**MANUAL DE TÉCNICO**

**Para:**

**Manual de Técnico**

**Sistema De Venta de Zapatos**

Versión 1.0

**Preparado por: ALVARADO GONZALEZ JEFFREY JEREMY**

**ARCE CELLAN DARLY NAYELI**

**CONTRERAS MALDONADO JORGE DANIEL**

**MONRROY ESPINOZA JORLY WILFRIDO**

**PINTAG SANGA GLEN EDISON**

**YEPEZ ALONSO ISAAC GUILLERMO**

**Fecha:** Guayaquil 2024

## Contenido

1. [INTRODUCCIÓN 4](#_bookmark0)
2. [OBJETIVO DE ESTE MANUAL 4](#_bookmark1)
3. [DIRIGIDO 4](#_bookmark2)
4. [CONOCIMIENTOS PREVIOS 4](#_bookmark3)
   1. [PASOS PARA ENCENDER LA COMPUTADORA 4](#_bookmark4)
5. [ESPECIFICACIONES TÉCNICAS 5](#_bookmark6)
   1. [HARDWARE 5](#_bookmark7)
   2. [SOFTWARE 5](#_bookmark8)
6. [DESCRIPCIÓN DEL SISTEMA REDIGAD 5](#_bookmark9)
7. [DESCRIPCIÓN DE COTENIDO DEL SISTEMA REDIGAD 6](#_bookmark10)
   1. [Microsoft Visual studio 6](#_bookmark11)
   2. [SQL SERVER 6](#_bookmark12)
8. [FUNCIONES PRINCIPALES DEL SISTEMA 6](#_bookmark13)
9. [CREACIÓN DEL SISTEMA EN 7](#_bookmark15)
10. [CONEXIÓN DE BASE DE DATOS 7](#_bookmark17)
11. [CREACION DEL PATRÓN DEL MODELO VISTA CONTROLADOR 7](#_bookmark19)
12. [CREACION DE CONTROLADORES 8](#_bookmark21)

[12.2. EJEMPLO DEL CONTROLADOR EN EL SISTEMA 8](#_bookmark23)

1. [INGRESO A MODELO 11](#_bookmark25)

[13.1 DEFINICIÓN 11](#_bookmark26)

[13.2. EJEMPLO DEL MODELO EN EL SISTEMA 12](#_bookmark28)

1. [INGRESO A VISTAS 14](#_bookmark30)
   1. [DEFINICIÓN 14](#_bookmark31)
   2. [EJEMPLO DE VISTA EN EL SISTEMA 14](#_bookmark33)
2. [DISTRIBUCIÓN DE ROLES DE USUARIO 15](#_bookmark36)

**TABLA DE ILUSTRACIONES**

[Ilustración 1 Botón de Encendido 5](#_bookmark5)

[Ilustración 2 Plataforma y servidor local para el desarrollo 7](#_bookmark14)

[Ilustración 5 Conexión con la base de datos utilizada 7](#_bookmark18)

[Ilustración 6 Estructura del MVC 8](#_bookmark20)

[Ilustración 7 Controladores creados para el sistema 8](#_bookmark22)

[Ilustración 8 Ejemplo del controlador de archivo documento 8](#_bookmark24)

[Ilustración 9 Modelos del sistema 11](#_bookmark27)

[Ilustración 10 Ejemplo dl modelo del sistema 12](#_bookmark29)

[Ilustración 11 Vistas de sistema 14](#_bookmark32)

[Ilustración 12 Ejemplo de vista del sistema 14](#_bookmark34)

[Ilustración 13 Diagrama General de Caso de Uso 15](#_bookmark37)

[Ilustración 14 Ingresos ala sistema de acuerdo al rol 16](#_bookmark38)

## INTRODUCCIÓN

## El sistema de Ventas de zapatos esta compuesto de forma que permita que la información contenida sea agregada , editada y/o eliminada por el personal encargado del departamento.

Para alcanzar estos propósitos se ha hecho uso de C Sharp, que es un lenguaje que se adecua a las nuevas necesidades de las aplicaciones actuales de este sistema, sin la necesidad de actualizaciones, a través de esta tecnología la administración y seguridad de la información se dará de forma centralizada y segura de datos, al mismo tiempo facilita la actualización eficiente de dicha información. En cuanto a este manual se ha considerado incluir todos los aspectos técnicos necesarios para el manejo y control de esta aplicación para el Departamento de Gestión Documental.

## OBJETIVO DE ESTE MANUAL

El objetivo primordial de este Manual es ayudar y guiar al técnico a informarse y utilizar herramientas para que el Sistema De ventas de zapatos del Departamento de Gestión Documental entre en producción (ejecución), para de esa manera poder hacer uso de la información deseada para poder despejar todas las dudas existentes y para poder comprender:

* Guía para gestión de herramientas para poner en funcionamiento el sistema para el sistema de ventas de zapatos
* Conocer cómo utilizar el sistema, mediante una descripción detallada e ilustrada de las opciones.
* Conocer el alcance de toda la información por medio de una explicación detallada e ilustrada de cada una de las páginas que lo conforman el manual técnico.

## DIRIGIDO

Este manual está orientado a los técnicos u otros tipos de personal encargado del de Gestión. Solamente dichas personas están autorizadas a realizar modificaciones en el sistema.

Una vez finalizado el proyecto, el Departamento o persona será el encargado de definir políticas, normas, etc. para la administración del sistema.

También a través de este manual el personal podrá estar en la capacidad de supervisar el cumplimiento de políticas, normas, etc. que permitan el correcto funcionamiento de los Sistemas.

Definir conjuntamente con los departamentos pertinentes, los contenidos o cambios para el o los sistema para que también de igual forma puedan ser capacitados en herramientas necesarias para el mantenimiento y ejecución

## CONOCIMIENTOS PREVIOS

Los conocimientos mínimos que deben tener las personas que operarán las páginas y deberán utilizar este manual son:

* Conocimientos básicos acerca de sistemas de gestión
* Conocimientos básicos en base de datos
* Conocimiento básico de Windows

## PASOS PARA ENCENDER LA COMPUTADORA

Encienda el C.P.U. presionar el botón Power Encienda el monitor presionar el botón Power



**Ilustración 1 Botón de Encendido**

Espere mientras carga el Sistema Operativo. La apariencia de la pantalla mientras se carga el sistema es de un color negro y se aprecia la frase de iniciar Windows.

Automáticamente aparecerá la pantalla de Windows. La pantalla de Windows puede ser de varios tipos o diseño.

## ESPECIFICACIONES TÉCNICAS

Para la implementación del Sistema Venta de zapatos para la preservación y validar la información obtenida, para lo cual requerimos lo siguiente:

## HARDWARE Cliente Requerido

El Software soporta sql server, Microsoft visual studio, C #, y las siguientes versiones.

## Servidor Windows

El sistema requiere básicamente todo lo que una instalación de un sistema de escritorio con su base de datos

Se recomienda que se utilicen los requerimientos expuestos anteriormente para la mejor funcionalidad del Sistema requerido

## SOFTWARE

* La Base de Datos requerida para el sistema es SQL server

## DESCRIPCIÓN DEL SISTEMA VENTAS DE ZAPATOS

El Sistema constará de toda la información preservada que se maneja en el departamento encargado de la gestión del sistema. Contendrá una la información necesaria para poder registrar usuarios con sus datos así también un historial de comprar realizadas por los clientes de este sistema, además de la cual será aprobada por el encardo de gestión

## Menú de Registro:

-Cedula

-Nombre

-Apellidos

-Teléfono

-Email

-Dirección

-Fecha

## Menú Cliente

* Buscar Cliente
* Modificar Cliente
* Eliminar cliente

## DESCRIPCIÓN DE COTENIDO DEL SISTEMA REDIGAD

Para la realización de este Sistema Venta de Zapatos se basó fundamentalmente en las necesidades que tenía el Departamento de gestión, aquí pueden preservar toda la información relacionada con su funcionamiento y descripción en general. Uno de los requerimientos principales es poder llevar un registro de los clientes. Con algunos de sus datos personales para contacto.

* Plataforma C#
  + Imágenes
  + Módulos
  + Plugins
* SQL Server

SQLQueryClientePCs

SQLQueryUsuarioPCs

SQLQueryZapatoPCS

A continuación, se tiene toda la información fundamental de cada uno de estos aspectos.

## Microsoft Visual studio y C#

Visual Studio Code es un editor de código fuente, ligero pero poderoso, desarrollado por Microsoft. Aunque es altamente personalizable y extensible, se destaca especialmente en el desarrollo de aplicaciones con C#. Al igual que NetBeans, VS Code ofrece una plataforma modular que permite a los desarrolladores ampliar sus funcionalidades de manera significativa.

Un aspecto clave de VS Code es su arquitectura basada en extensiones. Estas extensiones son paquetes de código que añaden nuevas características al editor, como soporte para lenguajes de programación adicionales (más allá de C#), herramientas de depuración, integración con sistemas de control de versiones, y mucho más. Cada extensión es desarrollada de forma independiente y puede ser instalada fácilmente desde el marketplace de VS Code.

**¿Por qué VS Code y C# son una combinación poderosa?**

* **Flexibilidad:** La combinación de VS Code y C# proporciona un entorno de desarrollo altamente flexible. Los desarrolladores pueden personalizar su entorno de trabajo de acuerdo a sus preferencias y necesidades específicas.
* **Extensibilidad:** Gracias a su arquitectura basada en extensiones, VS Code se puede adaptar a una amplia variedad de proyectos y flujos de trabajo.
* **Comunidad activa:** Tanto VS Code como C# cuentan con una gran comunidad de desarrolladores que contribuyen con nuevas extensiones, herramientas y recursos.

## Sql Sever

SQL Server es un sistema gestor de bases de datos relacionales (SGBDR) desarrollado por Microsoft. Se utiliza para almacenar, organizar y administrar grandes cantidades de información de manera estructurada. Imagina una biblioteca gigante, pero en lugar de libros, guarda datos sobre clientes, productos, ventas, etc.

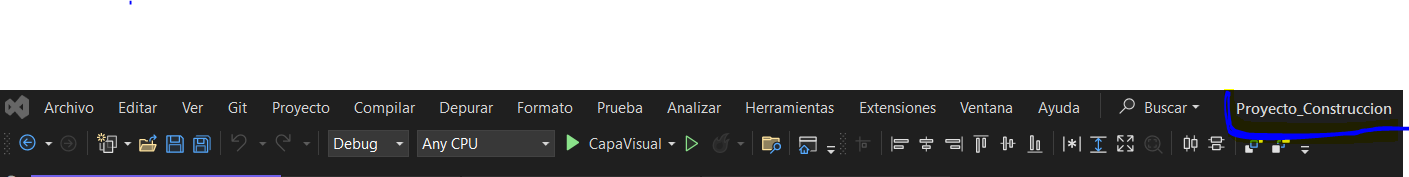
**¿Por qué es importante SQL Server?**

* **Organización:** Almacena datos de manera ordenada y accesible, facilitando su búsqueda y análisis.
* **Confiabilidad:** Ofrece herramientas para asegurar la integridad y disponibilidad de los datos, incluso en caso de fallos.
* **Flexibilidad:** Se adapta a una amplia variedad de aplicaciones, desde pequeñas bases de datos hasta grandes sistemas empresariales.
* **Potencia:** Permite realizar consultas complejas y procesar grandes volúmenes de datos de manera eficiente.

## FUNCIONES PRINCIPALES DEL SISTEMA VENTAS DE ZAPATOS

Se requiere instalar los dos programas bases para la ejecución del sistema Venta de Zapatos, entre ellos tenemos a C#. y SQL Server 2022.

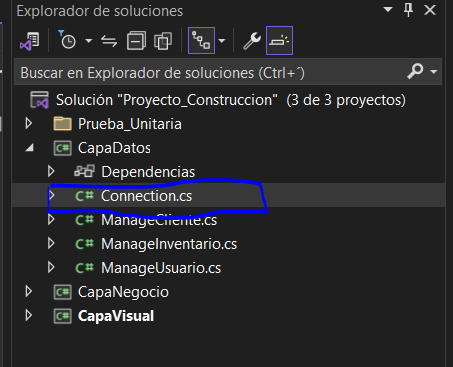
## CREACIÓN DEL SISTEMA EN Microsoft Visual studio.



**Debemos verificar que estemos en la ruta correcta con el proyecto correcto**

# Ilustración 3 Creación del Proyecto en la plataforma visual studio

## CONEXIÓN DE BASE DE DATOS.



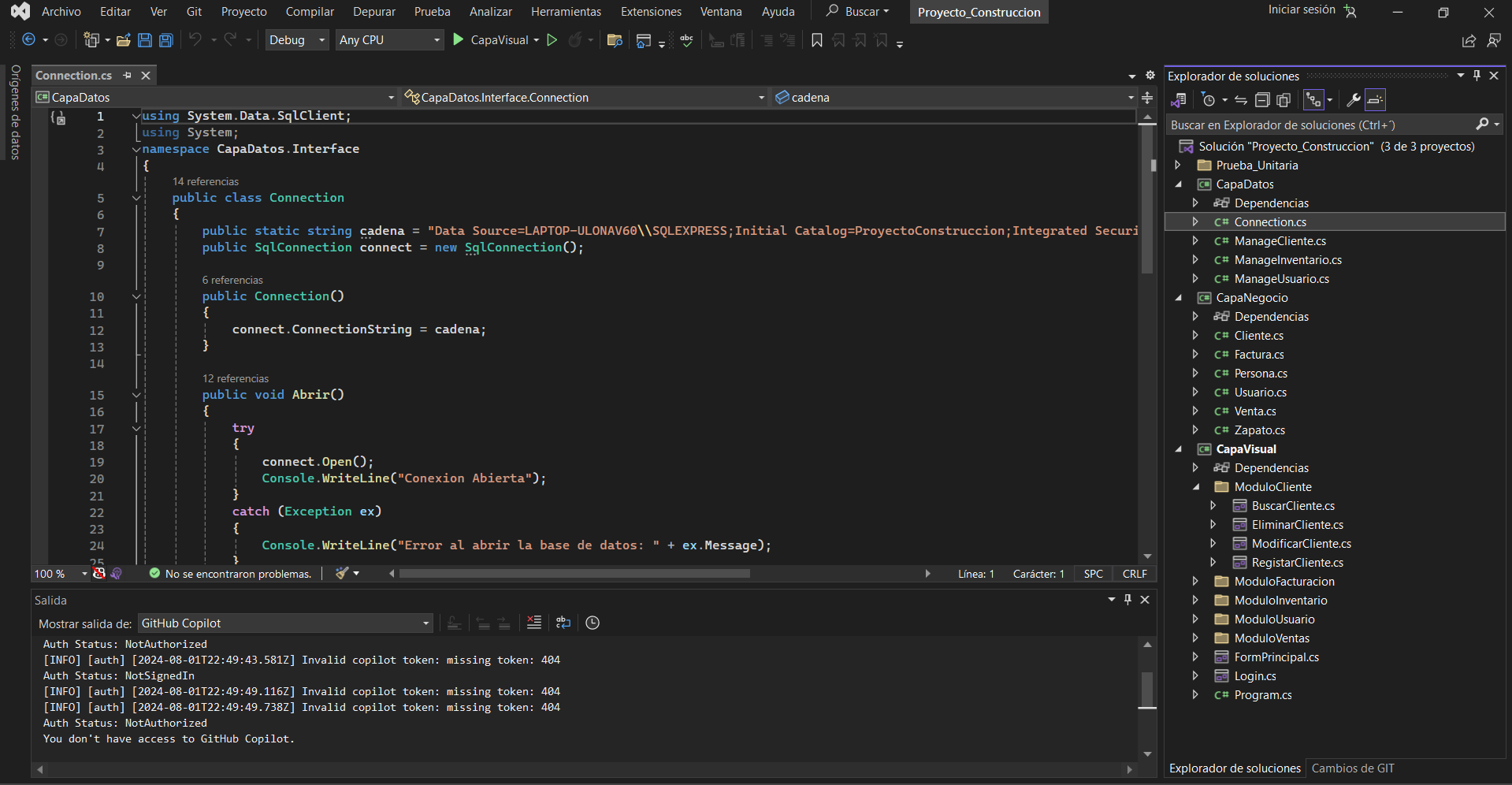
# Ilustración 4 Conexión con la base de datos utilizada

## CREACION DEL PATRÓN DEL MODELO VISTA CONTROLADOR

El Sistema está estructurado con un patrón de diseño de software utilizado para implementar sistemas donde se requiere el uso de interfaces de usuario. Se potencia la facilidad de mantenimiento, reutilización del código y la separación de conceptos.

Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad:

* Modelo.
* Vista.
* Controlador.



# Ilustración 5 Estructura del MVC

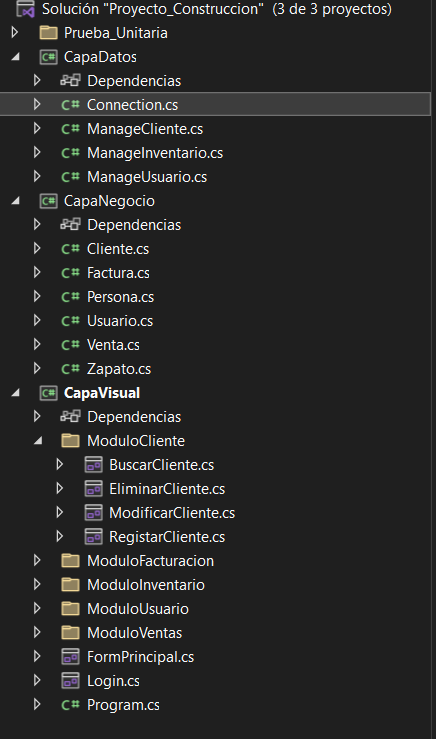
## CREACION DE CONTROLADORES

* 1. **DEFINICIÓN:**

Es el intermediario entre la Vista y el Modelo, se encarga de controlar las interacciones del usuario en la vista, contiene el código necesario para responder a las acciones que se solicitan en la aplicación.

Si se solicita desarrollar un sistema de administración de usuarios con un CRUD (Create, Read, Update, Delete).

* Pide los datos al modelo y los devuelve de nuevo a la vista para que ésta los muestre al usuario.
* Es decir las llamadas a clases y métodos, y los datos recibidos de formularios.
* La conexión entre la parte gráfica y los datos y los eventos que se producen cuando manejamos la aplicación.



# Ilustración 6 Controladores creados para el sistema

## EJEMPLO DEL CONTROL DE CONECCION EN EL SISTEMA CLIENTE

## CÓDIGO:

using CapaDatos.Interface;

using System.Data.SqlClient;

using System.Data;

namespace CapaDatos

{

public class ManageCliente

{

private Connection connection = new Connection();

public ManageCliente()

{

connection = new Connection();

}

// Método para insertar un cliente

public void InsertarCliente(string cedula, string nombre, string apellido, string telefono, string email, string direccion, DateTime fechaRegistro)

{

using (SqlCommand cmd = new SqlCommand("sp\_InsertarCliente", connection.connect))

{

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Cedula", cedula);

cmd.Parameters.AddWithValue("@Nombre", nombre);

cmd.Parameters.AddWithValue("@Apellido", apellido);

cmd.Parameters.AddWithValue("@Telefono", telefono);

cmd.Parameters.AddWithValue("@Email", email);

cmd.Parameters.AddWithValue("@Direccion", direccion);

cmd.Parameters.AddWithValue("@FechaRegistro", fechaRegistro);

try

{

connection.Abrir();

cmd.ExecuteNonQuery();

Console.WriteLine("Cliente insertado exitosamente.");

}

catch (Exception ex)

{

Console.WriteLine("Error al insertar el cliente: " + ex.Message);

}

finally

{

connection.Cerrar();

}

}

}

// Método para actualizar un cliente

public void ActualizarCliente(string cedula, string nombre, string apellido, string telefono, string email, string direccion, DateTime fechaRegistro)

{

using (SqlCommand cmd = new SqlCommand("sp\_ActualizarCliente", connection.connect))

{

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Cedula", cedula);

cmd.Parameters.AddWithValue("@Nombre", nombre);

cmd.Parameters.AddWithValue("@Apellido", apellido);

cmd.Parameters.AddWithValue("@Telefono", telefono);

cmd.Parameters.AddWithValue("@Email", email);

cmd.Parameters.AddWithValue("@Direccion", direccion);

cmd.Parameters.AddWithValue("@FechaRegistro", fechaRegistro);

try

{

connection.Abrir();

cmd.ExecuteNonQuery();

Console.WriteLine("Cliente actualizado exitosamente.");

}

catch (Exception ex)

{

Console.WriteLine("Error al actualizar el cliente: " + ex.Message);

}

finally

{

connection.Cerrar();

}

}

}

// Método para eliminar un cliente

public void EliminarCliente(string cedula)

{

using (SqlCommand cmd = new SqlCommand("sp\_EliminarCliente", connection.connect))

{

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Cedula", cedula);

try

{

connection.Abrir();

cmd.ExecuteNonQuery();

Console.WriteLine("Cliente eliminado exitosamente.");

}

catch (Exception ex)

{

Console.WriteLine("Error al eliminar el cliente: " + ex.Message);

}

finally

{

connection.Cerrar();

}

}

}

// Método para obtener un cliente por cédula

public DataTable ObtenerCliente(string cedula)

{

DataTable dt = new DataTable();

using (SqlCommand cmd = new SqlCommand("sp\_ObtenerCliente", connection.connect))

{

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Cedula", cedula);

try

{

connection.Abrir();

using (SqlDataAdapter da = new SqlDataAdapter(cmd))

{

da.Fill(dt);

}

}

catch (Exception ex)

{

Console.WriteLine("Error al obtener el cliente: " + ex.Message);

}

finally

{

connection.Cerrar();

}

}

return dt;

}

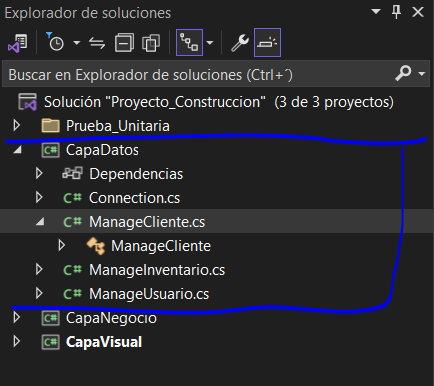
}

}

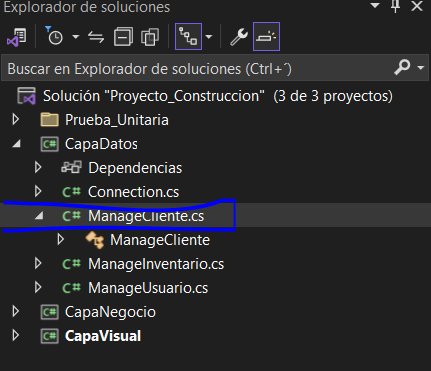
## INGRESO A MODELO

## 13.1 DEFINICIÓN

Contiene la lógica de la aplicación., son las clases y métodos que se comunican directamente con la base de datos, todas las funciones que accederán a las tablas y harán los correspondientes SELECT, UPDATE, INSERT, DELETE.



# Ilustración 8 Modelos del sistema



# Ilustración 9 Ejemplo del modelo del sistema de cliente

## CÓDIGO:

namespace CapaDatos

{

public class ManageCliente

{

private Connection connection = new Connection();

public ManageCliente()

{

connection = new Connection();

}

// Método para insertar un cliente

public void InsertarCliente(string cedula, string nombre, string apellido, string telefono, string email, string direccion, DateTime fechaRegistro)

{

using (SqlCommand cmd = new SqlCommand("sp\_InsertarCliente", connection.connect))

{

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Cedula", cedula);

cmd.Parameters.AddWithValue("@Nombre", nombre);

cmd.Parameters.AddWithValue("@Apellido", apellido);

cmd.Parameters.AddWithValue("@Telefono", telefono);

cmd.Parameters.AddWithValue("@Email", email);

cmd.Parameters.AddWithValue("@Direccion", direccion);

cmd.Parameters.AddWithValue("@FechaRegistro", fechaRegistro);

try

{

connection.Abrir();

cmd.ExecuteNonQuery();

Console.WriteLine("Cliente insertado exitosamente.");

}

catch (Exception ex)

{

Console.WriteLine("Error al insertar el cliente: " + ex.Message);

}

finally

{

connection.Cerrar();

}

}

}

// Método para actualizar un cliente

public void ActualizarCliente(string cedula, string nombre, string apellido, string telefono, string email, string direccion, DateTime fechaRegistro)

{

using (SqlCommand cmd = new SqlCommand("sp\_ActualizarCliente", connection.connect))

{

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Cedula", cedula);

cmd.Parameters.AddWithValue("@Nombre", nombre);

cmd.Parameters.AddWithValue("@Apellido", apellido);

cmd.Parameters.AddWithValue("@Telefono", telefono);

cmd.Parameters.AddWithValue("@Email", email);

cmd.Parameters.AddWithValue("@Direccion", direccion);

cmd.Parameters.AddWithValue("@FechaRegistro", fechaRegistro);

try

{

connection.Abrir();

cmd.ExecuteNonQuery();

Console.WriteLine("Cliente actualizado exitosamente.");

}

catch (Exception ex)

{

Console.WriteLine("Error al actualizar el cliente: " + ex.Message);

}

finally

{

connection.Cerrar();

}

}

}

// Método para eliminar un cliente

public void EliminarCliente(string cedula)

{

using (SqlCommand cmd = new SqlCommand("sp\_EliminarCliente", connection.connect))

{

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Cedula", cedula);

try

{

connection.Abrir();

cmd.ExecuteNonQuery();

Console.WriteLine("Cliente eliminado exitosamente.");

}

catch (Exception ex)

{

Console.WriteLine("Error al eliminar el cliente: " + ex.Message);

}

finally

{

connection.Cerrar();

}

}

}

// Método para obtener un cliente por cédula

public DataTable ObtenerCliente(string cedula)

{

DataTable dt = new DataTable();

using (SqlCommand cmd = new SqlCommand("sp\_ObtenerCliente", connection.connect))

{

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Cedula", cedula);

try

{

connection.Abrir();

using (SqlDataAdapter da = new SqlDataAdapter(cmd))

{

da.Fill(dt);

}

}

catch (Exception ex)

{

Console.WriteLine("Error al obtener el cliente: " + ex.Message);

}

finally

{

connection.Cerrar();

}

}

return dt;

}

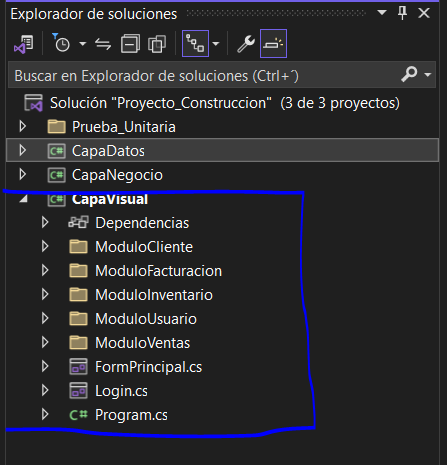
}

}

## INGRESO A VISTAS

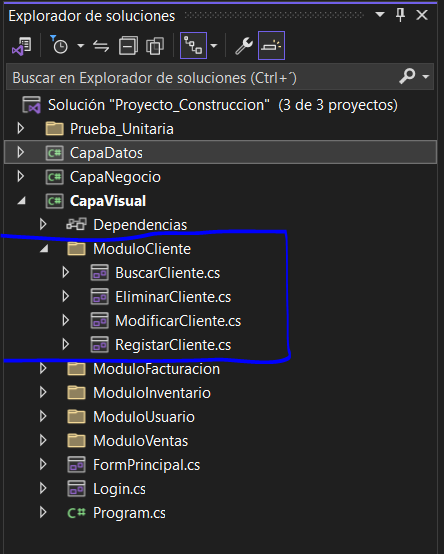
## DEFINICIÓN

Programaremos lo relativo a la interfaz gráfica, muestra la información al usuario e interactúa con él. Las vistas requerirán los datos a los modelos y ellas se generarán las salidas, tal como nuestra aplicación requiera.



# Ilustración 10 Vistas de sistema REDIGAD

## EJEMPLO DE VISTA EN EL SISTEMA



# Ilustración 11 Ejemplo de vista del sistema Cliente

## Verificación de SQL Server y Queri para ceración de base de datos

**Los Queri están divididos en tres que son Cliente Usuario Y Zapatos**

## 

# Ilustración 11 Ejemplo de vista De Queri SQL

**que contienen los métodos para**

Crear tabla

Crear procedimiento almacenado para insertar

Crear procedimiento almacenado para actualizar

Crear procedimiento almacenado para eliminar

Crear procedimiento almacenado para obtener un usuario

Ejemplo código De Sql Server para usuario

-- Crear base de datos

CREATE DATABASE ProyectoConstruccion;

GO

-- Usar base de datos

USE ProyectoConstruccion;

GO

-- Crear tabla Usuario

CREATE TABLE Usuario (

Id INT PRIMARY KEY IDENTITY(1,1),

Cedula VARCHAR(10) NOT NULL,

Nombre VARCHAR(50) NOT NULL,

Apellido VARCHAR(50) NOT NULL,

Telefono VARCHAR(10) NOT NULL,

Email VARCHAR(100) NOT NULL,

NombreUsuario VARCHAR(50) NOT NULL,

Contrasenia VARCHAR(100) NOT NULL,

Rol VARCHAR(20) NOT NULL

);

GO

-- Crear procedimiento almacenado para insertar un usuario

CREATE PROCEDURE sp\_InsertarUsuario

@Cedula VARCHAR(10),

@Nombre VARCHAR(50),

@Apellido VARCHAR(50),

@Telefono VARCHAR(10),

@Email VARCHAR(100),

@NombreUsuario VARCHAR(50),

@Contrasenia VARCHAR(100),

@Rol VARCHAR(20)

AS

BEGIN

INSERT INTO Usuario (Cedula, Nombre, Apellido, Telefono, Email, NombreUsuario, Contrasenia, Rol)

VALUES (@Cedula, @Nombre, @Apellido, @Telefono, @Email, @NombreUsuario, @Contrasenia, @Rol);

END;

GO

-- Crear procedimiento almacenado para actualizar un usuario

CREATE PROCEDURE sp\_ActualizarUsuario

@Cedula VARCHAR(10),

@Nombre VARCHAR(50),

@Apellido VARCHAR(50),

@Telefono VARCHAR(10),

@Email VARCHAR(100),

@NombreUsuario VARCHAR(50),

@Contrasenia VARCHAR(100),

@Rol VARCHAR(20)

AS

BEGIN

UPDATE Usuario

SET Cedula = @Cedula,

Nombre = @Nombre,

Apellido = @Apellido,

Telefono = @Telefono,

Email = @Email,

NombreUsuario = @NombreUsuario,

Contrasenia = @Contrasenia,

Rol = @Rol

WHERE Cedula = @Cedula;

END;

GO

-- Crear procedimiento almacenado para eliminar un usuario

CREATE PROCEDURE sp\_EliminarUsuario

@Cedula VARCHAR(10)

AS

BEGIN

DELETE FROM Usuario

WHERE Cedula = @Cedula;

END;

GO

-- Crear procedimiento almacenado para obtener un usuario por cédula

CREATE PROCEDURE sp\_ObtenerUsuario

@Cedula VARCHAR(10)

AS

BEGIN

SELECT \* FROM Usuario

WHERE Cedula = @Cedula;

END;

GO

## DISTRIBUCIÓN DE ROLES DE USUARIO

Para la programación de este aplicativo se definieron 3 tipos de usuarios o roles donde cada uno interactúa con el sistema de diferentes modos. En la figura 1 se ilustra los roles y casos de uso que realiza cada usuario dentro del sistema.

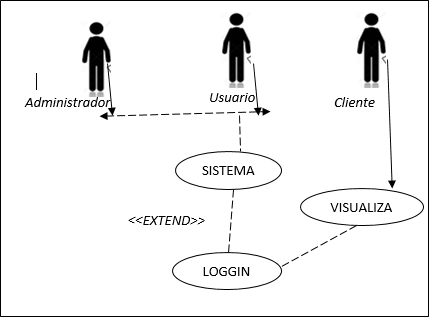
# Ilustración 12 Diagrama General de Caso de Uso

En el sistema encontramos 3 tipos de usuarios con son: administrador, usuario y cliente.

**Administrador:** Este usuario realiza tareas de preservar los archivos del sistema, registrar un nuevo usuario, editar información del usuario, asignar acciones y roles al usuario, visualizar a todos los usuarios registrados, acceder al sistema, administrar los archivos del sistema

**Usuario:** Este usuario realiza tareas de acceder al sistema, administrar los archivos del sistema, manipula información de los documentos

**Cliente:** este usuario realiza tareas de acceder al sistema y visualiza los documentos almacenados.



# Ilustración 13 Ingresos ala sistema de acuerdo al rol