

UNIVERSIDAD PRIVADA DE TACNA
FACULTAD DE INGENIERÍA DE SISTEMAS
ESCUELA PROFESIONAL DE INGENIERÍA SISTEMAS



PROYECTO TRABAJO FINAL
UNIDAD I

CURSO

Base de Datos II

DOCENTE

Mag. Patrick Cuadros Quiroga

INTEGRANTES

Tarqui Montalico, Risther Jaime - 2017057857

X Limache Victorio, Víctor Piero - 2017057857

Callata Flores, Martín Rafael - 2015053847

Sánchez Rodríguez, Bayron - 2017057859

Liendo Velasquez, Joaquín - 2016054463

TACNA - PERÚ

2020

Resumen

Un sistema de software de Escritorio basado en gestionar datos importantes de la empresa en las cuales podrá registrar el cliente y derivadamente a su mascota con la aplicación por parte del cliente, y por medio de la empresa podrá ver sus clientes, el registro de empleados, las ventas de medicamentos en las cuales se realizará de una forma concisa y clara a las necesidades de la empresa, y así la veterinaria podrá obtener un espacio de registro más sencillo y eficaz.

El software está basado en escritorio como antes mencionado, en un lenguaje de aplicación llamada “(C Sharp)” con la finalidad de adaptarse al sistema operativo Windows desde W7 hacia adelante, la aplicación está plenamente desarrollado para el uso dentro de la Empresa, mas no de los clientes. A su vez al desarrollar este programa puede ser que haya determinadas fallas del software en las cuales utilizaremos la aplicación SonarQube para el mejoramiento de dicha aplicación.

Abstract

A Desktop software system based on managing data important of the company in which you can register the client and, consequently, your pet with the application by the client, and through the company you can see its clients, the registration of employees, the sales of medicines in which the a concise and clear way to the needs of the company, and thus the veterinarian can obtain a simpler and more efficient registration space.

The software is based on the desktop as mentioned before, in an application language called “(C Sharp)” in order to adapt to the Windows operating system from W7 onwards, the application is fully developed for use within the Company, more not from customers. In turn, when developing this program, it may be that there are certain software flaws in which we will use the SonarQube application to improve said application.

PROYECTO TRABAJO FINAL

UNIDAD I

1. Introducción.

Una gran cantidad de empresas de nuestra región interactúan de forma directa con los clientes, no cuentan con un registro organizado ni automatizado, lo cual representa lentitud al momento de registrar algún pago o desbalance al momento de sacar las cuentas del día.

La investigación planteada en este trabajo está relacionada con el proceso de veterinaria Pet's Planet ubicada Gral. Varela 233, Tacna, para el desarrollo del sistema de veterinaria se tuvo que usar las herramientas y los instrumentos necesario para la obtención de información tales como la encuesta realizada al dueño de la empresa para así poder identificar los procesos de funcionamiento de la veterinaria Pet's Planet.

2. Título.

Sistema de Veterinaria "PET'S PLANET".

3. Autores.

- Limache Victorio, Víctor Piero
- Tarqui Montalico, Risther Jaime
- Sánchez Rodríguez, Bayron
- Liendo Velasquez, Joaquín
- Callata Flores, Rafael

4. Planteamiento del problema.

a) Problema.

El problema de todo estudiante del cual empieza con el desarrollo de software de una manera principiante, es el no uso de métricas que pueden ayudar a la calidad del código al cual nosotros queremos implementar, es por eso que el programa actual del cual ya está desarrollado no cuenta con un uso correcto de métricas, por lo tanto, este código siempre puede optar por mejorar.

La actualización de este código no debe afectar al funcionamiento del programa, pero sí a su rendimiento. Dicho esto, el código fuente actual puede ser mejorable.

b) Justificación.

Con el fin de mejorar la calidad del software" Veterinaria PET'S PLANET", estos serían nuestros planteamientos que nos haremos:

1) **¿Qué es lo que vamos de hacer?**

Una mejora al código fuente del sistema Veterinaria PET'S PLANET con la herramienta SonarQube.

2) **¿Por qué se va hacer?**

Para lograr la mejora del código actual, sin modificar el proyecto.

3) **¿Para qué se va hacer?**

Para tener un mejor rendimiento del programa, y utilizar la herramienta que aprendimos a lo largo de la unidad: SonarQube.

4) **¿Cómo se va hacer?**

Utilizando la herramienta SonarQube, y en la cual utilizaremos nuestros conocimientos aprendidos en el transcurso del curso para la mejora del software.

5) **¿Por qué es importante este trabajo de unidad?**

Porque mejora el actual código fuente y como consecuencia aumenta el rendimiento del sistema.

c) **Alcance.**

- El Sistema contará con una base de datos ,en la nube.
- El Sistema contará de un control de Registros.
- Contará con reportes de Servicios.

5. **Referentes teóricos.**

6. **Objetivos.**

a) **General.**

Diseñar e implementar un software para mejorar de interacción entre los clientes y la clínica veterinaria.

b) **Específicos.**

- 1) Generar una base de datos que maneje información de clientes, mascotas, productos y servicios para generar un análisis histórico.
- 2) Proveer una interfaz práctica y amigable para facilitar el uso de la plataforma a usuarios y médicos veterinarios.
- 3) Mostrar productos y servicios que ofrece la clínica.
- 4) Generar reportes de atenciones de cada médico veterinario, así como de los clientes y mascotas.

7. **Referentes teóricos.**

- a) Un componente de software debe diseñarse e implementarse de modo que pueda volverse a usar en muchos programas diferentes. Los modernos componentes reutilizables incorporan tanto los datos como el procesamiento que se les aplica, lo que permite que el ingeniero de software cree nuevas aplicaciones a partir de partes susceptibles de volverse a usar [1].
- b) El mejoramiento del proceso de Software abarca un conjunto de actividades que conducirán a un mejor proceso de software y, en consecuencia, a software de mayor calidad y a su entrega en forma más oportuna [2].

8. Desarrollo de la propuesta.

a) Tecnología de información

- **MySQL:** Es un sistema de gestión de base de datos relacional, desarrollado por la empresa Microsystems Oracle Corporation. El lenguaje de desarrollo utilizado es Transact-SQL, una implementación del estándar ANSI del lenguaje SQL, utilizado para manipular y recuperar datos, crear tablas y definir relaciones entre ellas.
- **C #:** lenguaje de programación multiparadigma desarrollado y estandarizado por Microsoft. Es un lenguaje de programación creado para diseñar aplicaciones en la plataforma.NET.
- **Visual Studio:** es un entorno de desarrollo en diferentes sistemas operativos y compatibles con múltiples lenguajes de programación al igual que entornos de desarrollo web.
- **SonarQube:** es una plataforma para evaluar código fuente. Es software libre y usa diversas herramientas de análisis estático de código fuente como Checkstyle, PMD o FindBugs para obtener métricas que pueden ayudar a mejorar la calidad del código de un programa.

b) Metodología, técnicas usadas

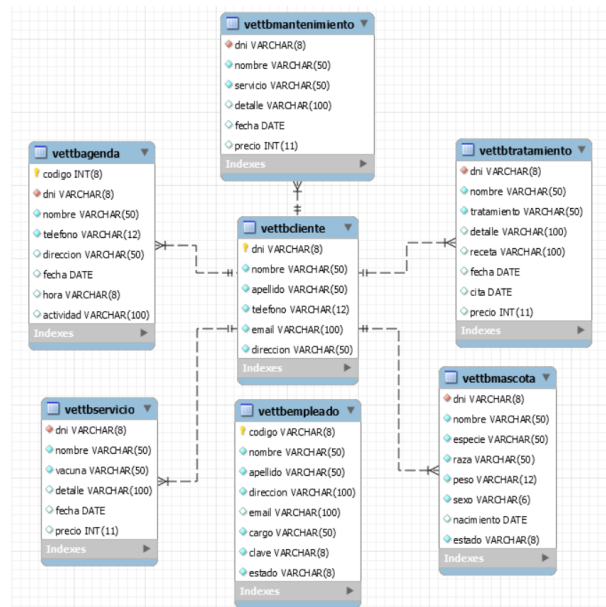
UML es un lenguaje para hacer modelos y es independiente de los métodos de análisis y diseño. Existen diferencias importantes entre un método y un lenguaje de modelado. Un método es una manera explícita de estructurar el pensamiento y las acciones de cada individuo.

Además, el método le dice al usuario qué hacer, cómo hacerlo, cuándo hacerlo y por qué hacerlo; mientras que el lenguaje de modelado carece de estas instrucciones.

Los métodos contienen modelos y esos modelos son utilizados para describir algo y comunicar los resultados del uso del método.

c) Aplicación

1) OR/M.



```

ClsEAgenda.cs  ClsNConexion.cs
D:\0000_017ciclo\UPT_2020_II\DB2\UNIDADII\SistemaVeterinaria\SistemaVeterinaria\Entit
1  using System.ComponentModel.DataAnnotations;
2  using System.ComponentModel.DataAnnotations.Schema;
3  namespace SistemaVeterinaria.Entidades
4  {
5      [Table("Agenda")]
6      12 referencias | 18/18/2020, Hace 23 días | 3 autores, 6 cambios
7      public partial class ClsEAgenda
8      {
9          [Column("IdAgenda")]
10         [Key]
11         [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
12         [Required]
13         public int IdAgenda { get; set; }
14         [Column("Codigo")]
15         [Required]
16         [StringLength(8)]
17         public string Codigo { get; set; }
18         [Column("Dni")]
19         [Required]
20         [StringLength(11)]
21         public string Dni { get; set; }
22         [Column("Nombres")]
23         [Required]
24         [StringLength(50)]
25         public string Nombres { get; set; }
26         [Column("Direccion")]
27         [Required]
28         [StringLength(100)]
29         public string Direccion { get; set; }
30         [Column("Telefono")]
31         [Required]
32         [StringLength(15)]
33         public string Telefono { get; set; }
34         [Column("Fecha")]
35         [Required]
36         [DataType(DataType.Date)]
37         [StringLength(10)]
38         public DateTime Fecha { get; set; }
39     }

```

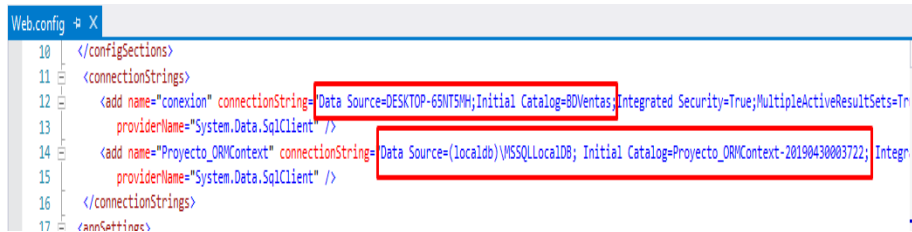
```

1 using System.ComponentModel.DataAnnotations;
2 using System.ComponentModel.DataAnnotations.Schema;
3 namespace SistemaVeterinaria.Entidades
4 {
5     [Table("Empleados")]
6     public partial class ClsEEmpleados
7     {
8
9         [Key]
10        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
11        [Required]
12        public int IdEmpleado { get; set; }
13        public string Codigo { get; set; }
14        public string Nombre { get; set; }
15        public string Apellidos { get; set; }
16        public string Direccion { get; set; }
17        public string Email { get; set; }
18        public string Cargo { get; set; }
19        public string Clave { get; set; }
20        public string Estado { get; set; }
21
22    }
23
24    [Table("Clientes")]
25    public partial class ClsEClientes
26    {
27
28        [Key]
29        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
30        [Required]
31        public int IdCliente { get; set; }
32        public string Oni { get; set; }
33        public string Nombre { get; set; }
34        public string Apellido { get; set; }
35        public string Telefono { get; set; }
36        public string Email { get; set; }
37        public string Direccion { get; set; }
38
39        public Principal.FrmClientes FrmClientes
40        {
41            get => default(Principal.FrmClientes);
42            set
43            {
44            }
45        }
46
47        internal static internal Negocios.ClsNClientes ClsNClientes
48        {
49            get => default(Negocios.ClsNClientes);
50            set
51            {
52            }
53        }
54    }
55 }

```

2) Migraciones.

Nos vamos al Archivo Web.config del proyecto y agregamos lo siguiente.

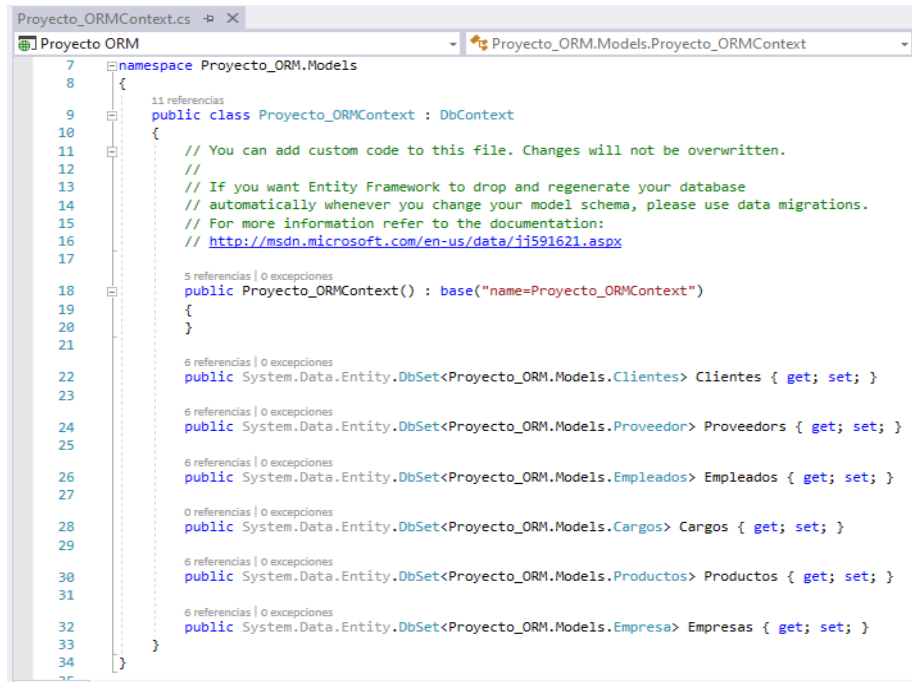


```

10 </configSections>
11 <connectionStrings>
12 <add name="conexion" connectionString="Data Source=DESKTOP-6SNTSMH;Initial Catalog=BDVentas;Integrated Security=True;MultipleActiveResultSets=True"
13     providerName="System.Data.SqlClient" />
14 <add name="Proyecto_ORMContext" connectionString="Data Source=(localdb)\MSSQLLocalDB; Initial Catalog=Proyecto_ORMContext-20190430003722; Integrated Security=True"
15     providerName="System.Data.SqlClient" />
16 </connectionStrings>
17 </annSettings>

```

La clase Proyecto-ORMContext-cs

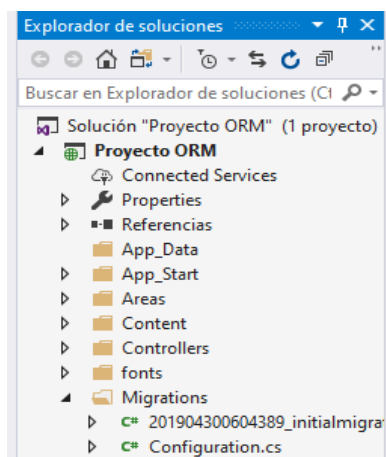


```

7 namespace Proyecto_ORM.Models
8 {
9     public class Proyecto_ORMContext : DbContext
10     {
11         // You can add custom code to this file. Changes will not be overwritten.
12         //
13         // If you want Entity Framework to drop and regenerate your database
14         // automatically whenever you change your model schema, please use data migrations.
15         // For more information refer to the documentation:
16         // http://msdn.microsoft.com/en-us/data/jj591621.aspx
17
18         public Proyecto_ORMContext() : base("name=Proyecto_ORMContext")
19         {
20         }
21
22         public System.Data.Entity.DbSet<Proyecto_ORM.Models.Clientes> Clientes { get; set; }
23
24         public System.Data.Entity.DbSet<Proyecto_ORM.Models.Proveedoror> Proveedores { get; set; }
25
26         public System.Data.Entity.DbSet<Proyecto_ORM.Models.Empleados> Empleados { get; set; }
27
28         public System.Data.Entity.DbSet<Proyecto_ORM.Models.Cargos> Cargos { get; set; }
29
30         public System.Data.Entity.DbSet<Proyecto_ORM.Models.Productos> Productos { get; set; }
31
32         public System.Data.Entity.DbSet<Proyecto_ORM.Models.Empresa> Empresas { get; set; }
33     }
34 }

```

La carpeta donde están las clases para la migración



La clase donde se inicia la migración 201904300604389-initialmigration-cs

```

201904300604389_initialmigration.cs
Proyecto ORM
namespace Proyecto ORM.Migrations
{
    using System;
    using System.Data.Entity.Migrations;

    2 referencias
    public partial class initialmigration : DbMigration
    {
        0 referencias | 0 excepciones
        public override void Up()
        {
            CreateTable(
                "dbo.Boletas",
                c => new
                {
                    Serie = c.String(nullable: false, maxLength: 128),
                    Numero = c.String(),
                    Fecha = c.String(),
                    Total = c.Double(nullable: false),
                    Estado = c.String(),
                    Ruc = c.String(maxLength: 128),
                    Dni = c.String(maxLength: 128),
                   Codigo = c.String(maxLength: 128),
                })
            .PrimaryKey(t => t.Serie)
            .ForeignKey("dbo.Clientes", t => t.Dni)
            .ForeignKey("dbo.Empleados", t => t.Codigo)
            .ForeignKey("dbo.Empleados", t => t.Ruc)
            .Index(t => t.Ruc)
            .Index(t => t.Dni)
            .Index(t => t.Codigo);
        }
    }
}

```

```

30
31
32 CreateTable(
33     "dbo.Clientes",
34     c => new
35     {
36         Dni = c.String(nullable: false, maxLength: 128),
37         Nombre = c.String(),
38         Apellidos = c.String(),
39         Telefono = c.String(),
40         Correo = c.String(),
41     })
42     .PrimaryKey(t => t.Dni);
43
44 CreateTable(
45     "dbo.Empleados",
46     c => new
47     {
48         Codigo = c.String(nullable: false, maxLength: 128),
49         Nombre = c.String(),
50         Apellido = c.String(),
51         Direccion = c.String(),
52         Telefono = c.String(),
53         Correo = c.String(),
54         Estado = c.String(),
55         Clave = c.String(),
56         Cargo = c.String(maxLength: 128),
57     })
58     .PrimaryKey(t => t.Codigo)
59     .ForeignKey("dbo.Cargos", t => t.Cargo)
60     .Index(t => t.Cargo);

```

```

61
62 CreateTable(
63     "dbo.Cargos",
64     c => new
65     {
66         Cargo = c.String(nullable: false, maxLength: 128),
67         Descripcion = c.String(),
68     })
69     .PrimaryKey(t => t.Cargo);
70
71 CreateTable(
72     "dbo.Empresas",
73     c => new
74     {
75         Ruc = c.String(nullable: false, maxLength: 128),
76         Nombre = c.String(),
77         Direccion = c.String(),
78         Telefono = c.String(),
79     })
80     .PrimaryKey(t => t.Ruc);
81
82 CreateTable(
83     "dbo.Detalles",
84     c => new
85     {
86         Serie = c.String(nullable: false, maxLength: 128),
87         Numero = c.String(),
88         Codigo = c.String(maxLength: 128),
89         Cantidad = c.Int(nullable: false),
90         Importe = c.Double(nullable: false),
91         Boletas_Serie = c.String(maxLength: 128),
92     })

```

```

92     .PrimaryKey(t => t.Serie)
93     .ForeignKey("dbo.Boletas", t => t.Boletas_Serie)
94     .ForeignKey("dbo.Productos", t => t.Codigo)
95     .Index(t => t.Codigo)
96     .Index(t => t.Boletas_Serie);
97
98     CreateTable(
99         "dbo.Productos",
100         c => new
101         {
102             Codigo = c.String(nullable: false, maxLength: 128),
103             Descripcion = c.String(),
104             Cantidad = c.Int(nullable: false),
105             Precio = c.Double(nullable: false),
106             Importe = c.Decimal(nullable: false, precision: 18, scale: 2),
107             Proveedor_Dni = c.String(maxLength: 128),
108         })
109     .PrimaryKey(t => t.Codigo)
110     .ForeignKey("dbo.Proveedores", t => t.Proveedor_Dni)
111     .Index(t => t.Proveedor_Dni);
112
113     CreateTable(
114         "dbo.Proveedores",
115         c => new
116         {
117             Dni = c.String(nullable: false, maxLength: 128),
118             Ruc = c.String(),
119             Descripcion = c.String(),
120             Telefono = c.String(),
121             Rubro = c.String(),
122             Direccion = c.String(),
123             Estado = c.String(),
124         })
125     .PrimaryKey(t => t.Dni);

```

```

127     CreateTable(
128         "dbo.Inventarios",
129         c => new
130         {
131             Codigo = c.String(nullable: false, maxLength: 128),
132             Descripcion = c.String(),
133             Cantidad = c.String(),
134             Precio = c.String(),
135             Estado = c.String(),
136             Producto = c.String(),
137             Productos_Codigo = c.String(maxLength: 128),
138             Proveedor_Dni = c.String(maxLength: 128),
139         })
140     .PrimaryKey(t => t.Codigo)
141     .ForeignKey("dbo.Productos", t => t.Productos_Codigo)
142     .ForeignKey("dbo.Proveedores", t => t.Proveedor_Dni)
143     .Index(t => t.Productos_Codigo)
144     .Index(t => t.Proveedor_Dni);
145
146     CreateTable(
147         "dbo.Pedidos",
148         c => new
149         {
150             Codigo = c.String(nullable: false, maxLength: 128),
151             Fecha = c.String(),
152             Descripcion = c.String(),
153             Total = c.String(),
154             Cliente = c.String(),
155             Empleado = c.String(),
156             Estado = c.String(),
157             Clientes_Dni = c.String(maxLength: 128),
158             Empleados_Codigo = c.String(maxLength: 128),
159         })

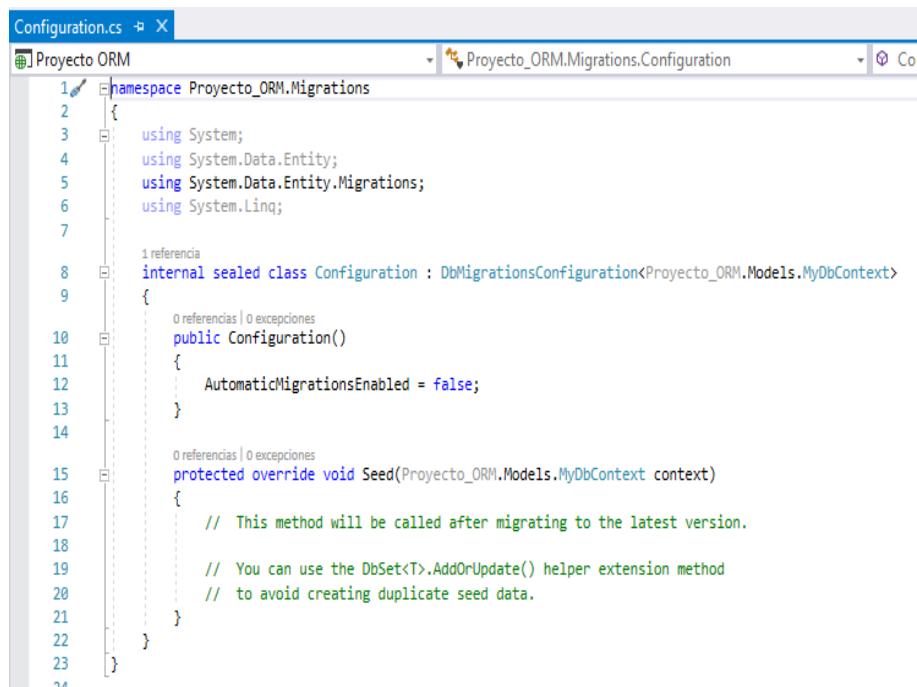
```

```

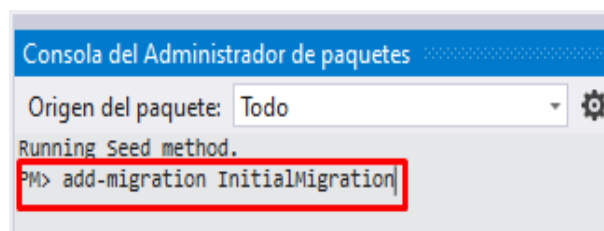
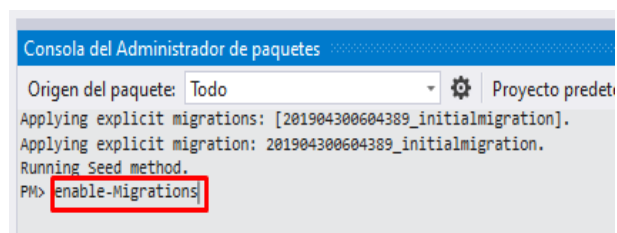
160         .PrimaryKey(t => t.Codigo)
161         .ForeignKey("dbo.Clientes", t => t.Clientes_Dni)
162         .ForeignKey("dbo.Empleados", t => t.Empleados_Codigo)
163         .Index(t => t.Clientes_Dni)
164         .Index(t => t.Empleados_Codigo);
165     }
166     0 referencias | 0 excepciones
167     public override void Down()
168     {
169         DropForeignKey("dbo.Pedidos", "Empleados_Codigo", "dbo.Empleados");
170         DropForeignKey("dbo.Pedidos", "Clientes_Dni", "dbo.Clientes");
171         DropForeignKey("dbo.Inventarios", "Proveedor_Dni", "dbo.Proveedores");
172         DropForeignKey("dbo.Inventarios", "Productos_Codigo", "dbo.Productos");
173         DropForeignKey("dbo.Productos", "Proveedor_Dni", "dbo.Proveedores");
174         DropForeignKey("dbo.Detalles", "Boletas_Serie", "dbo.Boletas");
175         DropForeignKey("dbo.Boletas", "Ruc", "dbo.Empleados");
176         DropForeignKey("dbo.Boletas", "Codigo", "dbo.Empleados");
177         DropForeignKey("dbo.Empleados", "Cargo", "dbo.Cargos");
178         DropForeignKey("dbo.Boletas", "Dni", "dbo.Clientes");
179         DropIndex("dbo.Pedidos", new[] { "Empleados_Codigo" });
180         DropIndex("dbo.Pedidos", new[] { "Clientes_Dni" });
181         DropIndex("dbo.Inventarios", new[] { "Proveedor_Dni" });
182         DropIndex("dbo.Inventarios", new[] { "Productos_Codigo" });
183         DropIndex("dbo.Productos", new[] { "Proveedor_Dni" });
184         DropIndex("dbo.Detalles", new[] { "Boletas_Serie" });
185         DropIndex("dbo.Detalles", new[] { "Codigo" });
186         DropIndex("dbo.Empleados", new[] { "Cargo" });
187         DropIndex("dbo.Boletas", new[] { "Codigo" });
188         DropIndex("dbo.Boletas", new[] { "Dni" });
189         DropIndex("dbo.Boletas", new[] { "Ruc" });
190         DropTable("dbo.Pedidos");
191         DropTable("dbo.Inventarios");
192         DropTable("dbo.Proveedores");
193         DropTable("dbo.Productos");
194         DropTable("dbo.Detalles");
195         DropTable("dbo.Empleados");
196         DropTable("dbo.Cargos");
197         DropTable("dbo.Clientes");
198         DropTable("dbo.Boletas");
199     }
200 }
201 }
202 }

```

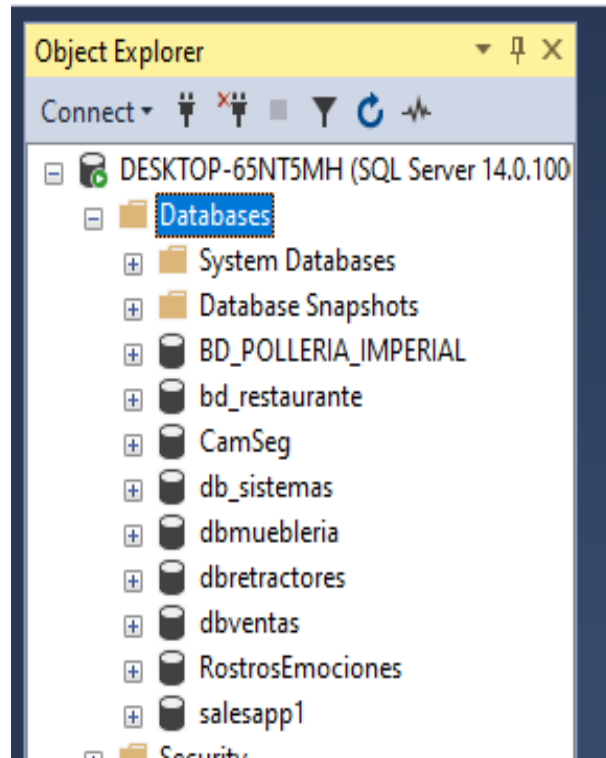
La clase donde se configura la migración Configuration.cs



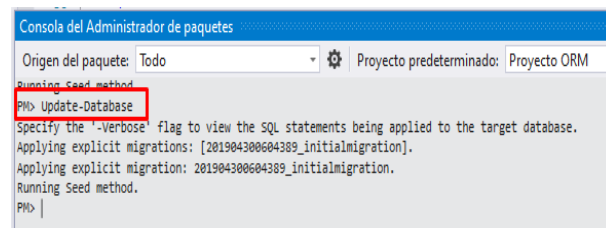
Comandos para la migración.

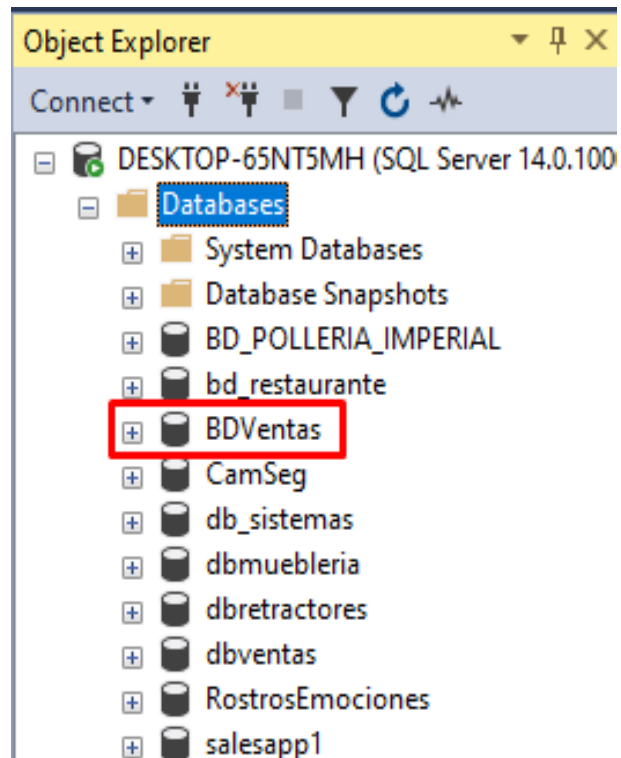


Se ve que el servidor no tiene la base de datos DBVentas.

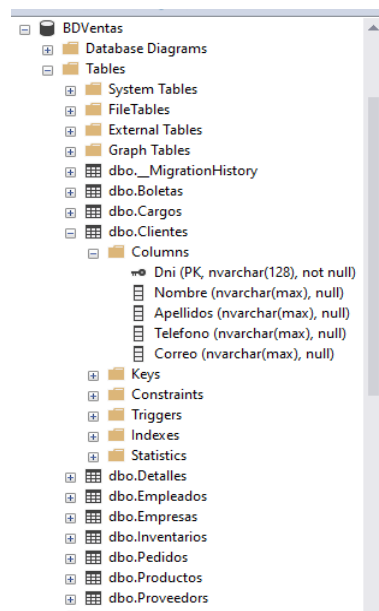


Con este comando logramos que migre y cree la base de datos del proyecto
Como se puede ver se crea la base de datos de nuestro proyecto





Dentro de la base de datos estan todas las propiedades de nuestras clases del proyecto asi como las clases mismas.



3) Diccionario de datos.

Formulario FrmClientes			
Campo	Tamaño	Tipo de Dato	Descripción
txtDni	8	varchar	DNI del cliente.
txtNombre	50	varchar	Nombres del cliente.
txtApellido	50	varchar	Apellido Paterno y Materno del cliente.
txtTelefono	12	varchar	Número telefónico del cliente.
txtEmail	100	varchar	Email del cliente.
txtDireccion	50	varchar	Dirección familiar del cliente.
txtBuscar	8	varchar	Búsqueda del cliente a través del DNI.
dgvCliente	-	general	Muestra todo el registro del FrmClientes.

Formulario FrmEmpleados			
Campo	Tamaño	Tipo de Dato	Descripción
txtCodigo	8	varchar	Código del empleado.
txtNombre	50	varchar	Nombres del empleado.
txtApellido	50	varchar	Apellido Paterno y Materno del empleado.
txtDireccion	50	varchar	Dirección familiar del empleado.
txtEmail	100	varchar	Email del empleado.
cbCargo	50	varchar	Cargo del empleado.
txtClave	8	varchar	Clave asignado al empleado.
rbEstado	8	varchar	Estado actual del empleado.
txtBuscar	8	varchar	Búsqueda al empleado por su código.
dgvEmpleado	-	general	Muestra todo el registro del FrmEmpleados.

Formulario FrmMascotas			
Campo	Tamaño	Tipo de Dato	Descripción
txtDni	8	varchar	Documento Nacional de Identidad del cliente.
txtNombre	50	varchar	Nombre de la mascota del cliente.
txtEspecie	50	varchar	Especie de la mascota del cliente.
txtRaza	50	varchar	Raza de la mascota del cliente.
txtPeso	12	varchar	Peso de la mascota del cliente.
txtSexo	6	varchar	Género de la mascota del cliente.
txtNacimiento	-	date	Fecha del nacimiento de la mascota.
txtEstado	8	varchar	Estado actual de la mascota del cliente.
txtBuscar	8	varchar	Búsqueda de la mascota por el DNI.
dgvCliente	max	general	Muestra todo el registro del FrmMascotas.

Formulario FrmServicioVacunación			
Campo	Tamaño	Tipo de Dato	Descripción
txtDni	8	varchar	Documento Nacional de Identidad del cliente.
txtNombre	50	varchar	Nombre de la mascota del cliente.
txtVacuna	50	varchar	Tipo de vacuna del servicio clínico.
txtDetalle	100	varchar	Detalle de la aplicación de la vacuna.
txtFecha	-	date	Fecha de la aplicación de la vacuna.
txtPrecio	11	int	Precio de la aplicación de la vacuna.
txtBuscar	8	varchar	Búsqueda de la vacuna por el DNI de cliente.
dgvVacunas	max	General	Muestra todo el registro del FrmServicios.

Formulario FrmMantenimientoMascota			
Campo	Tamaño	Tipo de Dato	Descripción
txtDni	8	varchar	Documento Nacional de Identidad.
txtNombre	50	varchar	Nombre de la mascota del cliente.
txtServicio	50	varchar	Tipo de servicio clínico.
txtDetalle	100	varchar	Detalle del servicio clínico ofrecido.
txtFecha	-	date	Fecha del servicio ofrecido.
txtPrecio	11	int	Precio del servicio ofrecido.
txtBuscar	8	varchar	Búsqueda del servicio por el DNI de cliente.
dgvMantenimiento	max	general	Muestra todo el registro del Formulario.

Formulario FrmTratamientoMascota			
Campo	Tamaño	Tipo de Dato	Descripción
txtDni	8	varchar	Documento Nacional de Identidad del cliente.
txtNombre	50	varchar	Nombre de la mascota del cliente.
txtTratamiento	50	varchar	Tipo de tratamiento clínico ofrecido.
txtDetalle	100	varchar	Detalle del tratamiento clínico ofrecido.
txtReceta	100	varchar	Receta del tratamiento clínico ofrecido.
txtFecha	-	date	Fecha del tratamiento ofrecido.
txtCita	-	date	Fecha de la próxima cita del tratamiento.
txtPrecio	11	int	Precio del servicio ofrecido.
txtBuscar	8	varchar	Búsqueda del servicio por el DNI de cliente.
dgvTratamiento	max	general	Muestra todo el registro del FrmTratamiento.

Formulario FrmAgenda			
Campo	Tamaño	Tipo de Dato	Descripción
txtDni	8	varchar	Documento Nacional de Identidad del cliente.
txtNombre	50	varchar	Nombres del cliente.
txtApellido	50	varchar	Apellido Paterno y Materno del cliente.
txtTelefono	12	varchar	Número telefónico del cliente.
txtFecha	-	date	Fecha del próximo evento programado.
txtHora	6	varchar	Hora del evento programado.
txtActividad	100	varchar	Descripción de la actividad programada.
txtBuscar	-	date	Búsqueda por fecha de la agenda.
dgvAgenda	max	general	Muestra el registro del formulario agenda.