Documentacion del Proyecto Sport Shop

Este Proyecto basado en realización de ordenes de pedidos de una tienda de deportes usando el framework de serenity, creado con el fin de que el admin cree las ordenes de los clientes y los clientes puedan solo pueden ver el estatus de la orden en tiempo real que hicieron en el local.

Donde se darán los detalles de funcionamiento que se usaron para elaborar el proyecto que serían:

En el apartado de migraciones esta:

DefaultDB_20240123_193200_TableProduct.cs: creación de la tabla product, creación del esquema usado.

```
using FluentMigrator;

namespace FinalPractice.Migrations.DefaultDB
{
    [Migration(20240123193200)]
    0 references
    public class DefaultDB_20240123_193200_TableProduct : AutoReversingMigration
    {
        0 references
        public override void Up()
        {
            Create.Schema("spt");
            Create.Table("Product").InSchema("spt")
            .WithColumn("ProductId").AsInt32().Identity().PrimaryKey().NotNullable()
            .WithColumn("Title").AsString(100).NotNullable()
            .WithColumn("Price").AsInt32().NotNullable();
        }
    }
}
```

DefaultDB_20240124_185400_TableCustomer.cs: Creación de la tabla de customers.

```cs
using FluentMigrator;

namespace FinalPractice.Migrations.DefaultDB
{
    [Migration(20240124185400)]
    0 references
    public class DefaultDB_20240124_185400_TableCustomer : AutoReversingMigration
    {
        0 references
        public override void Up()
        {
            Create.Table("Customer").InSchema("spt")
            .WithColumn("CustomerId").AsInt32().Identity().PrimaryKey().NotNullable()
            .WithColumn("Firstname").AsString(50).NotNullable()
            .WithColumn("Lastname").AsString(50).NotNullable()
            .WithColumn("Gender").AsInt32().NotNullable()
            .WithColumn("Email").AsString();
        }
    }
}
```

DefaultDB_20240124_185500_TableOrder.cs: Creación de la tabla de order.

```cs
using FluentMigrator;

namespace FinalPractice.Migrations.DefaultDB
{
    [Migration(20240124185500)]
    0 references
    public class DefaultDB_20240124_185500_TableOrder : AutoReversingMigration
    {
        0 references
        public override void Up()
        {
            Create.Table("Order").InSchema("spt")
            .WithColumn("OrderId").AsInt32().Identity().PrimaryKey().NotNullable()
            .WithColumn("CutormerId").AsInt32().NotNullable()
            .ForeignKey("FK_Order_CustomerId", "spt", "customer", "CustomerId")
            .WithColumn("Status").AsInt32().NotNullable().WithDefaultValue(1)
            .WithColumn("ReleaseDate").AsDateTime().Nullable()
            .WithColumn("Total").AsDouble().NotNullable();

        }
    }
}
```

DefaultDB_20240124_190200_TableOrderDetails.cs: Creación de la tabla OrderDetails.

```
using FluentMigrator;

namespace FinalPractice.Migrations.DefaultDB
{
    [Migration(20240124190200)]
    0 references
    public class DefaultDB_20240124_190200_TableOrderDetails : AutoReversingMigration
    {
        0 references
        public override void Up()
        {
            Create.Table("OrderDetails").InSchema("spt")
            .WithColumn("OrderDetailsId").AsInt32().Identity()
            .PrimaryKey().NotNullable()
            .WithColumn("OrderId").AsInt32().NotNullable()
            .ForeignKey("FK_OrderDetails_OrderId", "spt", "Order", "OrderId")
            .WithColumn("ProductId").AsInt32().NotNullable()
            .ForeignKey("FK_OrderDetails_CustomerId", "spt", "Product", "ProductId")
            .WithColumn("Quantity").AsInt32().NotNullable();
        }
    }
}
```

DefaultDB_20240125_174900_UpdateProduct.cs: elimina y agrega a la tabla producto una columna

Pero con otro dato.

```
using FluentMigrator;

namespace FinalPractice.Migrations.DefaultDB
{
    [Migration(20240125174900)]
    0 references
    public class DefaultDB_20240125_174900_UpdateProduct : Migration
    {
        0 references
        public override void Up()
        {
            Delete.Column("Price").FromTable("Product").InSchema("spt");

            Alter.Table("Product").InSchema("spt")
            .AddColumn("Price").AsDouble().Nullable();
        }
        0 references
        public override void Down()
        {
        }
    }
}
```

DefaultDB_20240125_202300_UpdateOrder.cs: elimina Columna de la tabla order.

```
using FluentMigrator;

namespace FinalPractice.Migrations.DefaultDB
{
    [Migration(20240125202300)]
    0 references
    public class DefaultDB_20240125_202300_UpdateOrder : Migration
    {
        0 references
        public override void Up()
        {
            Delete.Column("Total").FromTable("Order").InSchema("spt");
        }
        0 references
        public override void Down()
        {
        }
    }
}
```

DefaultDB_20240125_230200_UpdateOrderDetails.cs: Agrega a la tabla OrderDetails una nueva columna.

```
using FluentMigrator;

namespace FinalPractice.Migrations.DefaultDB
{
    [Migration(20240125230200)]
    0 references
    public class DefaultDB_20240125_230200_UpdateOrderDetails : AutoReversingMigration
    {
        0 references
        public override void Up()
        {
            Alter.Table("OrderDetails").InSchema("spt")
            .AddColumn("UnitPrice").AsDouble().Nullable();
        }
    }
}
```

DefaultDB_20240126_191100_UpdateCustomer.cs: Agrega a la tabla Customer una nueva columna.

```
using FluentMigrator;

namespace FinalPractice.Migrations.DefaultDB
{
    [Migration(20240126191100)]
    0 references
    public class DefaultDB_20240126_191100_UpdateCustomer : AutoReversingMigration
    {
        0 references
        public override void Up()
        {
            Alter.Table("Customer").InSchema("spt")
            .AddColumn("UserId").AsInt32().Nullable()
            .ForeignKey("FK_Customer_UserId","dbo", "Users", "UserId");
        }
    }
}
```

DefaultDB_20240126_221100_UpdateOrder2.cs: Se agrega otra columna a la tabla order.

```
using FluentMigrator;

namespace FinalPractice.Migrations.DefaultDB
{
    [Migration(20240126221100)]
    0 references
    public class DefaultDB_20240126_221100_UpdateOrder2 : AutoReversingMigration
    {
        0 references
        public override void Up()
        {
            Alter.Table("Order").InSchema("spt")
            .AddColumn("ShipCity").AsInt32().Nullable();
        }
    }
}
```

DefaultDB_20240126_230100_UpdateCustomer2.cs: Se le agrega otra columna a la tabla customer.

```
using FluentMigrator;

namespace FinalPractice.Migrations.DefaultDB
{
    [Migration(20240126230100)]
    0 references
    public class DefaultDB_20240126_230100_UpdateCustomer2 : AutoReversingMigration
    {
        0 references
        public override void Up()
        {
            Alter.Table("Customer").InSchema("spt")
            .AddColumn("Address").AsString(255).NotNullable()
            .WithDefaultValue("");
        }
    }
}
```

Uso de la aplicación sergen en el módulo **SportDB,** generando los siguientes códigos de las siguientes tablas previamente creadas en las migraciones:

Customer: Row, Service, UI, Custom

Order: Row, Service, UI, Custom

OrderDetails: Row, Service, Custom

Product: Row, Service, UI, Custom

En ../Module/SportDB esta lo creado por sergen y de esos datos se alteraron o agregados son estos:

Customer

CustomerColumn.cs: se le agrego Fullname y email definidos en CustomerRow.

```
namespace FinalPractice.SportDB.Columns
{
    using Serenity;
    using Serenity.ComponentModel;
    using Serenity.Data;
    using System;
    using System.ComponentModel;
    using System.Collections.Generic;
    using System.IO;

    [ColumnsScript("SportDB.Customer")]
    [BasedOnRow(typeof(Entities.CustomerRow), CheckNames = true)]
    0 references
    public class CustomerColumns
    {
        [EditLink, DisplayName("Db.Shared.RecordId"), AlignRight]
        0 references
        public Int32 CustomerId { get; set; }
        //Show full name
        [EditLink]
        0 references
        public String Fullname { get; set; }
        //show user Email
        0 references
        public String Email { get; set; }
        0 references
        public Gender Gender { get; set; }
        0 references
        public String Address { get; set; }
    }
}
```

CustomerDialog.ts: se agregó una función para pestaña para ver las ordenes en el formulario de Customer y un filtro que solo muestra las ordenes que tiene ese customer.

```
private orderGrid: CustomerOrderGrid;

constructor() {
    super();
    //show another tap in CustomerForm
    this.orderGrid = new CustomerOrderGrid(this.byId("OrderGrid"));
    this.tabs.on('tabsactivate', (e, i) => {
        this.arrange();
    });
}

//filtering orders by customerId
protected afterLoadEntity() {
    super.afterLoadEntity();
    this.orderGrid.customerId = this.entityId;
}
```

CustomerForm.cs: se le agrego los atributos category para mostrar una división en el formulario y la selección de genero.

```
namespace FinalPractice.SportDB.Forms
{
    using Serenity;
    using Serenity.ComponentModel;
    using Serenity.Data;
    using System;
    using System.ComponentModel;
    using System.Collections.Generic;
    using System.IO;

    [FormScript("SportDB.Customer")]
    [BasedOnRow(typeof(Entities.CustomerRow), CheckNames = true)]
    0 references
    public class CustomerForm
    {
        [Category("Customer")]
        0 references
        public String Firstname { get; set; }
        0 references
        public String Lastname { get; set; }
        //select gender in the form
        0 references
        public Gender Gender { get; set; }
        [Category("Other details")]
        0 references
        public String Address { get; set; }
    }
}
```

CustomerOrderColumns.cs: Formulario para las ordenes que están en la pestaña del formulario Customer.

```
namespace FinalPractice.SportDB.Columns
{
    using Serenity;
    using Serenity.ComponentModel;
    using Serenity.Data;
    using System;
    using System.ComponentModel;
    using System.Collections.Generic;
    using System.IO;

    // form for order tab in CustomerForm
    [ColumnsScript("SportDB.CustomerOrder")]
    [BasedOnRow(typeof(Entities.OrderRow))]
    0 references
    public class CustomerOrderColumns
    {
        [EditLink, DisplayName("Db.Shared.RecordId"), AlignRight]
        0 references
        public Int32 OrderId { get; set; }
        [EditLink, DisplayName("Full Name")]
        0 references
        public String CustomerFullname { get; set; }
        0 references
        public OrderStatus Status { get; set; }
        [Width(120)]
        0 references
        public OrderCity ShipCity { get; set; }
        0 references
        public DateTime ReleaseDate { get; set; }
    }
}
```

CustomerOrderDialog.ts: dialog que muestra el dato seleccionado en la pestaña de order en el formulario de Customer.

```
1   /// <reference path="../Order/OrderDialog.ts" />
2
3   namespace FinalPractice.SportDB {
4
5       //Use for show details in customerOrder
6
7       @Serenity.Decorators.registerClass()
8       export class CustomerOrderDialog extends OrderDialog {
9
10          constructor() {
11              super();
12          }
13
14          updateInterface() {
15              super.updateInterface();
16
17              Serenity.EditorUtils.setReadOnly(this.form.CutormerId, true);
18          }
19      }
20  }
21
```

CustomerOrderGrid.ts: Grid de la pestaña de ordenes en el formulario de Customer

```
namespace FinalPractice.SportDB {
    @Serenity.Decorators.registerClass()
    export class CustomerOrderGrid extends Serenity.EntityGrid<OrderRow, any>
    {
        protected getColumnsKey() { return "SportDB.CustomerOrder"; }
        protected getIdProperty() { return OrderRow.idProperty; }

        //see details in CustomerOrder
        protected getDialogType() { return CustomerOrderDialog; }

        protected getLocalTextPrefix() { return OrderRow.localTextPrefix; }
        protected getService() { return OrderService.baseUrl; }
        constructor(container: JQuery) {
            super(container);
        }
        //disable buttons in order tap in CustomerForm
        protected getButtons() {
            return null;
        }
        protected getInitialTitle() {
            return null;
        }
        protected usePager() {
            return false;
        }

        protected getGridCanLoad() {
            return super.getGridCanLoad() && !!this.customerId;
        }

        private _customerId: number;

        get customerId() {
            return this._customerId;
        }

        //validation filtering orders by customerId
        set customerId(value: number) {
            if (this._customerId != value) {
                this._customerId = value;
                this.setEquality(OrderRow.Fields.CutormerId, value);
                this.refresh();
            }
        }
    }
}

```

CustomerRepository.cs: se edito la función `ListRequestHandler` para que filtre si el usuario logueado no tiene permisos de admin solo muestre el customer de ese usuario.

```
43      private class MyListHandler : ListRequestHandler<MyRow> {
44
            0 references
45          protected override void ApplyFilters(SqlQuery query)
46          {
47
48              base.ApplyFilters(query);
49              //user Logged in
50              var user = (UserDefinition)Authorization.UserDefinition;
51
52              //condition to know if you have general permissions if it does not show you your orders
53
54              if (!Authorization.HasPermission(PermissionKeys.General))
55                  query.Where(fld.CustomerId == user.UserId);
56          }
57      }
```

CustomerRow.cs: cambio de permisos de View solo podrán leer los detalles, declaración de valores que no trajo sergen

```
14
15      //only read permission
16      [ReadPermission(PermissionKeys.Customer.View)]
17
18      [ModifyPermission("Administration:General")]
```

```
43      //Concat the firstname and lastname
44
45      [DisplayName("Full Name")]
46      [Expression("(t0.Firstname + ' ' + t0.Lastname)"), QuickSearch]
        0 references
47      public String Fullname
48      {
49          get { return Fields.Fullname[this]; }
50          set { Fields.Fullname[this] = value; }
51      }
52      // user email
53      [DisplayName("Email")]
54      [Expression("jUser.[Email]")]
        0 references
55      public String Email
56      {
57          get { return Fields.Email[this]; }
58          set { Fields.Email[this] = value; }
59      }
60
61      // customer gender (male or female)
62      [DisplayName("Gender"), NotNull]
        0 references
63      public Gender? Gender
64      {
65          get { return (Gender?)Fields.Gender[this]; }
66          set { Fields.Gender[this] = (Int32?)value; }
67      }
68      // Address
69      [DisplayName("Address"),Width(200),DefaultValue("none")]
        0 references
70      public String Address
71      {
72          get { return Fields.Address[this]; }
73          set { Fields.Address[this] = value; }
74      }
75
76      //concat Fullname and Username
77      [Expression("(t0.Firstname + ' ' + t0.Lastname +' ' + '('+ jUser.[Username] +')')")]
        0 references
78      public String FullUsername
79      {
80          get { return Fields.FullUsername[this]; }
81          set { Fields.FullUsername[this] = value; }
82      }
```

```
83          // id user
84          [DisplayName("User"), ForeignKey("[dbo].[Users]", "UserId"), LeftJoin("jUser"), TextualField("")]
85          [Expression("t0.[CustomerId]")]
86
            0 references
87          public Int32? UserId
88          {
89              get { return Fields.UserId[this]; }
90              set { Fields.UserId[this] = value; }
91          }
            0 references
92          IIdField IIdRow.IdField
93          {
94              get { return Fields.CustomerId; }
95          }
96
            0 references
97          StringField INameRow.NameField
98          {
99              //show in Lookup the "fullname (username)"
100             get { return Fields.FullUsername; }
101         }
102
103         public static readonly RowFields Fields = new RowFields().Init();
104
            0 references
105         public CustomerRow()
106             : base(Fields)
107         {
108         }
109
            3 references
110         public class RowFields : RowFieldsBase
111         {
112             public Int32Field CustomerId;
113             public Int32Field UserId;
114             public StringField Firstname;
115             public StringField Lastname;
116
117             public StringField Fullname;
118             public StringField FullUsername;
119             public StringField Address;
120             public Int32Field Gender;
121             public StringField Email;
122         }
123     }
124 }
```

Gender.cs: un enum para de los géneros usado en Gender en CustomerRow.

```
1   using Serenity.ComponentModel;
2   using System.ComponentModel;
3   namespace FinalPractice.SportDB
4   {
5       //enum of various Gender
6
7       [EnumKey("SportDB.Gender")]
            4 references
8       public enum Gender
9       {
10          [Description("Male")]
11          Male = 1,
12          [Description("Female")]
13          Female = 2
14      }
15  }
```

SportDB.CustomerDialog.Template.html: Plantilla para crear la pestaña de orders en el formulario de Customer.

```
<!-- Template to see other tap in CustomerForm -->

<div id="~_Tabs" class="s-DialogContent">
    <ul>
        <li><a href="#~_TabInfo"><span>Customer</span></a></li>
        <li><a href="#~_TabOrder"><span>Orders</span></a></li>
    </ul>
    <div id="~_TabInfo" class="tab-pane s-TabInfo">
        <div id="~_Toolbar" class="s-DialogToolbar">
        </div>
        <div class="s-Form">
            <form id="~_Form" action="">
                <div class="fieldset ui-widget ui-widget-content ui-corner-all">
                    <div id="~_PropertyGrid"></div>
                    <div class="clear"></div>
                </div>
            </form>
        </div>
    </div>
    <div id="~_TabOrder" class="tab-pane s-TabOrder">
        <div id="~_OrderGrid">
        </div>
    </div>
</div>
```

Order

OrderCity.cs: eum para las ciudades usada en ShipCity en OrderRow.

```
using Serenity.ComponentModel;
using System.ComponentModel;
namespace FinalPractice.SportDB
{
    // enum of various City

    [EnumKey("SportDB.City")]
    5 references
    public enum OrderCity
    {
        [Description("none")]
        none = 1,
        [Description("Santo Domingo")]
        SantoDomingo = 2,
        [Description("Santiago")]
        Santiago = 3,
        [Description("Azua")]
        Azua = 4,
        [Description("Samana")]
        Samana = 5,
        [Description("Higuey")]
        Higuey = 6
    }
}
```

OrderColumns.cs: Se agrego valores que no trajo la generación de código de sergen

```
namespace FinalPractice.SportDB.Columns
{
    using Serenity;
    using Serenity.ComponentModel;
    using Serenity.Data;
    using System;
    using System.ComponentModel;
    using System.Collections.Generic;
    using System.IO;

    [ColumnsScript("SportDB.Order")]
    [BasedOnRow(typeof(Entities.OrderRow), CheckNames = true)]
    1 reference
    public class OrderColumns
    {
        [EditLink, DisplayName("Db.Shared.RecordId"), AlignRight]
        0 references
        public Int32 OrderId { get; set; }
        [EditLink, DisplayName("Full Name")]
        0 references
        public String CustomerFullname { get; set; }

        //order status
        0 references
        public OrderStatus Status { get; set; }

        //Order city
        0 references
        public OrderCity ShipCity { get; set; }
        0 references
        public DateTime ReleaseDate { get; set; }
    }
}
```

OrderEndpoint.cs: extrae los datos de la Columnas de Order para Exportarlas a Excel.

```
        // EndPoint Excel
        0 references
        public FileContentResult ListExcel(IDbConnection connection, ListRequest request)
        {
            var data = List(connection, request).Entities;
            var report = new DynamicDataReport(data, request.IncludeColumns, typeof(Columns.OrderColumns));
            var bytes = new ReportRepository().Render(report);
            return ExcelContentResult.Create(bytes, "OrderList_" +
                DateTime.Now.ToString("yyyyMMdd_HHmmss") + ".xlsx");
        }
    }
}
```

OrderForm.cs: se agregó los atributos de category y la lista de productos de OrderDetail.

```
13      //Order Form
14
15      [FormScript("SportDB.Order")]
16      [BasedOnRow(typeof(Entities.OrderRow), CheckNames = true)]
        0 references
17      public class OrderForm
18      {
19          [Category("Order")]
            0 references
20          public Int32 CutormerId { get; set; }
            0 references
21          public OrderStatus Status { get; set; }
            0 references
22          public OrderCity ShipCity { get; set; }
23
24          // current date
25
26          [DefaultValue("now")]
            0 references
27          public DateTime ReleaseDate { get; set; }
28
29          [Category("Order Details")]
30
31          //Product List Form
32
33          [OrderrDetailsEditor, NotMapped]
            0 references
34          public List<Entities.OrderDetailsRow> ProductList { get; set; }
35      }
36  }
```

OrderGrid.ts: Implementación del botón para exportar las columnas a un documento de Excel.

```
16          //button Excel
17          protected getButtons() {
18              var buttons = super.getButtons();
19
20              buttons.push(Common.ExcelExportHelper.createToolButton({
21                  grid: this,
22                  service: OrderService.baseUrl + '/ListExcel',
23                  onViewSubmit: () => this.onViewSubmit(),
24                  separator: true
25              }));
26              return buttons;
27          }
```

OrderRepository.cs: se editó la función `ListRequestHandler para que filtre si el usuario logueado no tiene permisos de admin solo muestre la orden del usuario.`

```
44      private class MyListHandler : ListRequestHandler<MyRow> {
            0 references
45          protected override void ApplyFilters(SqlQuery query)
46          {
47
48              base.ApplyFilters(query);
49              //user Logged in
50              var user = (UserDefinition)Authorization.UserDefinition;
51
52              //condition to know if you have general permissions if it does not show you your orders
53
54              if (!Authorization.HasPermission(PermissionKeys.General))
55                  query.Where(fld.CutormerId == user.UserId);
56          }
57      }
```

`OrderRow.cs:` cambio de permisos de View solo podrán leer los detalles, declaración de valores que no trajo sergen.

```
16      //Permision Only read
17      [ReadPermission(PermissionKeys.Customer.View)]
18
```

```
31          //Lookup Customer
32          [LookupEditor(typeof(CustomerRow))]
            0 references
33          public Int32? CutormerId
34          {
35              get { return Fields.CutormerId[this]; }
36              set { Fields.CutormerId[this] = value; }
37          }
38          // Order Status (In Progress, In to deliver or Delivered)
39          [DisplayName("Status"), NotNull, DefaultValue(1),QuickFilter]
40          [Width(100)]
            0 references
41          public OrderStatus? Status
42          {
43              get { return (OrderStatus?)Fields.Status[this]; }
44              set { Fields.Status[this] = (Int32?)value; }
45          }
46          // show cities (santo domingo, azua,...)
47          [Description("Ship City"),DefaultValue(1),QuickFilter]
48          [Width(120)]
            0 references
49          public OrderCity? ShipCity
50          {
51              get { return (OrderCity?)Fields.ShipCity[this]; }
52              set { Fields.ShipCity[this] = (Int32?)value; }
53          }
54
55          [DisplayName("Order Date"),DefaultValue(1), QuickFilter]
            0 references
56          public DateTime? ReleaseDate
57          {
58              get { return Fields.ReleaseDate[this]; }
59              set { Fields.ReleaseDate[this] = value; }
60          }
```

```
76          //Concat customer first name and last name
77          [DisplayName("Customer FullName")]
78          [Expression("(jCutormer.[Firstname] + ' ' + jCutormer.[Lastname])")]
79
            0 references
80          public String CustomerFullname
81          {
82              get { return Fields.CustomerFullname[this]; }
83              set { Fields.CustomerFullname[this] = value; }
84          }
85
86          [DisplayName("Cutormer Gender"), Expression("jCutormer.[Gender]")]
            0 references
87          public Int32? CutormerGender
88          {
89              get { return Fields.CutormerGender[this]; }
90              set { Fields.CutormerGender[this] = value; }
91          }
92
93          // list of products (use only in OrderForm)
94
95          [DisplayName("Product List"), NotMapped]
96          [MasterDetailRelation(foreignKey: "OrderId", IncludeColumns = "ProductName,LineTotal")]
97
            0 references
98          public List<OrderDetailsRow> ProductList
99          {
100             get { return Fields.ProductList[this]; }
101             set { Fields.ProductList[this] = value; }
102         }
```

OrderStatus.cs: enum de los estados de la orden.

```
1   using Serenity.ComponentModel;
2   using System.ComponentModel;
3   namespace FinalPractice.SportDB
4   {
5       // enum of various status
6
7       [EnumKey("SportDB.Order")]
        5 references
8       public enum OrderStatus
9       {
10          [Description("In Progress")]
11          InProgress = 1,
12          [Description("In To Deliver")]
13          InToDeliver = 2,
14          [Description("Delivered")]
15          delivered = 3
16      }
17  }
```

OrderDetails

OrderDetailsColumns.cs: columna que se muestra en el formulario de Order.

```
namespace FinalPractice.SportDB.Columns
{
    using Serenity;
    using Serenity.ComponentModel;
    using Serenity.Data;
    using System;
    using System.ComponentModel;
    using System.Collections.Generic;
    using System.IO;

    [ColumnsScript("SportDB.OrderDetails")]
    [BasedOnRow(typeof(Entities.OrderDetailsRow), CheckNames = true)]
    0 references
    public class OrderDetailsColumns
    {
        //Column product list in the order
        [EditLink,Width(100)]
        0 references
        public Int32 ProductName { get; set; }
        0 references
        public Double UnitPrice { get; set; }
        0 references
        public Int32 Quantity { get; set; }
        0 references
        public Double LineTotal { get; set; }
    }
}
```

OrderDetailsForm.cs: seleccion de productos en el formulario de Order

```
namespace FinalPractice.SportDB.Forms
{
    using Serenity;
    using Serenity.ComponentModel;
    using Serenity.Data;
    using System;
    using System.ComponentModel;
    using System.Collections.Generic;
    using System.IO;

    [FormScript("SportDB.OrderDetails")]
    [BasedOnRow(typeof(Entities.OrderDetailsRow), CheckNames = true)]
    0 references
    public class OrderDetailsForm
    {
        //product selection form, Product List in the Order
        0 references
        public Int32 ProductId { get; set; }
        0 references
        public Double UnitPrice { get; set; }
        0 references
        public Int32 Quantity { get; set; }
    }
}
```

OrderDetailsRow.cs: Uso de lookup y declarando cosas que sergen no hizo.

```
32      [DisplayName("Product"), NotNull, ForeignKey("[spt].[Product]", "ProductId")]
33      [LeftJoin("jProduct"), TextualField("ProductTitle")]
34
35      //Lookup products
36      [LookupEditor(typeof(ProductRow))]
        0 references
37      public Int32? ProductId
38      {
39          get { return Fields.ProductId[this]; }
40          set { Fields.ProductId[this] = value; }
41      }
42
43      [DisplayName("Quantity"), NotNull]
        0 references
44      public Int32? Quantity
45      {
46          get { return Fields.Quantity[this]; }
47          set { Fields.Quantity[this] = value; }
48      }
49
50      [DisplayName("Unit Price"), NotNull]
        0 references
51      public Double? UnitPrice
52      {
53          get { return Fields.UnitPrice[this]; }
54          set { Fields.UnitPrice[this] = value; }
55      }
56      //total product price * quantity
57      [DisplayName("Line Total")]
58      [Expression("(t0.UnitPrice * t0.Quantity)")]
59
        0 references
60      public Double? LineTotal
61      {
62          get { return Fields.LineTotal[this]; }
63          set { Fields.LineTotal[this] = value; }
64      }
65
66      // Product Name
67      [DisplayName("Product Name")]
68      [Expression("jProduct.[Title]")]
        0 references
69      public String ProductName
70      {
71          get { return Fields.ProductName[this]; }
72          set { Fields.ProductName[this] = value; }
73      }
```

OrderrDetailsEditDialog.ts: dialog que muestra en el formulario de Order y función para poner el valor del precio mediante un lookup.

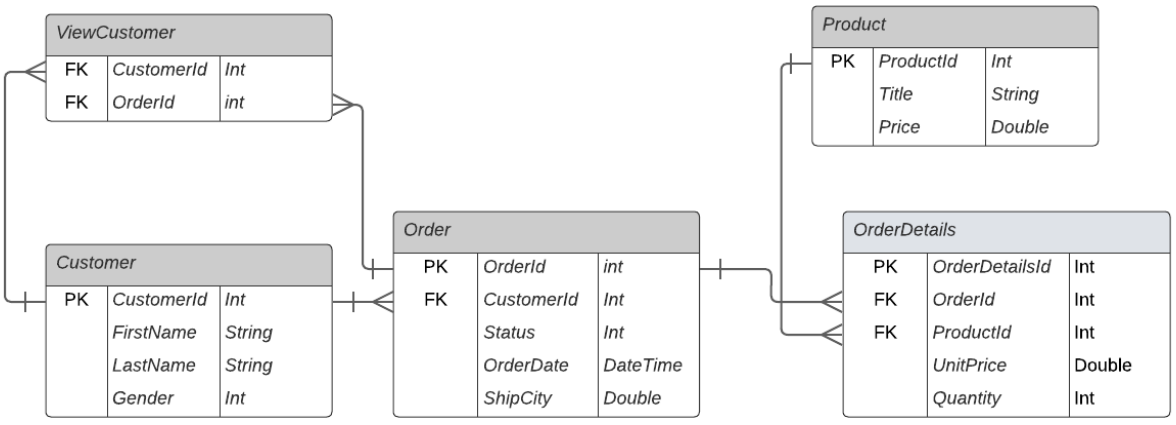
```
/// <reference path="../../Common/Helpers/GridEditorDialog.ts" />


namespace FinalPractice.SportDB {

    @Serenity.Decorators.registerClass()
    export class OrderrDetailsEditDialog extends
        Common.GridEditorDialog<OrderDetailsRow> {
        protected getFormKey() { return OrderDetailsForm.formKey; }
        protected getLocalTextPrefix() { return OrderDetailsRow.localTextPrefix; }
        protected form: OrderDetailsForm;
        constructor() {
            super();
            this.form = new OrderDetailsForm(this.idPrefix);

            //set value to Price in the form product list
            this.form.ProductId.changeSelect2(e => {
                var productId = Q.toId(this.form.ProductId.value);
                if (productId != null) {
                    this.form.UnitPrice.value = ProductRow.getLookup().itemById[productId].Price;
                }
            });
        }

    }
}
```

OrderrDetailsEditor.ts: grid para mostrar en el formulario de Order.

```
/// <reference path="../../Common/Helpers/GridEditorBase.ts" />


namespace FinalPractice.SportDB {

    @Serenity.Decorators.registerEditor()
    export class OrderrDetailsEditor
        extends Common.GridEditorBase<OrderDetailsRow> {
        protected getColumnsKey() { return "SportDB.OrderDetails"; }
        protected getDialogType() { return OrderrDetailsEditDialog }
        protected getLocalTextPrefix() { return OrderDetailsRow.localTextPrefix; }
        constructor(container: JQuery) {
            super(container);
        }
        // add and rename button to add new products
        protected getAddButtonCaption() {
            return "Add Product";
        }

        validateEntity(row, id) {
            row.ProductId = Q.toId(row.ProductId);

            // validation if the product is already there
            var sameProduct = Q.tryFirst(this.view.getItems(), x => x.ProductId === row.ProductId);
            if (sameProduct && this.id(sameProduct) !== id) {
                Q.alert('This product is already in order details!');
                return false;
            }
            //set value to product name using Lookup
            row.ProductName = ProductRow.getLookup().itemById[row.ProductId].Title;
            //set value to price total
            row.LineTotal = (row.Quantity || 0) * (row.UnitPrice || 0);
            Q.notifySuccess("it was executed correctly");
            return true;
        }


    }
}
```

Product

ProductRow.cs: se agrego el atributo lookup y a Price lookupInclude.



```
16      //lookup
17      [LookupScript("SportDB.Product")]
        26 references
18      public sealed class ProductRow : Row, IIdRow, INameRow
19      {
20          [DisplayName("Product Id"), Identity]
            0 references
21          public Int32? ProductId
22          {
23              get { return Fields.ProductId[this]; }
24              set { Fields.ProductId[this] = value; }
25          }
26
27          [DisplayName("Name"), Size(100), NotNull, QuickSearch]
            0 references
28          public String Title
29          {
30              get { return Fields.Title[this]; }
31              set { Fields.Title[this] = value; }
32          }
33          // Lookup include to be able to use the price
34          [DisplayName("Price"), NotNull, LookupInclude]
            0 references
35          public Double? Price
36          {
37              get { return Fields.Price[this]; }
38              set { Fields.Price[this] = value; }
39          }
```

Fuera de esas carpetas esta

SportDBNavigation.cs: orden e iconos que se usaran en el sideBar en la navigation.

```
1   using Serenity.Navigation;
2   using MyPages = FinalPractice.SportDB.Pages;
3
4   // order navigation in de SiteBar
5   [assembly: NavigationMenu(8000, "Sport Shop", icon: "fa-soccer-ball-o")]
6   [assembly: NavigationLink(8000, "Sport Shop/Product", typeof(MyPages.ProductController), icon: "fa-shopping-cart")]
7   [assembly: NavigationLink(8000, "Sport Shop/Customer", typeof(MyPages.CustomerController), icon: "fa-user")]
8   [assembly: NavigationLink(8000, "Sport Shop/Orders", typeof(MyPages.OrderController), icon: "fa-ticket")]
9
```

SportDBPermissionKeys.cs: se definen los permisos que se usan en los archivos.

```
using Serenity.Extensibility;
using System.ComponentModel;

namespace FinalPractice.SportDB
{
    // permissions to be able to manipulate in Sport Shop
    [NestedPermissionKeys]
    [DisplayName("Default")]
    4 references
    public class PermissionKeys
    {
        [DisplayName("Customer")]
        2 references
        public class Customer
        {
            [ImplicitPermission(General), ImplicitPermission(View)]
            public const string Delete = "Default:Customer:Delete";
            [Description("Create/Update"), ImplicitPermission(General), ImplicitPermission(View)]
            public const string Modify = "Default:Customer:Modify";
            public const string View = "Default:Customer:View";
        }

        // permission general in Sport Shop
        [Description("[General]")]
        public const string General = "Default:General";
    }
}
```

Relación de las tablas en la base de datos