



...ulita - copia\LaPaulita.Core\Common\EntityCommon.cs 1

```
1 namespace LaPaulita.Entity.Common
2 {
3     public class EntityCommon
4     {
5         public int Id { get; set; }
6         public int CreatedBy { get; set; }
7         public DateTime CreatedDate { get; set; } = DateTime.UtcNow;
8         public int UpdatedBy { get; set; }
9         public DateTime UpdatedDate { get; set; } = DateTime.UtcNow;
10    }
11 }
```

...lita - copia\LaPaulita.Core\Entities\OrderHeader.cs 1

```
1 using LaPaulita.Entity.Common;
2 using LaPaulita.Entity.Enums;
3
4 namespace LaPaulita.Entity.Entities
5 {
6     public class OrderHeader : EntityCommon
7     {
8         public int ClientId { get; set; }
9         public string ShippingAddress { get; set; }
10        public int ShippingCity { get; set; }
11        public int ShippingCountry { get; set; }
12        public string ShippingZip { get; set; }
13        public DateTime DateOrder { get; set; } = DateTime.UtcNow;
14        public TransportType TransportType { get; set; } = 
            TransportType.Road;
15        public DiscountType DiscountType { get; set; } = 
            DiscountType.Percentage;
16        public int Discount { get; set; } = 10;
17    }
18 }
```

...aulita - copia\LaPaulita.Core\Enums\DiscountType.cs 1

```
1 namespace LaPaulita.Entity.Enums
2 {
3     public enum DiscountType
4     {
5         Percentage = 1,
6         Flat = 2
7     }
8 }
9 }
```

...ulita - copia\LaPaulita.Core\Enums\TransportType.cs 1

```
1 namespace LaPaulita.Entity.Enums
2 {
3     public enum TransportType
4     {
5         Maritime = 1,
6         Air = 2,
7         Rail = 3,
8         Road = 4
9     }
10 }
```

```
1 namespace LaPaulita.Entity.Interfaces
2 {
3     public interface IUnitOfWork
4     {
5         Task SaveChange();
6     }
7 }
8
```

```
1 using LaPaulita.Sales.BusinessRules.DTOS;
2 using LaPaulita.Sales.BusinessRules.ValueObject;
3
4 namespace LaPaulita.Sales.BusinessRules.Agregates
5 {
6     public class CreateOrder : OrderHeader
7     {
8         //
9         // *****
10        // Esto si lo hago con una clase tipo record immutable.
11        // *****
12        //readonly List<OrdersDetails> records = new List<OrdersDetails>
13        //();
14        //public IReadOnlyList<OrdersDetails> Details => records;
15
16        //summary>
17        //summary> Método que agrega un nuevo record a <c>OrdersDetails</c>
18        //summary> de una orden de compra
19        //summary> <c>OrderHeader</c>.
20        //summary> Si el nuevo record a agregar tiene un <c>ProductId</c> que
21        //summary> ya existe en
22        //summary> <c>OrdersDetails</c>, solo se modifica la cantidad del
23        //summary> producto adicionando al
24        //summary> valor existente la nueva cantidad.
25        //summary> </summary>
26        //summary> <param name="record">Objeto que contiene los datos del
27        //summary> nuevo registro a agregar.</param>
28        //public void AddRecord(OrdersDetails record)
29        //{
30        //    var ExistingOrderDetail = records.FirstOrDefault(r =>
31        //        r.ProductId == record.ProductId);
32        //    if (ExistingOrderDetail != default)
33        //    {
34        //        // Con with creamos una copia del objeto que esta a su
35        //        // izquierda pero con
36        //        // las propiedades con valores modificado (entre la
37        //        // llaves)
38        //        // Es decir agrega a la lista records una copia del
39        //        // objeto ExistingOrderDetail
40        //        // pero modificando el valor de la propiedad
41        //        // ProductQuantity, asignandole como valor
42        //        // lo que tiene ExistingOrderDetail.ProductQuantity
43        //        // sumado a lo que tiene
44        //        // record.ProductQuantity
45        //        records.Add(ExistingOrderDetail with
46        //        {
47        //            ProductQuantity = (short)
48        //            (ExistingOrderDetail.ProductQuantity + record.ProductQuantity)
```

```
38         //     });
39         //     records.Remove(ExistingOrderDetail);
40         //     }
41         //     else
42         //     {
43         //         records.Add(record);
44         //     }
45         //}
46         //public void AddRecord(int _productId, decimal productPrice,
47         //    short productQuantity)
48         //{
49         //    AddRecord(new OrdersDetails(_productId, productPrice,
50         //        productQuantity));
51         //}
52         //
53         // *****
54         // *****
55         // Esto si lo hago con una clase que es preparada por nosotros
56         // para ser
57         // immutable.
58         //
59         // *****
60         // *****
61
62         // Campo
63         readonly List<OrderDetail> records = new List<OrderDetail>();
64         // Propiedad
65         public IReadOnlyList<OrderDetail> Details => records;
66
67         public void AddDetail(OrderDetail record)
68         {
69             var ExistingOrderDetail = records.FirstOrDefault(r =>
70                 r.ProductId == record.ProductId);
71             if (ExistingOrderDetail != default)
72             {
73                 records.Add(new OrderDetail(record.ProductId,
74                     record.ProductPrice, (short)(record.ProductQuantity +
75                     ExistingOrderDetail.ProductQuantity)));
76                 records.Remove(ExistingOrderDetail);
77             }
78             else
79             {
80                 records.Add(record);
81             }
82         }
83
84         public void AddRecord(int productId, decimal productPrice, short
85             productQuantity)
86         {
87             AddDetail(new OrderDetail(productId, productPrice,
88                 productQuantity));
89         }
90     }
```

```
79     public static CreateOrder From(OrderHeaderDto orderHeaderDto)
80     {
81         // Aqui realizamos el mapeo de las propiedades del DTO con la Entidad.
82         CreateOrder createOrder = new CreateOrder
83         {
84             ClientId = orderHeaderDto.ClientId,
85             ShippingAddress = orderHeaderDto.ShippingAddress,
86             ShippingCity = orderHeaderDto.ShippingCity,
87             ShippingCountry = orderHeaderDto.ShippingCountry,
88             ShippingZip = orderHeaderDto.ShippingZip
89         };
90
91         foreach (var item in orderHeaderDto.OrderDetails)
92         {
93             createOrder.AddRecord(item.ProductId, item.ProductPrice,
94                                   item.ProductQuantity);
95         }
96         return createOrder;
97     }
98 }
99
```

```
1 namespace LaPaulita.Sales.BusinessRules.DTOs
2 {
3     public class OrderDetailDto
4     {
5         public int ProductId { get; set; }
6         public decimal ProductPrice { get; set; }
7         public short ProductQuantity { get; set; }
8     }
9 }
```

```
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 }
```

```
1 namespace LaPaulita.Sales.BusinessRules.DTOs\ValidatorDTO
2 {
3     public class ValidationErrorDto
4     {
5         public string PropertyName { get; set; }
6         public string ErrorMessage { get; set; }
7     }
8 }
9
```



```
1 using LaPaulita.Sales.BusinessRules.DTOS;
2 using LaPaulita.Sales.BusinessRules.Wrappers;
3
4 namespace LaPaulita.Sales.BusinessRules.Interface.Controllers
5 {
6     public interface ICreateOrderController
7     {
8         Task<WrappersSalesOrder> CreateOrder(OrderHeaderDto order);
9     }
10 }
11
```

```
1 using LaPaulita.Sales.BusinessRules.DTOS;
2
3 namespace LaPaulita.Sales.BusinessRules.Interface.Getways
4 {
5     public interface ICreateOrderInputPort
6     {
7         Task Handle(OrderHeaderDto createOrderDto);
8     }
9 }
10
```

```
1 using LaPaulita.Sales.BusinessRules.Wrappers;
2
3 namespace LaPaulita.Sales.BusinessRules.Interface.Getways
4 {
5     public interface ICreateOrderOutputPort
6     {
7         //ValueTask Handle(int orderId);
8         ValueTask Handle(WrappersSalesOrder order);
9         ValueTask ValidationFaild(List<ValidationErrorDto>
                                validationError);
10    }
11 }
12
```



```
1 namespace LaPaulita.Sales.BusinessRules.ValueObject
2 {
3     public class OrderDetail : IEquatable<OrderDetail>
4     {
5         #region "Campos"
6         readonly int _productId;
7         readonly decimal _productPrice;
8         readonly short _productQuantity;
9         #endregion
10
11         #region "Propiedades"
12         //public int ProductId { get { return _productId; } }
13         public int ProductId => _productId; //expresión Lambda
14         public decimal ProductPrice => _productPrice;
15         public short ProductQuantity => _productQuantity;
16         #endregion
17
18         #region "Cosntructor"
19         public OrderDetail(int productId, decimal productPrice, short productQuantity)
20         {
21             _productId = productId;
22             _productPrice = productPrice;
23             _productQuantity = productQuantity;
24         }
25         #endregion
26
27         #region "Equals and GetHashCode"
28         public override bool Equals(object? obj)
29         {
30             return Equals(obj as OrderDetail);
31         }
32
33         public bool Equals(OrderDetail? other)
34         {
35             //return other is not null &&
36             //    ProductId == other.ProductId &&
37             //    ProductPrice == other.ProductPrice &&
38             //    ProductQuantity == other.ProductQuantity;
39
40             //*****
41             // Otra forma usando GetHashCode.
42             //*****
43             if (other != null) { return GetHashCode() ==
44                 other.GetHashCode(); }
45             else { return false; }
46         }
47
48         public override int GetHashCode()
49         {
50             return HashCode.Combine(ProductId, ProductPrice,
51                 ProductQuantity);
52         }
53     }
54 }
```

```
51         #endregion
52     }
53 }
54
55
```

```
... copia\LaPaulita.Sales\ValueObject\OrdersDetails.cs 1
1 namespace LaPaulita.Sales.BusinessRules.ValueObject
2 {
3     public record class OrdersDetails(int ProductId, decimal ProductPrice, short ProductQuantity);
4 }
5
```

```
1 namespace LaPaulita.Sales.BusinessRules.Wrappers
2 {
3     public class WrappersSalesOrder
4     {
5         public int OrderId { get; set; }
6         public List<ValidationErrorDto> Errors { get; set; }
7     }
8 }
9
```

```
1 using LaPaulita.Sales.BusinessRules.Agregates;
2 using LaPaulita.Sales.BusinessRules.Interface.Presenters;
3 using LaPaulita.Sales.BusinessRules.Interface.Repositories;
4 using LaPaulita.Sales.BusinessRules.Wrappers;
5 using LaPaulita.UsesCase.Specifications;
6
7 namespace LaPaulita.Sales.UsesCase.Create
8 {
9     /// <summary>
10     /// <b>Use Case Interactor</b>. Es el elemento que contiene el código con la lógica de
11     /// negocios que resuelve un caso de uso. Este elemento implementa la abstracción
12     /// representada por el elemento <i>Use Case Input Port</i>. En términos de programación
13     /// orientada a objetos, el <b>Interactor</b> es una clase que implementa una
14     /// Interface o clase abstracta <i>(InputPort)</i>.
15     /// </summary>
16     public class CreateOrderIterator : ICreateOrderInputPort
17     {
18         //readonly ICreateOrderOutputPort _outputPort;
19         readonly ISalesCommandRepository _repository;
20         readonly ICreateOrderPresenter _presenter;
21
22         public CreateOrderIterator(ICreateOrderOutputPort outputPort,
23             ISalesCommandRepository repository, ICreateOrderPresenter presenter)
24         {
25             //_outputPort = outputPort;
26             _repository = repository;
27             _presenter = presenter;
28         }
29
30         public async Task Handle(OrderHeaderDto createOrderDto)
31         {
32             // Instanciamos un objeto del tipo List<ValidationErrorDto> y le asignamos
33             // lo que nos devuelva el método privado ValidateOrder.
34             List<ValidationErrorDto> validationErrors = new List<ValidationErrorDto>();
35             validationErrors = ValidateOrder(createOrderDto);
36             WrappersSalesOrder order = new();
37             // Consultamos si la lista validationErrors posee algún elemento.
38             if (validationErrors.Count > 0)
39             {
40                 // Si la lista poseía algún elemento, es que hay por lo menos un error
41                 // Entonces retornamos el OutputPort al presentador con la lista de errores.
42             }
```





```
43         order.Errors = validationErrors;
44         await _presenter.Handle(order);
45         return;
46     }
47
48     // Si no hay errores contiunamos con la ejecución del código y
49     // creamos la orden, luego guardamos los cambios y finalmente retornamos
50     // el Id del registro creado.
51
52     CreateOrder createOrder = CreateOrder.From(createOrderDto);
53     try
54     {
55         await _repository.CreateOrder(createOrder);
56         await _repository.SaveChange();
57         order.OrderId = createOrder.Id;
58         await _presenter.Handle(order);
59     }
60     catch (Exception ex)
61     {
62         Console.WriteLine(ex.Message);
63     }
64
65 }
66 /// <summary>
67 /// Método que valida los datos de la Orden de compra antes de
68   ser presistida
69   /// en la base de datos.
70   /// <br/> <br/>
71   /// <i>Posteriormente deberemos realizar la validación de la
72   existencia de los
73   /// datos que debene existir previamente a la orden de compra,
74   como el id del
75   /// cliente o el id del producto.</i>
76   /// </summary>
77   /// <param name="createOrderDto">Objeto que contiene las
78   propiedades a validar</param>
79   /// <returns>La lista de propiedades que no cumplen con la
80   validación
81   /// y la descripción del error específico.</returns>
82   private List<ValidationErrorDto> ValidateOrder(OrderHeaderDto
83   createOrderDto)
84   {
85       var specification = new OrderHeaderSpecification
86           (createOrderDto);
87
88       return specification.IsValid();
89   }
90 }
```

```
1 namespace LaPaulita.UsesCase.Specifications
2 {
3     public partial class OrderHeaderSpecification : ISpecification<OrderHeaderDto> ➤
4     {
5         readonly List<ValidationErrorDto> validationErrors = new List<ValidationErrorDto>(); ➤
6         readonly OrderHeaderDto entity;
7
8         public OrderHeaderSpecification(OrderHeaderDto entity)
9         {
10             this.entity = entity;
11         }
12
13         public List<ValidationErrorDto> IsValid()
14         {
15             IsClientIdValid();
16             IsAddressValid();
17             IsCityValid();
18             IsCountryValid();
19             IsZipValid();
20
21             return validationErrors;
22         }
23
24         private partial void IsClientIdValid();
25         private partial void IsAddressValid();
26         private partial void IsCityValid();
27         private partial void IsCountryValid();
28         private partial void IsZipValid();
29     }
30 }
31
```

```
1 namespace LaPaulita.UsesCase.Specifications
2 {
3     public partial class OrderHeaderSpecification : ISpecification<OrderHeaderDto>
4     {
5         private partial void IsAddressValid()
6         {
7             if (string.IsNullOrEmpty(entity.ShippingAddress))
8             {
9                 validationErrors.Add(new ValidationErrorDto
10                 {
11                     PropertyName = "ShippingAddress",
12                     ErrorMessage = "La dirección de envío es requerida."
13                 });
14             }
15
16             if (entity.ShippingAddress.Length > 50)
17             {
18                 validationErrors.Add(new ValidationErrorDto
19                 {
20                     PropertyName = "ShippingAddress",
21                     ErrorMessage = "La dirección de envío no puede
22                                     exceder los 50 caracteres."
23                 });
24             }
25         }
26     }
27 }
```

```
1 using LaPaulita.Sales.BusinessRules.DTOs;
2 using LaPaulita.Sales.BusinessRules.Interface;
3
4 namespace LaPaulita.UsesCase.Specifications
5 {
6     public partial class OrderHeaderSpecification : ISpecification<OrderHeaderDto> ↗
7     {
8         private partial void IsCityValid()
9         {
10             if (entity.ShippingCity <= 0)
11             {
12                 validationErrors.Add(new ValidationErrorDto
13                 {
14                     PropertyName = "SippingCity",
15                     ErrorMessage = "La ciudad de entrega es obligatorio." ↗
16                 });
17             }
18         }
19     }
20 }
21
```

```
1 namespace LaPaulita.UsesCase.Specifications
2 {
3     public partial class OrderHeaderSpecification : 
4         ISpecification<OrderHeaderDto>
5     {
6         private partial void IsClientIdValid()
7         {
8             if (entity.ClientId <= 0)
9             {
10                 validationErrors.Add(new ValidationErrorDto
11                 {
12                     PropertyName = "ClientId",
13                     ErrorMessage = "El Id del cliente es obligatorio."
14                 });
15             }
16         }
17     }
18 }
```

```
1 namespace LaPaulita.UsesCase.Specifications
2 {
3     public partial class OrderHeaderSpecification : 
4         ISpecification<OrderHeaderDto>
5     {
6         private partial void IsCountryValid()
7         {
8             if (entity.ShippingCountry <= 0)
9             {
10                 validationErrors.Add(new ValidationErrorDto
11                 {
12                     PropertyName = "ShippingCountry",
13                     ErrorMessage = "El país de entrega es obligatorio."
14                 });
15             }
16         }
17     }
18 }
```

```
1 using System.Text.RegularExpressions;
2
3 namespace LaPaulita.UsesCase.Specifications
4 {
5     public partial class OrderHeaderSpecification : ISpecification<OrderHeaderDto> ➤
6     {
7         private partial void IsZipValid()
8         {
9             if (string.IsNullOrEmpty(entity.ShippingZip))
10             {
11                 validationErrors.Add(new ValidationErrorDto
12                 {
13                     PropertyName = "ShippingZip",
14                     ErrorMessage = "El código postal de entrega es obligatorio." ➤
15                 });
16             }
17
18             if (entity.ShippingZip.Length != 4)
19             {
20                 validationErrors.Add(new ValidationErrorDto
21                 {
22                     PropertyName = "ShippingZip",
23                     ErrorMessage = "El código postal de entrega debe contener 4 caracteres." ➤
24                 });
25             }
26
27             // Patrón de expresión regular para verificar si solo contiene números ➤
28             string pattern = @"^[0-9]+$";
29
30             // Verificar si la entrada coincide con el patrón
31             bool isMatch = Regex.IsMatch(entity.ShippingZip, pattern);
32
33             if (!isMatch)
34             {
35                 validationErrors.Add(new ValidationErrorDto
36                 {
37                     PropertyName = "ShippingZip",
38                     ErrorMessage = "El código postal de entrega debe contener solo números." ➤
39                 });
40             }
41         }
42     }
43 }
44
```


```
1 namespace LaPaulita.Sales.UsesCase
2 {
3     public static class DependencyContainer
4     {
5         public static IServiceCollection AddServicesUseCase(this IServiceCollection services)
6         {
7             services.AddScoped<ICreateOrderInputPort, CreateOrderInteractor>();
8
9             return services;
10        }
11    }
12 }
13
```



```
1 using LaPaulita.Sales.BusinessRules.Wrappers;
2
3 namespace LaPaulita.Sales.Controllers
4 {
5     public class CreateOrderController : ICreateOrderController
6     {
7         readonly ICreateOrderInputPort _inputPort;
8         readonly ICreateOrderPresenter _presenter;
9
10        //public CreateOrderController(ICreateOrderInputPort inputPort,
11        //    ICreateOrderPresenter presenter)
12        //{
13        //    _inputPort = inputPort;
14        //    _presenter = presenter;
15        //}
16        public CreateOrderController(ICreateOrderInputPort inputPort,
17            ICreateOrderPresenter presenter) => (_presenter, _inputPort) =
18            (presenter, inputPort); //Expresión lamda.
19
20        public async Task<WrappersSalesOrder> CreateOrder(OrderHeaderDto
21            order)
22        {
23            await _inputPort.Handle(order);
24            return _presenter.Order;
25        }
26    }
27 }
```

```
1 namespace LaPaulita.Sales.Controllers
2 {
3     public static class DependencyContainers
4     {
5         public static IServiceCollection AddServicesSalesControllers(
6             this IServiceCollection services)
7         {
8             services.AddScoped<ICreateOrderController,
9                 CreateOrderController>();
10            return services;
11        }
12    }
13 }
```

```
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
```

```
1 namespace LaPaulita.Sales.Repositories.EFCore.Configurations
2 {
3     public class OrderDetailConfiguration : 
4         IEntityTypeConfiguration<OrderDetail>
5     {
6         public void Configure(EntityTypeBuilder<OrderDetail> builder)
7         {
8             builder.HasKey(d => new { d.Id, d.ProductId });
9             builder.Property(d => d.Id)
10                 .UseIdentityColumn();
11             builder.Property(d => d.ProductPrice)
12                 .HasPrecision(8, 2);
13         }
14     }
15 }
```

```
1 namespace LaPaulita.Sales.Repositories.EFCore.Configurations
2 {
3     public class OrderHeaderConfiguration : IEntityTypeConfiguration<OrderHeader>
4     {
5         public void Configure(EntityTypeBuilder<OrderHeader> builder)
6         {
7             builder.Property(o => o.ClientId)
8                 .IsRequired()
9                 .HasMaxLength(5);
10
11             builder.Property(o => o.ShippingAddress)
12                 .IsRequired()
13                 .HasMaxLength(50);
14
15             builder.Property(o => o.ShippingCity)
16                 .IsRequired();
17
18             builder.Property(o => o.ShippingCountry)
19                 .IsRequired();
20
21             builder.Property(o => o.ShippingZip)
22                 .HasMaxLength(4);
23         }
24     }
25 }
26
```

```
...Sales.Repository\EFCore\Context\LaPaulitaContext.cs 1
1 namespace LaPaulita.Sales.Repositories.EFCore.Context
2 {
3     internal class LaPaulitaContext : DbContext
4     {
5         protected override void OnConfiguring(DbContextOptionsBuilder ➤
6             optionsBuilder)
7         {
8             optionsBuilder.UseSqlServer("Data Source=DanielPagano; ➤
9                 Initial Catalog=LaPaulitaDB; User ID=sa; ➤
10                 Password=MsSqlServer; TrustServerCertificate=True");
11
12             //"Data Source=(LocalDb)\\MSSQLLocalDB;Initial ➤
13                 Catalog=LaPaulitaDB;Integrated Security=SSPI;"
14         }
15
16         public DbSet<OrderHeader> OrderHeaders { get; set; }
17         public DbSet<OrderDetail> OrderDetails { get; set; }
18
19         protected override void OnModelCreating(ModelBuilder ➤
20             modelBuilder)
21         {
22             modelBuilder.ApplyConfigurationsFromAssembly(
23                 Assembly.GetExecutingAssembly());
24         }
25     }
26 }
```

```
....Repository\EFCore\Context\LaPaulitaSalesContext.cs 1
1 namespace LaPaulita.Sales.Repositories.EFCore.Context
2 {
3     public class LaPaulitaSalesContext : DbContext
4     {
5         public LaPaulitaSalesContext
6             (DbContextOptions<LaPaulitaSalesContext> options)
7             : base(options) { }
8
9         public DbSet<OrderHeader> OrderHeaders { get; set; }
10        public DbSet<OrderDetail> OrderDetails { get; set; }
11
12        protected override void OnModelCreating(ModelBuilder
13            modelBuilder)
14        {
15            modelBuilder.ApplyConfigurationsFromAssembly(
16                Assembly.GetExecutingAssembly());
17        }
18    }
19 }
```

```
...a\LaPaulita.Sales.Repository\EFCore\Migraciones.txt 1
1 Crear la migración
2 Add-Migration InitialCreate -p LaPaulita.Sales.Repositories -s 2
   LaPaulita.Sales.Repositories -c LaPaulitaContext
3
4 Actualización de la DB
5 Update-Database -p LaPaulita.Sales.Repositories -s 2
   LaPaulita.Sales.Repositories -context LaPaulitaContext
6
7
8 Cadena de conexión
9 Data Source=(LocalDb)\MSSQLLocalDB;Initial 2
   Catalog=LaPaulitaDB;Integrated Security=SSPI;
10
11 {
12     "clientId": 1,
13     "shippingAddress": "La Pampa 132",
14     "shippingCity": 1,
15     "shippingCountry": 1,
16     "shippingZip": "5620",
17     "orderDetails": [
18         {
19             "productId": 1,
20             "productPrice": 175.26,
21             "productQuantity": 1
22         },
23         {
24             "productId": 1,
25             "productPrice": 175.26,
26             "productQuantity": 5
27         },
28         {
29             "productId": 2,
30             "productPrice": 375.26,
31             "productQuantity": 1
32         }
33     ]
34 }
```



```
1 namespace LaPaulita.Sales.Repositories.Repositories
2 {
3     public class SalesCommandRepository : ISalesCommandRepository
4     {
5         readonly LaPaulitaSalesContext _context;
6
7         public SalesCommandRepository(LaPaulitaSalesContext context)
8         {
9             _context = context;
10        }
11
12        public async Task CreateOrder(CreateOrder order)
13        {
14            await _context.AddAsync(order);
15            foreach (var item in order.Details)
16            {
17                await _context.AddAsync(new OrderDetail
18                {
19                    Order = order,
20                    ProductId = item.ProductId,
21                    ProductPrice = item.ProductPrice,
22                    ProductQuantity = item.ProductQuantity
23                });
24            }
25        }
26
27        public async Task SaveChange()
28        {
29            await _context.SaveChangesAsync();
30        }
31    }
32 }
33
```

```
1 namespace LaPaulita.Sales.Repositories
2 {
3     public static class DependencyContainer
4     {
5         public static IServiceCollection AddRepositories(
6             this IServiceCollection services,
7             IConfiguration configuration,
8             string connectionStringName)
9         {
10             services.AddDbContext<LaPaulitaSalesContext>(options =>
11                 options.UseSqlServer(configuration
12                     .GetConnectionString(connectionStringName)));
13
14             services.AddScoped<ISalesCommandRepository,
15                 SalesCommandRepository>();
16
17             return services;
18         }
19     }
20 }
21
```

```
1 using LaPaulita.Sales.BusinessRules.Wrappers;
2
3 namespace LaPaulita.Sales.Presenters
4 {
5     public class CreateOrderPresenter : ICreateOrderPresenter
6     {
7         //public int OrderId { get; private set; }
8
9         //public List<ValidationErrorDto> ErrorsList { get; private set; }
10
11         public WrappersSalesOrder Order { get; private set; } = new WrappersSalesOrder();
12
13         //public ValueTask Handle(int orderId)
14         //{
15         //    OrderId = orderId;
16         //    return ValueTask.CompletedTask;
17         //}
18
19         public ValueTask Handle(WrappersSalesOrder order)
20         {
21             Order.OrderId = order.OrderId;
22             Order.Errors = order.Errors;
23             return ValueTask.CompletedTask;
24         }
25
26         public ValueTask ValidationFailed(List<ValidationErrorDto> validationError)
27         {
28             Order.Errors = validationError;
29             return ValueTask.CompletedTask;
30         }
31     }
32 }
33
```

```
1 namespace LaPaulita.Sales.Presenters
2 {
3     public static class DependencyContainer
4     {
5         public static IServiceCollection AddServicesPresenter(this IServiceCollection services)
6         {
7             services.AddScoped<CreateOrderPresenter>();
8
9             services.AddScoped<ICreateOrderOutputPort,
10                 CreateOrderPresenter>();
11
12             services.AddScoped<ICreateOrderPresenter,
13                 CreateOrderPresenter>();
14
15             return services;
16         }
17     }
```

```
1 using LaPaulita.Sales.BusinessRules.DTOs;
2 using LaPaulita.Sales.BusinessRules.Interface.Controllers;
3
4 namespace LaPaulita.Sales.WebApi
5 {
6     public static class EndPoints
7     {
8         public static WebApplication LaPaulitaSalesEndPoint(this WebApplication app)
9         {
10             app.MapPost("/create", async (OrderHeaderDto order,
11                 ICreateOrderController controller) =>
12             {
13                 var result = await controller.CreateOrder(order);
14                 if (result.Errors != null && result.Errors.Count > 0)
15                 {
16                     return Results.BadRequest(result);
17                 }
18                 else
19                 {
20                     return Results.Ok(result);
21                 }
22             });
23         }
24         return app;
25     }
26 }
27
28
29
```

```
1 using DependencyInversion;
2
3 namespace LaPaulita.Sales.WebApi
4 {
5     public static class WebApplicationHelper
6     {
7         public static WebApplication CreateWebApplication(this WebApplicationBuilder builder)
8         {
9             // Configurar APIExplorer para descubrir y exponer
10             // los metadatos de los endpoints de la aplicación.
11             builder.Services.AddEndpointsApiExplorer();
12
13             // Agregar el generador que construye los objetos de
14             // documentación de Swagger con la funcionalidad del
15             // APIExplorer.
16             builder.Services.AddSwaggerGen();
17
18             // Registrar los servicios de la aplicación
19             builder.Services.AddLaPaulitaSalesServices(
20                 builder.Configuration, "LaPaulitaDB");
21
22             // Agregar el servicio CORS para clientes que se ejecutan
23             // en el navegador Web (como Blazor WebAssembly).
24             builder.Services.AddCors(options =>
25             {
26                 options.AddDefaultPolicy(config =>
27                 {
28                     config.AllowAnyMethod();
29                     config.AllowAnyHeader();
30                     config.AllowAnyOrigin();
31                 });
32             });
33
34             return builder.Build();
35         }
36         public static WebApplication ConfigureWebApplication(
37             this WebApplication app)
38         {
39             // Habilitar el middleware para servir el documento
40             // JSON generado y la interfaz UI de Swagger en el
41             // ambiente de desarrollo.
42             if (app.Environment.IsDevelopment())
43             {
44                 app.UseSwagger();
45                 app.UseSwaggerUI();
46             }
47
48             // Registrar los endpoints de la aplicación
49             app.LaPaulitaSalesEndPoint();
50
51             // Agregar el Middleware CORS
52             app.UseCors();
```

```
...opia\LaPaulita.Sales.WebApi\WebApplicationHelper.cs 2
```

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

```
...LaPaulita - copia\LaPaulita.Sales.WebApi\Program.cs 1
```

```
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();
```

```
...ita - copia\LaPaulita.Sales.WebApi\appsettings.json 1
```

```
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 }
```