Documentación del Sistema de Gestión de Bibliotecas

Universidad: corporación universitaria iberoamericana

Nombre: Kevin Eduardo Cabarcas Quesedo

Profesor: José Ruiz

ACTIVIDAD 2 Proyecto: sistema de gestión de biblioteca

Fecha: 2/04/2025

Introducción

El sistema de gestión de bibliotecas desarrollado en ASP.NET permite la administración eficiente de los recursos bibliográficos y la gestión de préstamos y devoluciones de libros. Este sistema facilita el control de libros, categorías, editoriales, autores y usuarios, optimizando el flujo de información dentro de la biblioteca.

Objetivos

Objetivo General

Diseñar e implementar un sistema de gestión de bibliotecas utilizando ASP.NET y bases de datos SQL Server, que permita la administración eficiente del inventario de libros y el control de préstamos y devoluciones.

Objetivos Específicos

Diseñar una base de datos estructurada para almacenar información sobre libros, usuarios y préstamos.

Implementar una interfaz web intuitiva con ASP.NET para la gestión de la biblioteca.

Desarrollar un sistema de autenticación y control de acceso basado en roles.

Automatizar la gestión de préstamos, devoluciones y control de ejemplares disponibles.

Implementar reportes y consultas para mejorar la administración del sistema.

Alcance

Este sistema está diseñado para bibliotecas de tamaño mediano a grande, proporcionando herramientas para la administración eficiente del inventario de libros y usuarios. Su desarrollo en ASP.NET permite su integración con otras plataformas y su escalabilidad.

Requisitos Funcionales

4.1 Gestión de Categorías

Crear, actualizar y eliminar categorías de libros.

Listar todas las categorías registradas.

Filtrar categorías por descripción.

4.2 Gestión de Editoriales

Registrar nuevas editoriales en el sistema.

Modificar la información de editoriales existentes.

Eliminar editoriales si no tienen libros asociados.

4.3 Gestión de Autores

Registrar autores con su información básica.

Editar los datos de un autor.

Listar todos los autores disponibles en el sistema.

4.4 Gestión de Libros

Registrar libros con su título, autor, editorial y categoría.

Gestionar la cantidad de ejemplares disponibles.

Actualizar y eliminar información de los libros.

Subir y almacenar imágenes de las portadas de los libros.

Búsqueda avanzada por título, autor, editorial y categoría.

Visualización de detalles de cada libro.

4.5 Gestión de Usuarios

Registro de nuevos usuarios con correo y contraseña.

Asignación de roles (administrador o usuario regular).

Autenticación de usuarios mediante login en ASP.NET.

Cambio y recuperación de contraseñas.

4.6 Gestión de Préstamos

Registrar préstamos de libros a usuarios.

Asignar fechas de devolución y emitir alertas por vencimiento.

Registrar devoluciones de libros y actualizar su estado.

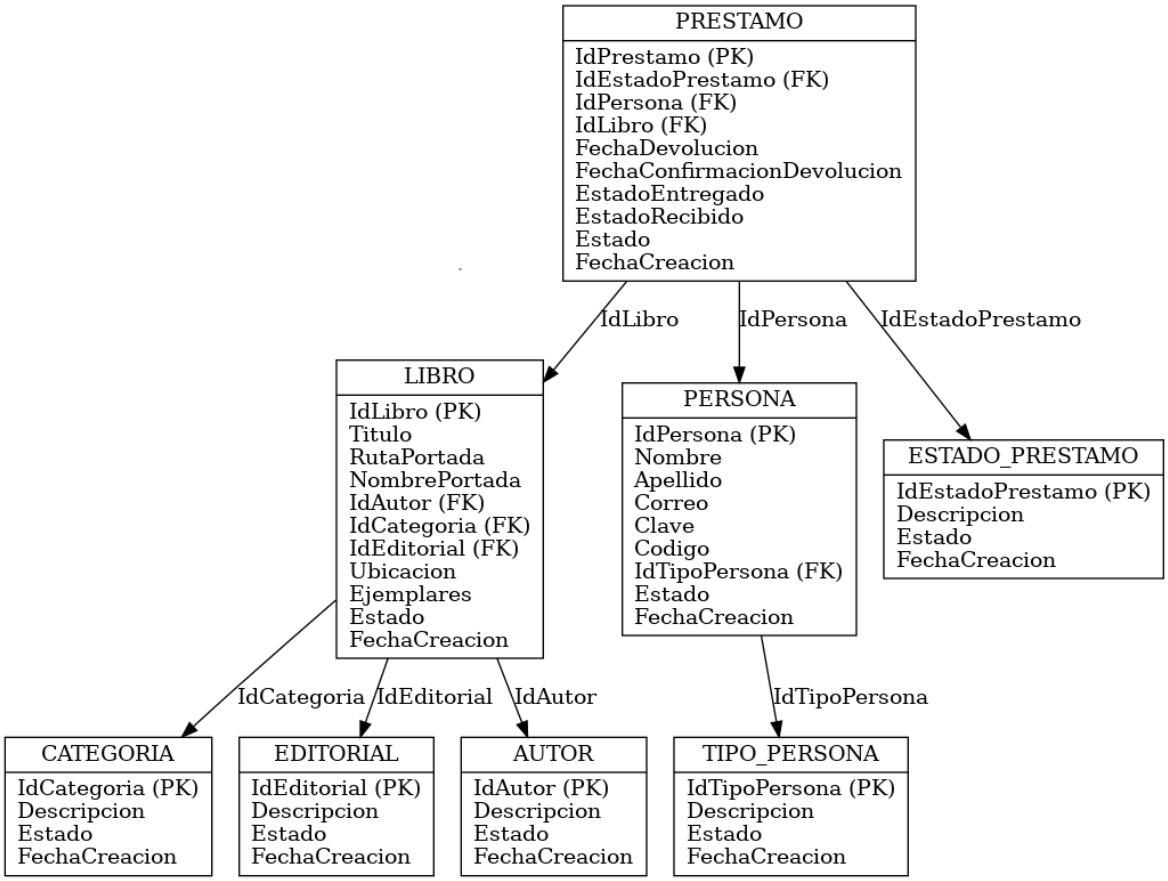
Controlar el estado de los préstamos (activo, vencido, devuelto).

4.7 Seguridad y Control de Acceso

Implementación de autenticación con ASP.NET Identity.

Restricción de acceso a funcionalidades según el rol del usuario.

Registro de actividad para auditoría.



Tecnologías Utilizadas

Lenguaje de Programación: C#

Framework: ASP.NET Core MVC

Base de Datos: SQL Server

ORM: Entity Framework Core

Frontend: HTML, CSS, Bootstrap, JavaScript

6. Modelo de Datos

El sistema cuenta con las siguientes tablas en la base de datos:

CATEGORIA: Gestiona las categorías de los libros.

EDITORIAL: Almacena información sobre editoriales.

AUTOR: Contiene información sobre los autores.

LIBRO: Contiene los datos de cada libro y su disponibilidad.

TIPO_PERSONA: Define los tipos de usuarios en el sistema.

PERSONA: Almacena información de los usuarios.

ESTADO_PRESTAMO: Define los estados de los préstamos.

PRESTAMO: Registra los préstamos y devoluciones de libros.

7. Seguridad

Implementación de ASP.NET Identity para autenticación de usuarios.

Cifrado de contraseñas y tokens de acceso.

Validación de entradas para prevenir inyecciones SQL y XSS.

8. Interfaces del Usuario

Panel de Administrador: Gestiona libros, usuarios y préstamos.

Panel de Usuario: Permite visualizar y solicitar préstamos.

Módulo de Reportes: Generación de informes sobre préstamos y disponibilidad de libros.

Proceso de desarrollo

Creación de la base de dato

1. Tabla CATEGORIA
Esta tabla almacena información sobre las categorías de los libros (por ejemplo: Ciencia, Literatura, Historia).

Columnas:

IdCategoria: Identificador único de la categoría, es una clave primaria que se genera automáticamente.

Descripcion: Descripción o nombre de la categoría (por ejemplo: "Ciencia", "Ficción").

Estado: Indica si la categoría está activa (1) o no (0).

FechaCreacion: Fecha en que se creó la categoría.

2. Tabla EDITORIAL
Aquí se almacenan los datos de las editoriales (por ejemplo: Penguin, Random House).

Columnas:

IdEditorial: Identificador único de la editorial (clave primaria autoincremental).

Descripcion: Nombre de la editorial.

Estado: Indica si la editorial está activa o no.

FechaCreacion: Fecha en que se registró la editorial.

3. Tabla AUTOR
Contiene información sobre los autores de los libros.

Columnas:

IdAutor: Identificador único del autor (clave primaria autoincremental).

Descripcion: Nombre del autor (puede ser un alias o nombre completo).

Estado: Estado de la autoría (activo o inactivo).

FechaCreacion: Fecha en que se registró al autor.

4. Tabla LIBRO
Aquí se almacenan los libros disponibles en la biblioteca. Cada libro tiene referencias a un autor, una categoría y una editorial.

Columnas:

IdLibro: Identificador único del libro (clave primaria autoincremental).

Titulo: Título del libro.

RutaPortada: Ruta donde se almacena la portada del libro (por ejemplo: imagenes/libro1.jpg).

NombrePortada: Nombre del archivo de la portada.

IdAutor: Relación con la tabla AUTOR (indica el autor del libro).

IdCategoria: Relación con la tabla CATEGORIA (indica la categoría del libro).

IdEditorial: Relación con la tabla EDITORIAL (indica la editorial del libro).

Ubicacion: Información sobre dónde se encuentra el libro en la biblioteca (por ejemplo: "Estante A3").

Ejemplares: Cantidad de ejemplares disponibles de ese libro.

Estado: Estado del libro (activo o no).

FechaCreacion: Fecha en que se registró el libro.

5. Tabla TIPO_PERSONA
Esta tabla almacena diferentes tipos de personas que pueden estar involucradas en el sistema (por ejemplo: "Estudiante", "Profesor", "Empleado").

Columnas:

IdTipoPersona: Identificador único del tipo de persona.

Descripcion: Descripción del tipo de persona (por ejemplo: "Estudiante", "Profesor").

Estado: Indica si el tipo de persona está activo o no.

FechaCreacion: Fecha en que se creó el tipo de persona.

6. Tabla PERSONA
Esta tabla almacena la información de las personas (usuarios del sistema), que pueden ser quienes toman los libros prestados.

Columnas:

IdPersona: Identificador único de la persona (clave primaria autoincremental).

Nombre: Nombre de la persona.

Apellido: Apellido de la persona.

Correo: Correo electrónico de la persona.

Clave: Contraseña o clave para acceder al sistema.

Codigo: Un código único que podría ser el número de matrícula o identificación de la persona.

IdTipoPersona: Relación con la tabla TIPO_PERSONA (indica el tipo de persona).

Estado: Estado de la persona (activo o no).

FechaCreacion: Fecha en que se registró a la persona.

7. Tabla ESTADO_PRESTAMO

Esta tabla define los estados en los que puede encontrarse un préstamo (por ejemplo: "En espera", "Confirmado", "Devuelto").

Columnas:

IdEstadoPrestamo: Identificador único del estado de préstamo (clave primaria).

Descripcion: Descripción del estado del préstamo (por ejemplo: "Pendiente", "Devuelto").

Estado: Indica si el estado está activo o no.

FechaCreacion: Fecha en que se registró el estado del préstamo.

8. Tabla PRESTAMO
Esta tabla registra los préstamos de libros a las personas. Relaciona a la persona que realiza el préstamo, el libro prestado y el estado del préstamo.

Columnas:

IdPrestamo: Identificador único del préstamo (clave primaria autoincremental).

IdEstadoPrestamo: Relación con la tabla ESTADO_PRESTAMO (indica el estado del préstamo).

IdPersona: Relación con la tabla PERSONA (indica la persona que realiza el préstamo).

IdLibro: Relación con la tabla LIBRO (indica el libro prestado).

FechaDevolucion: Fecha en la que se debe devolver el libro.

FechaConfirmacionDevolucion: Fecha en la que se confirma la devolución del libro.

EstadoEntregado: Información sobre si el libro fue entregado en buen estado.

EstadoRecibido: Información sobre si el libro fue recibido correctamente.

Estado: Estado del préstamo (activo o no).

FechaCreacion: Fecha en que se registró el préstamo.

Relaciones entre las tablas:
Libros están relacionados con Autores, Categorías y Editoriales.

Personas (usuarios) tienen un tipo asignado a través de la tabla TIPO_PERSONA.

Préstamos están relacionados con Personas (quién toma el libro) y Libros (qué libro se presta), además de tener un estado definido en la tabla ESTADO_PRESTAMO.

Conclusión
Este sistema de base de datos está diseñado para gestionar una biblioteca, permitiendo:

Registrar libros con detalles como autor, categoría y editorial.

Gestionar los usuarios (personas) y su tipo (por ejemplo, estudiantes, profesores).

Controlar los préstamos de libros, asegurándose de que se registre el estado de los préstamos y devoluciones.

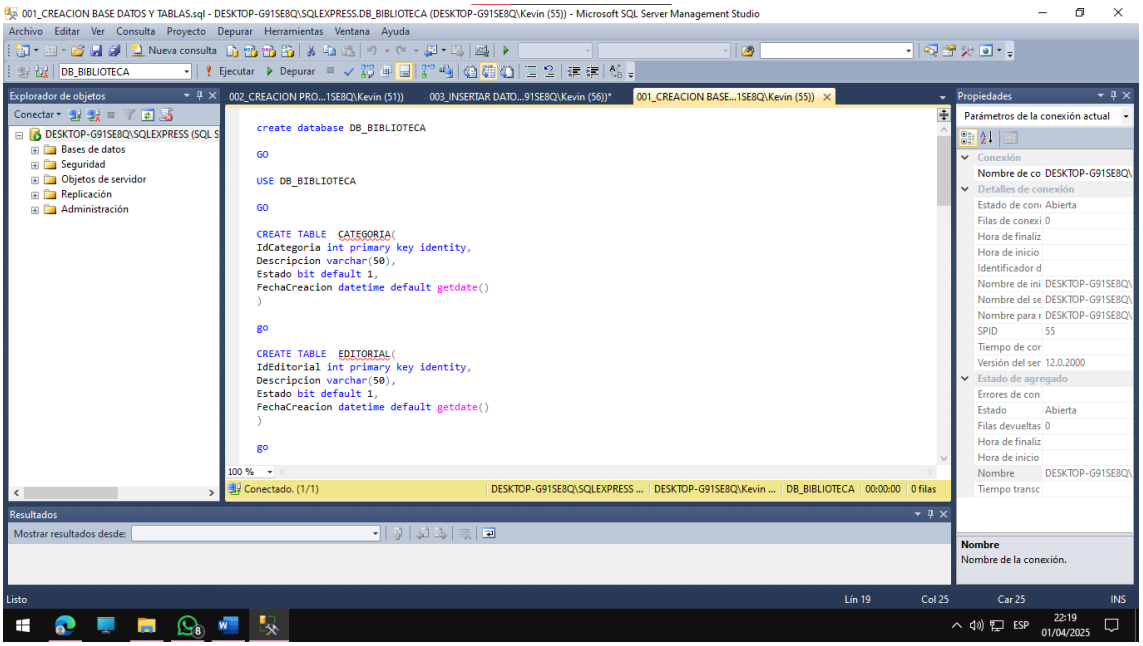
La estructura está organizada para soportar operaciones comunes en un sistema de biblioteca, como el registro de libros, la asignación de préstamos y la consulta de información sobre libros y personas.

Si tienes más preguntas sobre cómo implementar o mejorar este sistema, no dudes en preguntar.

```
create database DB_BIBLIOTECA

GO

USE DB_BIBLIOTECA

GO

CREATE TABLE  CATEGORIA(
IdCategoria int primary key identity,
Descripcion varchar(50),
Estado bit default 1,
FechaCreacion datetime default getdate()
)

go

CREATE TABLE  EDITORIAL(
IdEditorial int primary key identity,
Descripcion varchar(50),
Estado bit default 1,
FechaCreacion datetime default getdate()
)

go


CREATE TABLE  AUTOR(
IdAutor int primary key identity,
Descripcion varchar(50),
Estado bit default 1,
FechaCreacion datetime default getdate()
)

go

CREATE TABLE LIBRO(
IdLibro int primary key identity,
Titulo varchar(100),
RutaPortada varchar(100),
NombrePortada varchar(100),
IdAutor int references AUTOR(IdAutor),
IdCategoria int references CATEGORIA(IdCategoria),
IdEditorial int references EDITORIAL(IdEditorial),
Ubicacion varchar(50),
Ejemplares int,
Estado bit default 1,
FechaCreacion datetime default getdate()
)

GO

CREATE TABLE TIPO_PERSONA(
IdTipoPersona  int primary key,
Descripcion varchar(50),
```

```
Estado bit default 1,
FechaCreacion datetime default getdate()
)

GO

CREATE TABLE PERSONA(
IdPersona int primary key identity,
Nombre varchar(50),
Apellido varchar(50),
Correo varchar(50),
Clave varchar(50),
Codigo varchar(50),
IdTipoPersona int references TIPO_PERSONA(IdTipoPersona),
Estado bit default 1,
FechaCreacion datetime default getdate()
)

go

CREATE TABLE ESTADO_PRESTAMO(
IdEstadoPrestamo int primary key,
Descripcion varchar(50),
Estado bit default 1,
FechaCreacion datetime default getdate()
)
GO

CREATE TABLE PRESTAMO(
IdPrestamo int primary key identity,
IdEstadoPrestamo int references ESTADO_PRESTAMO(IdEstadoPrestamo),
IdPersona int references PERSONA(IdPersona),
IdLibro int references Libro(IdLibro),
FechaDevolucion datetime,
FechaConfirmacionDevolucion datetime,
EstadoEntregado varchar(100),
EstadoRecibido varchar(100),
Estado bit default 1,
FechaCreacion datetime default getdate()
)
```

Evidencia



Creación de procedimientos almacenado

Fue la parte más tediosa y desafiante en la elaboración del proyecto

. sp_ModificarCategoria

Este procedimiento permite modificar la información de una categoría en la base de datos.

Parámetros de entrada:

@IdCategoria: El identificador de la categoría a modificar.

@Descripcion: La nueva descripción para la categoría.

@Estado: El nuevo estado de la categoría.

@Resultado: Variable de salida que indica el resultado de la operación (1 si es exitosa, 0 si falla).

Función:

Verifica que no exista otra categoría con el mismo nombre (excepto la que se está modificando).

Si no existe, actualiza los datos de la categoría.

Si ya existe una categoría con ese nombre, establece el resultado a 0.

2. sp_RegistrarEditorial

Este procedimiento guarda una nueva editorial en la base de datos.

Parámetros de entrada:

@Descripcion: Nombre de la editorial.

@Resultado: Variable de salida que indica el resultado de la operación (1 si es exitosa, 0 si falla).

Función:

Verifica que no exista ya una editorial con el mismo nombre.

Si no existe, inserta la nueva editorial.

Si ya existe, establece el resultado a 0.

3. sp_ModificarEditorial

Este procedimiento permite modificar la información de una editorial existente.

Parámetros de entrada:

@IdEditorial: El identificador de la editorial a modificar.

@Descripcion: La nueva descripción para la editorial.

@Estado: El nuevo estado de la editorial.

@Resultado: Variable de salida que indica el resultado de la operación (1 si es exitosa, 0 si falla).

Función:

Verifica que no exista otra editorial con el mismo nombre (excepto la que se está modificando).

Si no existe, actualiza los datos de la editorial.

Si ya existe una editorial con ese nombre, establece el resultado a 0.

4. sp_RegistrarAutor

Este procedimiento permite registrar un nuevo autor en la base de datos.

Parámetros de entrada:

@Descripcion: Nombre del autor.

@Resultado: Variable de salida que indica el resultado de la operación (1 si es exitosa, 0 si falla).

Función:

Verifica que no exista un autor con el mismo nombre.

Si no existe, inserta el nuevo autor.

Si ya existe, establece el resultado a 0.

5. sp_ModificarAutor

Este procedimiento permite modificar la información de un autor existente.

Parámetros de entrada:

@IdAutor: El identificador del autor a modificar.

@Descripcion: El nuevo nombre del autor.

@Estado: El nuevo estado del autor.

@Resultado: Variable de salida que indica el resultado de la operación (1 si es exitosa, 0 si falla).

Función:

Verifica que no exista otro autor con el mismo nombre (excepto el que se está modificando).

Si no existe, actualiza los datos del autor.

Si ya existe, establece el resultado a 0.

6. sp_registrarLibro

Este procedimiento guarda un nuevo libro en la base de datos.

Parámetros de entrada:

@Titulo: Título del libro.

@RutaPortada: Ruta del archivo de la portada del libro.

@NombrePortada: Nombre del archivo de la portada.

@IdAutor: El identificador del autor del libro.

@IdCategoria: El identificador de la categoría del libro.

@IdEditorial: El identificador de la editorial del libro.

@Ubicacion: Ubicación del libro en la biblioteca.

@Ejemplares: Número de ejemplares disponibles del libro.

@Resultado: Variable de salida que contiene el ID del libro recién insertado si la operación es exitosa.

Función:

Verifica que no exista un libro con el mismo título.

Si no existe, inserta el libro en la base de datos y devuelve el ID del libro recién insertado.

7. sp_modificarLibro

Este procedimiento permite modificar la información de un libro existente.

Parámetros de entrada:

@IdLibro: El identificador del libro a modificar.

@Titulo: El nuevo título del libro.

@IdAutor: El identificador del autor del libro.

@IdCategoria: El identificador de la categoría del libro.

@IdEditorial: El identificador de la editorial del libro.

@Ubicacion: La nueva ubicación del libro.

@Ejemplares: El nuevo número de ejemplares disponibles.

@Estado: El nuevo estado del libro.

@Resultado: Variable de salida que indica el resultado de la operación (1 si es exitosa, 0 si falla).

Función:

Verifica que no exista otro libro con el mismo título (excepto el que se está modificando).

Si no existe, actualiza los datos del libro.

Si ya existe, establece el resultado a 0.

8. sp_actualizarRutaImagen

Este procedimiento actualiza la ruta de la portada de un libro.

Parámetros de entrada:

@IdLibro: El identificador del libro.

@NombrePortada: El nuevo nombre de la portada.

Función:

Actualiza el nombre de la portada del libro en la base de datos.

9. fn_obtenercorrelativo

Esta función genera un código correlativo para un número dado.

Parámetros de entrada:

@numero: El número para el cual se generará el correlativo.

Función:

Devuelve un código correlativo con el formato LE000000 seguido de un número de seis dígitos.

10. sp_RegistrarPersona

Este procedimiento guarda una nueva persona en la base de datos.

Parámetros de entrada:

@Nombre: Nombre de la persona.

@Apellido: Apellido de la persona.

@Correo: Correo electrónico de la persona.

@Clave: Contraseña de la persona.

@IdTipoPersona: El tipo de persona (por ejemplo, estudiante, profesor).

@Resultado: Variable de salida que indica el resultado de la operación (1 si es exitosa, 0 si falla).

Función:

Verifica que no exista una persona con el mismo correo.

Si no existe, inserta la persona en la base de datos y, si es de tipo 3, genera un código único para ella.

11. sp_ModificarPersona

Este procedimiento permite modificar la información de una persona existente.

Parámetros de entrada:

@IdPersona: El identificador de la persona a modificar.

@Nombre: El nuevo nombre de la persona.

@Apellido: El nuevo apellido de la persona.

@Correo: El nuevo correo de la persona.

@Clave: La nueva clave de la persona.

@IdTipoPersona: El nuevo tipo de persona.

@Estado: El nuevo estado de la persona.

@Resultado: Variable de salida que indica el resultado de la operación (1 si es exitosa, 0 si falla).

12. sp_RegistrarPrestamo

Este procedimiento registra un nuevo préstamo de un libro.

Parámetros de entrada:

@IdEstadoPrestamo: El estado del préstamo.

@IdPersona: El identificador de la persona que realiza el préstamo.

@IdLibro: El identificador del libro prestado.

@FechaDevolucion: La fecha en la que se espera que el libro sea devuelto.

@EstadoEntregado: El estado del libro entregado.

@Resultado: Variable de salida que indica el resultado de la operación (1 si es exitosa).

13. sp_existePrestamo

Este procedimiento verifica si una persona ya tiene un préstamo de un libro específico.

Parámetros de entrada:

@IdPersona: El identificador de la persona.

@IdLibro: El identificador del libro.

Función:

Verifica si ya existe un préstamo para la persona y el libro indicados. Si existe, devuelve 1, de lo contrario devuelve 0.

```
USE DB_BIBLIOTECA

GO

--PROCEDIMIENTO PARA GUARDAR CATEGORIA
CREATE PROC sp_RegistrarCategoria(
@Descripcion varchar(50),
@Resultado bit output
)as
begin
	SET @Resultado = 1
	IF NOT EXISTS (SELECT * FROM CATEGORIA WHERE Descripcion = @Descripcion)

		insert into CATEGORIA(Descripcion) values (
		@Descripcion
		)
	ELSE
		SET @Resultado = 0

end


go

--PROCEDIMIENTO PARA MODIFICAR CATEGORIA
create procedure sp_ModificarCategoria(
@IdCategoria int,
@Descripcion varchar(60),
@Estado bit,
@Resultado bit output
)
as
begin
	SET @Resultado = 1
	IF NOT EXISTS (SELECT * FROM CATEGORIA WHERE Descripcion =@Descripcion and
IdCategoria != @IdCategoria)

		update CATEGORIA set
		Descripcion = @Descripcion,
		Estado = @Estado
		where IdCategoria = @IdCategoria
	ELSE
		SET @Resultado = 0
```

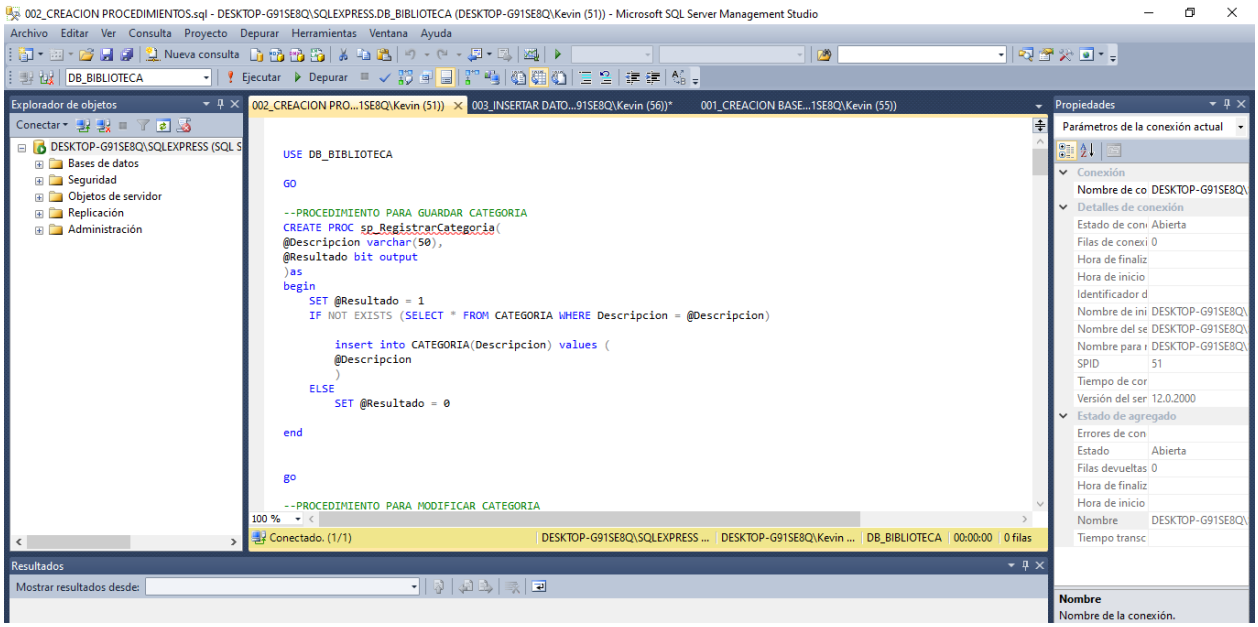
```
end

GO


--PROCEDIMIENTO PARA GUARDAR EDITORIAL
CREATE PROC sp_RegistrarEditorial(
@Descripcion varchar(50),
@Resultado bit output
)as
begin
	SET @Resultado = 1
	IF NOT EXISTS (SELECT * FROM EDITORIAL WHERE Descripcion = @Descripcion)

		insert into EDITORIAL(Descripcion) values (
		@Descripcion
		)
	ELSE
		SET @Resultado = 0

end


go

--PROCEDIMIENTO PARA MODIFICAR EDITORIAL
create procedure sp_ModificarEditorial(
@IdEditorial int,
@Descripcion varchar(60),
@Estado bit,
@Resultado bit output
)
as
begin
	SET @Resultado = 1
	IF NOT EXISTS (SELECT * FROM EDITORIAL WHERE Descripcion =@Descripcion and
IdEditorial != @IdEditorial)

		update EDITORIAL set
		Descripcion = @Descripcion,
		Estado = @Estado
		where IdEditorial = @IdEditorial
	ELSE
		SET @Resultado = 0

end


GO


--PROCEDIMIENTO PARA GUARDAR AUTOR
CREATE PROC sp_RegistrarAutor(
@Descripcion varchar(50),
@Resultado bit output
)as
begin
	SET @Resultado = 1
	IF NOT EXISTS (SELECT * FROM AUTOR WHERE Descripcion = @Descripcion)

		insert into AUTOR(Descripcion) values (
```

```
                @Descripcion
                )
        ELSE
                SET @Resultado = 0

end


go

--PROCEDIMIENTO PARA MODIFICAR AUTOR
create procedure sp_ModificarAutor(
@IdAutor int,
@Descripcion varchar(60),
@Estado bit,
@Resultado bit output
)
as
begin
        SET @Resultado = 1
        IF NOT EXISTS (SELECT * FROM AUTOR WHERE Descripcion =@Descripcion and
IdAutor != @IdAutor)

                update AUTOR set
                Descripcion = @Descripcion,
                Estado = @Estado
                where IdAutor = @IdAutor
        ELSE
                SET @Resultado = 0

end


go

create proc sp_registrarLibro(
@Titulo varchar(100),
@RutaPortada varchar(100),
@NombrePortada varchar(100),
@IdAutor int,
@IdCategoria int,
@IdEditorial int,
@Ubicacion varchar(100),
@Ejemplares int,
@Resultado int output
)
as
begin
        SET @Resultado = 0
        IF NOT EXISTS (SELECT * FROM LIBRO WHERE Titulo = @Titulo)
        begin
                insert into
LIBRO(Titulo,RutaPortada,NombrePortada,IdAutor,IdCategoria,IdEditorial,Ubicacion,
Ejemplares) values (

        @Titulo,@RutaPortada,@NombrePortada,@IdAutor,@IdCategoria,@IdEditorial,@Ub
icacion,@Ejemplares)

                SET @Resultado = scope_identity()
        end
end
```

```
go

create proc sp_modificarLibro(
@IdLibro int,
@Titulo varchar(100),
@IdAutor int,
@IdCategoria int,
@IdEditorial int,
@Ubicacion varchar(100),
@Ejemplares int,
@Estado bit,
@Resultado bit output
)
as
begin
	SET @Resultado = 0
	IF NOT EXISTS (SELECT * FROM LIBRO WHERE Titulo = @Titulo and IdLibro != 
@IdLibro)
	begin

		update LIBRO set
		Titulo = @Titulo,
		IdAutor = @IdAutor,
		IdCategoria = @IdCategoria,
		IdEditorial = @IdEditorial,
		Ubicacion = @Ubicacion,
		Ejemplares = @Ejemplares,
		Estado = @Estado
		where IdLibro = @IdLibro

		SET @Resultado = 1
	end
end

GO

create proc sp_actualizarRutaImagen(
@IdLibro int,
@NombrePortada varchar(500)
)
as
begin
	update libro set NombrePortada = @NombrePortada where IdLibro = @IdLibro
end



GO

CREATE FUNCTION fn_obtenercorrelativo(@numero int)

RETURNS varchar(100)
AS
BEGIN
	DECLARE @correlativo varchar(100)

	set @correlativo = 'LE' + RIGHT('000000' + CAST(@numero AS varchar), 6)

	RETURN @correlativo

END
```

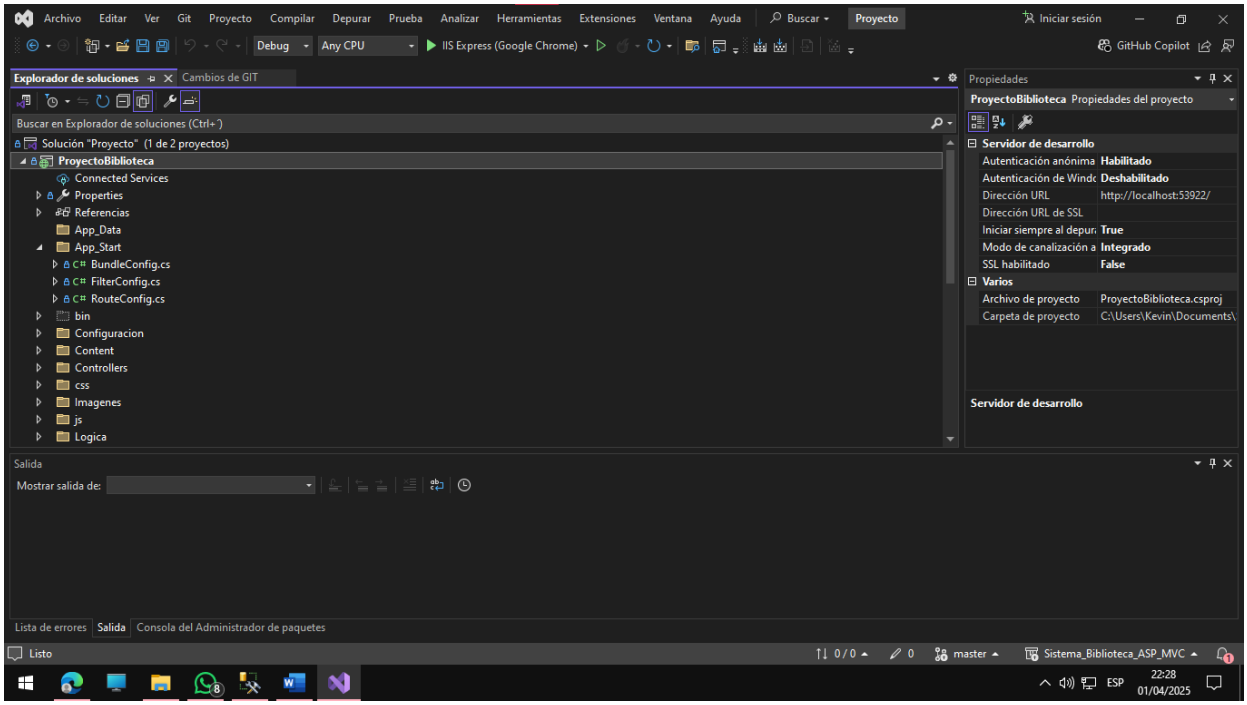
```
GO

--PROCEDIMIENTO PARA GUARDAR CATEGORIA
create PROC sp_RegistrarPersona(
@Nombre varchar(50),
@Apellido varchar(50),
@Correo varchar(50),
@Clave varchar(50),
@IdTipoPersona int,
@Resultado bit output
)as
begin
	SET @Resultado = 1
	DECLARE @IDPERSONA INT
	IF NOT EXISTS (SELECT * FROM persona WHERE correo = @Correo)
	begin
		insert into persona(Nombre,Apellido,Correo,Clave,IdTipoPersona)
values (
		@Nombre,@Apellido,@Correo,@Clave,@IdTipoPersona)

		SET @IDPERSONA = SCOPE_IDENTITY()
		print @IDPERSONA
		if(@IdTipoPersona = 3)
		begin
			print 'si es igual'
			UPDATE PERSONA SET
			Codigo = dbo.fn_obtenercorrelativo(@IDPERSONA),
			Clave = dbo.fn_obtenercorrelativo(@IDPERSONA)
			WHERE IdPersona = @IDPERSONA
		end
	end
	ELSE
		SET @Resultado = 0

end


go

--PROCEDIMIENTO PARA MODIFICAR CATEGORIA
create procedure sp_ModificarPersona(
@IdPersona int,
@Nombre varchar(50),
@Apellido varchar(50),
@Correo varchar(50),
@Clave varchar(50),
@IdTipoPersona int,
@Estado bit,
@Resultado bit output
)
as
begin
	SET @Resultado = 1
	IF NOT EXISTS (SELECT * FROM persona WHERE correo =@Correo and IdPersona
!= @IdPersona)

		update PERSONA set
		Nombre = @Nombre,
		Apellido = @Apellido,
		Correo = @Correo,
		IdTipoPersona = @IdTipoPersona,
		Estado = @Estado
```

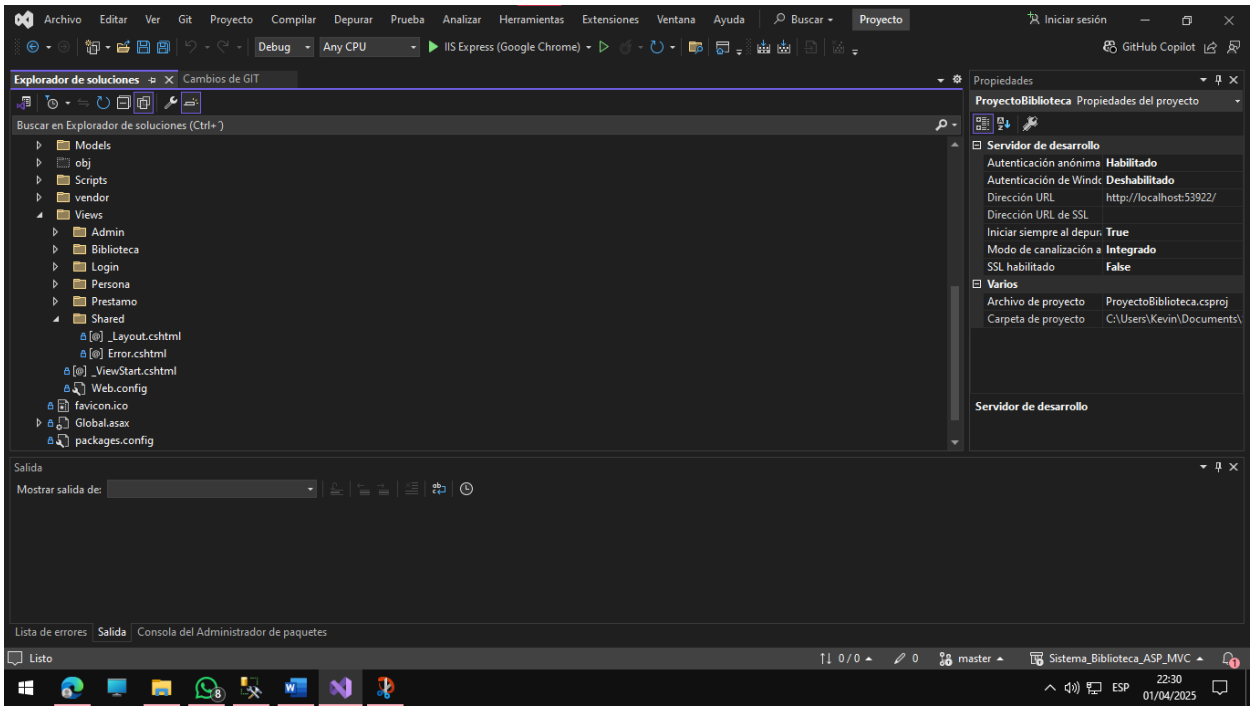
```
            where IdPersona = @IdPersona
      ELSE
            SET @Resultado = 0

end

GO


create PROC sp_RegistrarPrestamo(
@IdEstadoPrestamo int,
@IdPersona int,
@IdLibro int,
@FechaDevolucion datetime,
@EstadoEntregado varchar(500),
@Resultado bit output
)as
begin
      SET DATEFORMAT dmy;
      INSERT INTO
PRESTAMO(IdEstadoPrestamo,IdPersona,IdLibro,FechaDevolucion,EstadoEntregado)
      values(@IdEstadoPrestamo,@IdPersona,@IdLibro,@FechaDevolucion,@EstadoEntre
gado)

      SET @Resultado = 1
end


go

create PROC sp_existePrestamo(
@IdPersona int,
@IdLibro int,
@Resultado bit output
)as
begin
      SET @Resultado = 0

      if(exists(select * from PRESTAMO where IdPersona = @IdPersona and IdLibro
=@IdLibro ))
      begin
            SET @Resultado = 1
      end

end
```


002_CREACION PROCEDIMIENTOS.sql - DESKTOP-G91SE8Q\SQLEXPRESS.DB_BIBLIOTECA (DESKTOP-G91SE8Q\Kevin (51)) - Microsoft SQL Server Management Studio
Archivo Editar Ver Consulta Proyecto Depurar Herramientas Ventana Ayuda
Nueva consulta
DB_BIBLIOTECA
Ejecutar Depurar
Explorador de objetos
Conectar
DESKTOP-G91SE8Q\SQLEXPRESS (SQL S
Bases de datos
Seguridad
Objetos de servidor
Replicación
Administración
002_CREACION PRO...1SE8Q\Kevin (51))
003_INSERTAR DATO...91SE8Q\Kevin (56))*
001_CREACION BASE...1SE8Q\Kevin (55))
USE DB_BIBLIOTECA
GO
--PROCEDIMIENTO PARA GUARDAR CATEGORIA
CREATE PROC sp_RegistrarCategoria(
@Descripcion varchar(50),
@Resultado bit output
)as
begin
SET @Resultado = 1
IF NOT EXISTS (SELECT * FROM CATEGORIA WHERE Descripcion = @Descripcion)
insert into CATEGORIA(Descripcion) values (
@Descripcion
)
ELSE
SET @Resultado = 0
end
go
--PROCEDIMIENTO PARA MODIFICAR CATEGORIA
100 %
Conectado. (1/1)
DESKTOP-G91SE8Q\SQLEXPRESS ...
DESKTOP-G91SE8Q\Kevin ...
DB_BIBLIOTECA
00:00:00
0 filas
Resultados
Mostrar resultados desde:
Propiedades
Parámetros de la conexión actual
Conexión
Nombre de co DESKTOP-G91SE8Q\
Detalles de conexión
Estado de con Abierta
Filas de conexi 0
Hora de finaliz
Hora de inicio
Identificador d
Nombre de ini DESKTOP-G91SE8Q\
Nombre del se DESKTOP-G91SE8Q\
Nombre para r DESKTOP-G91SE8Q\
SPID 51
Tiempo de cor
Versión del ser 12.0.2000
Estado de agregado
Errores de con
Estado Abierta
Filas devueltas 0
Hora de finaliz
Hora de inicio
Nombre DESKTOP-G91SE8Q\
Tiempo transc
Nombre
Nombre de la conexión.

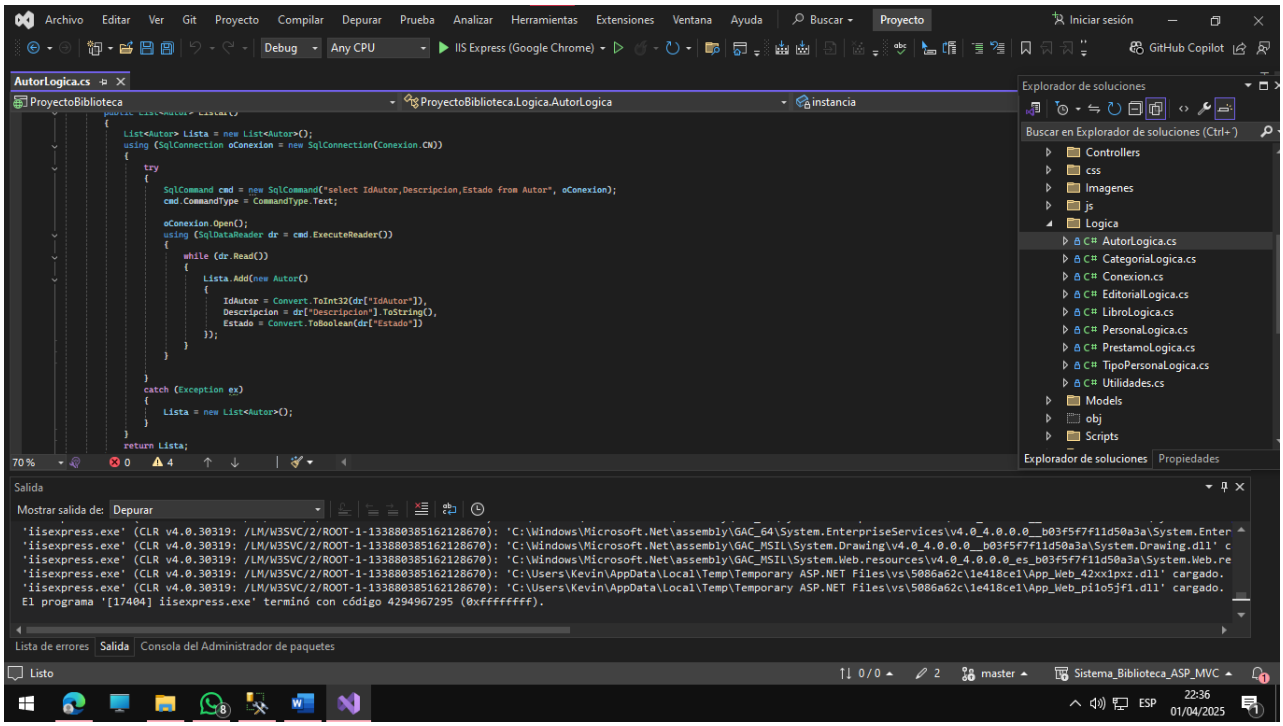
Evidencia

Estructura y organización del proyecto

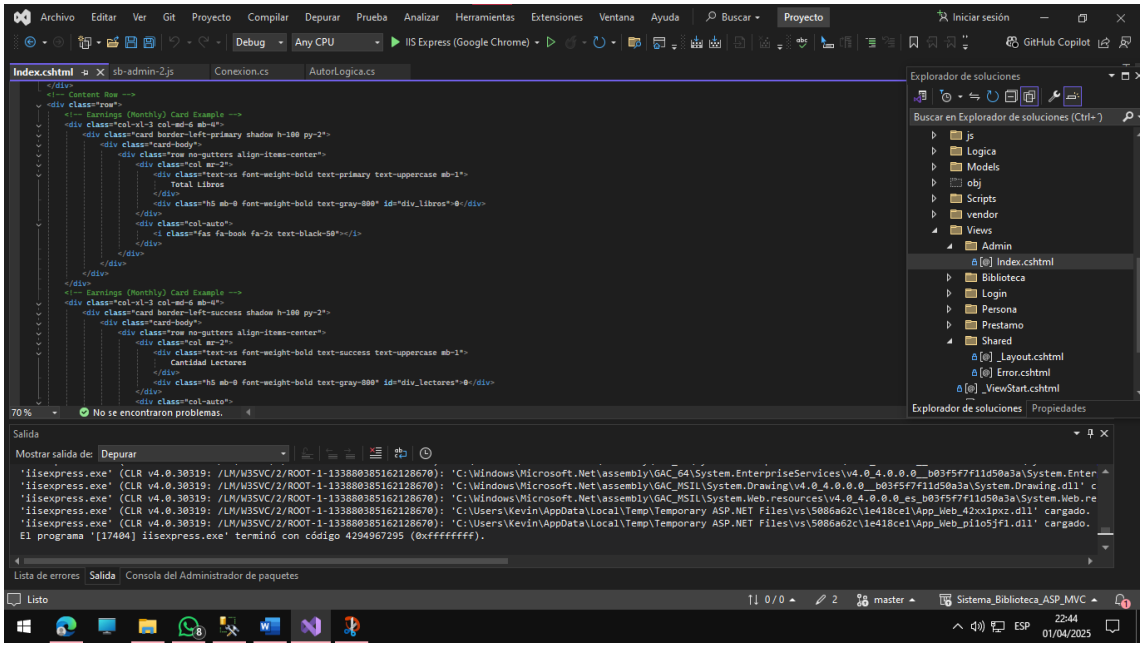




Utilizando framework asp.net mvc

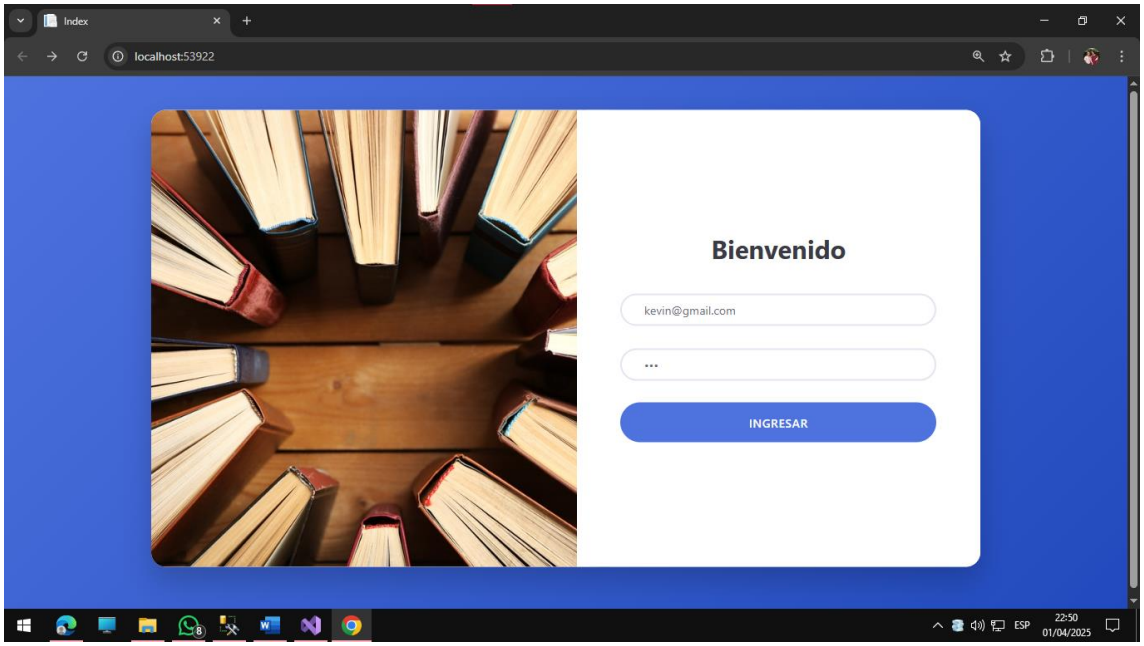


Tuve problemas de depuración en la carpeta de lógica ya que de allí tuve inconveniente con la base de datos basado a lo SP creado

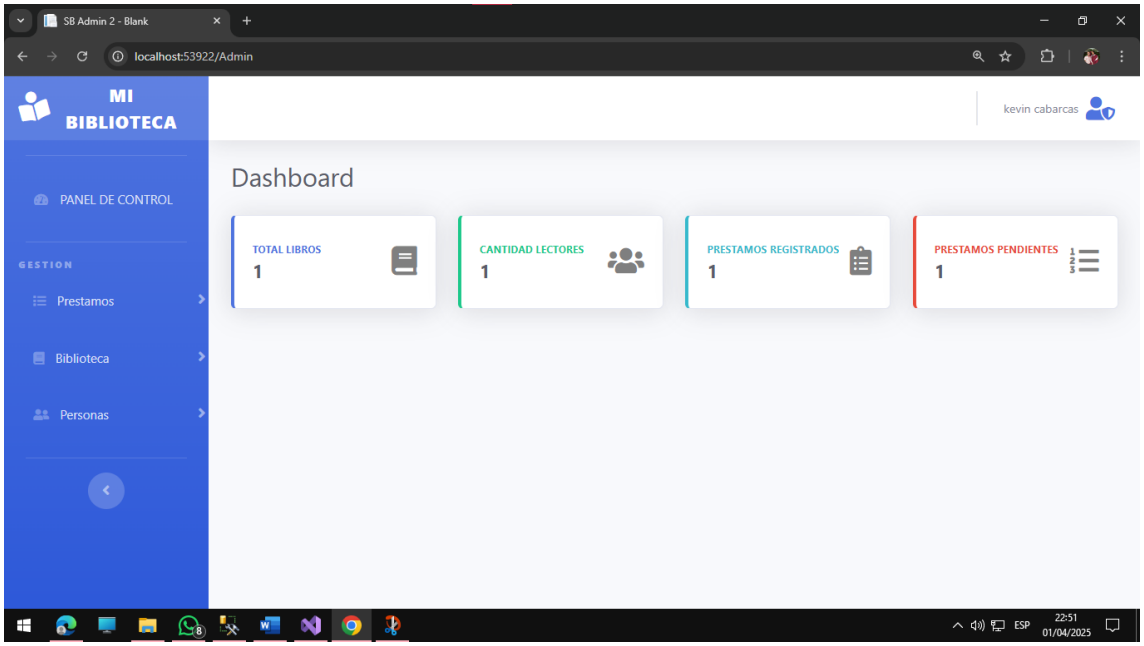


En la vistas no tuve tantos problemas ya que el MVC ya te la creas y solo es implementar codigo ya existente de diseño como paneles de administración o de control.

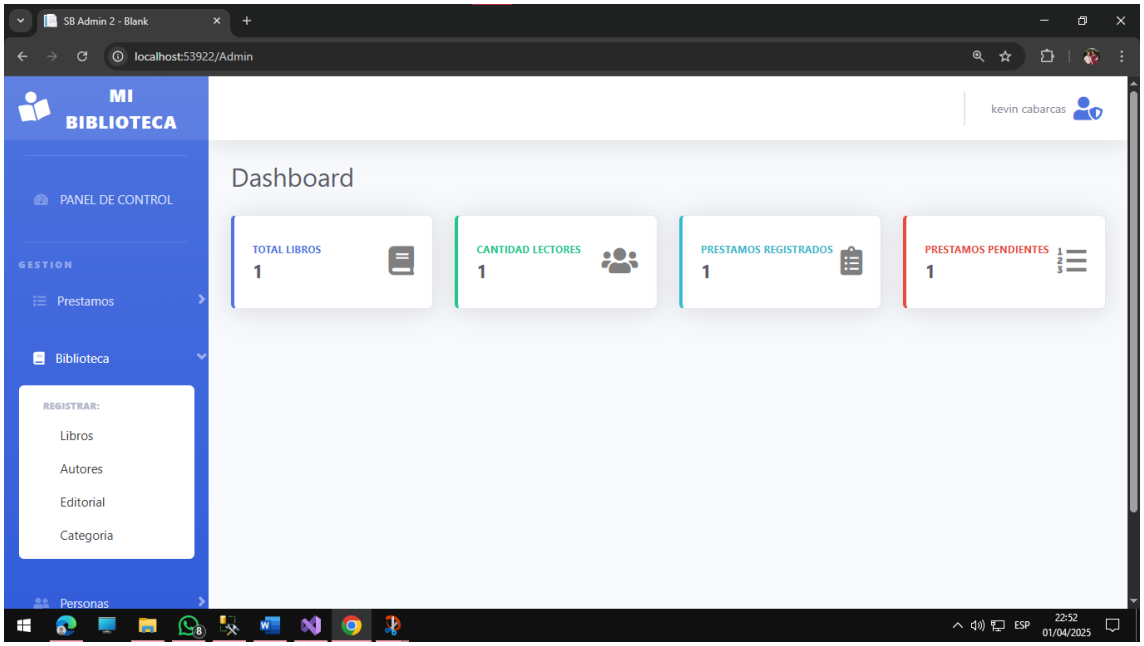
Login



Panel de control

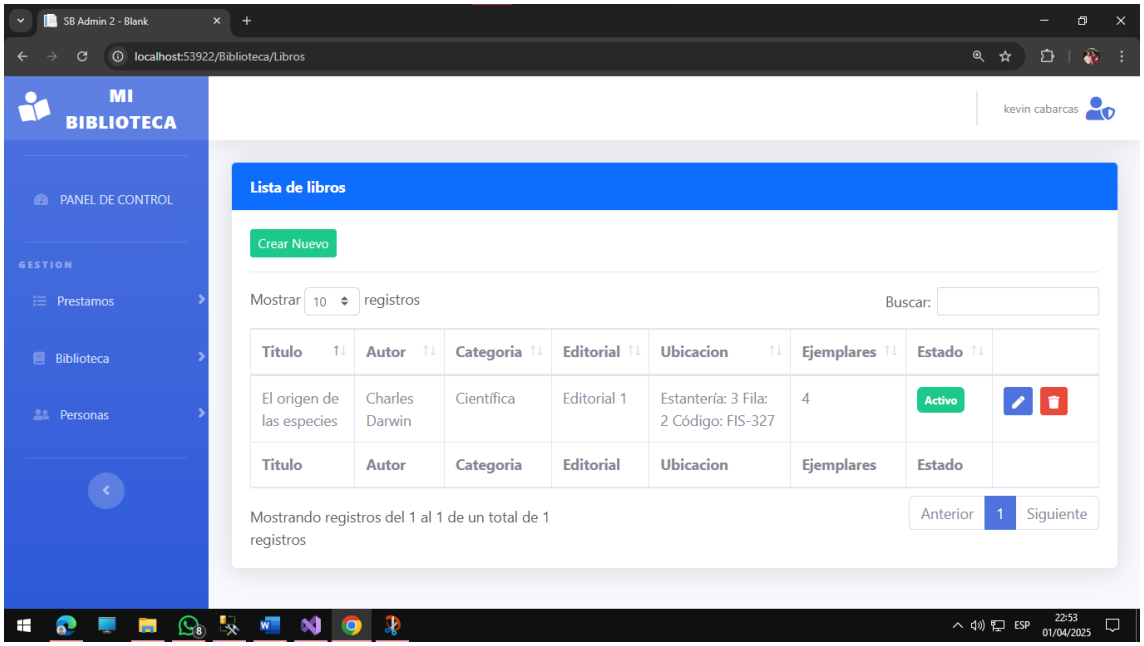


Opciones de biblioteca

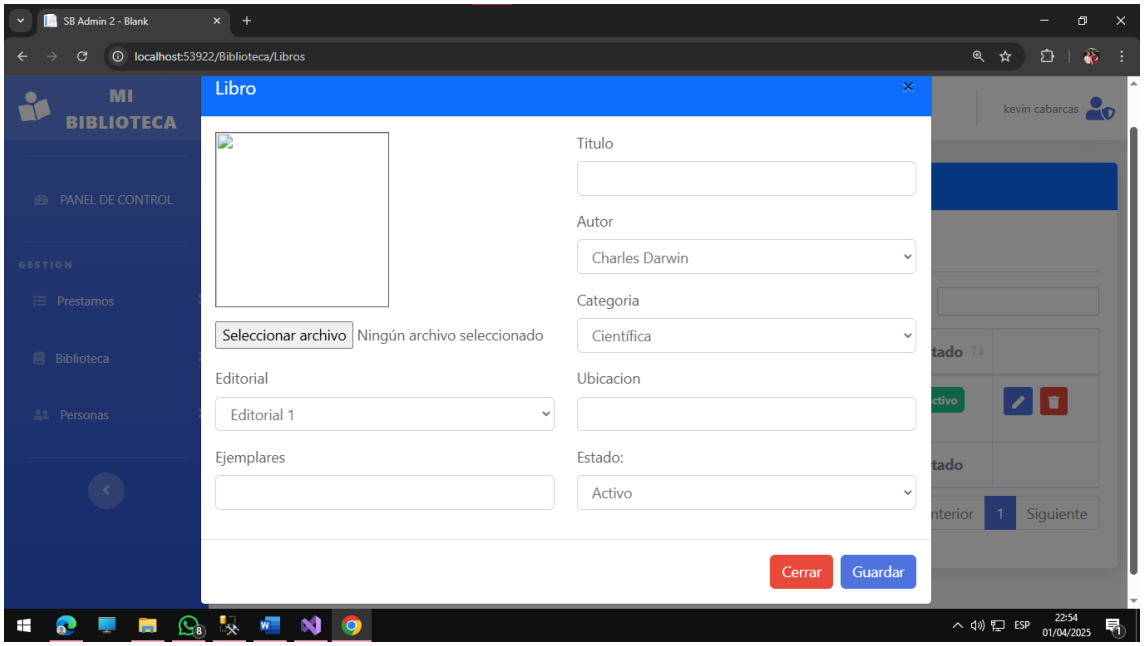


Panel donde se ve el control de la gestión de los libros

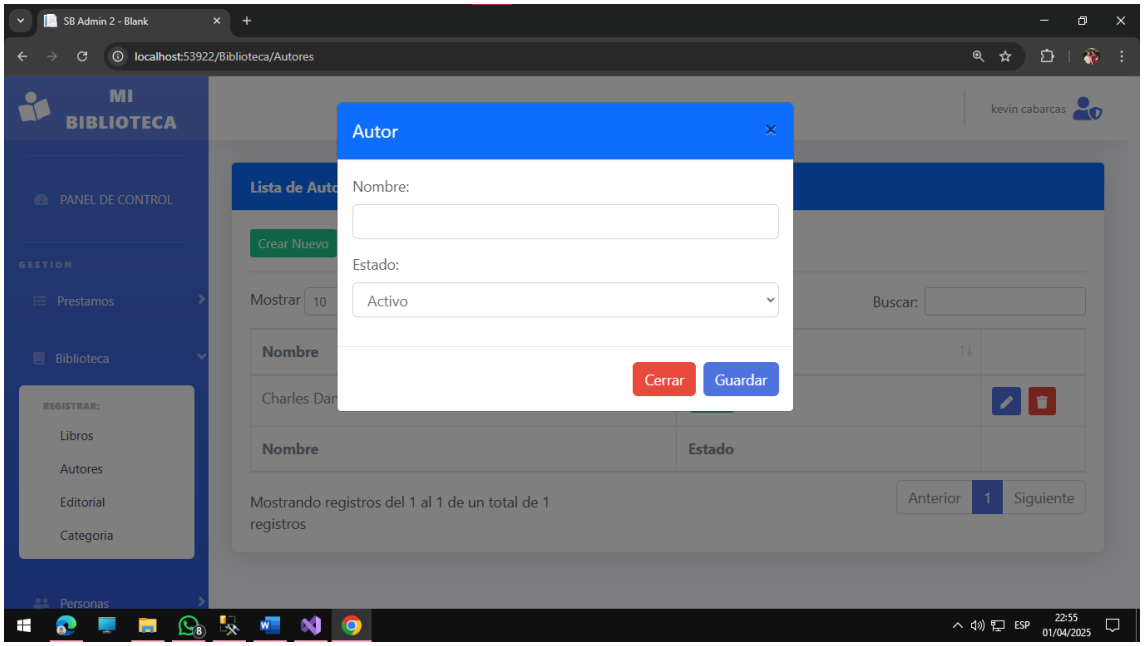
Opción de libro



Creación de libro

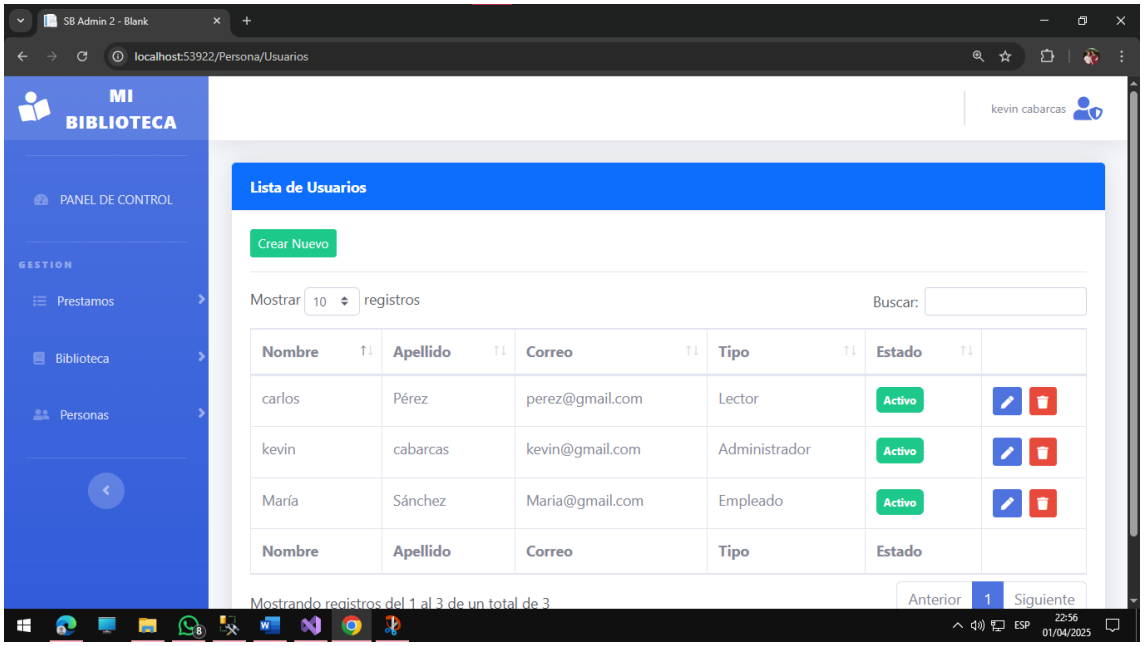


Creación de autores



Esto mismo aplica a categoría y editorial en la parte visual al momento de creación

En la opción persona



Creación de usuario y lista de usuario

Conclusiones

El sistema de gestión de bibliotecas en ASP.NET proporciona una solución completa para la administración de recursos bibliográficos. Su diseño modular facilita la escalabilidad, integración con otras plataformas y mejora la eficiencia en la gestión de préstamos y devoluciones.

Proyecto funcional

CABARCAS-Q/Sistema-de-gesti-n-de-biblioteca