Imagen que contiene interior, mamífero, gato, viendo

Descripción generada automáticamente

**Tema**

PATRONES DE SOFTWARE

**Tutor**

Ing. Eduardo Mauricio Campaña Ortega

MIS.MDU.CCNA.CCIA.

PhD. (c) Ingeniería de Software

PhD. (c) Seguridad Información

**Fecha**

22/02/2023

**PATRONES DE SOFTWARE**

[INTRODUCCIÓN 4](#_Toc127723167)

[OBJETIVOS 4](#_Toc127723168)

[OBJETIVO GENERAL 4](#_Toc127723169)

[OBJETIVOS ESPECIFICOS 4](#_Toc127723170)

[MARCO TEÓRICO 4](#_Toc127723171)

[¿QUE ES UN PATRÓN? 4](#_Toc127723172)

[¿QUÉ ES UN PATRON DE SOFTWARE? 6](#_Toc127723173)

[CARACTERISTICAS 7](#_Toc127723174)

[CATEGORIAS 7](#_Toc127723175)

[PATRONES DE DISEÑO 7](#_Toc127723176)

[PATRÓN DAO (DATA ACCESS OBJECT) 8](#_Toc127723177)

[TIPOS DE PATRONES DE DISEÑO 9](#_Toc127723178)

[ANTIPATRONES 10](#_Toc127723179)

[PATRÓN SINGLETON 10](#_Toc127723180)

[PATRÓN DTO (DATA TRANSFER OBJECT) 10](#_Toc127723181)

[RELACIÓN ENTRE PATRONES 10](#_Toc127723182)

[PARTE PRÁCTICA 10](#_Toc127723183)

[CREACIÓN DE LA SOLUCIÓN 10](#_Toc127723184)

[EJECUCIÓN DEL PROYECTO 10](#_Toc127723185)

[CONCLUSIONES 11](#_Toc127723186)

[RECOMENDACIONES 11](#_Toc127723187)

[BIBLIOGRAFÍA 11](#_Toc127723188)

**INDICE DE IMÁGENES**

[Figura 1. Patrón Greca Azteca 6](#_Toc127922225)

[Figura 2. Patrón de desbloqueo en celular 6](#_Toc127922226)

[Figura 3. Patrones de Diseño y Arquitectura 7](#_Toc127922227)

[Figura 4. Patrón de diseño Singleton 8](#_Toc127922228)

[Figura 5. Diagrama de Patrones de Diseño 9](#_Toc127922229)

[Figura 6. Patrón de Diseño DAO 10](#_Toc127922230)

[Figura 7. Clasificación de los Patrones de Diseño 11](#_Toc127922231)

[Figura 8. Creación de la base de datos 13](#_Toc127922232)

[Figura 9. Ingreso de registros a SQL Management Studio 13](#_Toc127922233)

[Figura 10. Ejecución del query para ingresar Clientes 14](#_Toc127922234)

[Figura 11. Consulta a la tabla Clientes 14](#_Toc127922235)

[Figura 12. Resultado de la consulta de Clientes 14](#_Toc127922236)

[Figura 13. Procedimiento de SQL para buscar Clientes 14](#_Toc127922237)

[Figura 14. Ventana de proyectos en Visual Studio 15](#_Toc127922238)

[Figura 15. Plantilla para el proyecto 15](#_Toc127922239)

[Figura 16. Nombre y ruta de alojamiento del proyecto 16](#_Toc127922240)

[Figura 17. Version de .NET para el proyecto 16](#_Toc127922241)

[Figura 18. Estructura para el proyecto 17](#_Toc127922242)

[Figura 19. Creación de nueva clase en DTO 17](#_Toc127922243)

[Figura 20. Creación de la clase ClienteDto 18](#_Toc127922244)

[Figura 21. Clase ClienteDto en la carpeta DTO 18](#_Toc127922245)

[Figura 22. Codificación e la clase ClienteDto 19](#_Toc127922246)

[Figura 23. Creación de una nueva clase en DAO 19](#_Toc127922247)

[Figura 24. Codificación de la clase ConexionDB para SQL Server 20](#_Toc127922248)

[Figura 25. Codificación de la Clase ClienteDaao 20](#_Toc127922249)

[Figura 26. Creación de las ventanas en UI 21](#_Toc127922250)

[Figura 27. Creación de la ventana FormPrincipal 21](#_Toc127922251)

[Figura 28. Diseño de la ventana FormPrincipal 22](#_Toc127922252)

[Figura 29. Codificación de la clase FormPrincipal 22](#_Toc127922253)

[Figura 30. Creación de la ventana FormClientes 23](#_Toc127922254)

[Figura 31. Diseño de la ventana FromClientes 23](#_Toc127922255)

[Figura 32. Codificación de la clase FormClientes 24](#_Toc127922256)

[Figura 33. Creación de la ventana FormLibros 24](#_Toc127922257)

[Figura 34. Diseño de la ventana FormLibros 25](#_Toc127922258)

[Figura 35. Codificación de la clase FormLibros 25](#_Toc127922259)

[Figura 36. Estructura final del proyecto 26](#_Toc127922260)

[Figura 37. Ejecución del proyecto 26](#_Toc127922261)

[Figura 38. Ventana principal con el panel de navegación 27](#_Toc127922262)

[Figura 39. Ventana para la búsqueda de Clientes 27](#_Toc127922263)

[Figura 40.Ventana para la búsqueda de Libros 28](#_Toc127922264)

**ÍNDICE DE TABLAS**

[Tabla 1. Ejemplo de implementación de clase conexión 31](#_Toc126525405)

# INTRODUCCIÓN

En la Ingeniera de Software un patrón es un elemento recurrente en un contexto particular que se puede utilizar como modelo para generar objetos o partes de ellos. La estructura única que se crea se conoce como el patrón fundamental. Los patrones se utilizan comúnmente en el desarrollo de software, como la arquitectura en capas, MVC y Repositorio. Los patrones de software describen estructuras y mecanismos específicos que se han demostrado que funcionan bien en situaciones particulares.

Un patrón de software se define como una herramienta para describir problemas recurrentes y sus soluciones en un contexto específico. Las soluciones recomendadas pueden ser una colaboración entre dos o más clases, objetos, servicios, procesos o componentes que trabajan juntos para resolver el problema identificado por el patrón. El patrón no resuelve un problema específico, sino que proporciona una solución general que puede ser aplicada en múltiples contextos. Los patrones de software se dividen en tres categorías principales: patrones de arquitectura, patrones de diseño y patrones de lenguaje.

Para ser considerado un patrón, debe ser un concepto probado y tener al menos tres implementaciones reales. La solución proporcionada por el patrón no debe ser obvia, y debe surgir de un proceso de resolución de problemas efectivo y comprobado. En general, los patrones de software ofrecen una solución general para problemas recurrentes que puede ser reutilizada y deben describir estructuras y mecanismos específicos que se han demostrado que funcionan bien en situaciones particulares.

# OBJETIVOS

## OBJETIVO GENERAL

Investigar sobre los patrones en la ingeniera de software para resolver problemas en el desarrollo de sistemas y aplicaciones, para esto realizar una práctica con el lenguaje de programación C#, SQL Server y Programación Orientada a Objetos.

## OBJETIVOS ESPECIFICOS

* Investigar sobre los patrones en la Ingeniería y desarrollo de Software, los tipos y categorías.
* Realizar una práctica para la búsqueda y filtrado de datos en el que se usen los patrones de diseño DTO, DAO, Singleton y MVC.
* Documentar todo el proceso de la parte práctica para usarlo como manual y guía en futuros proyectos.

# MARCO TEÓRICO

## ¿QUE ES UN PATRÓN?

Un patrón es un elemento que se repite de manera predecible en un entorno, y puede ser utilizado como modelo o plantilla para generar objetos o partes de ellos. Si los objetos que se crean tienen algo en común, se puede inferir la estructura del patrón fundamental y se dice que los objetos exhiben un único patrón. Un ejemplo claro de esto es una Greca Azteca.

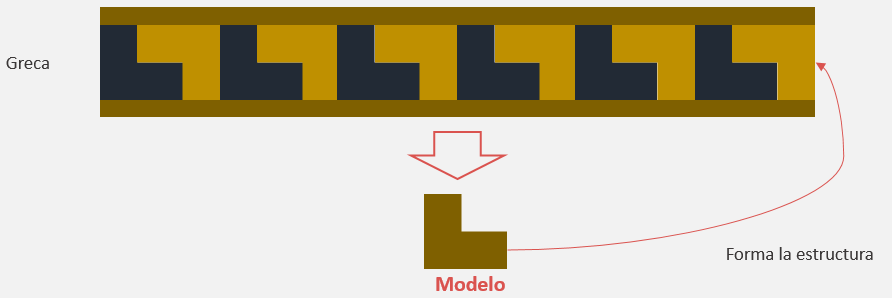


Figura 1. Patrón Greca Azteca

Como se puede observar, el elemento recurrente de un patrón forma una estructura única. Otro ejemplo de patrón es el de desbloqueo de un celular, donde la línea es el elemento recurrente que forma una estructura mediante un patrón único en el desarrollo de software.

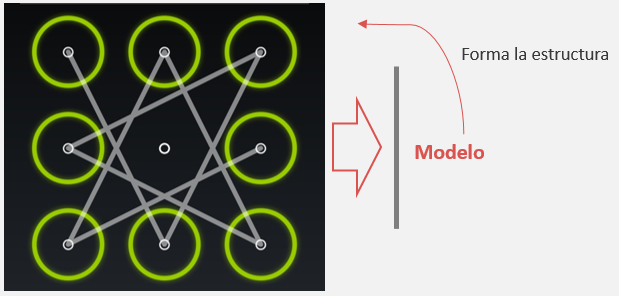


Figura 2. Patrón de desbloqueo en celular

En la construcción de una aplicación, se suelen utilizar muchos patrones, como: Arquitectura en Capas, MVC (Modelo-Vista-Controlador), DTO (Objeto de Transferencia de Datos), Repositorio.

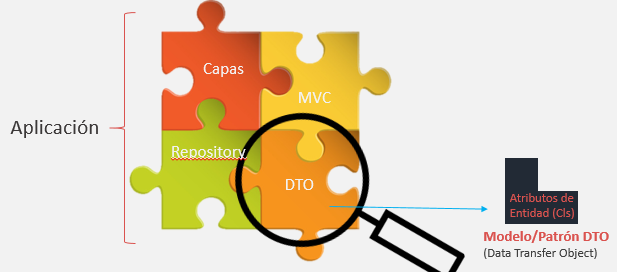


Figura 3. Patrones de Diseño y Arquitectura

Se define el patrón como un modelo recurrente en un contexto, y se puede decir que el patrón de Teo es un modelo que contiene solo atributos de una entidad, y que puede ser utilizado tantas veces como sea necesario. A pesar de que no es fácil establecer una definición definitiva de un patrón, se sabe que éstos tienen raíces en muchas áreas, especialmente en la arquitectura civil, basada en el libro en inglés "The Timeless Way of Building", escrito por Christopher Alexander en el año 1979.

El arquitecto Christopher Alexander define al patrón como una regla de tres partes que expresa una relación entre un contexto, un problema y una solución como un elemento en el mundo. Cada patrón es una relación entre un contexto, un sistema de fuerzas que ocurren repetidamente en ese contexto y una configuración espacial que permite que esas fuerzas se resuelvan entre sí. Aunque la teoría original de los patrones se aplica a la construcción de casas y planeamiento urbanístico, también se puede aplicar al software.

## ¿QUÉ ES UN PATRON DE SOFTWARE?

En el libro "The Sins", escrito por "Four" en 1995, se define el patrón como una herramienta para describir problemas recurrentes y sus soluciones en un contexto específico. En el mismo sentido, el libro "Enterprise Development Reference Architecture", escrito por Microsoft en 2004, ofrece una definición más técnica. Según esta definición, un patrón es una descripción de un problema recurrente que se presenta en un contexto particular y que recomienda una solución. La solución puede ser una colaboración entre dos o más clases, objetos, servicios, procesos o componentes que trabajan juntos para resolver el problema identificado por el patrón. La solución es usualmente simple y puede ser usada múltiples veces. Es decir, un patrón ofrece una solución general para problemas recurrentes que puede ser reutilizada.

Para ilustrar un ejemplo de problema recurrente, se menciona la creación de instancias múltiples, lo cual puede ser un problema en ciertos contextos, por ejemplo, al abrir un formulario. En este caso, se puede aplicar el patrón Singleton, que permite tener una instancia única del formulario. De esta manera, se resuelve el problema de crear instancias múltiples de un formulario.

Shape

Description automatically generated

Figura 4. Patrón de diseño Singleton

En conclusión, los patrones de software son modelos o plantillas que facilitan la reutilización de diseños y arquitecturas para resolver problemas recurrentes en la ingeniería de software. No resuelven un problema específico, sino que proporcionan una solución general que puede ser aplicada en múltiples contextos.

### CARACTERISTICAS

En cuanto a acelerar el proceso de construcción de software, se espera que un buen patrón de software tenga ciertas características. En primer lugar, debe resolver un problema específico y concreto. Es decir, los patrones capturan soluciones reales, en lugar de principios abstractos o estrategias teóricas. Además, para ser considerado un patrón, debe ser un concepto probado y tener al menos tres implementaciones reales. La solución proporcionada por el patrón no debe ser obvia, y debe surgir de un proceso de resolución de problemas efectivo y comprobado.

En general, los patrones de software describen estructuras y mecanismos específicos que se han demostrado que funcionan bien en situaciones particulares. Si bien muchas técnicas de resolución de problemas, como los paradigmas y métodos de diseño de software, pueden derivar soluciones a problemas, los mejores patrones de software generan soluciones efectivas de manera indirecta, a través de la descripción de estructuras y mecanismos específicos.

### CATEGORIAS

Los diferentes tipos de patrones de software que se utilizan en el desarrollo de sistemas de software se dividen en tres categorías principales: patrones de arquitectura, patrones de diseño y patrones de lenguaje. Los patrones de arquitectura son aquellos que afectan a la estructura global del sistema, mientras que los patrones de diseño definen microarquitecturas de subsistemas mediante clases y objetos. Los patrones de lenguaje son específicos de los lenguajes de programación y describen cómo implementar ciertos aspectos de un problema utilizando las características de un lenguaje de programación.

Además, se utiliza una representación gráfica para ilustrar estos patrones de software, donde los cuadrados representan a los patrones de arquitectura, los círculos representan a los patrones de diseño y los triángulos representan a los patrones de lenguaje. Se destaca que un patrón de arquitectura puede tener varios patrones de diseño dentro de ella, así como también se pueden utilizar varios patrones de lenguaje.

Existen también otros tipos de patrones de software, como patrones de análisis, patrones de integración de aplicaciones, patrones de interfaz de usuario y patrones de pruebas. Estos patrones se utilizan para diferentes propósitos en el desarrollo de software.

Se utiliza el ejemplo de la construcción de una casa para ilustrar cómo los patrones de arquitectura y diseño se aplican a la construcción de sistemas de software. Se destaca que una casa está construida por una serie de patrones y que se deben aplicar los mejores conocimientos y patrones de arquitectura civil para asegurar su estabilidad y seguridad. En resumen, los patrones de software son herramientas importantes para el desarrollo de sistemas de software efectivos y eficientes.

## PATRONES DE DISEÑO

Los patrones de diseño permiten definir estructuras de diseño o relaciones entre componentes a través de esquemas. De esta forma, se establecen microarquitecturas de subsistemas de componentes mediante clases y objetos, lo que permite describir una estructura recurrente de componentes que resuelven un problema general de diseño en un contexto determinado. Cabe destacar que un patrón de diseño se considera como un modelo o plantilla que ayuda a solucionar problemas comunes en el desarrollo de software.

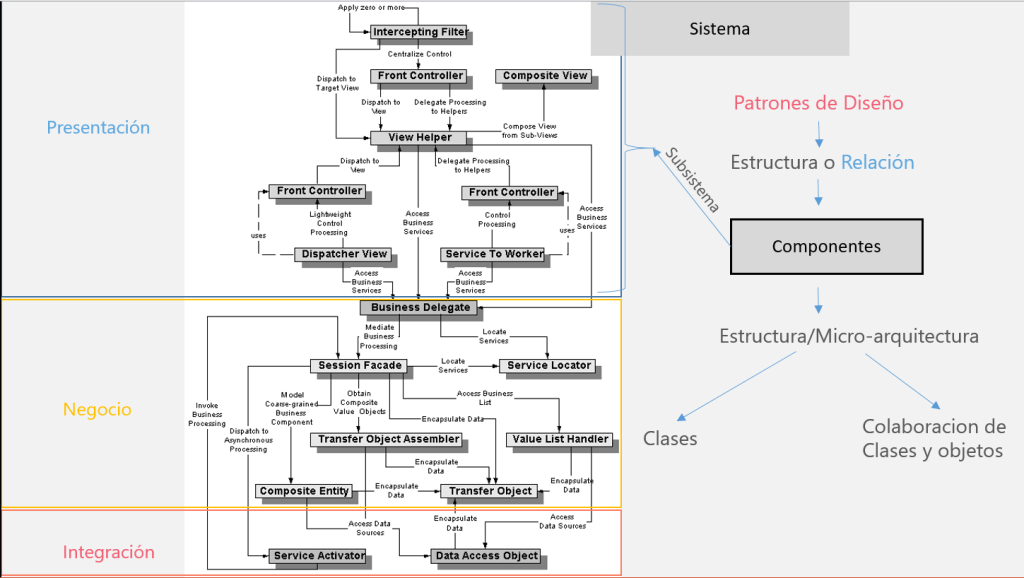


Figura 5. Diagrama de Patrones de Diseño

### PATRÓN DAO (DATA ACCESS OBJECT)

El patrón DAO es ampliamente utilizado y se enfoca en resolver problemas de diseño orientados a datos. El patrón DAO se describe como una clase que tiene definidos los métodos para insertar o eliminar datos de una base de datos, lo que lo convierte en una estructura muy común en la capa de datos. Sin embargo, muchos han utilizado este patrón sin saber que se trataba de un patrón de diseño específico.

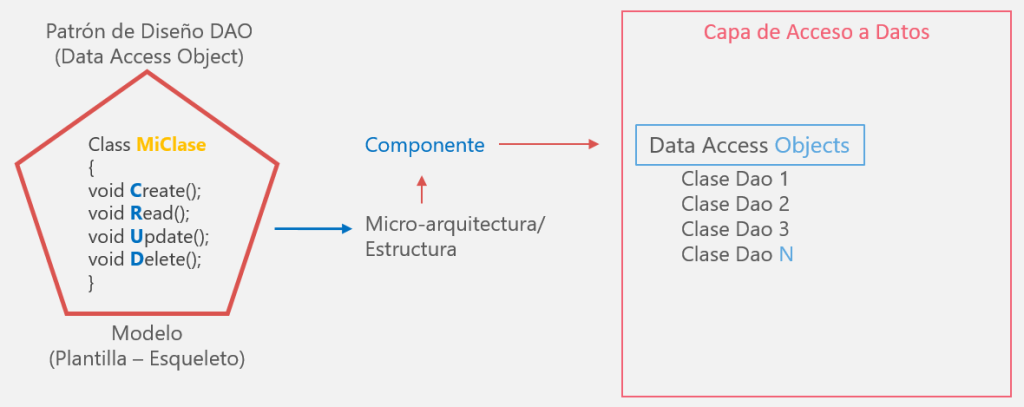


Figura 6. Patrón de Diseño DAO

Un patrón de diseño es una microarquitectura o estructura de un componente, en este caso, una clase que se encarga de acceder a los datos. Además, se menciona que un componente puede ser un objeto creado a partir de una colaboración de clases y objetos, como sucede con el patrón de diseño de método de fábrica. En este caso, la clase o interfaz base es implementada por una serie de subclases y una clase fábrica que se encarga de crear un objeto concreto, lo que forma un componente único que abarca todos los objetos de acceso a datos.

El componente DAO pertenece a la capa de datos en el patrón de arquitectura de capas, y su función es encapsular el acceso a la base de datos y crear una capa de persistencia. Se destaca que el nombre del patrón, la solución, el problema y la consecuencia son elementos fundamentales en cualquier patrón de diseño.

### TIPOS DE PATRONES DE DISEÑO

Hay una gran variedad de patrones de diseño que son ampliamente aceptados por la comunidad. Los más conocidos son los presentados en el libro de Saint Patterns, escrito por Gang of Four, que incluye 23 patrones de diseño divididos en tres categorías. Los patrones creacionales se encargan de la inicialización y configuración de objetos, mientras que los patrones estructurales separan la interfaz de la implementación y se ocupan de cómo las clases y objetos se agrupan para formar estructuras más grandes. Los patrones de comportamiento se encargan de describir objetos o clases y la comunicación entre ellos.

En el libro Jds escrito por Java se presentan otros patrones de diseño, divididos en cinco capas de arquitectura y quince patrones de diseño. Estos se dividen en tres categorías: presentación, negocios e integración. Los patrones de diseño orientados a la capa de presentación se encargan de la interfaz gráfica y la interacción con el usuario, mientras que los patrones de diseño orientados a la capa de negocios se encargan de las reglas de negocio, validaciones y transferencia de datos. Los patrones orientados a la integración se encargan de explotar los datos de cualquier fuente, ya sea archivos o bases de datos, y todos estos patrones pueden interactuar entre sí.

Los patrones de diseño definen las estructuras de diseño o sus relaciones en los componentes, y la colaboración de estos componentes forma subsistemas. Los patrones de diseño se encargan de la estructura de clases o definen microarquitecturas mediante la colaboración de clases y objetos. En resumen, los patrones de diseño son fundamentales para definir los detalles de cómo está construido un sistema de software, y se dividen en dos categorías: los que definen estructuras de componentes o sus relaciones y los que están orientados netamente a clases y objetos.



Figura 7. Clasificación de los Patrones de Diseño

### ANTIPATRONES

Es importante tener en cuenta que no es obligatorio utilizar los patrones de diseño en la programación de software, ya que sólo se aconseja su uso en caso de presentarse un problema similar. Además, es esencial recordar que, aunque los patrones de diseño puedan solucionar un problema, en algunos casos particulares puede que no sean aplicables. Forzar el uso o abusar de los patrones puede llevar a la creación de antipatrones, lo que representa un grave error. Por esto que se debe tener mucho cuidado al utilizar los patrones de diseño en el desarrollo de software.

### PATRÓN SINGLETON

El patrón Singleton restringe la instanciación de una clase o valor de un tipo a un solo objeto, este patrón se puede usar para solucionar el problema de instancias múltiples de un formulario.

### PATRÓN DTO (DATA TRANSFER OBJECT)

El patrón de diseño DTO se trata de un objeto se realizable para la transferencia de datos, el patrón de diseño DTO consiste en utilizar un objeto de acceso a datos para abstraer y encapsular todos los accesos a la fuente de datos. DAO maneja la conexión con la fuente de datos para obtener y almacenar datos.

### RELACIÓN ENTRE PATRONES

El patrón DAO siempre va de la mano con el patrón de DTO, ya que necesita objetos de valor para extraer datos e insertar a la base de datos, o viceversa. Es decir, extrae datos de una fuente y necesita un objeto se realizable para almacenar dichos datos y así poder transportar los datos a su destino. El patrón DAO indica que debe ver una clase con métodos CRUD por cada tabla de la base de datos.

# PARTE PRÁCTICA

Realizar una pequeña aplicación para mostrar búsqueda y filtro de datos. Se hará uso del lenguaje C#, SQL Server y programación orientada a objetos.

## CREACIÓN DE LA SOLUCIÓN

Para la parte práctica, se necesita crear una base de datos que guarde varios registros en una tabla Clientes. Para crear la base de datos abrir SQL Management Studio y crear una nueva base de datos con el nombre “Practica\_Patrones”.

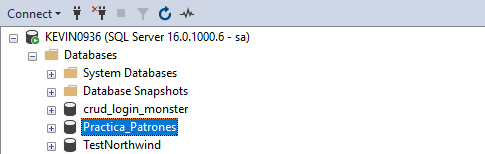


Figura 8. Creación de la base de datos

Para cargar los registros se debe crear un nuevo query y pegar el código SQL que se encuentra en el archivo “10 MIL Registros.sql”.

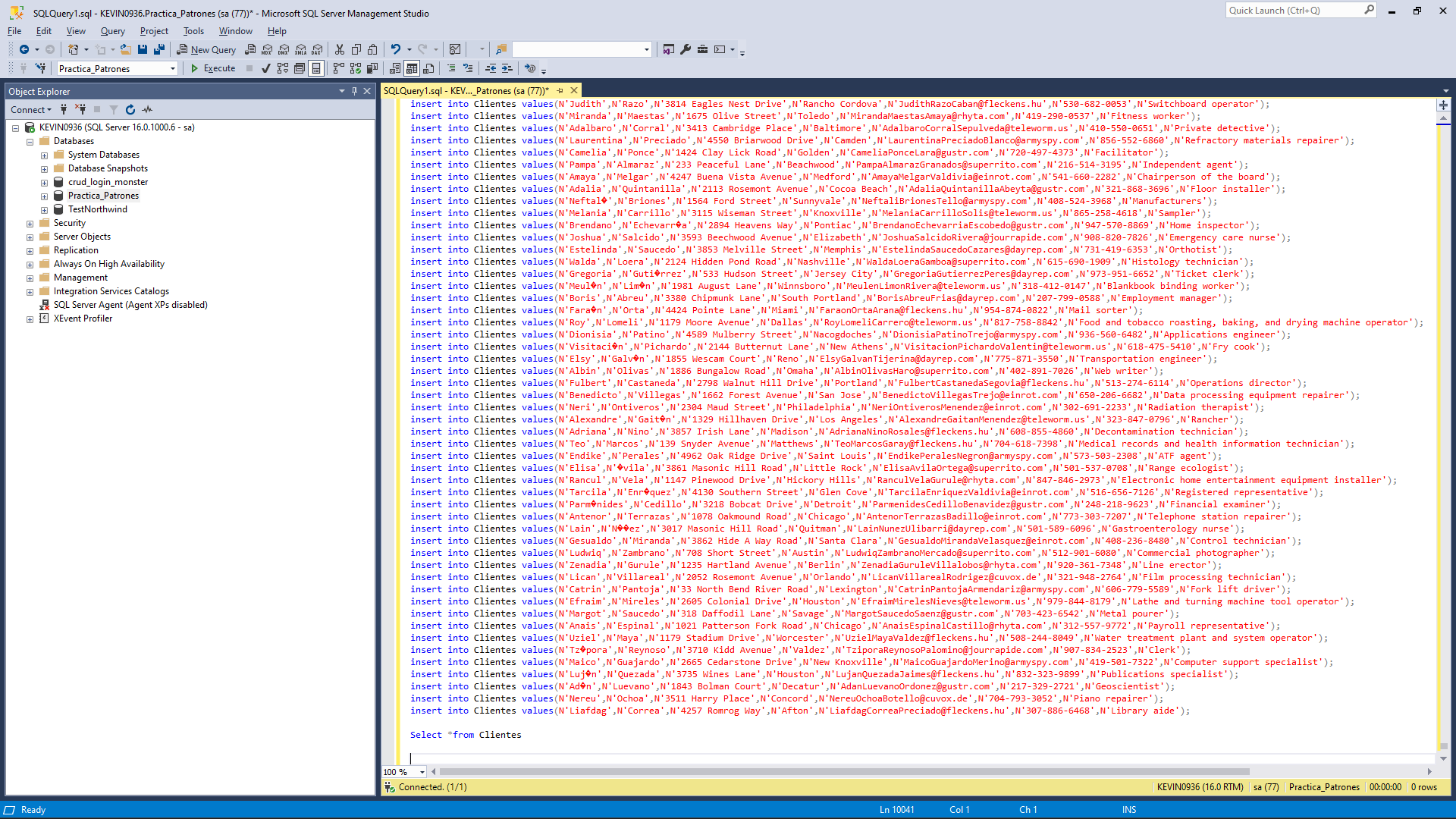


Figura 9. Ingreso de registros a SQL Management Studio

Para ejecutar el query y guardar los registros se debe dar clic en el botón Execute.

Graphical user interface, application, Word

Description automatically generated

Figura 10. Ejecución del query para ingresar Clientes

Para verificar que los registros se ingresaron con éxito, ejecutar el siguiente query.

Graphical user interface, text, application, chat or text message

Description automatically generated

Figura 11. Consulta a la tabla Clientes

Si se registraron con éxito los datos, la consulta anterior mostrará los siguientes resultados.

Graphical user interface, application, table

Description automatically generated

Figura 12. Resultado de la consulta de Clientes

Luego de ingresar los registros de clientes, se debe ejecutar el siguiente procedimiento SQL marcado en azul.

Graphical user interface, application, Word

Description automatically generated

Figura 13. Procedimiento de SQL para buscar Clientes

Una vez creada y configurada la base de datos, abrir Visual Studio y crear un nuevo proyecto.

Graphical user interface, text

Description automatically generated

Figura 14. Ventana de proyectos en Visual Studio

Seleccionar la plantilla de proyecto “Windows From App” y clic en Next.

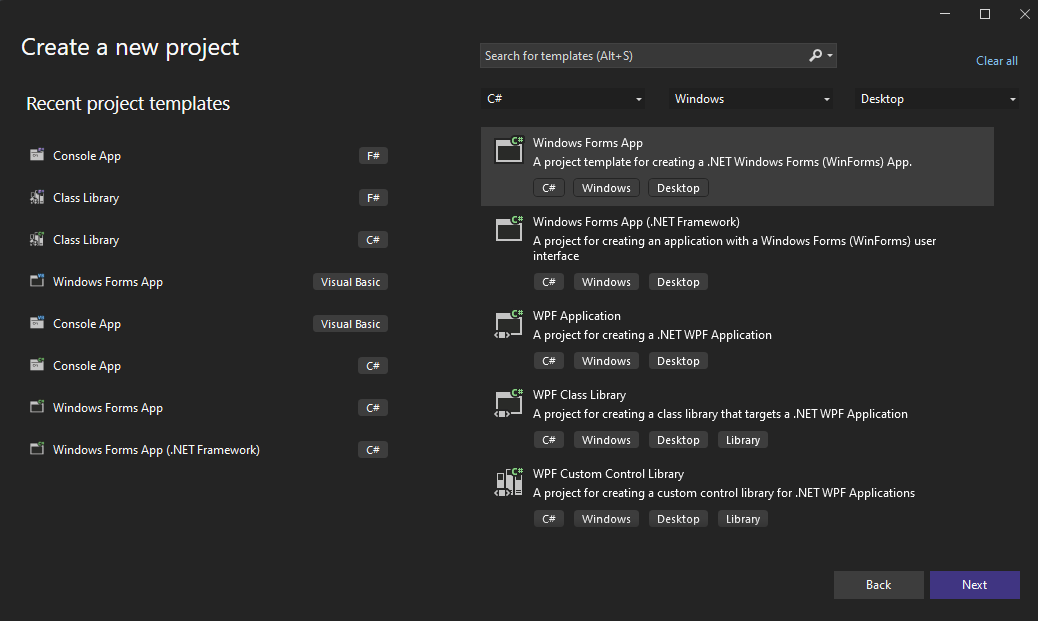


Figura 15. Plantilla para el proyecto

En la siguiente ventana, ingresar el nombre del proyecto y la ruta en donde se guardará el proyecto, por último, clic en Next.

Graphical user interface, text

Description automatically generated

Figura 16. Nombre y ruta de alojamiento del proyecto

Se debe seleccionar la versión de .NET para el proyecto y clic en Create.

Graphical user interface, application

Description automatically generated

Figura 17. Version de .NET para el proyecto

En esta parte de la práctica se hará uso de los patrones de diseño DAO y DTO para gestionar la base de datos, para esto se deben crear las carpetas como se muestra en la Figura 18.

Graphical user interface, text

Description automatically generated

Figura 18. Estructura para el proyecto

Una vez creada la estructura del proyecto, dar clic en la carpeta DTO, clic derecho y seleccionar “New Item”.

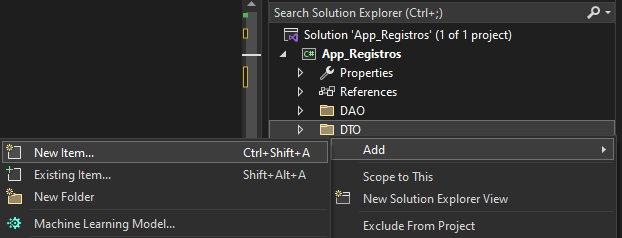


Figura 19. Creación de nueva clase en DTO

Seleccionar el tipo de archivo C# Class con el nombre ClienteDto y clic en Add.

A screenshot of a computer

Description automatically generated with medium confidence

Figura 20. Creación de la clase ClienteDto

Una vez creado el archivo dirigirse al panel de solución y abrir el archivo ClienteDto.

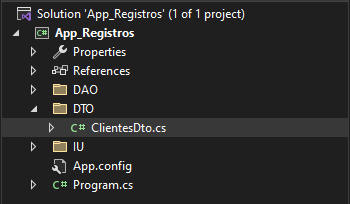


Figura 21. Clase ClienteDto en la carpeta DTO

En esta clase se define la clase que contendrá los campos de la tabla Clientes creada en SQL Server.

Text

Description automatically generated

Figura 22. Codificación e la clase ClienteDto

Tabla 1. Codificación del archivo RegistroDTO.cs

namespace App\_Registros.DTO

{

class ClientesDto

{

int \_ID;

string \_Nombre;

string \_Apellido;

string \_Direccion;

string \_Ciudad;

string \_Email;

string \_Telefono;

string \_Ocupacion;

public int ID

{

get

{

return \_ID;

}

set

{

\_ID = value;

}

}

public string Nombre

{

get

{

return \_Nombre;

}

set

{

\_Nombre = value;

}

}

public string Apellido

{

get

{

return \_Apellido;

}

set

{

\_Apellido = value;

}

}

public string Direccion

{

get

{

return \_Direccion;

}

set

{\_Direccion = value;}

}

public string Ciudad{

get{return \_Ciudad;}

set{\_Ciudad = value;}

}

public string Email

{

get{return \_Email;}

set{\_Email = value;}

}

public string Telefono{

get{return \_Telefono;}

set{\_Telefono = value;}

}

public string Ocupacion

{get{return \_Ocupacion;}

set

{

\_Ocupacion = value;

}

}

}

}

En la carpeta DAO dar clic derecho y “New Item” para crear las clases ConexionDB y ClienteDao.

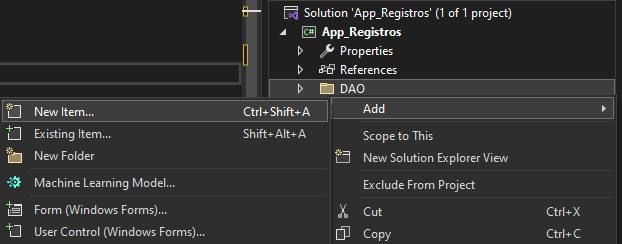


Figura 23. Creación de una nueva clase en DAO

La clase ConexionDB será la encargada de implementar la conexión a la base de datos en SQL Server.

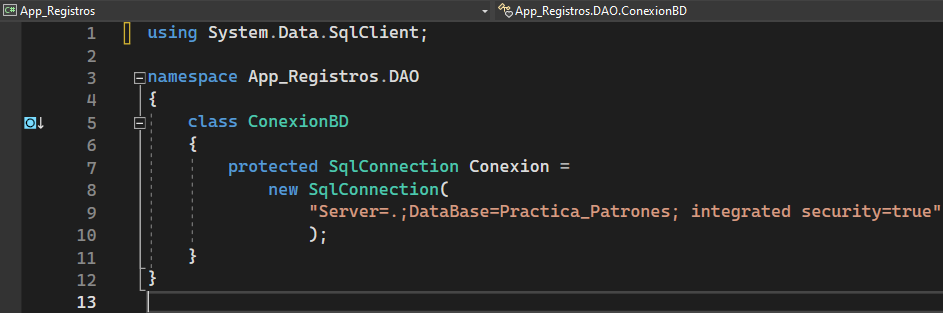


Figura 24. Codificación de la clase ConexionDB para SQL Server

Tabla 2. Codificación del archivo ConexionDB.cs

using System.Data.SqlClient;

namespace App\_Registros.DAO

{

class ConexionBD

{

protected SqlConnection Conexion =

new SqlConnection(

"Server=.;DataBase=Practica\_Patrones; integrated security=true"

);

}

}

La clase ClienteDao se encarga de definir los métodos para acceder a la base de datos y gestionarla.

Text

Description automatically generated

Figura 25. Codificación de la Clase ClienteDao

Tabla 3. Codificación del archivo ClienteDao.cs

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using App\_Registros.DTO;

namespace App\_Registros.DAO

{

class ClienteDao : ConexionBD

{

SqlDataReader LeerFilas;

SqlCommand Comando = new SqlCommand();

//METODOS CRUD

public List<ClientesDto> VerRegistros(string Condicion)

{

Comando.Connection = Conexion;

Comando.CommandText = "VerRegistros";

Comando.CommandType = CommandType.StoredProcedure;

Comando.Parameters.AddWithValue("@Condicion", Condicion);

Conexion.Open();

LeerFilas = Comando.ExecuteReader();

List<ClientesDto> ListaGenerica = new List<ClientesDto>();

while (LeerFilas.Read())

{

ListaGenerica.Add(new ClientesDto

{

ID = LeerFilas.GetInt32 (0),

Nombre = LeerFilas.GetString(1),

Apellido = LeerFilas.GetString(2),

Direccion = LeerFilas.GetString(3),

Ciudad = LeerFilas.GetString(4),

Email = LeerFilas.GetString(5),

Telefono = LeerFilas.GetString(6),

Ocupacion = LeerFilas.GetString(7),

});

}

LeerFilas.Close();

Conexion.Close();

return ListaGenerica;

}

public void Insert() { }

public void Edit() { }

public void Delete() { }

}

}

En la carpeta UI dar clic derecho y clic en “New Item”.

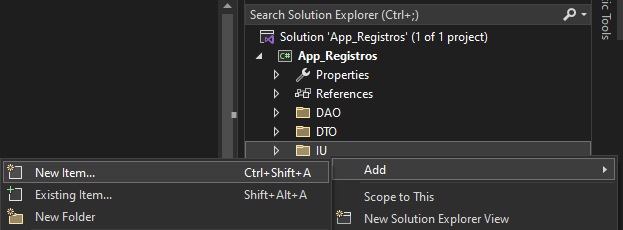


Figura 26. Creación de las ventanas en UI

En la siguiente ventana, seleccionar el tipo de archivo “Form (Windows Forms)” con el nombre FormPrincipal y clic en Add.

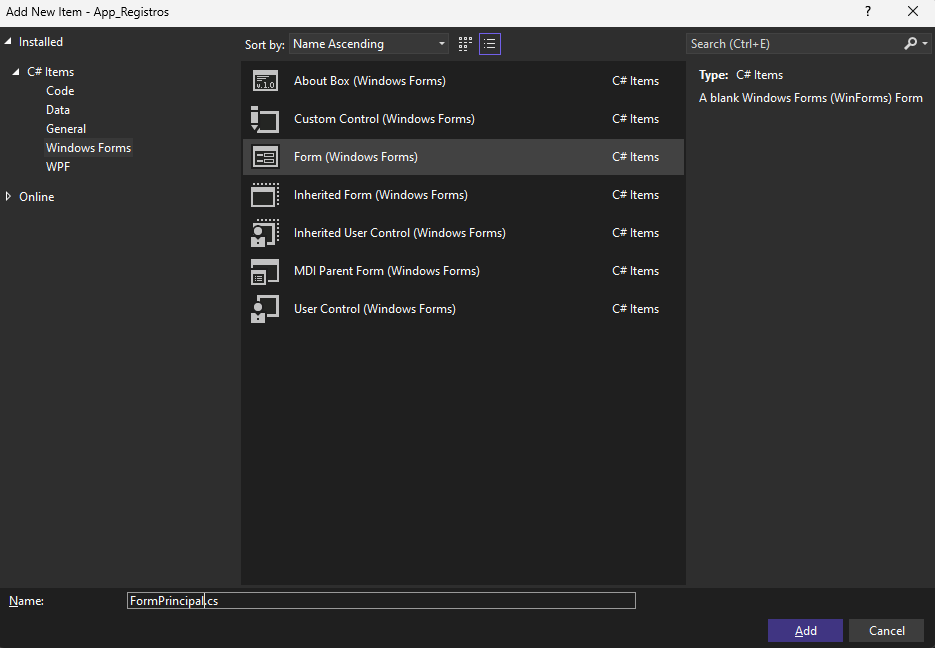


Figura 27. Creación de la ventana FormPrincipal

Esta ventana contiene el menú de navegación para la búsqueda de Clientes y Libros.

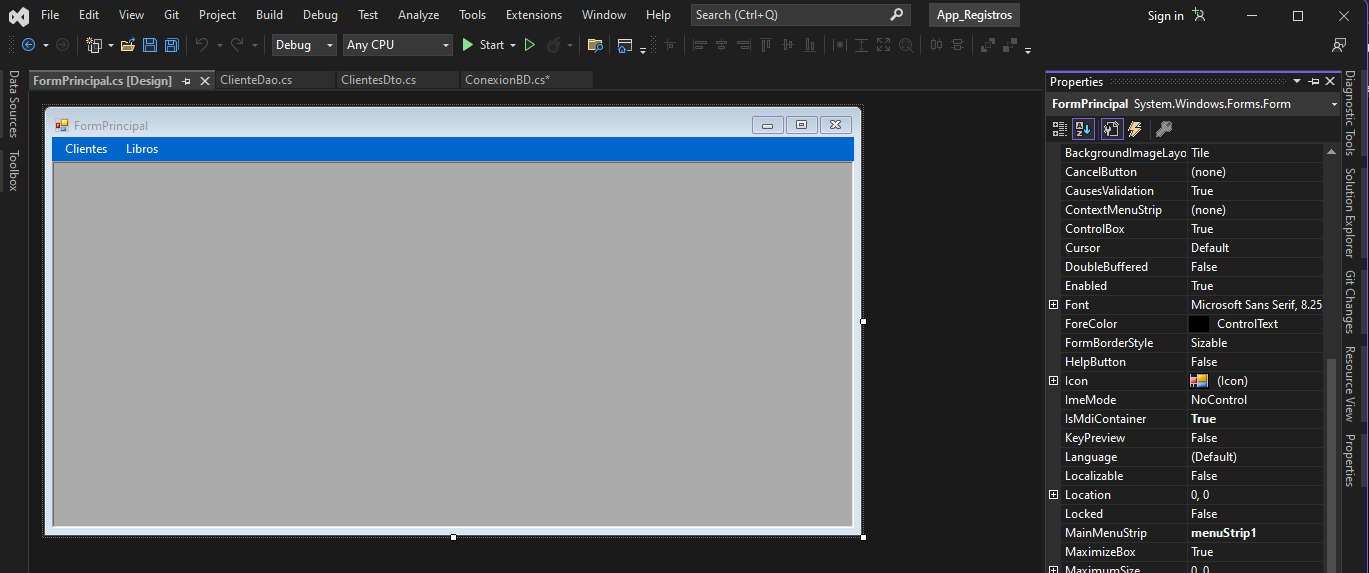


Figura 28. Diseño de la ventana FormPrincipal

En la clase FormPrincipal asociada a la ventana se definen los eventos clic para acceder a las ventanas de Clientes y Libros.

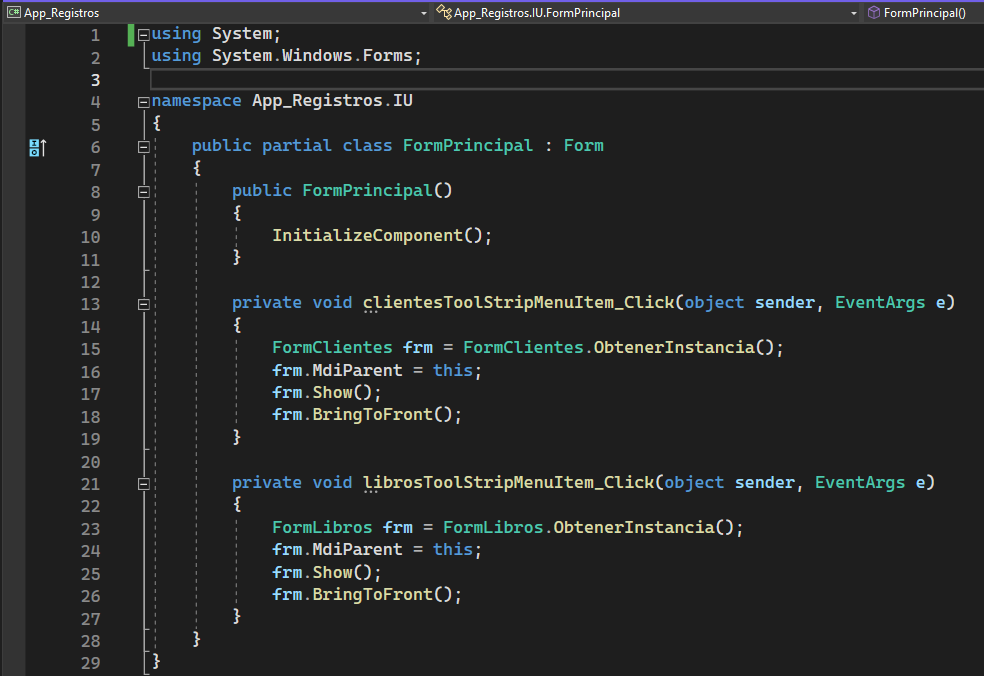


Figura 29. Codificación de la clase FormPrincipal

Tabla 4. Codificación del archivo FormPrincipal.cs

using System;

using System.Windows.Forms;

namespace App\_Registros.IU

{

public partial class FormPrincipal : Form

{

public FormPrincipal()

{

InitializeComponent();

}

private void clientesToolStripMenuItem\_Click(object sender, EventArgs e)

{

FormClientes frm = FormClientes.ObtenerInstancia();

frm.MdiParent = this;

frm.Show();

frm.BringToFront();

}

private void librosToolStripMenuItem\_Click(object sender, EventArgs e)

{

FormLibros frm = FormLibros.ObtenerInstancia();

frm.MdiParent = this;

frm.Show();

frm.BringToFront();

}}

}

Para la ventana de Clientes, se crea un nuevo Windows Form con el nombre FormClientes.

A screenshot of a computer

Description automatically generated with medium confidence

Figura 30. Creación de la ventana FormClientes

Se agrega una entrada de texto para la búsqueda, un botón y un panel para mostrar el resultado.

A picture containing text, screenshot, monitor, computer

Description automatically generated

Figura 31. Diseño de la ventana FromClientes

En la clase asociada a la vista FormClientes se definen los métodos y eventos para mostrar los resultados de la búsqueda.

Text

Description automatically generated

Figura 32. Codificación de la clase FormClientes

Tabla 5. Codificación del archivo FormClientes.cs

using System;

using System.Windows.Forms;

using App\_Registros.DAO;

namespace App\_Registros.IU

{

public partial class FormClientes : Form

{

//PATRON SINGLETON

private FormClientes()

{

InitializeComponent();

}

private static FormClientes Instancia = null;

public static FormClientes ObtenerInstancia()

{

if (Instancia == null)

{

Instancia = new FormClientes();

Instancia.FormClosed += new FormClosedEventHandler(reset);//SOLO PARA FORMULARIOS

}

return Instancia;

}

//FIN SINGLETON

private static void reset(object sender, FormClosedEventArgs e)//SOLO PARA FORMULARIOS

{

Instancia = null;

}

private void FormClientes\_Load(object sender, EventArgs e)

{

//MOSTRAR TODO

VerRegistros("");

}

//METOOD VER REGISTROS

private void VerRegistros(string condicion)

{

ClienteDao DAO = new ClienteDao();

dataGridView1.DataSource = DAO.VerRegistros(condicion);

}

//BUSCAR

private void btnBuscar\_Click(object sender, EventArgs e)

{

VerRegistros(txtBuscar.Text);

}

//FILTRAR

private void txtBuscar\_TextChanged(object sender, EventArgs e)

{

VerRegistros(txtBuscar.Text);

}

}

}

Por último, se crea otra ventana para los libros con el nombre FormLibros.

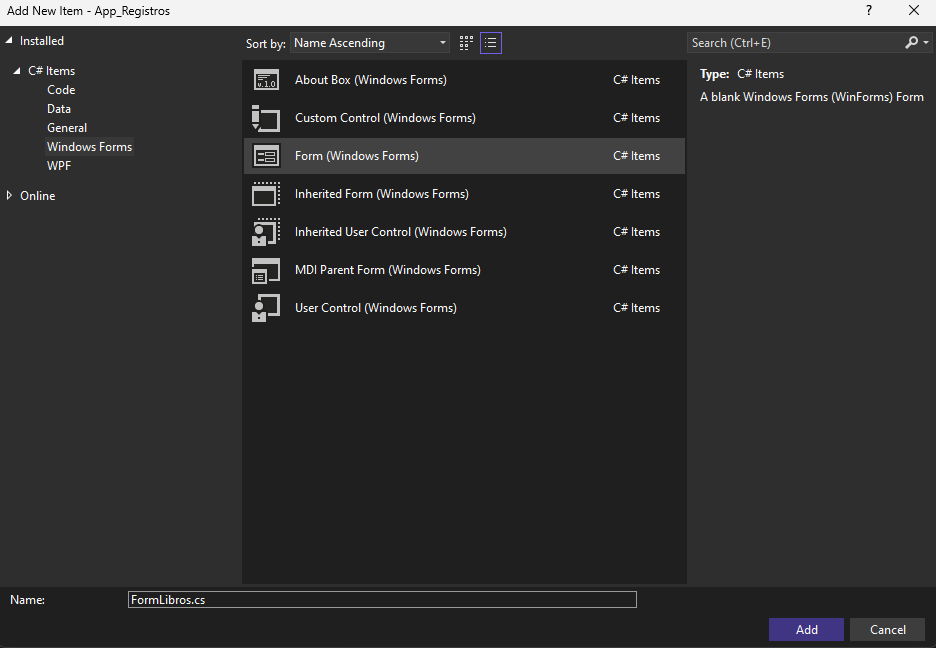


Figura 33. Creación de la ventana FormLibros

Para esta ventana, se definirá su diseño en otra práctica, por lo tanto, no contendrá ningún elemento en su ventana.

A screenshot of a computer

Description automatically generated with medium confidence

Figura 34. Diseño de la ventana FormLibros

En la clase asociada a la ventana ClienteLibros se definen los métodos para recuperar y mostrar los datos en pantalla.

Text

Description automatically generated

Figura 35. Codificación de la clase FormLibros

Tabla 6. Codificación del archivo FormLibros.cs

using System;

using System.Windows.Forms;

namespace App\_Registros.IU

{

public partial class FormLibros : Form

{

private FormLibros()

{

InitializeComponent();

}

private static FormLibros instancia = null;

public static FormLibros ObtenerInstancia() {

if (instancia == null)

{

instancia = new FormLibros();

instancia.FormClosed += new FormClosedEventHandler(reset);//SOLO PARA FORMULARIOS

}

return instancia;

}

private static void reset(object sender, FormClosedEventArgs e)//SOLO PARA FORMULARIOS

{

instancia = null;

}

private void FormLibros\_Load(object sender, EventArgs e)

{

}

}

}

La estructura final del proyecto para esta practica es la siguiente, DAO será el encargado de acceder a la base de datos y enviar o recuperar información, DTO será el encargado de convertir la información de la base de datos a objetos de C#, UI recibe las peticiones del usuario y muestra los resultados por pantalla.

Graphical user interface, text

Description automatically generated

Figura 36. Estructura final del proyecto

# EJECUCIÓN DEL PROYECTO

Para ejecutar el proyecto, dirigirse al panel de herramientas de Visual Studio y dar clic en el botón Start.

Graphical user interface, text, application, chat or text message

Description automatically generated

Figura 37. Ejecución del proyecto

Al ejecutar el proyecto, se muestra la ventana principal con el panel de navegación a la ventana de Clientes y Libros.

Shape, rectangle

Description automatically generated

Figura 38. Ventana principal con el panel de navegación

Al navegar a la ventana de Clientes se muestra una tabla con todos los registros de la tabla Clientes. El campo de entrada de texto permite buscar a un cliente por su ID o Nombre, por lo tanto, al ingresar texto, se muestran los registros en la tabla filtrándose por el texto ingresado.

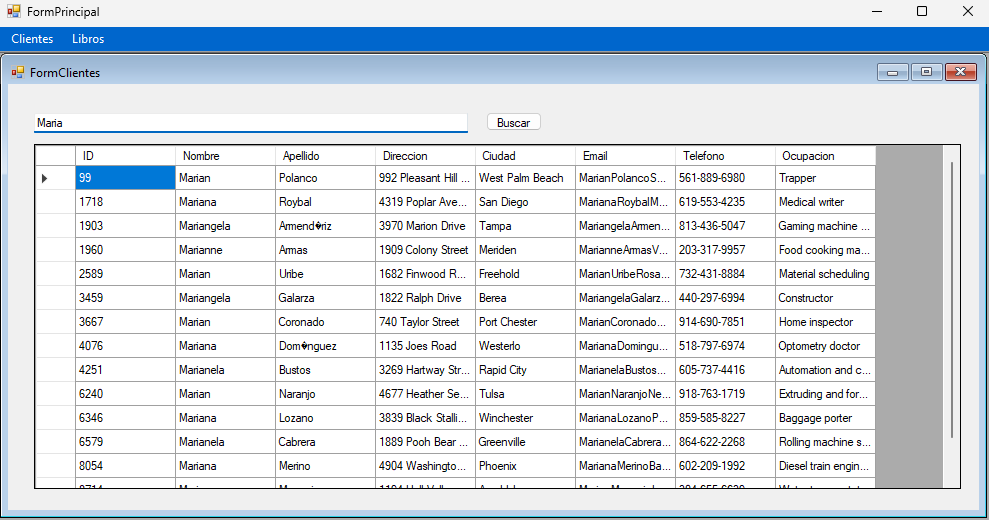


Figura 39. Ventana para la búsqueda de Clientes

Al navegar a la ventana de Libros, no se muestra nada aun, pues su implementación se realizaría en una próxima práctica.

Graphical user interface, application

Description automatically generated

Figura 40.Ventana para la búsqueda de Libros

# CONCLUSIONES

* Los patrones son elementos recurrentes que se utilizan como modelos para organizar y obtener un código de mejor calidad, además, permiten generar objetos o partes de ellos para que puedan ser utilizados en diferentes contextos.
* Los patrones de software son modelos o plantillas que facilitan la reutilización de diseños y arquitecturas para resolver problemas comunes o recurrentes en la ingeniería de software.
* Los patrones de software describen estructuras y mecanismos específicos que se han demostrado que funcionan bien en situaciones particulares.

# RECOMENDACIONES

* Utilizar patrones de software en el proceso de construcción de aplicaciones permite acelerar el proceso y mejorar la calidad del resultado final.
* Se recomienda buscar patrones de software que hayan sido probados y tengan al menos tres implementaciones reales para garantizar que sean efectivos.
* Clasificar los patrones de software en categorías como patrones de arquitectura, patrones de diseño y patrones de lenguaje para una mejor comprensión de su aplicación y uso.

# BIB**LIOGRAFÍA**

RJ Code Advance. (15 de Noviembre de 2018). *C2/ Patrones de Diseño + Ejemplo DAO, DTO, Singleton (Buscar/Filtrar Datos) C#, SQL*. Obtenido de YouTube: https://www.youtube.com/watch?v=9NjtWyjjLKQ&t=265s