliente)
        {
            this._repositorioPedido = repositorioPedido;
            this._repositorioCliente = repositorioCliente;
        }
        public double EncontrarPrecioPedido(PedidoDTO pedido, Boolean
esExpress)
        {
            if (pedido == null) return 0;
            if (esExpress)
            {
                PedidoExpress elPedido =
                PedidoDTOMapper.FromDtoExpress(pedido);
                elPedido.ClienteObj =
                _repositorioCliente.FindByID(elPedido.ClienteId);
                return _repositorioPedido.CalcularPrecio(elPedido);
            }
            else
            {
                PedidoComun elPedido = PedidoDTOMapper.FromDtoComun(pedido);
                elPedido.ClienteObj =
                _repositorioCliente.FindByID(elPedido.ClienteId);
                return _repositorioPedido.CalcularPrecio(elPedido);
            }
        }
    }
}

GetPedidosDescCU
using LogicaNegocio.Entidades;
using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Articulo;
using Papeleria.LogicaAplicacion.Mappers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Pedidos
{
    public class GetPedidosDescCU : IGetPedidosDesc
    {
        private IRepositoryPedido _repositorioPedido;
        public GetPedidosDescCU(IRepositoryPedido repositorioPedido)
        {
            _repositorioPedido = repositorioPedido;
        }
    }
}

```

```

        public IEnumerable<PedidoDTO> GetPedidosDesc()
        {
            IEnumerable<Pedido> pedidos =
            _repositorioPedido.FindAll().Where(pedido =>
            pedido.Anulado).OrderByDescending(pedido => pedido.Fecha);
            List<PedidoDTO> pedidosDto = new List<PedidoDTO>();
            foreach (Pedido pedido in pedidos)
            {
                if (pedido is PedidoComun)
                pedidosDto.Add(PedidoDTOMapper.FromComun(pedido as PedidoComun));
                else pedidosDto.Add(PedidoDTOMapper.FromExpress(pedido as
                PedidoExpress));
            }
            return pedidosDto;
        }
    }
}

GetPedidosPorFechaCU
using LogicaNegocio.Entidades;
using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Pedido;
using Papeleria.LogicaAplicacion.Mappers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.CasosDeUso.Pedidos
{
    public class GetPedidosPorFechaCU : IGetPedidosPorFecha
    {
        private IRepositoryPedido _repositorioPedido;
        public GetPedidosPorFechaCU(IRepositoryPedido repositorioPedido)
        {
            this._repositorioPedido = repositorioPedido;
        }

        public IEnumerable<PedidoDTO> GetPedidosPorFecha(DateTime fecha)
        {
            IEnumerable<Pedido> pedidos =
            _repositorioPedido.FindAll().Where(pedido => pedido.FechaPrometida.Date ==
            fecha.Date);
            List<PedidoDTO> pedidosDto = new List<PedidoDTO>();
            foreach (Pedido pedido in pedidos)
            {
                if (pedido is PedidoComun)
                pedidosDto.Add(PedidoDTOMapper.FromComun(pedido as PedidoComun));
                else pedidosDto.Add(PedidoDTOMapper.FromExpress(pedido as
                PedidoExpress));
            }
            return pedidosDto;
        }
    }
}

```

DTOs

AdministradorDTO

```
using LogicaNegocio.Entidades;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.DTOs
{
    [Index(nameof(Email), IsUnique = true)]

    public class AdministradorDTO : UsuarioDTO
    {
        [Required(ErrorMessage = "Requerido.")]
        [EmailAddress]
        public string Email { get; set; }
        [Required(ErrorMessage = "Requerido.")]
        [RegularExpression(@"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[.,;!])[a-zA-Z\d.,;!]{6,}$",
            ErrorMessage = "La contraseña debe tener al menos una letra en minúscula, mayúscula, un número, y uno de estos símbolos: . ; , ! ")
        ]
        [StringLength(200, MinimumLength = 6, ErrorMessage = "La Contraseña debe ser de al menos 6 caracteres")]
        public string Password { get; set; }
        public string PasswordSinEncript { get; set; }
        public AdministradorDTO() { }
        public AdministradorDTO(Administrador admin)
        {
            if (admin != null)
            {
                this.Id = admin.Id;
                this.Email = admin.Email;
                this.Password = admin.Password;
                this.PasswordSinEncript = admin.PasswordSinEncript;
                this.Nombre = admin.NombreCompleto.Nombre;
                this.Apellido = admin.NombreCompleto.Apellido;
            }
        }
    }
}
```

ArticuloDTO

```
using LogicaNegocio.Entidades;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.DTOs
{
    public class ArticuloDTO
    {
        public int Id { get; set; }
        public static int UltimoId { get; set; }
        public string Nombre { get; set; }
        public stringCodigo { get; set; }
        public string Descripcion { get; set; }
        public double Precio { get; set; }
    }
}
```



```

        public int Stock { get; set; }
        public ArticuloDTO() { }
        public ArticuloDTO(Articulo articulo)
        {
            if(articulo != null)
            {
                Id = articulo.Id;
                Descripcion = articulo.Descripcion;
                Precio = articulo.Precio;
                Codigo = articulo.Codigo;
                Nombre = articulo.Nombre;
                Stock = articulo.Stock;
            }
        }
    }
}

ClienteDTO
using LogicaNegocio.Entidades;
using LogicaNegocio.ValueObjects;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.DTOs
{
    public class ClienteDTO
    {
        public int Id;
        public string Rut;
        public string RazonSocial;
        public string Nombre;
        public string Apellido;
        public string nombreCalle;
        public string numeroPuerta;
        public string ciudad;
        public double distanciaKm;
        public ClienteDTO() { }
        public ClienteDTO(Cliente cliente)
        {
            if (cliente != null)
            {
                this.Id = cliente.Id;
                this.RazonSocial = cliente.RazonSocial;
                this.Rut = cliente.Rut;
                this.Nombre = cliente.NombreCliente.Nombre;
                this.Apellido = cliente.NombreCliente.Apellido;
                this.nombreCalle = cliente.Direccion.NombreCalle;
                this.numeroPuerta = cliente.Direccion.NumeroPuerta;
                this.ciudad = cliente.Direccion.Ciudad;
                this.distanciaKm = cliente.Direccion.DistanciaKm;
            }
        }
    }
}

ConfiguracionDTO
using LogicaNegocio.Entidades;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

```

```

using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.DTOs
{
    public class ConfiguracionDTO
    {
        public int Id { get; set; }
        public string Nombre { get; set; }
        public int Valor { get; set; }
        public ConfiguracionDTO(Configuracion config)
        {
            Id = config.Id;
            Nombre = config.Nombre;
            Valor = config.Valor;
        }
    }
}

LineaDTO
using LogicaNegocio.Entidades;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.DTOs
{
    public class LineaDTO
    {
        public int Id { get; set; }
        public double Precio { get; set; }
        public int CantUnidades { get; set; }
        public int ArtículoId { get; set; }
    }
}

PedidoDTO
using LogicaNegocio.Entidades;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.DTOs
{
    public class PedidoDTO
    {
        public static int UltimoId { get; set; }
        public int Id { get; set; }
        public int ClienteId { get; set; }
        public DateTime Fecha { get; set; }
        public DateTime FechaPrometida { get; set; }
        public List<LineaDTO> Lineas { get; set; }
        public Boolean Anulado { get; set; }
    }
}

UsuarioDTO
using LogicaNegocio.Entidades;
using LogicaNegocio.ValueObjects;
using System;

```

```

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

namespace Papeleria.LogicaAplicacion.DTOs
{
    public abstract class UsuarioDTO
    {
        public int Id { get; set; }
        public static int UltimoId { get; set; }
        [Required(ErrorMessage = "Requerido.")]
        [RegularExpression(@"^[a-zA-Z][-a-zA-Z ']*(?![-])$", ErrorMessage
= "El nombre o apellido es inválido.")]
        public string Nombre { get; set; }
        [Required(ErrorMessage = "Requerido.")]
        [RegularExpression(@"^[a-zA-Z][-a-zA-Z ']*(?![-])$", ErrorMessage
= "El nombre o apellido es inválido.")]

        public string Apellido { get; set; }
        public UsuarioDTO(){}
        public UsuarioDTO(Usuario usuario)
        {
            if (usuario != null)
            {
                this.Id = usuario.Id;
                this.Nombre = usuario.NombreCompleto.Nombre;
                this.Apellido = usuario.NombreCompleto.Apellido;
            }
        }
    }
}

```

Encriptado

Hash

```

using System;
using System.Security.Cryptography;
using System.Text;

public class Hash
{
    public Hash() { }
    public string GetHashSha256(string text)
    {
        byte[] bytes = Encoding.UTF8.GetBytes(text);
        using (SHA256 hashAlgorithm = SHA256.Create())
        {
            byte[] hash = hashAlgorithm.ComputeHash(bytes);
            string hashString = string.Empty;
            foreach (byte x in hash)
            {
                hashString += String.Format("{0:x2}", x);
            }
            return hashString;
        }
    }
}

```

InterfacesCasosDeUso

IBorrarAdmin

```
using Papeleria.LogicaAplicacion.DTOs;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Administrador
{
    public interface IBorrarAdmin
    {
        void BorrarAdmin(int id);
    }
}
```

ICrearAdmin

```
using Papeleria.LogicaAplicacion.DTOs;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Administrador
{
    public interface ICrearAdmin
    {
        void CrearAdmin(AdministradorDTO aCrear);
    }
}
```

IEditarAdmin

```
using Papeleria.LogicaAplicacion.DTOs;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Administrador
{
    public interface IEditarAdmin
    {
        bool EditarAdmin(AdministradorDTO aModificar);
    }
}
```

IEncontrarAdmin

```
using LogicaNegocio.Entidades;
using Papeleria.LogicaAplicacion.DTOs;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Administrador
{
    public interface IEncontrarAdmins
    {
        IEnumerable<AdministradorDTO> FindAllAdmins();
    }
}
```

```

}
IFindAdminById
using Papeleria.LogicaAplicacion.DTOs;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Administrador
{
    public interface IFindAdminById
    {
        AdministradorDTO FindAdminById(int id);
    }
}

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

namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Administrador
{
    public interface ILogin
    {
        bool Login(string email, string password);
    }
}

ICrearArticulo
using Papeleria.LogicaAplicacion.DTOs;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Articulo
{
    public interface ICrearArticulo
    {
        void CrearArticulo(ArticuloDTO aCrear);
    }
}

IEncontrarArticulos
using Papeleria.LogicaAplicacion.DTOs;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Articulo
{
    public interface IEncontrarArticulos
    {
        IEnumerable<ArticuloDTO> EncontrarArticulos();
    }
}

```

```

IEncontrarArticulosOrd
using Papeleria.LogicaAplicacion.DTOs;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Articulo
{
    public interface IEncontrarArticulosOrd
    {
        IEnumerable<ArticuloDTO> GetArticulosOrd();
    }
}

IEncontrarXIdArticulo
using Papeleria.LogicaAplicacion.DTOs;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Articulo
{
    public interface IEncontrarXIdArticulo
    {
        ArticuloDTO FindById(int id);
    }
}

IEncontrarClientes
using Papeleria.LogicaAplicacion.DTOs;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Cliente
{
    public interface IEncontrarClientes
    {
        IEnumerable<ClienteDTO> FindAllClientes();
    }
}

IgetClientesXMonto
using Papeleria.LogicaAplicacion.DTOs;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Cliente
{
    public interface IGetClientesXmonto
    {
        IEnumerable<ClienteDTO> GetClientesXmonto(double monto);
    }
}

```

```

IgetClientesXNombreYApellido
using Papeleria.LogicaAplicacion.DTOs;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Cliente
{
    public interface IGetClientesXnombreYapellido
    {
        IEnumerable<ClienteDTO> GetClientesXnombreYapellido(string
especifica);
    }
}

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

namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Pedido
{
    public interface IAnularPedido
    {
        void AnularPedido(int id);
    }
}

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

namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Pedido
{
    public interface IAnularPedido
    {
        void AnularPedido(int id);
    }
}

IEncontrarPedidos
using Papeleria.LogicaAplicacion.DTOs;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Pedido
{
    public interface IEncontrarPedidos
    {
        IEnumerable<PedidoDTO> EncontrarPedidos();
    }
}

IEncontrarPrecioPedido
using Papeleria.LogicaAplicacion.DTOs;

```

```

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

namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Pedido
{
    public interface IEncontrarPrecioPedido
    {
        double EncontrarPrecioPedido(PedidoDTO pedido, Boolean esExpress);
    }
}

IGetPedidosDesc
using Papeleria.LogicaAplicacion.DTOs;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Articulo
{
    public interface IGetPedidosDesc
    {
        IEnumerable<PedidoDTO> GetPedidosDesc();
    }
}

```

IGetPedidosPorFecha

```

using Papeleria.LogicaAplicacion.DTOs;

```

```

namespace Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Pedido
{
    public interface IGetPedidosPorFecha
    {
        IEnumerable<PedidoDTO> GetPedidosPorFecha(DateTime fecha);
    }
}

```

Mappers

AdministradorDTOMapper

```

using Papeleria.LogicaAplicacion.DTOs;
using LogicaNegocio.Entidades;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace Papeleria.LogicaAplicacion.Mappers
{
    public class AdministradorDTOMapper
    {
        public static AdministradorDTO ToDto(Administrador admin)
        {
            return new AdministradorDTO(admin);
        }

        public static Administrador FromDto(AdministradorDTO adminDto)
        {
            if(adminDto == null)
            {

```



```

        throw new NotImplementedException();
    }
    return new Administrador(admindto.Id, admindto.Nombre,
admindto.Apellido, admindto.Email, admindto.Password);
    }
}

ArticuloDTOMapper
using LogicaNegocio.Entidades;
using Papeleria.LogicaAplicacion.DTOs;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.Mappers
{
    public class ArticuloDTOMapper
    {
        public static ArticuloDTO ToDto(Articulo articulo)
        {
            return new ArticuloDTO(articulo);
        }

        public static Articulo FromDto(ArticuloDTO articuloDTO)
        {
            if (articuloDTO == null)
            {
                throw new NotImplementedException();
            }
            return new Articulo(articuloDTO.Id, articuloDTO.Nombre,
articuloDTO.Codigo, articuloDTO.Descripcion, articuloDTO.Precio,
articuloDTO.Stock);
        }
    }
}

ClienteDTOMapper
using LogicaNegocio.Entidades;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaNegocio.Excepciones;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.Mappers
{
    public class ClienteDTOMapper
    {
        public static ClienteDTO ToDto(Cliente cliente)
        {
            return new ClienteDTO(cliente);
        }

        public static Cliente FromDto(ClienteDTO clienteDTO)
        {
            if (clienteDTO == null)
            {
                throw new ClienteNoValidoException("Cliente invalido.");
            }
        }
    }
}

```

```

        return new Cliente(clienteDTO.Id, clienteDTO.RazonSocial,
            clienteDTO.Rut, clienteDTO.Nombre, clienteDTO.Apellido,
            clienteDTO.nombreCalle, clienteDTO.numeroPuerta,
            clienteDTO.ciudad, clienteDTO.distanciaKm);
    }
}

```

```

}

```

ConfiguracionDTOMapper

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using LogicaNegocio.Entidades;
using Papeleria.LogicaAplicacion.DTOs;

```

namespace Papeleria.LogicaAplicacion.Mappers

```

{
    public class ConfiguracionDTOMapper
    {
        public ConfiguracionDTO ToDto(Configuracion config)
        {
            return new ConfiguracionDTO(config);
        }
    }
}

```

LineaDTOMapper

```

using LogicaNegocio.Entidades;
using Papeleria.LogicaAplicacion.DTOs;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

namespace Papeleria.LogicaAplicacion.Mappers

```

{
    public class LineaDTOMapper
    {
        public static LineaDTO toDTO(Linea linea)
        {
            return new LineaDTO
            {
                Id = linea.Id,
                Precio = linea.Precio,
                CantUnidades = linea.CantUnidades,
                ArticuloId = linea.ArticuloId
            };
        }

        public static Linea FromDTO(LineaDTO linea)
        {
            return new Linea
            {
                Id = linea.Id,
                Precio = linea.Precio,
                CantUnidades = linea.CantUnidades,
                ArticuloId = linea.ArticuloId,
            };
        }
    }
}

```

PedidoDTOMapper

```
using LogicaNegocio.Entidades;
using Papeleria.LogicaAplicacion.DTOs;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaAplicacion.Mappers
{
    public class PedidoDTOMapper
    {
        public static PedidoDTO FromComun(PedidoComun pedido)
        {
            if (pedido == null) return null;
            return new PedidoDTO
            {
                Id = pedido.Id,
                ClienteId = pedido.ClienteId,
                Fecha = pedido.Fecha,
                FechaPrometida = pedido.FechaPrometida,
                Lineas = pedido.Lineas.Select(linea =>
LineaDTOMapper.toDTO(linea)).ToList(),
                Anulado = pedido.Anulado,
            };
        }
        public static PedidoDTO FromExpress(PedidoExpress pedido)
        {
            if (pedido == null) return null;
            return new PedidoDTO
            {
                Id = pedido.Id,
                ClienteId = pedido.ClienteId,
                Fecha = pedido.Fecha,
                FechaPrometida = pedido.FechaPrometida,
                Lineas = pedido.Lineas.Select(linea =>
LineaDTOMapper.toDTO(linea)).ToList(),
                Anulado = pedido.Anulado,
            };
        }
        public static PedidoComun FromDtoComun(PedidoDTO pedido)
        {
            if (pedido != null)
            {
                return new PedidoComun
                {
                    Id = pedido.Id,
                    ClienteId = pedido.ClienteId,
                    Fecha = pedido.Fecha,
                    FechaPrometida = pedido.FechaPrometida,
                    Lineas = pedido.Lineas.Select(linea =>
LineaDTOMapper.FromDTO(linea)).ToList(),
                    Anulado = pedido.Anulado,
                };
            }
            else throw new NotImplementedException();
        }
        public static PedidoExpress FromDtoExpress(PedidoDTO pedido)
        {
            if (pedido != null)
            {
                return new PedidoExpress
```

```

        {
            Id = pedido.Id,
            ClienteId = pedido.ClienteId,
            Fecha = pedido.Fecha,
            FechaPrometida = pedido.FechaPrometida,
            Lineas = pedido.Lineas.Select(linea =>
                LineaDTOMapper.FromDTO(linea)).ToList(),
            Anulado = pedido.Anulado,
        };
    }
    else throw new NotImplementedException();
}
}
}

```

LogicaNegocio

Entidades

Administrador

```

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

namespace LogicaNegocio.Entidades
{
    public class Administrador : Usuario
    {
        public Administrador() { }
        public Administrador(int id, string nombre, string apellido, string email, string password)
        {
            Id = id;
            NombreCompleto = new NombreCompleto(nombre, apellido);
            this.Email = email;
            this.Password = password;
        }
        public Administrador(string nombre, string apellido, string email, string password) {
            NombreCompleto = new NombreCompleto(nombre, apellido);
            this.Email = email;
            this.Password = password;
        }
    }
}

```

Articulo

```

using Microsoft.EntityFrameworkCore;
using Papeleria.LogicaNegocio.Excepciones;
using Papeleria.LogicaNegocio.InterfacesEntidades;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.Entidades
{
    [Index(nameof(Nombre), IsUnique = true)]
    public class Articulo : IValid
    {
        public int Id { get; set; }
    }
}

```

```

        public static int UltimoId { get; set; }

        public string Nombre { get; set; }
        public stringCodigo { get; set; }
        public string Descripcion { get; set; }
        public double Precio { get; set; }
        public int Stock { get; set; }
        public Articulo() { }
        public Articulo(int id, string nombre, string codigo, string
descripcion, double precio, int stock)
        {
            Id = id;
            Nombre = nombre;
            Codigo = codigo;
            Descripcion = descripcion;
            Precio = precio;
            Stock = stock;
        }
        public Articulo( string nombre, string codigo, string descripcion,
double precio, int stock)
        {
            Nombre = nombre;
            Codigo = codigo;
            Descripcion = descripcion;
            Precio = precio;
            Stock = stock;
        }
        public void IsValid()
        {
            if (Nombre == "" && Nombre.Length < 10 || Nombre.Length > 200)
throw new ArticuloNoValidoException("Nombre del articulo invalido.");
            if (Codigo == "") throw new ArticuloNoValidoException("Codigo
invalido.");
            if (Descripcion == "" && Descripcion.Length < 5) throw new
ArticuloNoValidoException("Descripcion no valida");
            if (Precio <= 0) throw new ArticuloNoValidoException("Precio no
valido");
            if (Stock <= 0) throw new ArticuloNoValidoException("Stock no
valido.");
        }
    }
}

```

Cliente

```

using LogicaNegocio.ValueObjects;
using Papeleria.LogicaNegocio.Excepciones;
using Papeleria.LogicaNegocio.InterfacesEntidades;
using Papeleria.LogicaNegocio.ValueObjects;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.Entidades
{
    public class Cliente : IValid
    {
        public int Id { get; set; }
        public static int UltimoId { get; set; }
        public string RazonSocial { get; set; }
        public string Rut { get; set; }
        public NombreCliente NombreCliente { get; set; }
        public Direccion Direccion { get; set; }
    }
}

```

```

        public Cliente() { }
        public Cliente(int id, string razonSocial, string rut, string
nombre, string apellido, string nombreCalle, string numeroPuerta, string
ciudad, double distanciaKm)
        {
            Id = id;
            RazonSocial = razonSocial;
            Rut = rut;
            NombreCliente = new NombreCliente(nombre, apellido);
            Direccion = new Direccion(nombreCalle, numeroPuerta, ciudad,
distanciaKm);
        }
        public Cliente(string razonSocial, string rut, string nombre, string
apellido, string nombreCalle, string numeroPuerta, string ciudad, double
distanciaKm)
        {
            RazonSocial = razonSocial;
            Rut = rut;
            NombreCliente = new NombreCliente(nombre, apellido);
            Direccion = new Direccion(nombreCalle, numeroPuerta, ciudad,
distanciaKm);
        }

        public void IsValid()
        {
            if (RazonSocial == null) throw new
ClienteNoValidoException("Razon Social no valida");
            if (Rut.Length != 12 || !int.TryParse(Rut, out int x)) throw new
ClienteNoValidoException("RUT no valido. Asegurese de que sea de 12 digitos
y no tenga letras.");
            NombreCliente.IsValid();
            Direccion.IsValid();
        }
    }
}

```

Configuracion

```

using Papeleria.LogicaNegocio.InterfacesEntidades;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace LogicaNegocio.Entidades

```

```

{
    public class Configuracion
    {
        public int Id { get; set; }
        public string Nombre { get; set; }
        public int Valor { get; set; }
        public Configuracion() { }
        public Configuracion(int id, string nombre, int valor)
        {
            Id = id;
            Nombre = nombre;
            Valor = valor;
        }
        public Configuracion(string nombre, int valor)
        {
            Nombre = nombre;
            Valor = valor;
        }
    }
}

```

```

    }
}

Linea
using Papeleria.LogicaNegocio.Excepciones;
using Papeleria.LogicaNegocio.InterfacesEntidades;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.Entidades
{
    public class Linea : IValid
    {
        public int Id { get; set; }
        public double Precio { get; set; }
        public int CantUnidades { get; set; }
        [ForeignKey(nameof(Articulo))] public int ArticuloId { get; set; }
        public Articulo ArticuloObj { get; set; }

        public Linea() { }
        public Linea(int id, int stock, Articulo articuloObj, int
cantUnidades)
        {
            Id = id;
            CantUnidades = cantUnidades;
            ArticuloObj = articuloObj;
            Precio = articuloObj.Precio * cantUnidades;
        }
        public Linea(double precio, int stock, Articulo articuloObj, int
cantUnidades)
        {
            Precio = precio;
            CantUnidades = cantUnidades;
            ArticuloObj = articuloObj;
        }

        public void IsValid()
        {
            if (ArticuloObj == null) throw new
LineaNoValidaException("Articulo no es valido.");
            if (CantUnidades > ArticuloObj.Stock) throw new
LineaNoValidaException("Cantidad de unidades supera el stock.");
        }
    }
}

Pedido
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.Entidades
{
    public abstract class Pedido
    {
        public static int UltimoId { get; set; }
        public int Id { get; set; }
    }
}

```

```

[ForeignKey(nameof(Cliente))] public int ClienteId { get; set; }
public Cliente ClienteObj { get; set; }
public DateTime Fecha { get; set; }
public DateTime FechaPrometida { get; set; }
public List<Linea> Lineas { get; set; }
public Boolean Anulado { get; set; }
public Pedido() { }

public abstract double CalcularPrecio(double impuesto, int
modificadorRecargo, int modificadorRecargoExp, int modificadorMismoDia);
public abstract void IsValid(int fechaPrometida);
}
}

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

namespace LogicaNegocio.Entidades
{
    public class PedidoComun : Pedido
    {
        public static int ModificadorRecargo { get; set; }
        public PedidoComun(){}
        public PedidoComun(int id, Cliente obj, DateTime fechaPrometida)
        {
            Id = id;
            ClienteObj = obj;
            FechaPrometida = fechaPrometida;
            Lineas = new List<Linea>();
            Anulado = false;
        }

        public PedidoComun(Cliente obj, DateTime fechaPrometida)
        {
            ClienteObj = obj;
            FechaPrometida = fechaPrometida;
            Lineas = new List<Linea>();
            Anulado = false;
        }

        public override double CalcularPrecio(double impuesto, int
modificadorRecargo, int modificadorRecargoExp, int modificadorMismoDia)
        {
            {
                double suma = 0;
                foreach (Linea linea in Lineas)
                {
                    suma += linea.Precio;
                }
                if (ClienteObj.Direccion.DistanciaKm > 100) suma += suma *
modificadorRecargo / 100;
                suma += suma * impuesto / 100;
                return suma;
            }
        }

        public override void IsValid(int fechaPrometida)
        {
            if (this.FechaPrometida.Day - DateTime.Today.Day <
fechaPrometida) throw new PedidoNoValidoException("Fecha no valida.");

```



```

        if (this.FechaPrometida < DateTime.Today) throw new
PedidoNoValidoException("Fecha no valida.");
        if (Lineas.Count == 0) throw new PedidoNoValidoException("Debe
tener al menos una linea.");
    }

}

}

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

namespace LogicaNegocio.Entidades
{
    public class PedidoExpress : Pedido
    {
        public static int ModificadorRecargo { get; set; }
        public static int ModificadorMismoDia { get; set; }
        public PedidoExpress() { }

        public PedidoExpress(int id, Cliente obj, DateTime fechaPrometida)
        {
            Id = id;
            ClienteObj = obj;
            FechaPrometida = fechaPrometida;
            Lineas = new List<Linea>();
            Anulado = false;
        }

        public PedidoExpress(Cliente obj, DateTime fechaPrometida)
        {
            ClienteObj = obj;
            FechaPrometida = fechaPrometida;
            Lineas = new List<Linea>();
            Anulado = false;
        }

        public override double CalcularPrecio(double impuesto, int
modificadorRecargo, int modificadorRecargoExp, int modificadorMismoDia)
        {
            {
                double suma = 0;
                foreach (Linea linea in Lineas)
                {
                    suma += linea.Precio;
                }
                if (FechaPrometida.Day == DateTime.Today.Day) suma += suma *
modificadorMismoDia / 100;
                else suma += suma * modificadorRecargoExp / 100;
                suma += suma * impuesto / 100;
                return suma;
            }
        }

        public override void IsValid(int fechaPrometida)
        {
            if (this.FechaPrometida.Day - DateTime.Today.Day >
fechaPrometida) throw new PedidoNoValidoException("Fecha no valida.");
            if (this.FechaPrometida < DateTime.Today) throw new
PedidoNoValidoException("Fecha no valida.");
        }
    }
}

```

```

        if (Lineas.Count == 0) throw new PedidoNoValidoException("Debe
tener al menos una linea.");
    }
}

Usuario
using LogicaNegocio.ValueObjects;
using Microsoft.EntityFrameworkCore;
using Papeleria.LogicaNegocio.Excepciones;
using Papeleria.LogicaNegocio.InterfacesEntidades;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace LogicaNegocio.Entidades
{
    [Index(nameof(Email), IsUnique = true)]

    public abstract class Usuario : IValid
    {
        public int Id { get; set; }
        [Required(ErrorMessage = "Requerido.")]
        [EmailAddress]
        public string Email { get; set; }
        [Required(ErrorMessage = "Requerido.")]
        [RegularExpression(@"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[.,;!])?[a-
zA-Z\d.,;!]{6,}$",
        ErrorMessage = "La contraseña debe tener al menos una letra en
minúscula, mayúscula, un número, y uno de estos símbolos: . ; , ! ")]
        [StringLength(200, MinimumLength = 6, ErrorMessage = "La Contraseña
debe ser de al menos 6 caracteres")]
        public string Password { get; set; }

        [Required(ErrorMessage = "Requerido.")]
        [RegularExpression(@"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[.,;!])?[a-
zA-Z\d.,;!]{6,}$",
        ErrorMessage = "La contraseña debe tener al menos una letra en
minúscula, mayúscula, un número, y uno de estos símbolos: . ; , ! ")]
        [StringLength(200, MinimumLength = 6, ErrorMessage = "La Contraseña
debe ser de al menos 6 caracteres")]
        public string PasswordSinEncript { get; set; }
        public static int UltimoId { get; set; }
        [Required(ErrorMessage = "Requerido.")]
        [RegularExpression(@"^[a-zA-Z]([-a-zA-Z ']*(?![ '-])$)", ErrorMessage
= "El nombre o apellido es inválido.")]
        public NombreCompleto NombreCompleto { get; set; }

        public void IsValid()
        {
            if (this != null)
            {
                if (Email == null) throw new
AdministradorNoValidoException("Email requerido.");
                if (Password == null) throw new
AdministradorNoValidoException("Password requerido.");
                if (NombreCompleto.Nombre == null) throw new
AdministradorNoValidoException("Nombre requerido.");
                if (NombreCompleto.Apellido == null) throw new
AdministradorNoValidoException("Apellido requerido.");
            }
        }
    }
}

```

```

        if (!Regex.IsMatch>Password, @"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[.,;!])[a-zA-Z\d.,;!]{6,}$"))
        {
            throw new AdministradorNoValidoException("La contraseña debe tener al menos una letra en minúscula, mayúscula, un número, y uno de estos símbolos: . ; , ! ");
        }
        NombreCompleto.IsValid();
    }
    else throw new AdministradorNoValidoException("Admin no válido");
}
}
}

```

Excepciones

AdministradorNoValidoException

```

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

```

```

namespace Papeleria.LogicaNegocio.Excepciones

```

```

{
    public class AdministradorNoValidoException : Exception
    {
        public AdministradorNoValidoException()
        {
        }

        public AdministradorNoValidoException(string? message) :
base(message)
        {
        }

        public AdministradorNoValidoException(string? message, Exception?
innerException) : base(message, innerException)
        {
        }

        protected AdministradorNoValidoException(SerializationInfo info,
StreamingContext context) : base(info, context)
        {
        }
    }
}

```

ArticuloNoValidoException

```

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

```

```

namespace Papeleria.LogicaNegocio.Excepciones

```

```

{
    public class ArticuloNoValidoException : Exception
    {
        public ArticuloNoValidoException()
        {
        }
    }
}

```

```

        public ArticuloNoValidoException(string? message) : base(message)
        {
        }

        public ArticuloNoValidoException(string? message, Exception?
innerException) : base(message, innerException)
        {
        }

        protected ArticuloNoValidoException(SerializationInfo info,
StreamingContext context) : base(info, context)
        {
        }
    }
}

ClienteNoValidoException
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaNegocio.Excepciones
{
    public class ClienteNoValidoException : Exception
    {
        public ClienteNoValidoException()
        {
        }

        public ClienteNoValidoException(string? message) : base(message)
        {
        }

        public ClienteNoValidoException(string? message, Exception?
innerException) : base(message, innerException)
        {
        }

        protected ClienteNoValidoException(SerializationInfo info,
StreamingContext context) : base(info, context)
        {
        }
    }
}

DireccionNoValidaException
using System.Runtime.Serialization;

namespace Papeleria.LogicaNegocio.Excepciones
{
    [Serializable]
    internal class DireccionNoValidaException : Exception
    {
        public DireccionNoValidaException()
        {
        }

        public DireccionNoValidaException(string? message) : base(message)
        {
        }
    }
}

```

```

        public DireccionNoValidaException(string? message, Exception?
innerException) : base(message, innerException)
        {
        }

        protected DireccionNoValidaException(SerializationInfo info,
StreamingContext context) : base(info, context)
        {
        }
    }
}

```

LineaNoValidaException

```

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

```

```

namespace Papeleria.LogicaNegocio.Excepciones
{

```

```

    public class LineaNoValidaException : Exception
    {
        public LineaNoValidaException()
        {
        }

        public LineaNoValidaException(string? message) : base(message)
        {
        }

        public LineaNoValidaException(string? message, Exception?
innerException) : base(message, innerException)
        {
        }

        protected LineaNoValidaException(SerializationInfo info,
StreamingContext context) : base(info, context)
        {
        }
    }
}

```

NombreClienteNoValidoException

```

using System.Runtime.Serialization;

```

```

namespace Papeleria.LogicaNegocio.Excepciones
{

```

```

    [Serializable]
    internal class NombreClienteNoValidoException : Exception
    {
        public NombreClienteNoValidoException()
        {
        }

        public NombreClienteNoValidoException(string? message) :
base(message)
        {
        }

        public NombreClienteNoValidoException(string? message, Exception?
innerException) : base(message, innerException)

```

```

        {
        }

        protected NombreClienteNoValidoException(SerializationInfo info,
StreamingContext context) : base(info, context)
        {
        }
    }
}

NombreCompletoNoValidoException
using System.Runtime.Serialization;

namespace Papeleria.LogicaNegocio.Excepciones
{
    [Serializable]
    internal class NombreCompletoNoValidoException : Exception
    {
        public NombreCompletoNoValidoException()
        {
        }

        public NombreCompletoNoValidoException(string? message) :
base(message)
        {
        }

        public NombreCompletoNoValidoException(string? message, Exception?
innerException) : base(message, innerException)
        {
        }

        protected NombreCompletoNoValidoException(SerializationInfo info,
StreamingContext context) : base(info, context)
        {
        }
    }
}

PedidoNoValidoException
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaNegocio.Excepciones
{
    public class PedidoNoValidoException : Exception
    {
        public PedidoNoValidoException()
        {
        }

        public PedidoNoValidoException(string? message) : base(message)
        {
        }

        public PedidoNoValidoException(string? message, Exception?
innerException) : base(message, innerException)
        {
        }
    }
}

```

```

        protected PedidoNoValidoException(SerializationInfo info,
StreamingContext context) : base(info, context)
        {
        }
    }
}

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

namespace Papeleria.LogicaNegocio.InterfacesEntidades
{
    public interface IValid
    {
        public void IsValid();
    }
}

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

namespace LogicaNegocio.InterfacesRepositorio
{
    public interface IRepository<T> where T : class
    {
        IEnumerable<T> FindAll();
        T FindByID(int id);
        bool Add(T aAgregar);
        bool Remove(int id);
        bool Update(T aModificar);
    }
}

IRepositorioAdminitrador
using LogicaNegocio.Entidades;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.InterfacesRepositorio
{
    public interface IRepositoryAdministrador : IRepository<Administrador>
    {
        Administrador FindByEmail(string email);
    }
}

IRepositorioArticulo
using LogicaNegocio.Entidades;

```

```

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

namespace LogicaNegocio.InterfacesRepositorio
{
    public interface IRepositoryArticulo : IRepository<Articulo>
    {
    }
}

//RepositorioCliente
using LogicaNegocio.Entidades;
using LogicaNegocio.InterfacesRepositorio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaNegocio.InterfacesRepositorio
{
    public interface IRepositoryCliente : IRepository<Cliente>
    {
        public IEnumerable<Cliente> ClientesXnombreYapellido(string
especifica);
        public IEnumerable<Cliente> ClientesXmonto(double monto);
    }
}

//RepositorioConfig
using LogicaNegocio.Entidades;
using LogicaNegocio.InterfacesRepositorio;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Papeleria.LogicaNegocio.InterfacesRepositorio
{
    public interface IRepositoryConfig : IRepository<Configuracion>
    {
        Configuracion FindByNombre(string nombre);
    }
}

//RepositorioPedido
using LogicaNegocio.Entidades;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.InterfacesRepositorio
{
    public interface IRepositoryPedido : IRepository<Pedido>
    {
        public double CalcularPrecio(Pedido pedido);
    }
}

```


ValueObjects

Direccion

```
using Microsoft.EntityFrameworkCore;
using Papeleria.LogicaNegocio.Excepciones;
using Papeleria.LogicaNegocio.InterfacesEntidades;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogicaNegocio.ValueObjects
{
    [Owned]
    public class Direccion : IValid
    {
        public string NombreCalle { get; private set; }
        public string NumeroPuerta { get; private set; }
        public string Ciudad { get; private set; }
        public double DistanciaKm { get; private set; }
        public Direccion(string nombreCalle, string numeroPuerta, string
ciudad, double distanciaKm)
        {
            NombreCalle = nombreCalle;
            NumeroPuerta = numeroPuerta;
            Ciudad = ciudad;
            DistanciaKm = distanciaKm;
        }

        public void IsValid()
        {
            if (DistanciaKm < 0) throw new
DireccionNoValidaException("Distancia negativa.");
            if (!int.TryParse(NumeroPuerta, out int x)) throw new
DireccionNoValidaException("Numero de puerta invalido");
        }
    }
}
```

NombreCliente

```
using Microsoft.EntityFrameworkCore;
using Papeleria.LogicaNegocio.Excepciones;
using Papeleria.LogicaNegocio.InterfacesEntidades;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace Papeleria.LogicaNegocio.ValueObjects
{
    [Owned]
    public class NombreCliente : IValid
    {
        public string Nombre { get; private set; }
        public string Apellido { get; private set; }
        public NombreCliente() { }
        public NombreCliente(string nombre, string apellido)
        {
            this.Nombre = nombre;
            this.Apellido = apellido;
        }
    }
}
```

```

        public void IsValid()
        {
            if (!Regex.IsMatch(Nombre, @"^[a-zA-Z][-a-zA-Z ']*(?<![ '-])$"))
                throw new NombreClienteNoValidoException("Nombre no es valido");
            if (!Regex.IsMatch(Apellido, @"^[a-zA-Z][-a-zA-Z ']*(?<![ '-])$"))
                throw new NombreClienteNoValidoException("Apellido no es valido");
        }
    }
}

```

NombreCompleto

```

using Microsoft.EntityFrameworkCore;
using Papeleria.LogicaNegocio.Excepciones;
using Papeleria.LogicaNegocio.InterfacesEntidades;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

```

```

namespace LogicaNegocio.ValueObjects

```

```

{
    [Owned]
    public class NombreCompleto : IValid
    {
        public string Nombre { get; private set; }
        public string Apellido { get; private set; }
        public NombreCompleto()
        {
            Nombre = "Sin Nombre";
            Apellido = "Sin Apellido";
        }
        public NombreCompleto(string nombre, string apellido) {
            this.Nombre = nombre;
            this.Apellido = apellido;
        }

        public void IsValid()
        {
            if (!Regex.IsMatch(Nombre, @"^[a-zA-Z][-a-zA-Z ']*(?<![ '-])$"))
                throw new NombreCompletoNoValidoException("Nombre no es valido");
            if (!Regex.IsMatch(Apellido, @"^[a-zA-Z][-a-zA-Z ']*(?<![ '-])$"))
                throw new NombreCompletoNoValidoException("Apellido no es valido");
        }
    }
}

```

WebApp

AdminController

```

using Microsoft.AspNetCore.Mvc;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Administrador;

```

```

namespace Papeleria.Web.Controllers

```

```

{
    public class AdminController : Controller
    {
        private ICrearAdmin _crearAdmin;
        private IEditarAdmin _editarAdmin;
        private IEncontrarAdmins _encontrarAdmins;
        private IFindAdminById _findAdminById;
        private IBorrarAdmin _borrarAdmin;
    }
}

```

```

        public AdminController(ICrearAdmin crearAdmin, IEditarAdmin
editarAdmin,
        IEncontrarAdmins encontrarAdmins, IFindAdminById findAdminById,
IBorrarAdmin borrarAdmin)
        {
            this._encontrarAdmins = encontrarAdmins;
            this._crearAdmin = crearAdmin;
            this._editarAdmin = editarAdmin;
            this._findAdminById = findAdminById;
            this._borrarAdmin = borrarAdmin;
        }
        // GET: AdminController
        public ActionResult Index()
        {
            if
(string.IsNullOrEmpty(HttpContext.Session.GetString("usuario")))
            {
                return RedirectToAction("Index", "Login", new { mensaje =
"Por favor inicie sesion" });
            }
            return View(_encontrarAdmins.FindAllAdmins());
        }

        // GET: AdminController/Create
        public ActionResult Create(string mensaje)
        {
            if
(string.IsNullOrEmpty(HttpContext.Session.GetString("usuario")))
            {
                return RedirectToAction("Index", "Login", new { mensaje =
"Por favor inicie sesion" });
            }
            ViewBag.mensaje = mensaje;
            return View();
        }

        // POST: AdminController/Create
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(AdministradorDTO adminDto)
        {
            try
            {
                this._crearAdmin.CrearAdmin(adminDto);
                return RedirectToAction(nameof(Index));
            }
            catch(Exception ex)
            {
                return RedirectToAction("Create", new {mensaje =
ex.Message});
            }
        }

        // GET: AdminController/Edit/5
        public ActionResult Edit(int id)
        {
            if
(string.IsNullOrEmpty(HttpContext.Session.GetString("usuario")))
            {
                return RedirectToAction("Index", "Login", new { mensaje =
"Por favor inicie sesion" });
            }
            AdministradorDTO admin = this._findAdminById.FindAdminById(id);

```

```

        return View(admin);
    }

    // POST: AdminController/Edit/5
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Edit(AdministradorDTO adminDto)
    {
        this._editarAdmin.EditarAdmin(adminDto);
        return View(adminDto);
    }

    // GET: AdminController/Delete/5
    public ActionResult Delete(int id)
    {
        if
(string.IsNullOrEmpty(HttpContext.Session.GetString("usuario")))
        {
            return RedirectToAction("Index", "Login", new { mensaje =
"Por favor inicie sesion" });
        }
        AdministradorDTO admin = this._findAdminById.FindAdminById(id);
        return View(admin);
    }

    // POST: AdminController/Delete/5
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Delete(AdministradorDTO admin, int id)
    {
        this._borrarAdmin.BorrarAdmin(id);
        return RedirectToAction("Index");
    }
}

```

ArticulosController

```

using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Articulo;

namespace Papeleria.Web.Controllers
{
    public class ArticulosController : Controller
    {
        private IEncontrarArticulos _encontrarArticulos;
        private ICrearArticulo _crearArticulo;
        public ArticulosController(IEncontrarArticulos encontrarArticulos,
ICrearArticulo crearArticulo)
        {
            _encontrarArticulos = encontrarArticulos;
            _crearArticulo = crearArticulo;
        }

        // GET: ArticulosController
        public ActionResult Index()
        {
            if
(string.IsNullOrEmpty(HttpContext.Session.GetString("usuario")))
            {

```

```

        return RedirectToAction("Index", "Login", new { mensaje =
"Por favor inicie sesion" });
    }
    return View(this._encontrarArticulos.EncontrarArticulos());
}

// GET: ArticulosController/Create
public ActionResult Create()
{
    if
(string.IsNullOrEmpty(HttpContext.Session.GetString("usuario")))
    {
        return RedirectToAction("Index", "Login", new { mensaje =
"Por favor inicie sesion" });
    }
    return View();
}

// POST: ArticulosController/Create
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create(ArticuloDTO articulo)
{
    try
    {
        this._crearArticulo.CrearArticulo(articulo);
        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}
}
}

```

ClientesController

```

using Microsoft.AspNetCore.Mvc;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Cliente;

namespace Papeleria.Web.Controllers
{
    public class ClientesController : Controller
    {
        private IEncontrarClientes _encontrarClientes;
        private IGetClientesXnombreYapellido _xnombreYapellido;
        private IGetClientesXmonto _xMonto;

        public ClientesController(IEncontrarClientes encontrarClientes,
IGetClientesXnombreYapellido xnombreYapellido, IGetClientesXmonto xMonto)
        {
            _encontrarClientes = encontrarClientes;
            _xnombreYapellido = xnombreYapellido;
            _xMonto = xMonto;
        }

        // GET: ClientesController
        public ActionResult Index(string mensaje, string filtro)
        {

```

```

        if
(string.IsNullOrEmpty(HttpContext.Session.GetString("usuario")))
        {
            return RedirectToAction("Index", "Login", new { mensaje =
"Por favor inicie sesion" });
        }
        IEnumerable<ClienteDTO> toShow = new List<ClienteDTO>();
        ViewBag.Mensaje = mensaje;
        ViewBag.Clientes = this._encontrarClientes.FindAllClientes();

        if (string.IsNullOrEmpty(filtro))
        {
            toShow = _encontrarClientes.FindAllClientes();
        }
        if (filtro == "PorNombre")
        {
            string nombreCliente = (string)TempData["NombreCliente"];
            toShow =
this._xnombreYapellido.GetClientesXnombreYapellido(nombreCliente);
        }
        if(filtro == "PorMonto")
        {
            double monto;
            if(double.TryParse(TempData["Monto"].ToString(), out monto))
toShow = this._xMonto.GetClientesXmonto(monto);
        }
        return View(toShow);
    }

    [HttpPost]
    public ActionResult FiltrarPorNombre (string nombreCliente)
    {
        if
(string.IsNullOrEmpty(HttpContext.Session.GetString("usuario")))
        {
            return RedirectToAction("Index", "Login", new { mensaje =
"Por favor inicie sesion" });
        }
        if (nombreCliente == null)
        {
            return RedirectToAction("Index");
        }
        TempData["NombreCliente"] = nombreCliente;
        return RedirectToAction("Index", new { filtro = "PorNombre" });
    }

    [HttpPost]
    public ActionResult FiltrarPorMonto (double monto)
    {
        if
(string.IsNullOrEmpty(HttpContext.Session.GetString("usuario")))
        {
            return RedirectToAction("Index", "Login", new { mensaje =
"Por favor inicie sesion" });
        }
        if (monto <= 0)
        {
            return RedirectToAction("Index", new { mensaje = "Monto
invalido." });
        }
        TempData["Monto"] = monto.ToString();
        return RedirectToAction("Index", new { filtro = "PorMonto" });
    }
}

```

```

    }
}
HomeController
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace Papeleria.Web.Controllers
{
    public class HomeController : Controller
    {
        // GET: HomeController
        public ActionResult Index(string mensaje)
        {
            if
(string.IsNullOrEmpty(HttpContext.Session.GetString("usuario")))
            {
                return RedirectToAction("Index", "Login", new { mensaje =
"No tienes acceso" });
            }
            else
            {
                ViewBag.Mensaje = mensaje;
                if (HttpContext.Session.GetString("rol") == "Miembro")
                {
                    ViewBag.usuario =
(HttpContext.Session.GetString("usuario"));
                }
                return View();
            }
        }
    }
}

LoginController
using LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Administrador;
using Microsoft.AspNetCore.Mvc;

namespace Papeleria.Web.Controllers
{
    public class LoginController : Controller
    {
        private IRepositoryoAdministrador _repositorioAdmins;
        private ILogin _loginCU;
        public LoginController(IRepositoryoAdministrador repositorioAdmins,
            ILogin login)
        {
            this._repositorioAdmins = repositorioAdmins;
            this._loginCU = login;
        }
        // GET: LoginController
        public ActionResult Index(string mensaje)
        {
            ViewBag.mensaje = mensaje;
            return View();
        }

        public ActionResult Login(string email, string password)
        {
            if (this._loginCU.Login(email, password))
            {
                HttpContext.Session.SetString("usuario", email);
            }
        }
    }
}

```

```

        return RedirectToAction("Index", "Home");
    }
    return RedirectToAction("Index", new { mensaje = "Nombre de
usuario o contraseña incorrecta." });
    //manda esto para el CULogin, que primero encripta la
contraseña, la desencripta en la capa de datos,
    //la verifica con los usuarios de la BD, y devuelve true o
false.
    }
    public IActionResult Logout()
    {
        HttpContext.Session.SetString("usuario", "");
        return RedirectToAction("Index");
    }
}
}

```

PedidosController

```

using Microsoft.AspNetCore.Mvc;
using Microsoft.CodeAnalysis.Elfi.Serialization;
using Papeleria.LogicaAplicacion.DTOs;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Articulo;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Cliente;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Pedido;

namespace Papeleria.Web.Controllers
{
    public class PedidosController : Controller
    {
        private ICrearPedido _crearPedido;
        private IEncontrarPedidos _encontrarPedidos;
        private IEncontrarClientes _encontrarClientes;
        private IEncontrarArticulos _encontrarArticulos;
        private IEncontrarPrecioPedido _encontrarPrecioPedido;
        private IEncontrarXIdArticulo _encontrarXIdArticulo;
        private IAnularPedido _anularPedido;
        private IGetPedidosPorFecha _getPedidosPorFecha;
        private static PedidoDTO tempPedido;
        private static List<ArticuloDTO> tempArticulos;

        public PedidosController(ICrearPedido crearPedido,
            IEncontrarArticulos encontrarArticulos, IEncontrarPedidos
encontrarPedidos,
            IEncontrarClientes encontrarClientes, IEncontrarPrecioPedido
encontrarPrecioPedido,
            IEncontrarXIdArticulo encontrarXIdArticulo, IAnularPedido
anularPedido, IGetPedidosPorFecha getPedidosPorFecha)
        {
            _crearPedido = crearPedido;
            _encontrarArticulos = encontrarArticulos;
            _encontrarPedidos = encontrarPedidos;
            _encontrarClientes = encontrarClientes;
            _encontrarPrecioPedido = encontrarPrecioPedido;
            _encontrarXIdArticulo = encontrarXIdArticulo;
            _anularPedido = anularPedido;
            _getPedidosPorFecha = getPedidosPorFecha;
        }

        // GET: PedidosController
        public ActionResult Index(string filtro, string mensaje)
        {
            if
(string.IsNullOrEmpty(HttpContext.Session.GetString("usuario")))

```



```

        {
            return RedirectToAction("Index", "Login", new { mensaje =
"Necesita hacer login" });
        }
        IEnumerable<PedidoDTO> aMostrar = new List<PedidoDTO>();
        if (string.IsNullOrEmpty(filtro))
        {
            aMostrar = this._encontrarPedidos.EncontrarPedidos();
            ViewBag.Filtro = "Ninguno";
        }
        if (filtro == "PorFecha")
        {
            DateTime FechaPrometida =
DateTime.Parse(TempData["FechaPrometida"].ToString());
            aMostrar =
this._getPedidosPorFecha.GetPedidosPorFecha(FechaPrometida);
            if(aMostrar.Count() == 0)
            {
                return RedirectToAction("Index", new { filtro = "",
mensaje = "No hay pedidos sin entregar con esa fecha." });
            }
            ViewBag.Filtro = "Fecha";
        }
        ViewBag.Mensaje = mensaje;
        ViewBag.Clientes = this._encontrarClientes.FindAllClientes();
        ViewBag.Articulos =
this._encontrarArticulos.EncontrarArticulos();
        return View(aMostrar);
    }

    // GET: PedidosController/Create
    public ActionResult Create(Boolean esExpress)
    {
        if
(string.IsNullOrEmpty(HttpContext.Session.GetString("usuario")))
        {
            return RedirectToAction("Index", "Login", new { mensaje =
"Por favor inicie sesion" });
        }
        ViewBag.Clientes = this._encontrarClientes.FindAllClientes();
        ViewBag.Articulos =
this._encontrarArticulos.EncontrarArticulos();
        ViewBag.PrecioPedido = 0;
        if (tempPedido != null)
        {
            ViewBag.Lineas = tempPedido.Lineas;
            ViewBag.PrecioPedido =
this._encontrarPrecioPedido.EncontrarPrecioPedido(tempPedido, esExpress);
        }
        return View();
    }

    // POST: PedidosController/Create
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Create(PedidoDTO pedido, Boolean esExpress)
    {
        try
        {
            if(tempPedido != null && tempPedido.Lineas.Count > 0)
            {
                pedido.Lineas = tempPedido.Lineas;
            }
        }
    }

```

```

        this._crearPedido.CrearPedido(pedido, esExpress);
        tempPedido = null;
        tempArticulos = null;
        return RedirectToAction(nameof(Index));
    }
    catch (Exception ex)
    {
        return RedirectToAction("Index", "Pedidos", new {mensaje =
ex.Message});
    }
}

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult AddLinea(int articuloId, int cantUnidades, int
idCliente)
{
    ArticuloDTO articulo =
_encontrarXIdArticulo.FindById(articuloId);
    if (tempArticulos == null)
    {
        tempArticulos = new List<ArticuloDTO>();
        if (articulo.Stock >= cantUnidades && cantUnidades > 0)
        {
            articulo.Stock -= cantUnidades;
            tempArticulos.Add(articulo);
            LineaDTO linea = new LineaDTO { ArticuloId = articuloId,
CantUnidades = cantUnidades, Precio = articulo.Precio * cantUnidades };
            if (tempPedido == null)
            {
                tempPedido = new PedidoDTO { Lineas = new
List<LineaDTO>() };
            }
            tempPedido.ClienteId = idCliente;
            tempPedido.Lineas.Add(linea);
            return this.RedirectToAction(nameof(Create));
        }
        else return this.RedirectToAction(nameof(Create));
    }
    else {
        foreach (ArticuloDTO unArticulo in tempArticulos)
        {
            if (unArticulo.Id == articuloId)
            {
                if (unArticulo.Stock >= cantUnidades && cantUnidades
> 0)
                {
                    unArticulo.Stock = unArticulo.Stock -
cantUnidades;
                    LineaDTO linea = new LineaDTO { ArticuloId =
articuloId, CantUnidades = cantUnidades, Precio = articulo.Precio *
cantUnidades };
                    if (tempPedido == null)
                    {
                        tempPedido = new PedidoDTO { Lineas = new
List<LineaDTO>() };
                    }
                    tempPedido.Lineas.Add(linea);
                    return this.RedirectToAction(nameof(Create));
                }
                else return this.RedirectToAction(nameof(Create));
            }
        }
    }
}

```

```

        if (articulo.Stock >= cantUnidades && cantUnidades > 0)
        {
            tempArticulos.Add(articulo);
            LineaDTO linea = new LineaDTO { ArticuloId = articuloId,
CantUnidades = cantUnidades, Precio = articulo.Precio * cantUnidades };
            if (tempPedido == null)
            {
                tempPedido = new PedidoDTO { Lineas = new
List<LineaDTO>() };
            }
            tempPedido.Lineas.Add(linea);
            return this.RedirectToAction(nameof(Create));
        }
        else return this.RedirectToAction(nameof(Create));
    }
}

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult AnularPedido(int idPedido)
{
    if
(string.IsNullOrEmpty(HttpContext.Session.GetString("usuario")))
    {
        return RedirectToAction("Index", "Login", new { mensaje =
"Por favor inicie sesion" });
    }
    try
    {
        this._anularPedido.AnularPedido(idPedido);
        return this.RedirectToAction(nameof(Index));
    }
    catch (Exception ex)
    {
        return this.RedirectToAction("Index", "Pedidos", new
{mensaje = "Error al anular" });
    }
}

[HttpPost]
public ActionResult FiltrarPorFecha(DateTime FechaPrometida)
{
    if
(string.IsNullOrEmpty(HttpContext.Session.GetString("usuario")))
    {
        return RedirectToAction("Index", "Login", new { mensaje =
"Por favor inicie sesion" });
    }
    if (FechaPrometida < DateTime.Today)
    {
        return RedirectToAction("Index", new { mensaje = "Fecha
invalida." });
    }
    TempData["FechaPrometida"] = FechaPrometida.ToString();
    return RedirectToAction("Index", new { filtro = "PorFecha" });
}
}
}

```

Program

```

using Papeleria.AccesoDatos.EntityFramework.Repositorios;
using Papeleria.LogicaAplicacion.CasosDeUso.Administradores;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Administrador;
using LogicaNegocio.InterfacesRepositorio;

```

```

using AccesoDatos.EntityFramework;
using Papeleria.LogicaNegocio.InterfacesRepositorio;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Cliente;
using Papeleria.LogicaAplicacion.CasosDeUso.Clientes;
using Papeleria.LogicaAplicacion.CasosDeUso.Articulos;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Articulo;
using AccesoDatos.EntityFramework.Repositorios;
using Papeleria.LogicaAplicacion.InterfacesCasosDeUso.Pedido;
using Papeleria.LogicaAplicacion.CasosDeUso.Pedidos;

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

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

            builder.Services.AddScoped<IRepositorioAdministrador,
RepositorioAdministradorEF>();
            builder.Services.AddScoped<IRepositorioConfig,
RepositorioConfiguracionEF>();
            builder.Services.AddScoped<IRepositorioCliente,
RepositorioClienteEF>();
            builder.Services.AddScoped<IRepositorioArticulo,
RepositorioArticuloEF>();
            builder.Services.AddScoped<IRepositorioPedido,
RepositorioPedidoEF>();

            builder.Services.AddScoped<ILogin, LoginCU>();
            builder.Services.AddScoped<IEncontrarArticulosOrd,
EncontrarArticulosOrdCU>();
            builder.Services.AddScoped<IGetPedidosDesc, GetPedidosDescCU>();
            builder.Services.AddScoped<ICrearAdmin, CrearAdminCU>();
            builder.Services.AddScoped<IEditarAdmin, EditarAdminCU>();
            builder.Services.AddScoped<IFindAdminById, FindAdminByIdCU>();
            builder.Services.AddScoped<IEncontrarAdmins,
EncontrarAdminsCU>();
            builder.Services.AddScoped<IBorrarAdmin, BorrarAdminCU>();
            builder.Services.AddScoped<IEncontrarClientes,
EncontrarClientesCU>();
            builder.Services.AddScoped<IGetClientesXnombreYapellido,
GetClientesXnombreYapellidoCU>();
            builder.Services.AddScoped<IGetClientesXmonto,
GetClientesXmontoCU>();
            builder.Services.AddScoped<ICrearArticulo, CrearArticuloCU>();
            builder.Services.AddScoped<IEncontrarArticulos,
EncontrarArticulosCU>();
            builder.Services.AddScoped<IEncontrarPedidos,
EncontrarPedidosCU>();
            builder.Services.AddScoped<ICrearPedido, CrearPedidoCU>();
            builder.Services.AddScoped<IEncontrarPrecioPedido,
EncontrarPrecioPedidoCU>();
            builder.Services.AddScoped<IEncontrarXIdArticulo,
FindArticuloByIdCU>();
            builder.Services.AddScoped<IAnularPedido, AnularPedidoCU>();
            builder.Services.AddScoped<IGetPedidosPorFecha,
GetPedidosPorFechaCU>();

```

```

builder.Services.AddSession(option =>
{
    option.IdleTimeout = TimeSpan.FromSeconds(5000);
    option.Cookie.HttpOnly = true;
    option.Cookie.IsEssential = true;
});

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change
this for production scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();
app.UseSession();

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

app.Run();
}
}
}

```