```
namespace LaPaulita.Entity.Common
{
    public class EntityCommon
    {
        public int Id { get; set; }
        public int CreatedBy { get; set; }
        public DateTime CreatedDate { get; set; } = DateTime.UtcNow;
        public int UpdatedBy { get; set; }
        public DateTime UpdatedDate { get; set; } = DateTime.UtcNow;
    }
}
```



```
using LaPaulita.Entity.Common;
using LaPaulita.Entity.Enums;

namespace LaPaulita.Entity.Entities
{
    public class OrderHeader : EntityCommon
    {
        public int ClientId { get; set; }
        public string ShippingAddress { get; set; }
        public int ShippingCity { get; set; }
        public int ShippingCountry { get; set; }
        public string ShippingZip { get; set; }
        public DateTime DateOrder { get; set; } = DateTime.UtcNow;
        public TransportType TransportType { get; set; } = TransportType.Road;
        public DiscountType DiscountType { get; set; } = DiscountType.Percentage;
        public int Discount { get; set; } = 10;
    }
}
```



```
namespace LaPaulita.Entity.Enums
{
    public enum DiscountType
    {
        Percentage = 1,
        Flat = 2

    }
}
```



```
namespace LaPaulita.Entity.Enums
{
    public enum TransportType
    {
        Maritime = 1,
        Air = 2,
        Rail = 3,
        Road = 4
    }
}
```

```
namespace LaPaulita.Entity.Interfaces
{
    public interface IUnitOfWork
    {
        Task SaveChange();
    }
}

```

```
using LaPaulita.Sales.BusinessRules.DTOs;
using LaPaulita.Sales.BusinessRules.ValueObject;

namespace LaPaulita.Sales.BusinessRules.Agregates
{
    public class CreateOrder : OrderHeader
    {
        //*************************************************************************
        // Esto si lo hago con una clase tipo record inmutable.
        //*************************************************************************

        //readonly List<OrdersDetails> records = new List<OrdersDetails>();
        //public IReadOnlyList<OrdersDetails> Details => records;

        ///// <summary>
        ///// Método que agrega un nuevo record a <c>OrdersDetails</c> de una orden de compra
        ///// <c>OrderHeader</c>.
        ///// Si el nuevo record a agregar tiene un <c>ProductId</c> que ya existe en
        ///// <c>OrdersDetails</c>, solo se modifica la cantidad del producto adicinando al
        ///// valor existente la nueva cantidad.
        ///// </summary>
        ///// <param name="record">Objeto que contiene los datos del nuevo registro a agregar.</param>
        //public void AddRecord(OrdersDetails record)
        //{
        //    var ExistingOrderDetail = records.FirstOrDefault(r => r.ProductId == record.ProductId);
        //    if (ExistingOrderDetail != default)
        //    {
        //        // Con with creamos una copia del objetoque esta a su izquierda pero con
        //        // las propiedades con valores modificado (entre la llaves)
        //        // Es decir agrega a la lista records una copia del objeto ExistingOrderDetail
        //        // pero modificando el valor de la propiedad ProductQuantity, asignandole como valor
        //        // lo que tiene ExistingOrderDetail.ProductQuantity sumado a lo que tiene
        //        // record.ProductQuantity

        //        records.Add(ExistingOrderDetail with
        //        {
        //            ProductQuantity = (short)(ExistingOrderDetail.ProductQuantity + record.ProductQuantity)
```

```
        //        });
        //        records.Remove(ExistingOrderDetail);
        //    }
        //    else
        //    {
        //        records.Add(record);
        //    }
        //}
        //public void AddRecord(int _productId, decimal productPrice, short productQuantity)
        //{
        //    AddRecord(new OrdersDetails(_productId, productPrice, productQuantity));
        //}

        //*************************************************************************
        // Esto si lo hago con una clase que es preparada por nosotros para ser
        // inmutable.
        //*************************************************************************


        // Campo
        readonly List<OrderDetail> records = new List<OrderDetail>();
        // Propiedad
        public IReadOnlyList<OrderDetail> Details => records;

        public void AddDetail(OrderDetail record)
        {
            var ExistingOrderDetail = records.FirstOrDefault(r => r.ProductId == record.ProductId);
            if (ExistingOrderDetail != default)
            {
                records.Add(new OrderDetail(record.ProductId, record.ProductPrice, (short)(record.ProductQuantity + ExistingOrderDetail.ProductQuantity)));
                records.Remove(ExistingOrderDetail);
            }
            else
            {
                records.Add(record);
            }
        }
        public void AddRecord(int productId, decimal productPrice, short productQuantity)
        {
            AddDetail(new OrderDetail(productId, productPrice, productQuantity));
        }
```

```
        public static CreateOrder From(OrderHeaderDto orderHeaderDto)
        {
            // Aqui realizamos el mapeo de las propiedades del DTO con la Entidad.
            CreateOrder createOrder = new CreateOrder
            {
                ClientId = orderHeaderDto.ClientId,
                ShippingAddress = orderHeaderDto.ShippingAddress,
                ShippingCity = orderHeaderDto.ShippingCity,
                ShippingCountry = orderHeaderDto.ShippingCountry,
                ShippingZip = orderHeaderDto.ShippingZip
            };

            foreach (var item in orderHeaderDto.OrderDetails)
            {
                createOrder.AddRecord(item.ProductId, item.ProductPrice, item.ProductQuantty);
            }
            return createOrder;
        }
    }
}
```

```
namespace LaPaulita.Sales.BusinessRules.DTOs
{
    public class OrderDetailDto
    {
        public int ProductId { get; set; }
        public decimal ProductPrice { get; set; }
        public short ProductQuantty { get; set; }
    }
}
```

```
1  namespace LaPaulita.Sales.BusinessRules.DTOs
2  {
3      public class OrderHeaderDto
4      {
5          public int ClientId { get; set; }
6          public string ShippingAddress { get; set; }
7          public int ShippingCity { get; set; }
8          public int ShippingCountry { get; set; }
9          public string ShippingZip { get; set; }
10
11         public List<OrderDetailDto> OrderDetails { get; set; }
12     }
13 }
```

```
namespace LaPaulita.Sales.BusinessRules.DTOs.ValidatorDTO
{
    public class ValidationErrorDto
    {
        public string PropertyName { get; set; }
        public string ErrorMessage { get; set; }
    }
}

```

```
using LaPaulita.Sales.BusinessRules.DTOs;
using LaPaulita.Sales.BusinessRules.Wrappers;

namespace LaPaulita.Sales.BusinessRules.Interface.Controllers
{
    public interface ICreateOrderController
    {
        Task<WrappersSalesOrder> CreateOrder(OrderHeaderDto order);
    }
}

```

```
using LaPaulita.Sales.BusinessRules.DTOs;

namespace LaPaulita.Sales.BusinessRules.Interface.Getways
{
    public interface ICreateOrderInputPort
    {
        Task Handle(OrderHeaderDto createOrderDto);
    }
}

```

```
using LaPaulita.Sales.BusinessRules.Wrappers;

namespace LaPaulita.Sales.BusinessRules.Interface.Getways
{
    public interface ICreateOrderOutputPort
    {
        //ValueTask Handle(int orderId);
        ValueTask Handle(WrappersSalesOrder order);
        ValueTask ValidationFaild(List<ValidationErrorDto>
          validationError);
    }
}

```

```
namespace LaPaulita.Sales.BusinessRules.ValueObject
{
    public class OrderDetail : IEquatable<OrderDetail>
    {
        #region "Campos"
        readonly int _productId;
        readonly decimal _productPrice;
        readonly short _productQuantity;
        #endregion

        #region "Propiedades"
        //public int ProductId { get { return _productId; } }
        public int ProductId => _productId; //expresión Lambda
        public decimal ProductPrice => _productPrice;
        public short ProductQuantity => _productQuantity;
        #endregion

        #region "Cosntructor"
        public OrderDetail(int productId, decimal productPrice, short productQuantity)
        {
            _productId = productId;
            _productPrice = productPrice;
            _productQuantity = productQuantity;
        }
        #endregion

        #region"Equals and GetHashCode"
        public override bool Equals(object? obj)
        {
            return Equals(obj as OrderDetail);
        }

        public bool Equals(OrderDetail? other)
        {
            //return other is not null &&
            //       ProductId == other.ProductId &&
            //       ProductPrice == other.ProductPrice &&
            //       ProductQuantity == other.ProductQuantity;

            //*****************************************************
            // Otra forma usando GetHashCode.
            //*****************************************************
            if (other != null) { return GetHashCode() == other.GetHashCode(); }
            else { return false; }
        }

        public override int GetHashCode()
        {
            return HashCode.Combine(ProductId, ProductPrice, ProductQuantity);
        }
```

```
        #endregion
    }
}


```

```
namespace LaPaulita.Sales.BusinessRules.ValueObject
{
    public record class OrdersDetails(int ProductId, decimal
      ProductPrice, short ProductQuantity);
}

```

```
namespace LaPaulita.Sales.BusinessRules.Wrappers
{
    public class WrappersSalesOrder
    {
        public int OrderId { get; set; }
        public List<ValidationErrorDto> Errors { get; set; }
    }
}

```

```
using LaPaulita.Sales.BusinessRules.Agregates;
using LaPaulita.Sales.BusinessRules.Interface.Presenters;
using LaPaulita.Sales.BusinessRules.Interface.Repositories;
using LaPaulita.Sales.BusinessRules.Wrappers;
using LaPaulita.UsesCase.Specifications;

namespace LaPaulita.Sales.UsesCase.Create
{
    /// <summary>
    /// <b>Use Case Interactor</b>. Es el elemento que contiene el código con la lógica de
    /// negocios que resuelve un caso de uso. Este elemento implementa la abstracción
    /// representada por el elemento <i>Use Case Input Port</i>. En términos de programación
    /// orientada a objetos, el <b>Interactor</b> es una clase que implementa una
    /// Interface o clase abstracta <i>(InputPort)</i>.
    /// </summary>
    public class CreateOrderIteractor : ICreateOrderInputPort
    {
        //readonly ICreateOrderOutputPort _outputPort;
        readonly ISalesCommandRepository _repository;
        readonly ICreateOrderPresenter _presenter;

        public CreateOrderIteractor(ICreateOrderOutputPort outputPort, ISalesCommandRepository repository, ICreateOrderPresenter presenter)
        {
            //_outputPort = outputPort;
            _repository = repository;
            _presenter = presenter;
        }

        public async Task Handle(OrderHeaderDto createOrderDto)
        {
            // Instanciamos un objeto del tipo List<ValidationErrorDto> y le asignamos
            // lo que nos devuelva el método privado ValidateOrder.

            List<ValidationErrorDto> validationErrors = new List<ValidationErrorDto>();
            validationErrors = ValidateOrder(createOrderDto);
            WrappersSalesOrder order = new();
            // Consultamos si la lista validationErrors posee algún elemento.
            if (validationErrors.Count > 0)
            {
                // Si la lista poseía algún elemento, es que hay por lo menos un error
                // Entonses retormaos el OutputPort al presentador con la lista de errores.

```

```
                order.Errors = validationErrors;
                await _presenter.Handle(order);
                return;
            }

            // Si no hay errores contiunuamos con la ejecución del código y
            // creamos la orden, luego guardamos los cambios y finalmente retornamos
            // el Id del registro creado.

            CreateOrder createOrder = CreateOrder.From(createOrderDto);
            try
            {
                await _repository.CreateOrder(createOrder);
                await _repository.SaveChange();
                order.OrderId = createOrder.Id;
                await _presenter.Handle(order);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }

        }
        /// <summary>
        /// Método que valida los datos de la Orden de compra antes de ser presistida
        /// en la base de datos.
        /// <br/> <br/>
        /// <i>Posteriormente deberemos realizar la validación de la existencia de los
        /// datos que debene existir previamente a la orden de compra, como el id del
        /// cliente o el id del producto.</i>
        /// </summary>
        /// <param name="createOrderDto">Objeto que contiene las propiedades a validar</param>
        /// <returns>La lista de propiedades que no cumplen con la validación
        /// y la descripción del error específico.</returns>
        private List<ValidationErrorDto> ValidateOrder(OrderHeaderDto createOrderDto)
        {
            var specification = new OrderHeaderSpecification(createOrderDto);

            return specification.IsValid();
        }
    }
}

```

```
namespace LaPaulita.UsesCase.Specifications
{
    public partial class OrderHeaderSpecification : ISpecification<OrderHeaderDto>
    {
        readonly List<ValidationErrorDto> validationErrors = new List<ValidationErrorDto>();
        readonly OrderHeaderDto entity;

        public OrderHeaderSpecification(OrderHeaderDto entity)
        {
            this.entity = entity;
        }

        public List<ValidationErrorDto> IsValid()
        {
            IsClientIdValid();
            IsAddressValid();
            IsCityValid();
            IsCountryValid();
            IsZipValid();

            return validationErrors;
        }

        private partial void IsClientIdValid();
        private partial void IsAddressValid();
        private partial void IsCityValid();
        private partial void IsCountryValid();
        private partial void IsZipValid();
    }
}

```

```
namespace LaPaulita.UsesCase.Specifications
{
    public partial class OrderHeaderSpecification : ISpecification<OrderHeaderDto>
    {
        private partial void IsAddressValid()
        {
            if (string.IsNullOrEmpty(entity.ShippingAddress))
            {
                validationErrors.Add(new ValidationErrorDto
                {
                    PropertyName = "ShippingAddress",
                    ErrorMessage = "La dirección de envío es requerida."
                });
            }

            if (entity.ShippingAddress.Length > 50)
            {
                validationErrors.Add(new ValidationErrorDto
                {
                    PropertyName = "ShippingAddres",
                    ErrorMessage = "La dirección de envío no puede exceder los 50 caracteres."
                });
            }
        }
    }
}

```

```
using LaPaulita.Sales.BusinessRules.DTOs;
using LaPaulita.Sales.BusinessRules.Interface;

namespace LaPaulita.UsesCase.Specifications
{
    public partial class OrderHeaderSpecification : ISpecification<OrderHeaderDto>
    {
        private partial void IsCityValid()
        {
            if (entity.ShippingCity <= 0)
            {
                validationErrors.Add(new ValidationErrorDto
                {
                    PropertyName = "SippingCity",
                    ErrorMessage = "La ciudad de entrega es obligatorio."
                });
            }
        }
    }
}

```

```
namespace LaPaulita.UsesCase.Specifications
{
    public partial class OrderHeaderSpecification : ISpecification<OrderHeaderDto>
    {
        private partial void IsClientIdValid()
        {
            if (entity.ClientId <= 0)
            {
                validationErrors.Add(new ValidationErrorDto
                {
                    PropertyName = "ClientId",
                    ErrorMessage = "El Id del cliente es obligatorio."
                });
            }
        }
    }
}

```

```
namespace LaPaulita.UsesCase.Specifications
{
    public partial class OrderHeaderSpecification : ISpecification<OrderHeaderDto>
    {
        private partial void IsCountryValid()
        {
            if (entity.ShippingCountry <= 0)
            {
                validationErrors.Add(new ValidationErrorDto
                {
                    PropertyName = "SippingCountry",
                    ErrorMessage = "El país de entrega es obligatorio."
                });
            }
        }
    }
}

```

```
using System.Text.RegularExpressions;

namespace LaPaulita.UsesCase.Specifications
{
    public partial class OrderHeaderSpecification : ISpecification<OrderHeaderDto>
    {
        private partial void IsZipValid()
        {
            if (string.IsNullOrEmpty(entity.ShippingZip))
            {
                validationErrors.Add(new ValidationErrorDto
                {
                    PropertyName = "ShippingZip",
                    ErrorMessage = "El código postal de entrega es obligatorio."
                });
            }

            if (entity.ShippingZip.Length != 4)
            {
                validationErrors.Add(new ValidationErrorDto
                {
                    PropertyName = "ShippingZip",
                    ErrorMessage = "El código postal de entrega debe contener 4 caracteres."
                });
            }

            // Patrón de expresión regular para verificar si solo contiene números
            string pattern = @"^[0-9]+$";

            // Verificar si la entrada coincide con el patrón
            bool isMatch = Regex.IsMatch(entity.ShippingZip, pattern);

            if (!isMatch)
            {
                validationErrors.Add(new ValidationErrorDto
                {
                    PropertyName = "SippingZip",
                    ErrorMessage = "El código postal de entrega debe contener solo números."
                });
            }
        }
    }
}

```

```
namespace LaPaulita.Sales.UsesCase
{
    public static class DependencyContainer
    {
        public static IServiceCollection AddServicesUseCase(this IServiceCollection services)
        {
            services.AddScoped<ICreateOrderInputPort, CreateOrderIteractor>();

            return services;
        }
    }
}

```

```
using LaPaulita.Sales.BusinessRules.Wrappers;

namespace LaPaulita.Sales.Controllers
{
    public class CreateOrderController : ICreateOrderController
    {
        readonly ICreateOrderInputPort _inputPort;
        readonly ICreateOrderPresenter _presenter;

        //public CreateOrderController(ICreateOrderInputPort inputPort, ICreateOrderPresenter presenter)
        //{
        //    _inputPort = inputPort;
        //    _presenter = presenter;
        //}
        public CreateOrderController(ICreateOrderInputPort inputPort, ICreateOrderPresenter presenter) => (_presenter, _inputPort) = (presenter, inputPort); //Expresión lamda.

        public async Task<WrappersSalesOrder> CreateOrder(OrderHeaderDto order)
        {
            await _inputPort.Handle(order);
            return _presenter.Order;

        }
    }
}

```

```
namespace LaPaulita.Sales.Controllers
{
    public static class DependencyContiners
    {
        public static IServiceCollection AddServicesSalesControllers(
            this IServiceCollection services)
        {
            services.AddScoped<ICreateOrderController,
              CreateOrderController>();
            return services;
        }
    }
}

```

```
1
2  namespace DependencyInversion
3  {
4      public static class DependencyContainer
5      {
6          public static IServiceCollection AddLaPaulitaSalesServices(this ↵
             IServiceCollection services, IConfiguration configuration, ↵
             string ConnectionString)
7          {
8              services
9                  .AddRepositories(configuration, ConnectionString)
10                 .AddServicesUseCase()
11                 .AddServicesPresenter()
12                 .AddServicesSalesControllers();
13
14             return services;
15         }
16     }
17 }
18
```

```
namespace LaPaulita.Sales.Repositories.EFCore.Configurations
{
    public class OrderDetailConfiguration : IEntityTypeConfiguration<OrderDetail>
    {
        public void Configure(EntityTypeBuilder<OrderDetail> builder)
        {
            builder.HasKey(d => new { d.Id, d.ProductId });
            builder.Property(d => d.Id)
                .UseIdentityColumn();
            builder.Property(d => d.ProductPrice)
                .HasPrecision(8, 2);
        }
    }
}
```

```
namespace LaPaulita.Sales.Repositories.EFCore.Configurations
{
    public class OrderHeaderConfiguration : IEntityTypeConfiguration<OrderHeader>
    {
        public void Configure(EntityTypeBuilder<OrderHeader> builder)
        {
            builder.Property(o => o.ClientId)
                .IsRequired()
                .HasMaxLength(5);

            builder.Property(o => o.ShippingAddress)
                .IsRequired()
                .HasMaxLength(50);

            builder.Property(o => o.ShippingCity)
                .IsRequired();

            builder.Property(o => o.ShippingCountry)
                .IsRequired();

            builder.Property(o => o.ShippingZip)
                .HasMaxLength(4);
        }
    }
}
```

```
namespace LaPaulita.Sales.Repositories.EFCore.Context
{
    internal class LaPaulitaContext : DbContext
    {
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer("Data Source=DanielPagano; Initial Catalog=LaPaulitaDB; User ID=sa; Password=MsSqlServer; TrustServerCertificate=True");

            //"Data Source=(LocalDb)\\MSSQLLocalDB;Initial Catalog=LaPaulitaDB;Integrated Security=SSPI;"
        }

        public DbSet<OrderHeader> OrderHeaders { get; set; }
        public DbSet<OrderDetail> OrderDetails { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.ApplyConfigurationsFromAssembly(
                Assembly.GetExecutingAssembly());
        }
    }
}

```

```
namespace LaPaulita.Sales.Repositories.EFCore.Context
{
    public class LaPaulitaSalesContext : DbContext
    {
        public LaPaulitaSalesContext (DbContextOptions<LaPaulitaSalesContext> options)
            : base(options) { }

        public DbSet<OrderHeader> OrderHeaders { get; set; }
        public DbSet<OrderDetail> OrderDetails { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.ApplyConfigurationsFromAssembly(
                Assembly.GetExecutingAssembly());
        }
    }
}

```

```
Crear la migración
Add-Migration InitialCreate -p LaPaulita.Sales.Repositories -s LaPaulita.Sales.Repositories -c LaPaulitaContext

Actualización de la DB
Update-Database -p LaPaulita.Sales.Repositories -s LaPaulita.Sales.Repositories -context LaPaulitaContext


Cadena de conexión
Data Source=(LocalDb)\\MSSQLLocalDB;Initial Catalog=LaPaulitaDB;Integrated Security=SSPI;

{
  "clientId": 1,
  "shippingAddress": "La Pampa 132",
  "shippingCity": 1,
  "shippingCountry": 1,
  "shippingZip": "5620",
  "orderDetails": [
    {
      "productId": 1,
      "productPrice": 175.26,
      "productQuantty": 1
    },
    {
      "productId": 1,
      "productPrice": 175.26,
      "productQuantty": 5
    },
    {
      "productId": 2,
      "productPrice": 375.26,
      "productQuantty": 1
    }
  ]
}
```

```
namespace LaPaulita.Sales.Repositories.Repositories
{
    public class SalesCommandRepository : ISalesCommandRepository
    {
        readonly LaPaulitaSalesContext _context;

        public SalesCommandRepository(LaPaulitaSalesContext context)
        {
            _context = context;
        }

        public async Task CreateOrder(CreateOrder order)
        {
            await _context.AddAsync(order);
            foreach (var item in order.Details)
            {
                await _context.AddAsync(new OrderDetail
                {
                    Order = order,
                    ProductId = item.ProductId,
                    ProductPrice = item.ProductPrice,
                    ProductQuantity = item.ProductQuantity
                });
            }
        }

        public async Task SaveChange()
        {
            await _context.SaveChangesAsync();
        }
    }
}

```

```
namespace LaPaulita.Sales.Repositories
{
    public static class DependencyContainer
    {
        public static IServiceCollection AddRepositories(
            this IServiceCollection services,
            IConfiguration configuration,
            string connectionStringName)
        {
            services.AddDbContext<LaPaulitaSalesContext>(options =>
            options.UseSqlServer(configuration
            .GetConnectionString(connectionStringName)));

            services.AddScoped<ISalesCommandRepository,
                SalesCommandRepository>();

            return services;
        }
    }
}

```

```
using LaPaulita.Sales.BusinessRules.Wrappers;

namespace LaPaulita.Sales.Presenters
{
    public class CreateOrderPresenter : ICreateOrderPresenter
    {
        //public int OrderId { get; private set; }

        //public List<ValidationErrorDto> ErrorsList { get; private set; }

        public WrappersSalesOrder Order { get; private set; } = new WrappersSalesOrder();

        //public ValueTask Handle(int orderId)
        //{
        //    OrderId = orderId;
        //    return ValueTask.CompletedTask;
        //}

        public ValueTask Handle(WrappersSalesOrder order)
        {
            Order.OrderId = order.OrderId;
            Order.Errors = order.Errors;
            return ValueTask.CompletedTask;
        }

        public ValueTask ValidationFaild(List<ValidationErrorDto> validationError)
        {
            Order.Errors = validationError;
            return ValueTask.CompletedTask;
        }
    }
}

```

```
namespace LaPaulita.Sales.Presenters
{
    public static class DependencyContainer
    {
        public static IServiceCollection AddServicesPresenter(this IServiceCollection services)
        {
            services.AddScoped<CreateOrderPresenter>();

            services.AddScoped<ICreateOrderOutputPort, CreateOrderPresenter>();

            services.AddScoped<ICreateOrderPresenter, CreateOrderPresenter>();

            return services;
        }
    }
}

```

```
using LaPaulita.Sales.BusinessRules.DTOs;
using LaPaulita.Sales.BusinessRules.Interface.Controllers;

namespace LaPaulita.Sales.WebApi
{
    public static class EndPoints
    {
        public static WebApplication LaPaulitaSalesEndPoint(this WebApplication app)
        {
            app.MapPost("/create", async (OrderHeaderDto order, ICreateOrderController controller) =>
            {
                var result = await controller.CreateOrder(order);

                if (result.Errors != null && result.Errors.Count > 0)
                {
                    return Results.BadRequest(result);
                }
                else
                {
                    return Results.Ok(result);
                }

            });

            return app;
        }
    }
}

```

```
using DependencyInversion;

namespace LaPaulita.Sales.WebApi
{
    public static class WebApplicationHelper
    {
        public static WebApplication CreateWebApplication(this WebApplicationBuilder builder)
        {
            // Configurar APIExplorer para descubrir y exponer
            // los metadatos de los endpoints de la aplicación.
            builder.Services.AddEndpointsApiExplorer();

            // Agregar el generador que construye los objetos de
            // documentación de Swagger con la funcionalidad del APIExplorer.
            builder.Services.AddSwaggerGen();

            // Registrar los servicios de la aplicación
            builder.Services.AddLaPaulitaSalesServices(
                builder.Configuration, "LaPaulitaDB");

            // Agregar el servicio CORS para clientes que se ejecutan
            // en el navegador Web (como Blazor WebAssembly).
            builder.Services.AddCors(options =>
            {
                options.AddDefaultPolicy(config =>
                {
                    config.AllowAnyMethod();
                    config.AllowAnyHeader();
                    config.AllowAnyOrigin();
                });
            });

            return builder.Build();
        }
        public static WebApplication ConfigureWebApplication(
            this WebApplication app)
        {
            // Habilitar el middleware para servir el documento
            // JSON generado y la interfaz UI de Swagger en el
            // ambiente de desarrollo.
            if (app.Environment.IsDevelopment())
            {
                app.UseSwagger();
                app.UseSwaggerUI();
            }

            // Registrar los endpoints de la aplicación
            app.LaPaulitaSalesEndPoint();

            // Agregar el Middleware CORS
            app.UseCors();
```

```
52
53              return app;
54          }
55      }
56  }
```



```
1  using LaPaulita.Sales.WebApi;
2
3  var builder = WebApplication.CreateBuilder(args)
4      .CreateWebApplication()
5      .ConfigureWebApplication();
6  builder.Run();
7  //var app = builder.Build();
8
9  //app.MapGet("/", () => "Hello World!");
10
11 //app.Run();
```



```
1  {
2    "Logging": {
3      "LogLevel": {
4        "Default": "Information",
5        "Microsoft.AspNetCore": "Warning"
6      }
7    },
8    "AllowedHosts": "*",
9    "ConnectionStrings": { "LaPaulitaDB": "Data Source=DanielPagano;
      Initial Catalog=LaPaulitaDB; User ID=sa; Password=MsSqlServer;
      TrustServerCertificate=True" }
10 }
```