

Práctica integradora 2

DAM 2



logo.png

26-02-2020

Desarrollo de Interfaces

Indice

*Introducción.....	3
*Objetivos	4
1.Pasos	4
2.Base de datos.	5
3.Entidad Relación	7
4.POJOS	7
5.Clases Y Interfaces	8
6.Funciones + QUERYS SQL	10
7.Desarrollo De la Interfaz	11
UserControl Items	13
8.Pantalla Principal	13
9.Froms	14
Produstos	14
Botón Añadir	16
Botón Buscar	16
Botón Limpiar	17
Botón Borrar	17
Botón Actualizar	17
Botón Actualizar	17
Ventanas:	19
10.Conclusiones	22

Desarrollo de Interfaces

*Introducción

Se a intentado realizar un proyecto, basado en una en una tienda de pedidos de productos para videojuegos.

Lamentablemente no se a podido finalizar la aplicación de una forma eficiente, ya que el proceso de desarrollo es compleja, ademas de desconocer el por que de numerosos error mientras se trabajaba en ello.

El proyecto se a realizo en varias fases desde la captura y análisis de requisitos hasta la fase de implementación. En este periplo se describirán los detalles de cada fase.

Tomando como referencia el proyecto anterior, se ha decido mantener los funcionamiento y si es posible añadir alguno mas, para mejorar los resultados de la misma.

Se trata de una tienda de pedidos cuya gestión se clientes y proveedores he intentado implementado en la interfaz visual que el usuario puede manejar de manera sencilla y rápida en la que puede insertar, modificar , borrar y actualizar obteniendo así un barios (CRUD).

Aunque el proyecto no esta tan pulido como se hubiera deseado, ha sido una gran forma de aprender como funciona internamente este tipo de aplicaciones, ademas de conocer de primero mano, la sintaxis de uno de los lenguajes mas populares en la industria del software.

Ademas pude trabajar con uno de los IDE mas potentes de Microsoft, estamos hablando del IDE con el mayor apoyo en la comunidad de desarrolladores, Visual Studio.

Esta IDE nos permitirá trabajar de con una mayor facilidad gracias a sus numerosas virtudes en los formularios y establecer un solo entorno de trabajo.

Desarrollo de Interfaces

*Objetivos

El objetivo de la practica tiene como objetivo dar el alumno el conocimiento básico para que puede desenvolverse con soltura a la hora de desarrollar aplicaciones .NET.

Para esto, el alumno tiene que tener cierto conocimiento de base de datos, ya que la aplicación necesita un diseño previo para el desarrollo de un modelo entidad relación, para mas tarde crear la base de datos, con un script o de forma visual.

El objetivo principal es intentar crear una aplicación tipo CRUD con base de datos, que intente ayudar al tipo de negocio seleccionado, esta labor se podrá realizar con una herramienta muy potente llamada DAPPER, que tiene una funcionalidad muy similar a Hibernate en java.

Dapper nos ayudara a realizar operaciones con nuestra base de datos y nuestro POJOS

La practica también tiene el objetivo que el alumno conozca un modelo de estructura de diseño de aplicaciones, este modelo de llama modelo vista controlador, modelo que se utiliza para mantener un orden a la hora de desarrollar ademas de poder poder reciclar código con esta forma.

1.Pasos

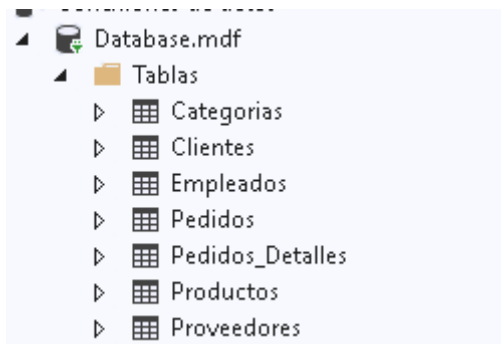
Para comenzar se tiene que tener un orden de acciones, entre ellas se encuentran:

- 1.Realizar la base de datos.
- 2.Crear Nuestros POJOS
- 3.Crear Las diferentes interfaces que hacen que la conexión sea posible y clases dapper.
- 4.Dar al proyecto Nucleo funciones que nos permitan trabajar con la base de datos en la solución.
- 5.Desarrollar una interfaz que nos permita poder trabajar con la base de datos previamente creada.

Desarrollo de Interfaces

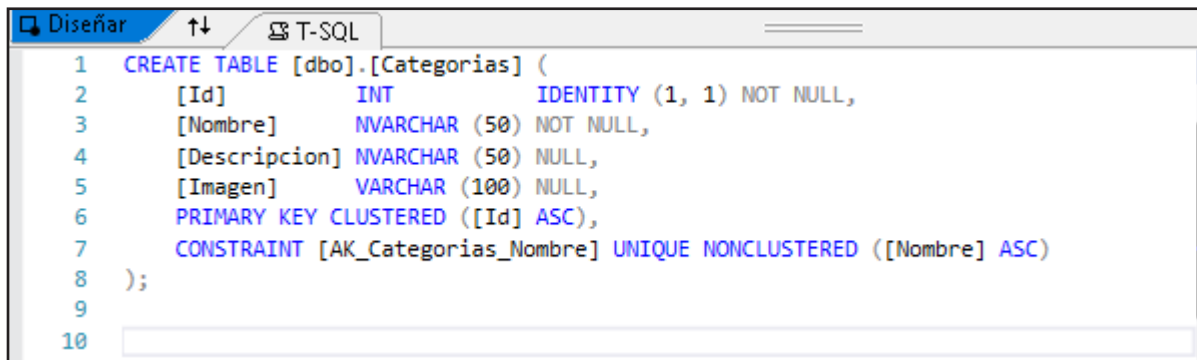
2.Base de datos.

Para la base de datos de han creado las siguientes tablas.

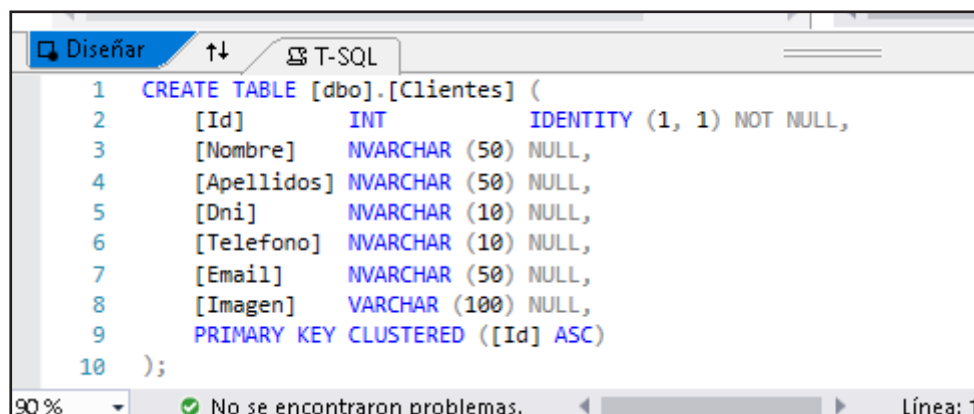


Base_de_datos.PNG

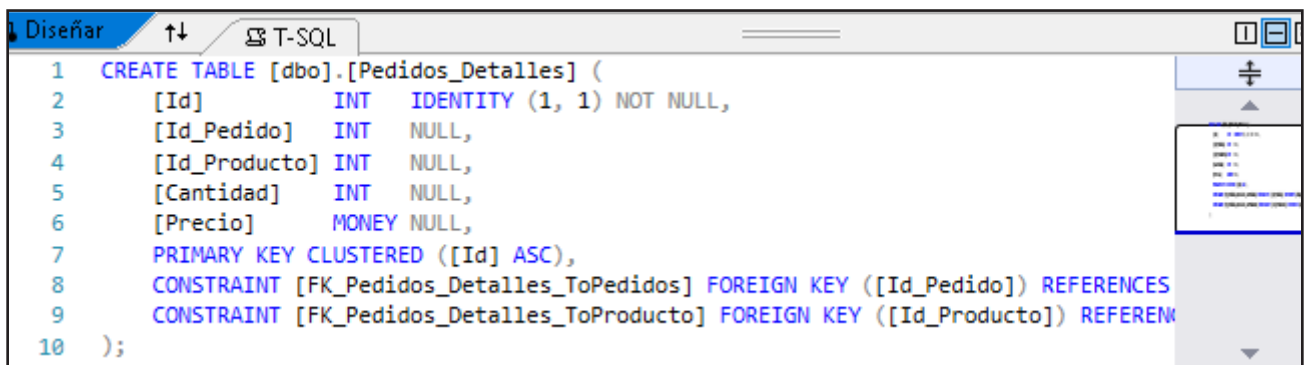
Se realiza de forma visual, como rellenar un formulario en resultado nos genera un script SQL que después se ejecutara.



Categorías_SQL.PNG



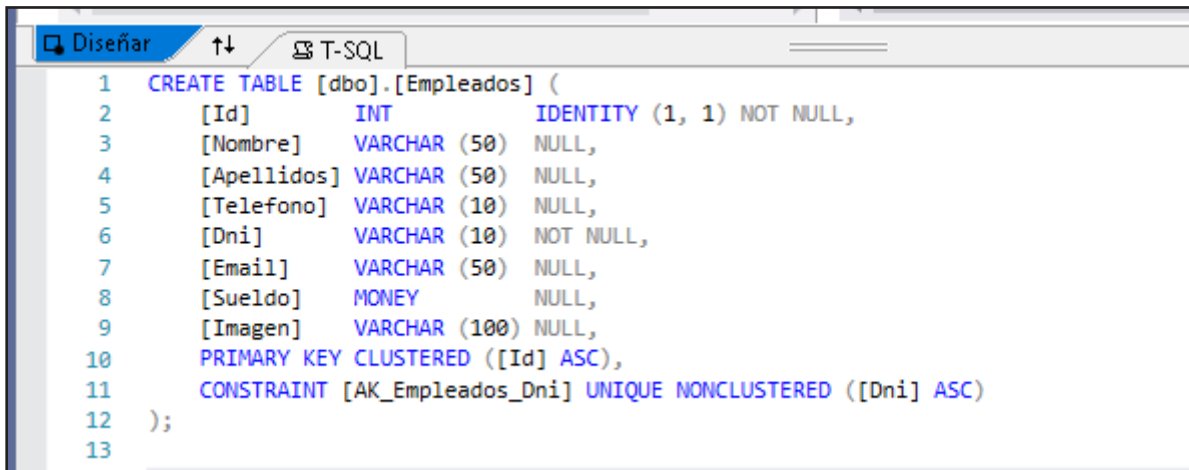
Clientes.PNG



Detalles_Pedido.PNG

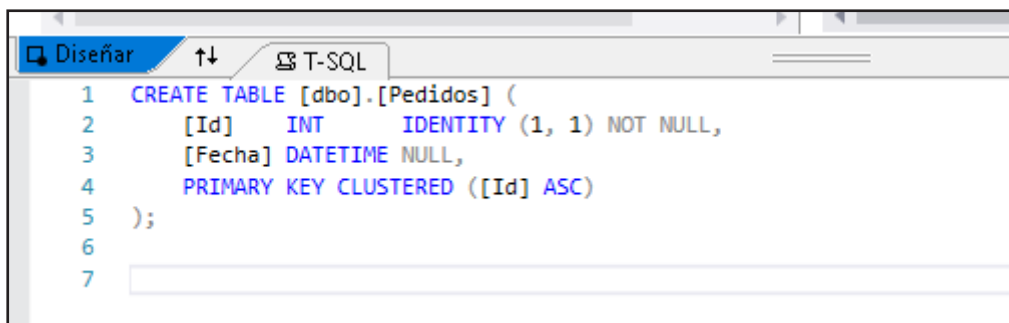
Desarrollo de Interfaces

Con ayuda de Visual Studio podres realizar las tablas de una forma visual, ya que nos proporciona una interfaz de fácil uso.



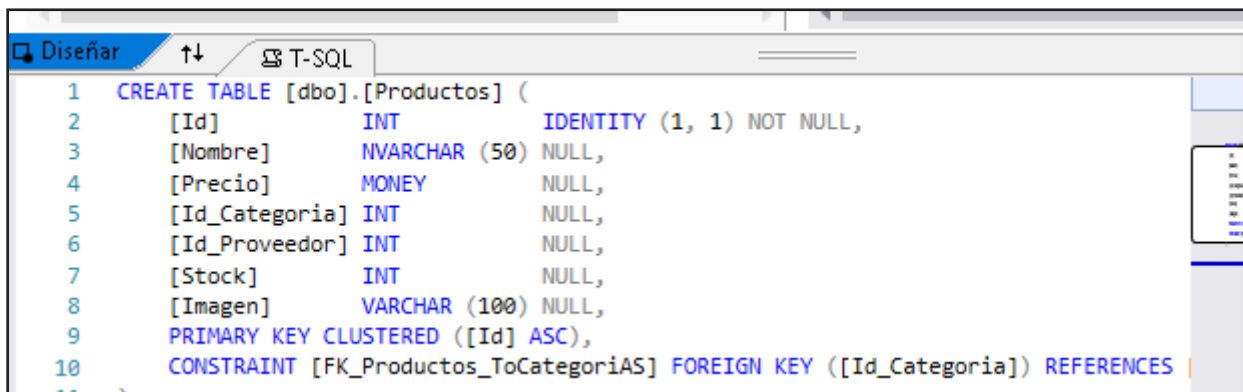
```
1 CREATE TABLE [dbo].[Empleados] (  
2     [Id] INT IDENTITY (1, 1) NOT NULL,  
3     [Nombre] VARCHAR (50) NULL,  
4     [Apellidos] VARCHAR (50) NULL,  
5     [Telefono] VARCHAR (10) NULL,  
6     [Dni] VARCHAR (10) NOT NULL,  
7     [Email] VARCHAR (50) NULL,  
8     [Sueldo] MONEY NULL,  
9     [Imagen] VARCHAR (100) NULL,  
10    PRIMARY KEY CLUSTERED ([Id] ASC),  
11    CONSTRAINT [AK_Empleados_Dni] UNIQUE NONCLUSTERED ([Dni] ASC)  
12 );  
13
```

Empleados.PNG



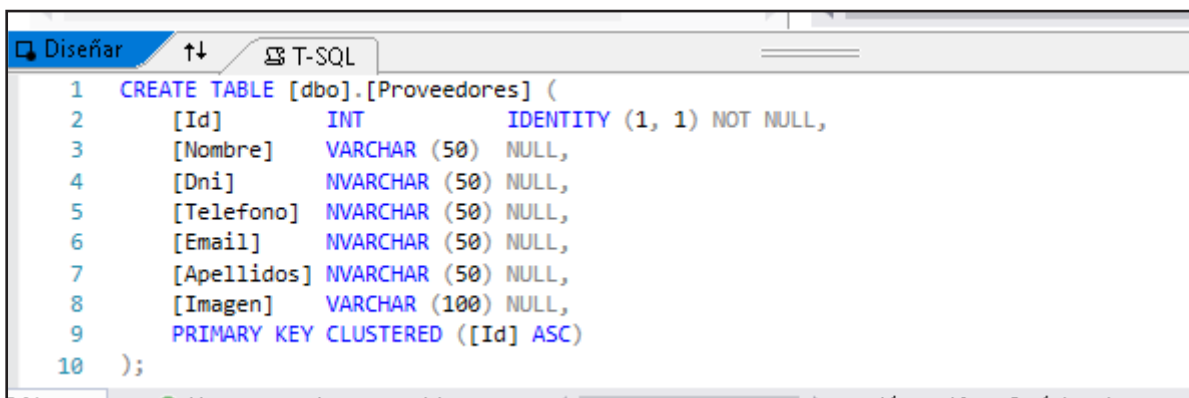
```
1 CREATE TABLE [dbo].[Pedidos] (  
2     [Id] INT IDENTITY (1, 1) NOT NULL,  
3     [Fecha] DATETIME NULL,  
4     PRIMARY KEY CLUSTERED ([Id] ASC)  
5 );  
6  
7
```

Pedidos.PNG



```
1 CREATE TABLE [dbo].[Productos] (  
2     [Id] INT IDENTITY (1, 1) NOT NULL,  
3     [Nombre] NVARCHAR (50) NULL,  
4     [Precio] MONEY NULL,  
5     [Id_Categoria] INT NULL,  
6     [Id_Proveedor] INT NULL,  
7     [Stock] INT NULL,  
8     [Imagen] VARCHAR (100) NULL,  
9     PRIMARY KEY CLUSTERED ([Id] ASC),  
10    CONSTRAINT [FK_Productos_ToCategorias] FOREIGN KEY ([Id_Categoria]) REFERENCES
```

Productos.PNG



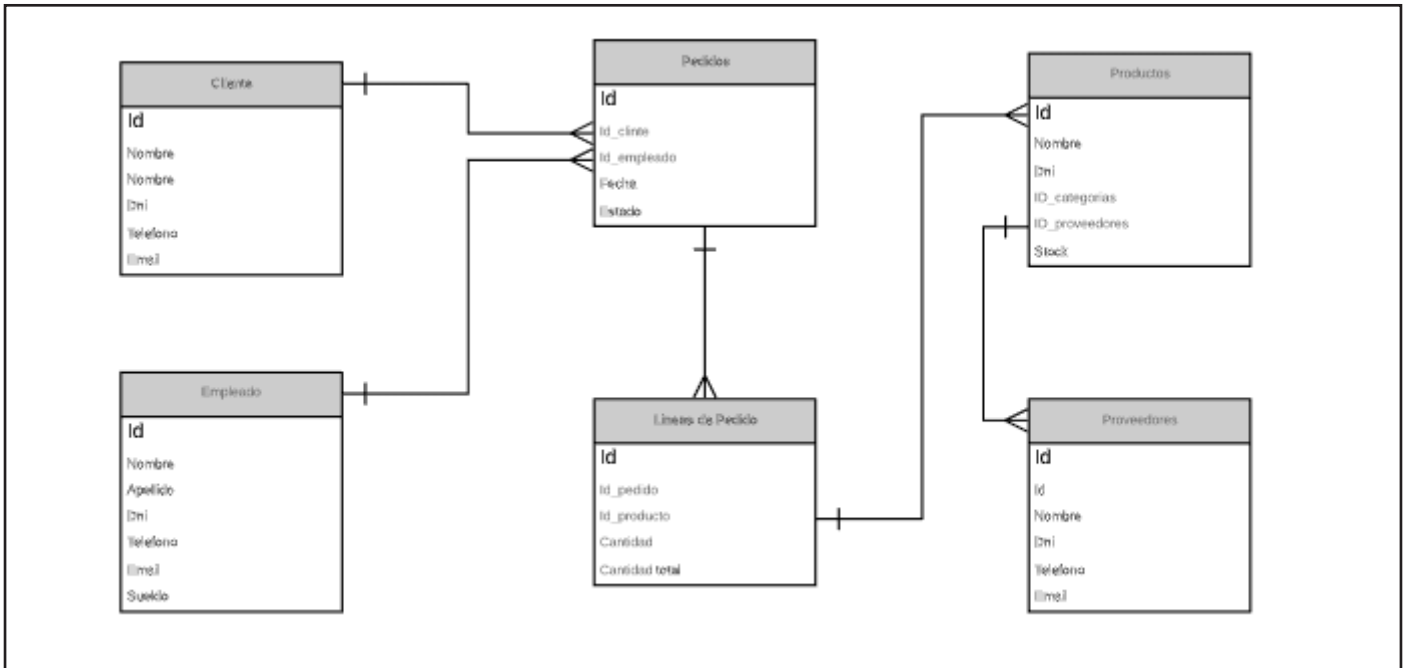
```
1 CREATE TABLE [dbo].[Proveedores] (  
2     [Id] INT IDENTITY (1, 1) NOT NULL,  
3     [Nombre] VARCHAR (50) NULL,  
4     [Dni] NVARCHAR (50) NULL,  
5     [Telefono] NVARCHAR (50) NULL,  
6     [Email] NVARCHAR (50) NULL,  
7     [Apellidos] NVARCHAR (50) NULL,  
8     [Imagen] VARCHAR (100) NULL,  
9     PRIMARY KEY CLUSTERED ([Id] ASC)  
10 );
```

Proveedores.PNG

Desarrollo de Interfaces

3.Entidad Relación

El diagrama entidad relación se a realizado con ayuda de una herramienta Online muy completa de forma online, la herramienta se encuentra en la pagina <https://www.lucidchart.com/>.

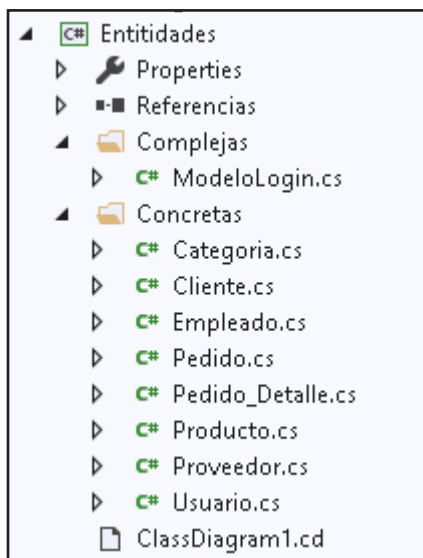


Diagrama_ER.PNG

4.POJOS

Los POJOS se encuentran en el proyecto de Entidades >Concretas, es aquí en donde tendremos que crear nuestras clases de C# que nos permitan seguir con el proyecto.

Los POJOS se crean teniendo como referencia las tablas de nuestra base de datos.



Pojos.PNG

```
1 using Nucleo.Entidades;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace Entidades.Concretas
9 {
10     16 referencias
11     public class Cliente : BaseEntidad, IEntidad
12     {
13         22 referencias
14         public int Id { get; set; }
15         6 referencias
16         public string Nombre { get; set; }
17         4 referencias
18         public string Apellidos { get; set; }
19         3 referencias
20         public string Dni { get; set; }
21         4 referencias
22         public string Telefono { get; set; }
23         4 referencias
24         public string Email { get; set; }
25         6 referencias
26         public string Imagen { get; set; }
27     }
28 }
```

Ejemplo_pojo.PNG

Desarrollo de Interfaces

5.Clases Y Interfaces

Para que la comunicación se produzca con éxito tenemos que crear ciertas clases y interfaces que estarán distribuidas en diferentes partes de la solución.

En este apartado se vera el ejemplo con clientes, es importante resaltar que esto se debe realizar con cada una de las entidades de nuestras tablas.

Para la creación de cada uno de estos es imprescindible haber creado nuestro POJOS.

```
1  using Entidades.Concretas;
2  using Nucleo.AccesoDatos.Abstractas;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace AccesoDatos.Abstractas
10 {
11     4 referencias
12     public interface IClienteDal : IRepositoryEntidad<Cliente>
13     {
14     }
15 }
```

IClienteDal.PNG

```
1  using AccesoDatos.Abstractas;
2  using Entidades.Concretas;
3  using Nucleo.AccesoDatos.Concretos.Dapper;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace AccesoDatos.Concretas.Dapper
11 {
12     2 referencias
13     public class DapClienteDal : DapBaseRepositorioEntidad<Cliente, DapperContexto>, IClienteDal
14     {
15         0 referencias
16         public DapClienteDal() : base(
17             tableName: "Clientes",
18             columns: "Nombre,Apellidos,Dni,Telefono,Email,Imagen",
19             parameters: "@Nombre,@Apellidos,@Dni,@Telefono,@Email,@Imagen")
20         {
21         }
22     }
23 }
```

DapClienteDal.PNG

Desarrollo de Interfaces

```
1 using Entidades.Concretas;
2 using Nucleo.Negocios.Abstractas;
3 using System;
4 using System.Collections.Generic;
5 using System.Linq;
6 using System.Text;
7 using System.Threading.Tasks;
8
9 namespace Negocios.Abstractas
10 {
11     4 referencias
12     public interface IServicioCliente : IServicio<Cliente>
13     {
14         1 referencia
15         bool EliminarCliente(int Id);
16     }
17 }
```

IServicioCliente.PNG

```
10
11 namespace Negocios.Concretas
12 {
13     2 referencias
14     public class GestorCliente : BaseService<Cliente>, IServicioCliente
15     {
16         1 referencia
17         public IClienteDal ClienteDal { get; }
18         0 referencias
19         public GestorCliente(IClienteDal ClienteDal) : base(ClienteDal)
20         {
21             this.ClienteDal = ClienteDal;
22         }
23
24         1 referencia
25         public bool EliminarCliente(int Id)
26         {
27             bool dev = true;
28             Cliente ant = GetById(Id);
29             if (ant != null)
30                 Delete(ant);
31             else
32                 dev = false;
33             return dev;
34         }
35     }
36 }
```

No se encontraron problemas. Línea: 1 Carácter: 1

GestorCliente.PNG

```
3 referencias
public class ModuloNinject : NinjectModule
{
    0 referencias
    public override void Load()
    {
        Bind<IUsuarioDal>().To<DapUsuarioDal>().InSingletonScope();
        Bind<IServicioUsuario>().To<GestorUsuario>().InSingletonScope();

        Bind<IEmpleadoDal>().To<DapEmpleadoDal>().InSingletonScope();
        Bind<IServicioEmpleado>().To<GestorEmpleado>().InSingletonScope();

        Bind<IPedidoDal>().To<DapPedidoDal>().InSingletonScope();
        Bind<IServicioPedido>().To<GestorPedido>().InSingletonScope();

        Bind<IPedido_DetalleDal>().To<DapPedido_DetalleDal>().InSingletonScope();
        Bind<IServicioPedido_Detalle>().To<GestorPedido_Detalle>().InSingletonScope();

        Bind<IClienteDal>().To<DapClienteDal>().InSingletonScope();
        Bind<IServicioCliente>().To<GestorCliente>().InSingletonScope();

        Bind<IProductoDal>().To<DapProductoDal>().InSingletonScope();
        Bind<IServicioProducto>().To<GestorProducto>().InSingletonScope();

        Bind<IProveedorDal>().To<DapProveedorDal>().InSingletonScope();
        Bind<IServicioProveedor>().To<GestorProveedor>().InSingletonScope();
    }
}
```

Por ultimo se tiene que declararlo en el Ninje.

Ninje_cliente.PNG

Desarrollo de Interfaces

6.Funciones + QUERYS SQL

Aunque se nos proporciono funciones para poder trabajar con la base de datos se han tenido que modificar, esta clase se encuentra en el apartado del Nucleo de la solución .AccesoDatos>Concretos>Dapper.

```
where TContexto : class, IContexto, new()
{
    11 referencias
    private string TableName { get; set; }
    5 referencias
    private string Columns { get; set; }
    2 referencias
    private string Params { get; set; }

    8 referencias
    public DapBaseRepositorioEntidad(string tableName, string columns, string parameters)
    {
        TableName = tableName;
        Columns = columns;
        Params = parameters;
    }

    private const string INSERT_QUERY = "INSERT INTO {0} ({1}) VALUES ({2}) SELECT * FROM {0} WHERE [{0}].Id = @@IDENTITY";
    private const string UPDATE_QUERY = "UPDATE {0} SET {1} WHERE [{0}].Id = @Id";
    private const string DELETE_QUERY = "DELETE FROM {0} WHERE [{0}].Id = @Id";
    private const string SELECT_ALL_QUERY = "SELECT * FROM {0}";
    private const string SELECT_FIRST_QUERY = "SELECT * FROM {0} WHERE [{0}].Id=@Id";
    private const string SELECT_CUSTOM_WHERE_QUERY = "SELECT * FROM {0} WHERE [{0}].{1}";
    private const string COUNT_QUERY = "SELECT COUNT(Id) FROM {0}";

    protected SqlConnection connection = new SqlConnection(new TContexto().CadenaConexion);

    3 referencias
    public TEntidad Add(TEntidad entity)
    {
        if (entity is BaseEntidad)
        {
            (entity as BaseEntidad).Creado = DateTime.Now;
            (entity as BaseEntidad).Estado = Entidades.Enumerados.EstadoDeFila.Nuevo;
        }
        TEntidad addedEntity = connection.QueryFirstOrDefault<TEntidad>(string.Format(INSERT_QUERY, TableName, Columns, Params), entity);
        return addedEntity;
    }

    3 referencias
    public int Delete(TEntidad entity)
    {
        var affectedRow = connection.Execute(string.Format(DELETE_QUERY, TableName), entity);
        return affectedRow;
    }

    3 referencias
    public TEntidad Update(TEntidad entity)
    {
        if (entity is BaseEntidad)
        {
            (entity as BaseEntidad).Estado = Entidades.Enumerados.EstadoDeFila.Actualizado;
        }
        var updateCmd = string.Join(",", Columns.Split(','), 'Select(x => string.Format("[{0}].[{1}] = @{1}", TableName, x));
        var entityList = connection.Query<TEntidad>(string.Format(UPDATE_QUERY, TableName, updateCmd), entity);
        return GetById(entity.Id);
    }

    3 referencias
    public List<TEntidad> GetAll(Expression filter = null)
    {
        List<TEntidad> entityList = connection.Query<TEntidad>(string.Format(SELECT_ALL_QUERY, TableName), filter).ToList();
        return entityList;
    }

    4 referencias
    public TEntidad GetById(int entityId)
    {
        var entity = connection.QueryFirstOrDefault<TEntidad>(string.Format(SELECT_FIRST_QUERY, TableName), new { Id = entityId });
        return entity;
    }

    3 referencias
    public TEntidad Get(string filter)
    {
        var entity = connection.QueryFirstOrDefault<TEntidad>(string.Format(SELECT_CUSTOM_WHERE_QUERY, TableName, filter));
        return entity;
    }

    3 referencias
    public TEntidad Get(Expression filter)
    {
        var entity = connection.QueryFirstOrDefault< TEntidad >(string.Format(SELECT_CUSTOM_WHERE_QUERY, TableName, filter));
        return entity;
    }

    3 referencias
    public List<TEntidad> GetAll(string filter)
    {
        var entity = connection.Query<TEntidad>(string.Format(SELECT_CUSTOM_WHERE_QUERY, TableName, filter)).ToList();
        return entity;
    }

    3 referencias
    public int Count()
    {
        return connection.ExecuteScalar<int>(string.Format(COUNT_QUERY, TableName));
    }

    3 referencias
    public void SaveData()
    {
        throw new NotImplementedException();
    }
}
```

DapBaseRepositorioEntidad.PNG

DapBaseRepositorioEntidad_2.PNG

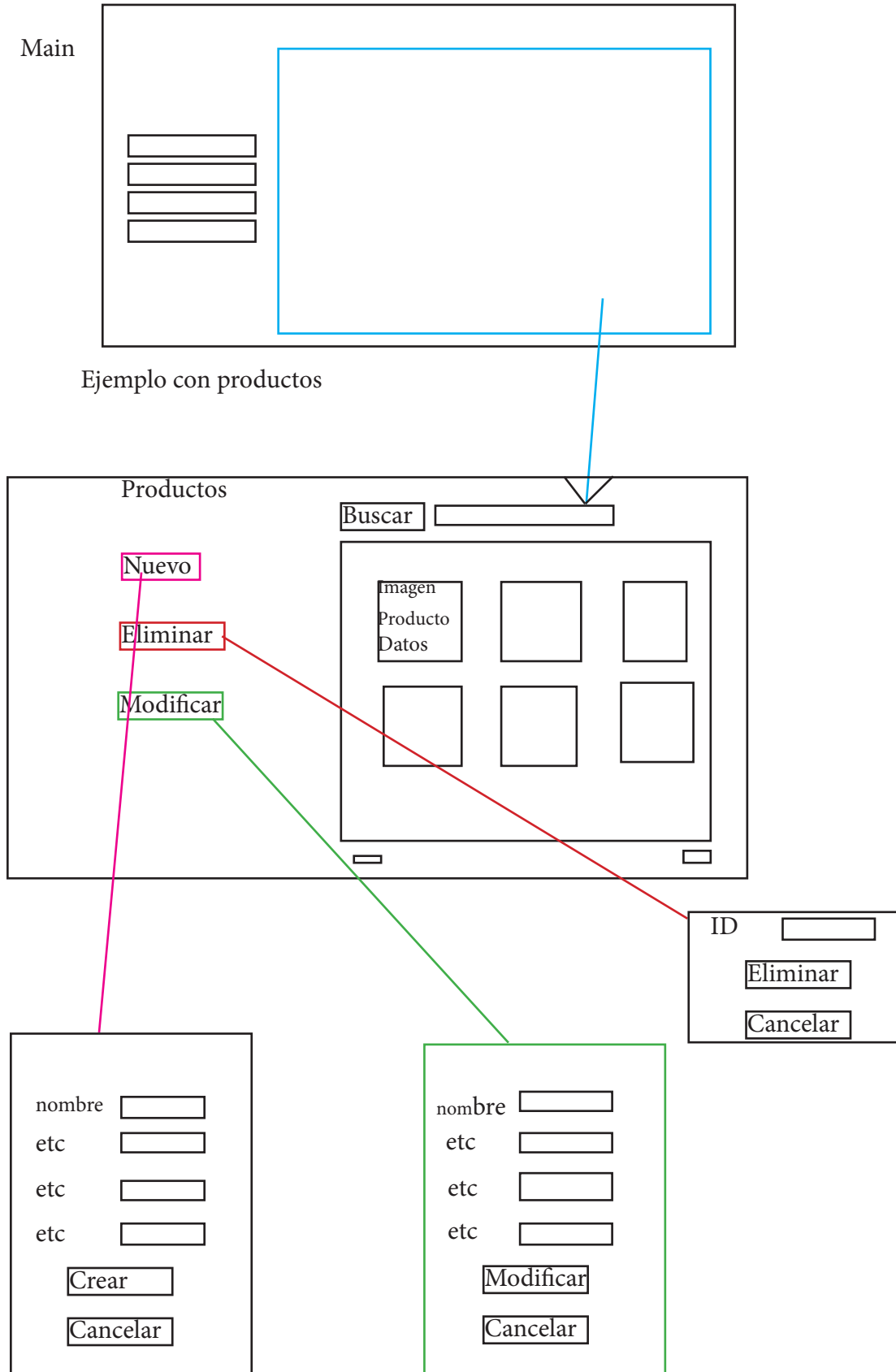
DapBaseRepositorioEntidad_3.PNG

DapBaseRepositorioEntidad_4.PNG

Desarrollo de Interfaces

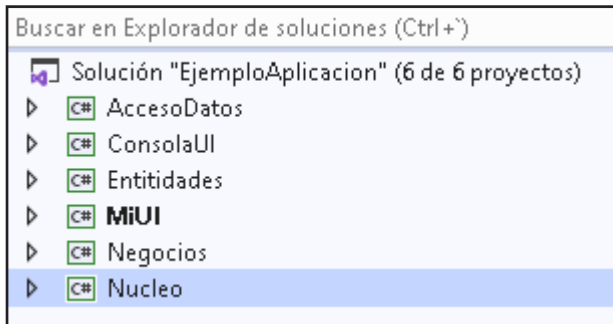
7.Desarrollo De la Interfaz

Para el diseño de la interfaz se ha realizado un pequeño boceto que me ha servido a la hora de distribuir los elementos en los diferentes paneles.



Desarrollo de Interfaces

En la solución el desarrollo de los Form se desarrolla en MIUI, un proyecto en donde además de capa visual se encuentran clases que servirán para filtrar la información recibida.

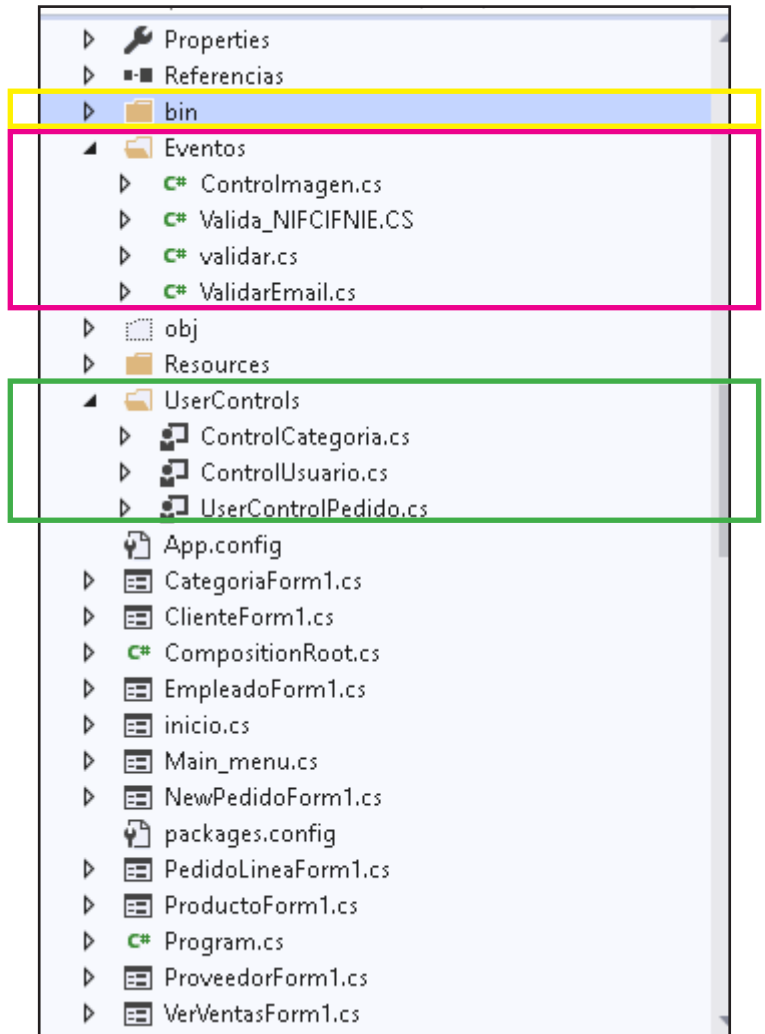


Solucion.PNG

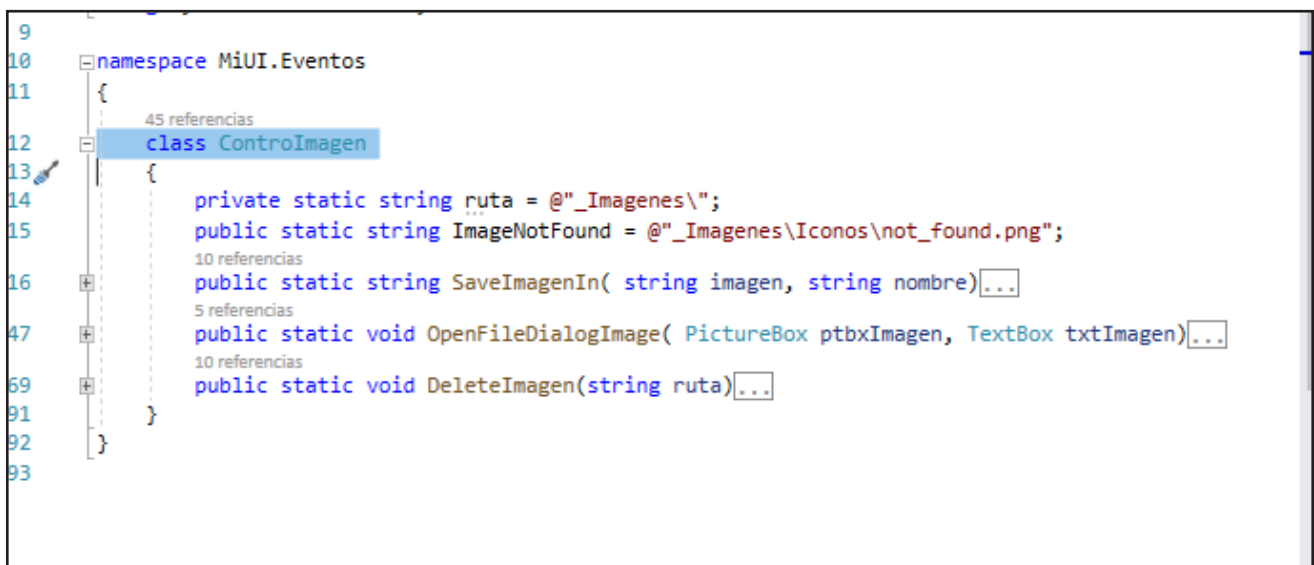
En el directorio bin del proyecto MUI se encuentran las imágenes que se utilizar para el proyecto.

Esto se realiza con ayuda de una clase que se encuentra en una carpeta llamada Eventos, la clase se llama Controlmagen en ella se encuentran métodos que nos ayudaren a guardar las imágenes seleccionadas.

Además en MUI se encuentran en el directorio UserControl, diferentes controladores, Items que nos ayudaran a mostrar la información guardada en la base de datos.



MIUI.PNG

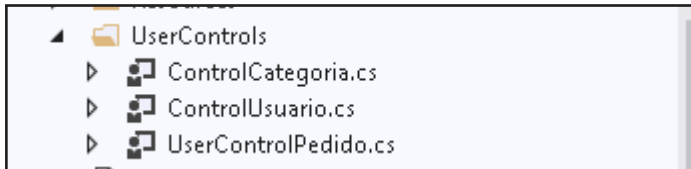


Metodos_imagenes.PNG

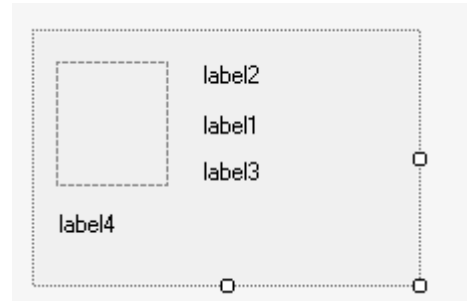
Desarrollo de Interfaces

UserControl Items

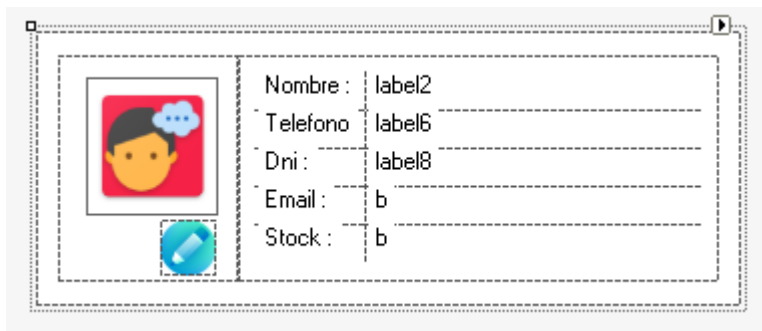
Para que el FlowLayoutPanel no solo muestra una tabla simple, se a añadido este tipo de elementos que nos ayudan a visualizar la información de los registros de cada una de las tablas de una forma mas atractiva.



MIU.PNG



Control_Item.PNG



Control.PNG



Item_categoria.PNG

8.Pantalla Principal

Como se puede ver en el boceto la pantalla principal es donde podremos visualizar los diferentes Form creados para nuestras tablas, a primera vista parece muy simple, pero que conlleva un cierto trabajo de diseño y atributos extras como la hora o la fecha en la parte inferior izquierda.



Inicio.PNG

Desarrollo de Interfaces

9.Froms Produstos

Con ayuda de los diferentes botos que se encuentran en la parte izquierda del menu de inicio podremos acceder si dificultad a solos diferentes forms creados en MUI.

Los Form los servirán para introducir y validar datos que en la base de datos creada en el punto numero uno según la cronología del proyecto, los diferentes elementos de inserción son los mismos que se utilizaron el la practica anterior así como los métodos de validación.

Los métodos de validacion se los a separado del los form ya que en la practica pasado se encontraban dentro del propio form que los utilizaba.

En el siguiente apartado se va a ver el Form llamado ProductoForm1, para ir a ella tenemos que pulsar el botón de artículos.

PRODUCTOS

Nombre:

Precio:

Stock:

Categoría:

Proveedores:

Imagen

Añadir

Clear

Buscar por Nombre

	Nombre: 565	Precio: 56,0000	Id: 3	Id: 2	Stock: 676817
	Nombre: ghg	Precio: 56,6700	Id: 3	Id: 2	Stock: 7005
	Nombre: lgdl	Precio: 56,8700	Id: 3	Id: 2	Stock: 85
	Nombre: mochalka	Precio: 257,0000	Id: 3	Id: 0	Stock: 66

Producto_form.

Con el podremos añadir un nuevo registro en la tabla productos.

Para poder realizar un nuevo registro tenemos que rellenar los campos del form y pulsar añadir este nos creara un nuevo objeto producto y con ayuda del metodo add() de nuestra interfaz IServicioProducto se insertara el nuevo registro.

Antes de que la inserción sea completada con éxito se debe comprobar que el producto existe o no, ademas si la información dada es valida y se introduce los diferentes proveedores y categorías en su correspondiente combobox, esto se realiza capturando el un List el resultado de ejecutar el método getAll() de su correspondiente servicio_X

Desarrollo de Interfaces

```
1 referencia
private void LoadProovedores()
{
    List<Proveedor> proveedores = _servicioProveedor.GetAll();
    cbProveedor.DataSource = proveedores;
}

1 referencia
private void LoadCategorias()
{
    List<Categoria> proveedores = _servicioCategoria.GetAll();
    cbCategoria.DataSource = proveedores;
}
```

combobox.

```
1 referencia
private bool ProductoExiste(string nombre)
{
    Producto e = _servicioProducto.Get("Nombre='" + nombre + "'");
    if (e != null)
    {
        MessageBox.Show("Ya existe un Producto con este nombre");
        return true;
    }
    return false;
}
```

Metodo_existe_producto.

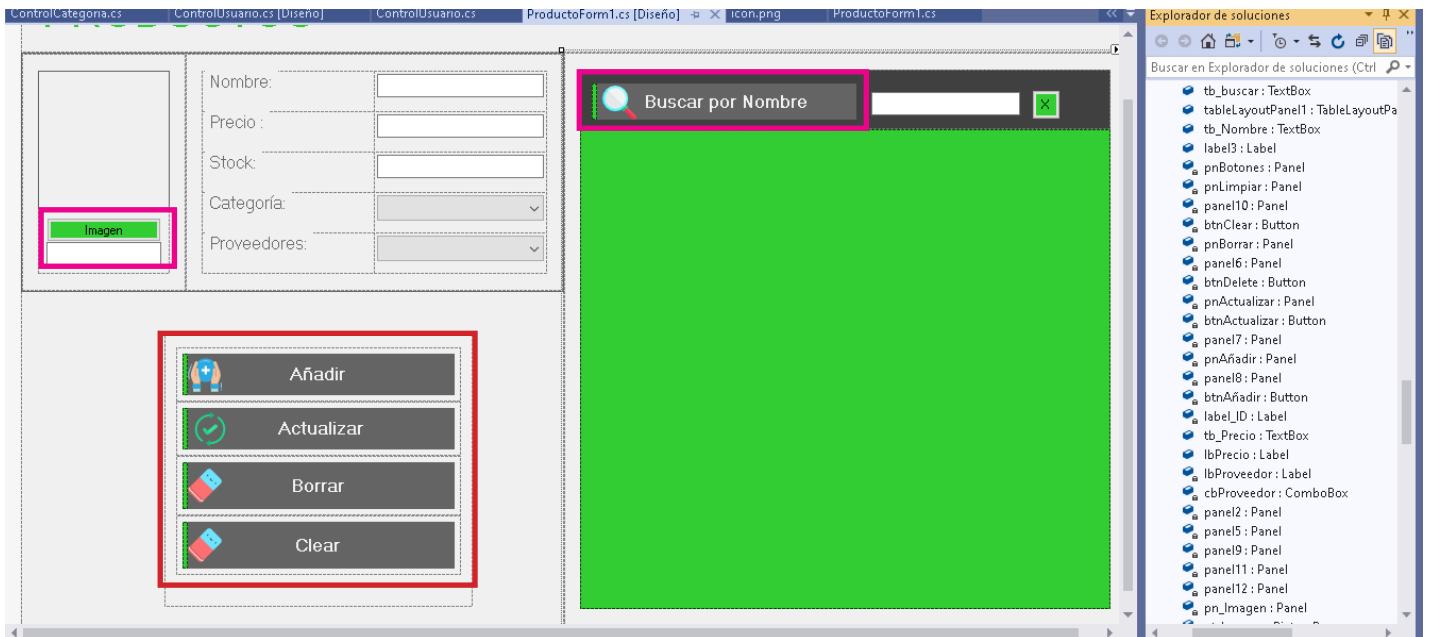
Estos dos métodos nos ayudaran a visualizar en contenido de nuestras tablas.

```
5 referencias
private void LoadUserControls()
{
    pnProductos.Controls.Clear();
    List<Producto> list = _servicioProducto.GetAll();
    foreach (Producto e in list)
    {
        ControlUsuario control = new ControlUsuario(e);
        pnProductos.Controls.Add(control);
        control._MostrarProducto += Control_MostrarProducto;
    }
}

2 referencias
private void Control_MostrarProducto(Producto e)
{
    seleccionado = e;
    tb_Nombre.Text = e.Nombre;
    tb_stock.Text = e.Stock.ToString();
    tb_Precio.Text = e.Precio.ToString();
    e.Categoria = _servicioCategoria.GetById(e.Id_Categoria);
    cbCategoria.SelectedItem = e.Categoria;
    e.Proveedor = _servicioProveedor.GetById(e.Id_Proveedor);
    cbProveedor.SelectedItem = e.Proveedor;
    pnAñadir.Visible = false;
    pnActualizar.Visible = true;
    pnBorrar.Visible = true;
    try
    {
        pt_Imagen.Image = Image.FromFile(e.Imagen);
        ImagenInicial = e.Imagen;
    }
    catch (FileNotFoundException)
    {
        pt_Imagen.Image = Image.FromFile(ControlImagen.ImageNotFound);
        ImagenInicial = ControlImagen.ImageNotFound;
    }
}
```

Metodo_producto_Control.

Desarrollo de Interfaces



Diseño_producto.

Botón Añadir

```
1 referencia
private void btnAñadir_Click(object sender, EventArgs e)
{
    if (!ComprobarCampoVacio() && !ProductoExiste(tb_Nombre.Text))
    {
        string pre = tb_Precio.Text.Replace(".", ",");
        Producto nuevo = new Producto()
        {
            Nombre = tb_Nombre.Text,
            Precio = Convert.ToDecimal(pre),
            Stock = Convert.ToInt32(tb_stock.Text),
            Proveedor = cbProveedor.SelectedItem as Proveedor,
            Categoria = cbCategoria.SelectedItem as Categoria,
            Imagen = ControImagen.SaveImagenIn(tb_Ruta_Imagen.Text, tb_Nombre.Text)
        };
        _servicioProducto.Add(nuevo);
        MessageBox.Show("Se ha Añadido una categoria. " + tb_Nombre.Text);
        LoadUserControls();
    }
}
```

Añadir.

Botón Buscar

```
1 referencia
private void btn_CancelarBusqueda_Click(object sender, EventArgs e)
{
    LoadUserControls();
    tb_buscar.Clear();
}

1 referencia
private void btn_buscar_Click(object sender, EventArgs e)
{
    Producto buscado = _servicioProducto.Get("Nombre='" + tb_buscar.Text + "'");
    if (buscado == null)
    {
        MessageBox.Show("No se ha encontrado ningun proveedor con ese nombre");
    }
    else
    {
        pnProductos.Controls.Clear();
        ControlUsuario control = new ControlUsuario(buscado);
        pnProductos.Controls.Add(control);
        control._MostrarProducto += Control_MostrarProducto;
    }
}
```

Buscar.

Desarrollo de Interfaces

Botón Limpiar

```
1 referencia
private void btnClear_Click(object sender, EventArgs e)
{
    tb_buscar.Clear();

    tb_Nombre.Clear();
    tb_Precio.Clear();
    tb_stock.Clear();
    pt_Imagen.Image.Dispose();

    pnActualizar.Visible = false;
    pnBorrar.Visible = false;
    pnAñadir.Visible = true;
}
```

Limpiar.

Botón Borrar

```
1 referencia
private void btnDelete_Click(object sender, EventArgs e)
{
    try
    {
        ControImagen.DeleteImagen(seleccionado.Imagen);
        _servicioProducto.Delete(seleccionado);
        LoadUserControls();
    }
    catch (Exception )
    {
        MessageBox.Show("El producto esta pedido ");
    }
}
```

Borrar.

Botón Actualizar

```
1 referencia
private void btnActualizar_Click(object sender, EventArgs e)
{
    if (!ComprobarCampoVacio())
    {
        seleccionado.Precio = Convert.ToDecimal(tb_Precio.Text.Replace(".", ","));
        seleccionado.Stock = Convert.ToInt32(tb_stock.Text);
        seleccionado.Proveedor = cbProveedor.SelectedItem as Proveedor;
        seleccionado.Categoria = cbCategoria.SelectedItem as Categoria;
        if (!ImagenInicial.Equals(tb_Ruta_Imagen))
        {
            ControImagen.DeleteImagen(ImagenInicial);
        }
        seleccionado.Imagen = ControImagen.SaveImagenIn(tb_Ruta_Imagen.Text, seleccionado.Nombre);
        _servicioProducto.Update(seleccionado);
        MessageBox.Show("Se ha actualizado un registro ");
        LoadUserControls();
    }
}
```

Actualizar.

Botón Actualizar

```
1 referencia
private void btnOpen_Click_2(object sender, EventArgs e)
{
    ControImagen.OpenFileDialogImage(pt_Imagen, tb_Ruta_Imagen);
}
```

Abri_imagen.

Desarrollo de Interfaces

Al insertar el un nuevo registro se actualizara la vista de los artículos se podrá borrar, con ayuda de un boton el en Control.



Nombre : botasblancas
Precio : 10,0000
Id : 3
Id : 5
Stock : 10

Imagen

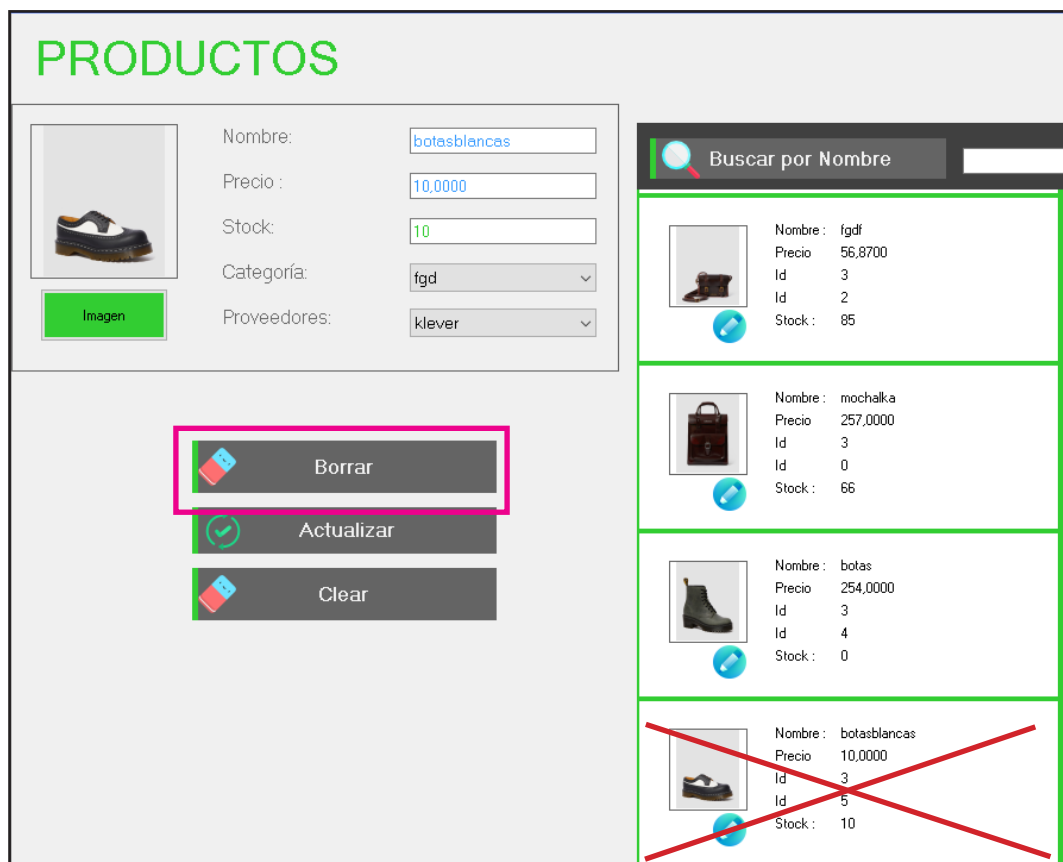
Botón de borrar (icono de basura) resaltado con un recuadro rojo.

```
1 referencia
private void Btn_MostrarProducto(object sender, EventArgs e)
{
    _MostrarProducto?.Invoke(_Producto);
}
```

Boton_en_Control_Iten.

Añadir_2.

Esto nos dará la opción de borrar o actualizar nuestro producto.
Si el producto no tiene ninguna dependencia se podrá borrar el registro.



PRODUCTOS

Nombre: botasblancas
Precio : 10,0000
Stock: 10
Categoría: fgd
Proveedores: klever

Imagen

Borrar
Actualizar
Clear

Buscar por Nombre

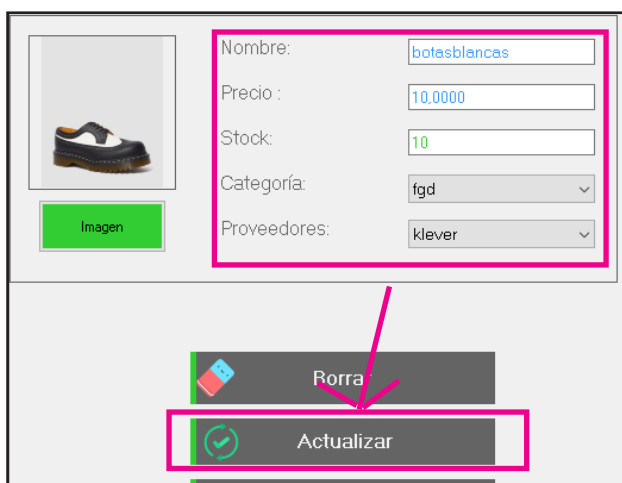
Nombre : fgdf
Precio : 56,8700
Id : 3
Id : 2
Stock : 85

Nombre : mochalka
Precio : 257,0000
Id : 3
Id : 0
Stock : 66

Nombre : botas
Precio : 254,0000
Id : 3
Id : 4
Stock : 0

Nombre : botasblancas
Precio : 10,0000
Id : 3
Id : 5
Stock : 10

Añadir_2.PNG



Nombre: botasblancas
Precio : 10,0000
Stock: 10
Categoría: fgd
Proveedores: klever

Imagen

Borra
Actualizar

Ala hora de actualizar el proceso es el mismo, se pulsa el botón para seleccionar el producto y después modificamos los datos que nos interese, y finalizamos pulsando el botón de actualizar.

Añadir_2.PNG

Desarrollo de Interfaces

Sobre las demás ventanas tienen la misma estructura a nivel interno, en diseño de ellas varia en cuanto a su estética nada mas.

Ventanas:

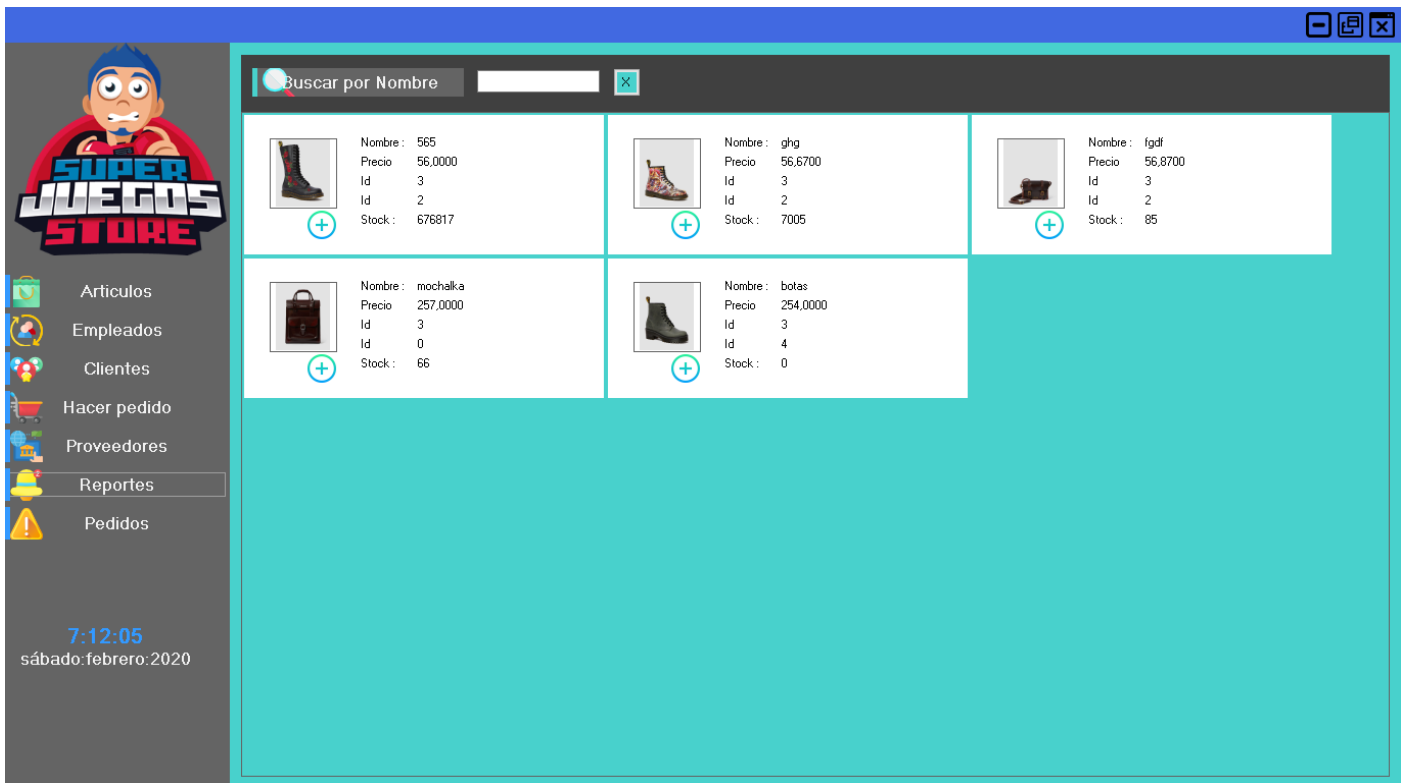
The screenshot shows the 'EMPLEADOS' (Employees) window. On the left is a sidebar with the 'SUPER JUEGOS STORE' logo and a menu: Articulos, Empleados, Clientes, Hacer pedido, Proveedores, Reportes, and Pedidos. Below the menu is a timestamp: 7:11:00 sábado: febrero: 2020. The main area has a title 'EMPLEADOS' in blue. It contains a form with fields for Apellido, Nombre, DNI-NIF-CIF, and Numero telefono, each with a corresponding input box. There is an 'Imagen' button below the first field. To the right of the form are two buttons: 'Añadir' (Add) and 'Clear'. On the far right is a search bar with the text 'Buscar por Nombre' and a search icon, followed by a large blue rectangular area representing the search results.

Ventana_empleados.

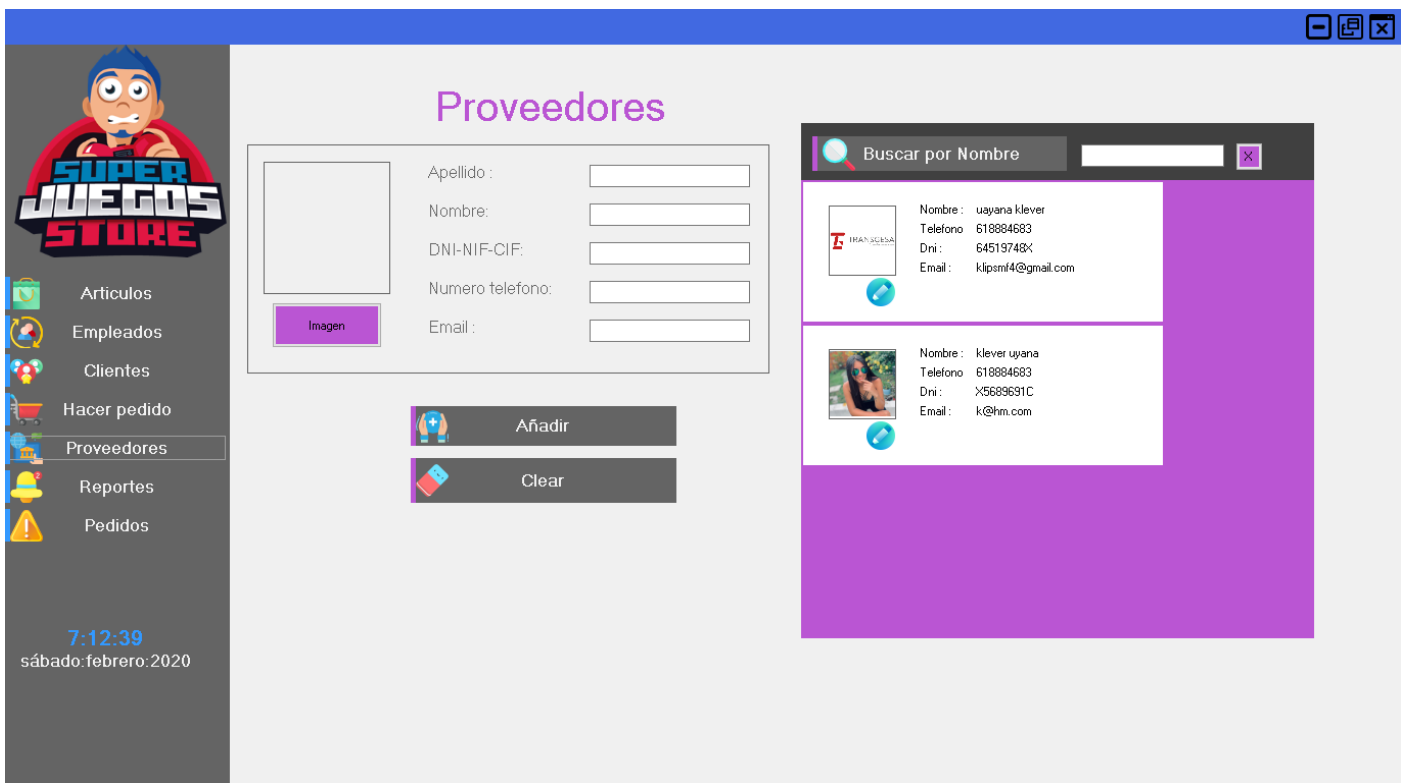
The screenshot shows the 'CLIENTES' (Clients) window. It has the same sidebar as the previous window. The main area has a title 'CLIENTES' in red. It contains a form with fields for Apellido, Nombre, DNI-NIF-CIF, Numero telefono, and Email, each with a corresponding input box. There is an 'Imagen' button below the first field. To the right of the form are two buttons: 'Añadir' (Add) and 'Clear'. On the far right is a search bar with the text 'Buscar por Nombre' and a search icon, followed by a large red rectangular area representing the search results.

Ventana_clientes.

Desarrollo de Interfaces

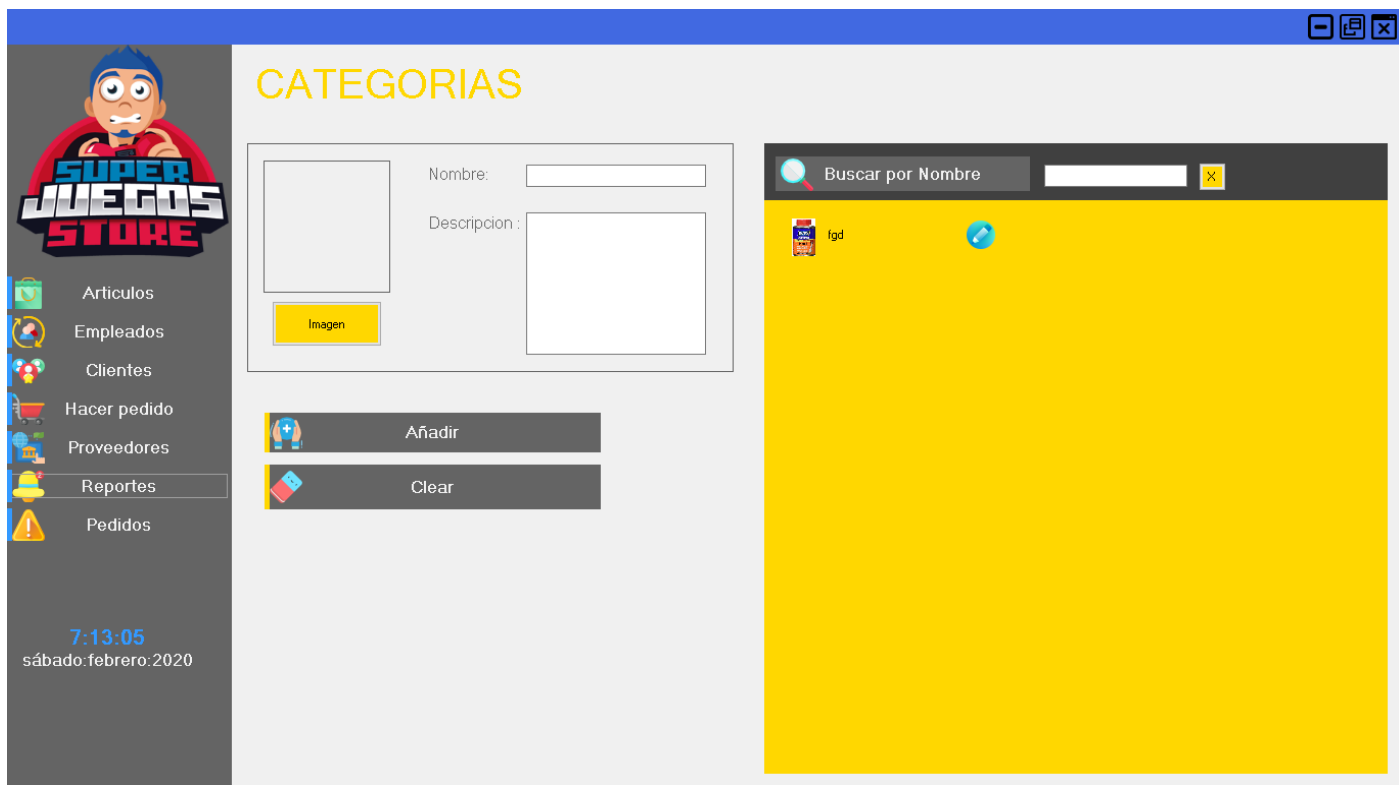


Ventana_Hacer_pedido.

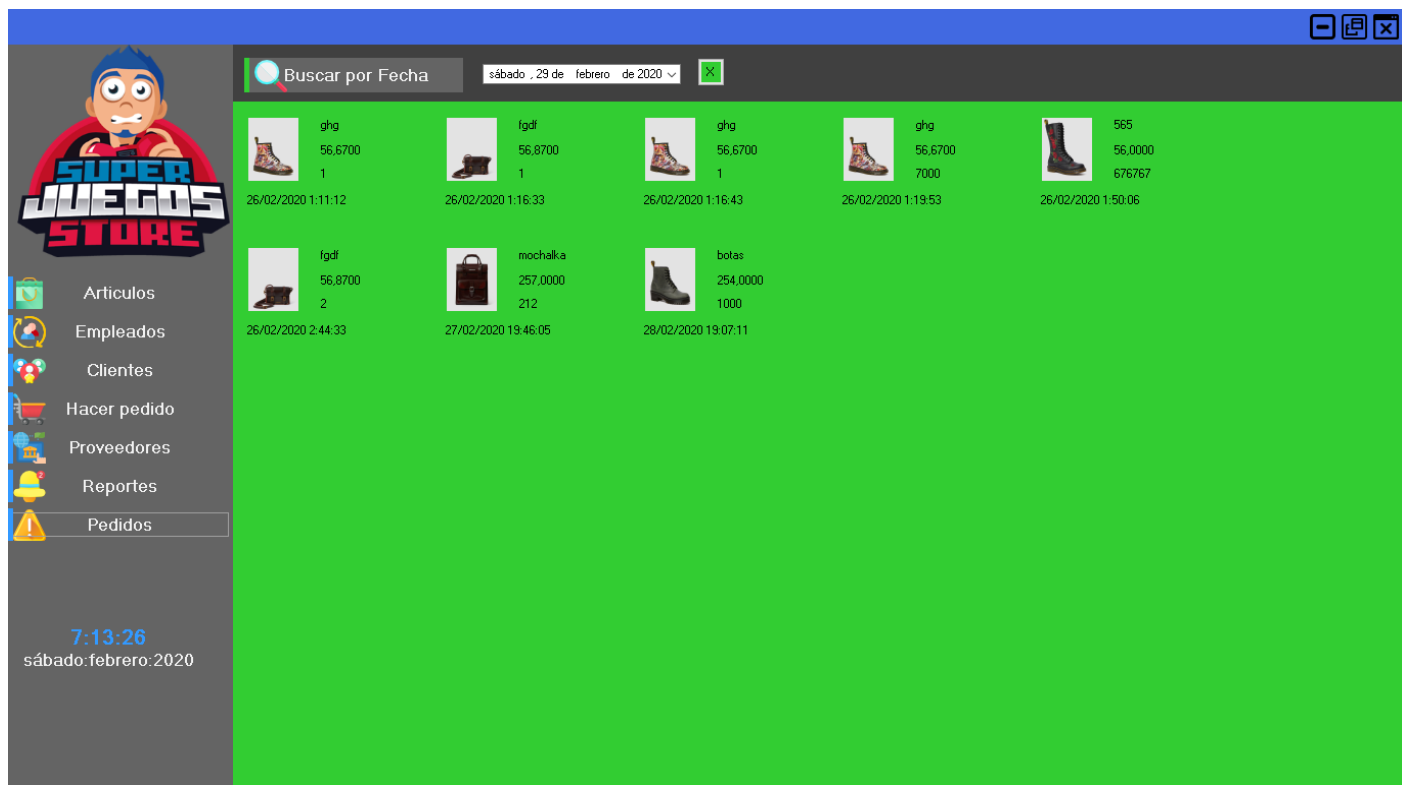


Ventana_proveedores.

Desarrollo de Interfaces



Ventana_categorias.



Ventana_pedidos.

Desarrollo de Interfaces

10.Conclusiones

Aunque a primera vista la aplicación pueda parecer fácil de desarrollar , a resultado todo lo contrario , ya que sobre todo al principio a la hora de instalar el Dapper a generado un numero de errores en las que no podía solucionar, por ello decidí realizar el proyecto desde 0, teniendo como base el proyecto base que se nos proporciona en clase.

En ocasiones el trabajo a sido frustrarte , ya que la idea principal del inicio no se puedo llevar a cabo de la mejor forma.

No optaste se he aprendido mucho sobre como llevar un proyecto de este calibre, he aprendido que los proyectos de este nivel hay que llevarlos de forma ordenado y marcándote unos tiempos.

Sin duda he disfrutado realizando este proyecto, pero me da pena no finalizar el proyecto con una estética y funcionalidad mas pulida.