UNIVERSIDAD TECNOLÓGICA DE TULA-TEPEJI

INGENIERÍA EN DESARROLLO Y GESTIÓN DE SOFTWARE

ADMINISTRACIÓN DE BASES DE DATOS

JOSE LUIS HERRERA GALLARDO

PRESENTA: BRIAN EMMANUEL FLORES HERNANDEZ

8IDGS-G1

FECHA DE ENTREGA: 10/03/2025

Tabla de Contenido

Informe Técnico

1.Creación Dinámica de Base de Datos

1.1. Descripción del Procedimiento

El procedimiento almacenado **sp_CreateDatabase** permite la creación dinámica de bases de datos mediante el uso de SQL Server. Este procedimiento recibe varios parámetros como el nombre de la base de datos, la ubicación de los archivos de datos y logs, el tamaño inicial de los archivos, el crecimiento de los mismos y el tamaño máximo permitido.

Parámetros del Procedimiento

- **@DBName**: Nombre de la base de datos.
- **@DataFilePath**: Ruta del archivo MDF (archivo principal de datos).
- **@LogFilePath**: Ruta del archivo LDF (archivo de logs).
- **@DataFileSizeMB**: Tamaño inicial del archivo MDF en MB.
- **@LogFileSizeMB**: Tamaño inicial del archivo LDF en MB.
- **@DataFileGrowth**: Crecimiento del archivo MDF.
- **@LogFileGrowth**: Crecimiento del archivo LDF.
- **@DataFileMaxSize**: Tamaño máximo del archivo MDF.

1.2. Validaciones Implementadas

Para garantizar la correcta creación de las bases de datos y evitar errores, se implementaron las siguientes validaciones:

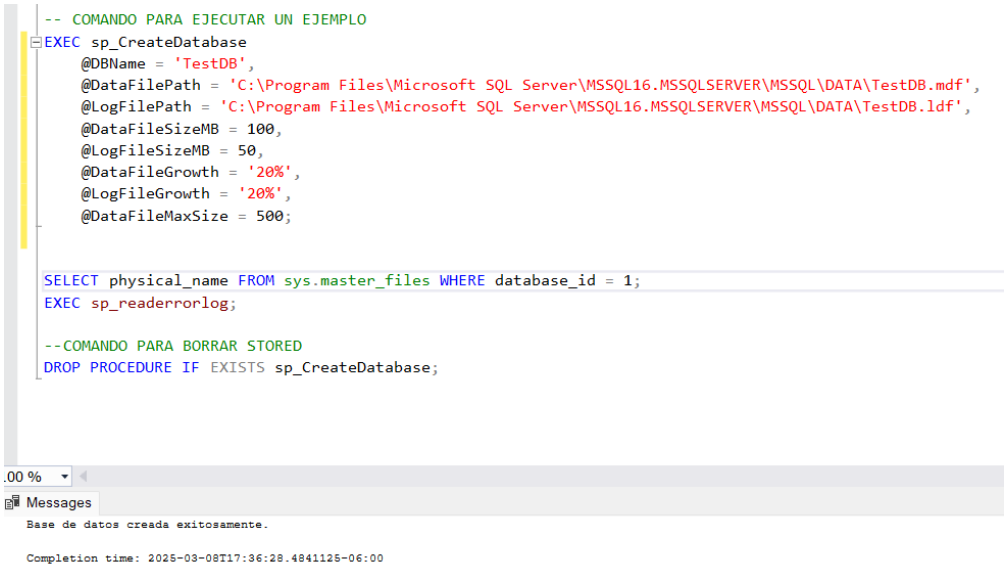
- **Verificación de existencia**: Se valida si la base de datos ya existe en sys.databases, en cuyo caso se devuelve un mensaje de error.
- **Evitar nombres vacíos o nulos**: Se verifica que el nombre de la base de datos no esté vacío o compuesto solo por espacios.
- **Validación de rutas de archivos**: Se asegura que las rutas proporcionadas para los archivos MDF y LDF no estén vacías.

- **Tamaños de archivos**: Se impide la creación de bases de datos con tamaños de archivo negativos o iguales a cero.
- **Crecimiento de archivos**: Se verifica que el crecimiento de los archivos sea un porcentaje mayor a 0%.
- **Manejo de errores**: Se captura cualquier error inesperado y se muestra un mensaje con el detalle del problema.

1.3. Pruebas Realizadas

Para verificar el correcto funcionamiento del procedimiento, se realizaron las siguientes pruebas:

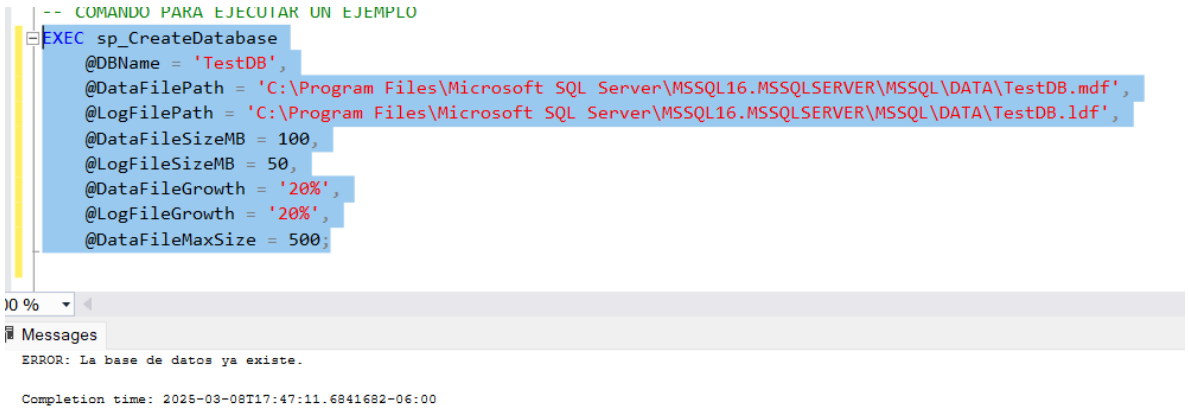
- **Prueba con datos válidos**: Se ejecutó el procedimiento con un nombre de base de datos único y configuraciones válidas. Resultado: Base de datos creada exitosamente.



```
-- COMANDO PARA EJECUTAR UN EJEMPLO
EXEC sp_CreateDatabase
    @DBName = 'TestDB',
    @DataFilePath = 'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\TestDB.mdf',
    @LogFilePath = 'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\TestDB.ldf',
    @DataFileSizeMB = 100,
    @LogFileSizeMB = 50,
    @DataFileGrowth = '20%',
    @LogFileGrowth = '20%',
    @DataFileMaxSize = 500;


SELECT physical_name FROM sys.master_files WHERE database_id = 1;
EXEC sp_readerrorlog;

--COMANDO PARA BORRAR STORED
DROP PROCEDURE IF EXISTS sp_CreateDatabase;
```

```
.00 %
Messages
Base de datos creada exitosamente.

Completion time: 2025-03-08T17:36:28.4841125-06:00
```

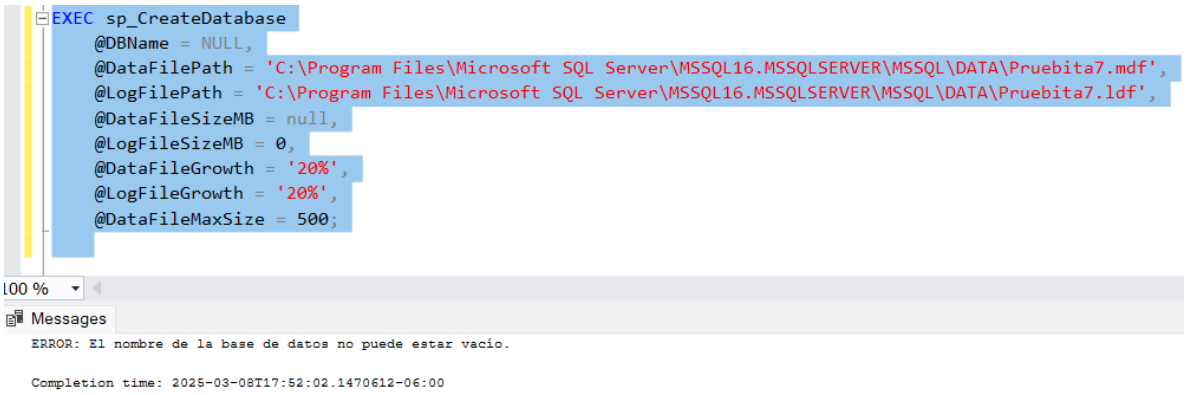
- **Prueba con nombre duplicado**: Se intentó crear una base de datos con un nombre ya existente. Resultado: Mensaje de error "La base de datos ya existe".



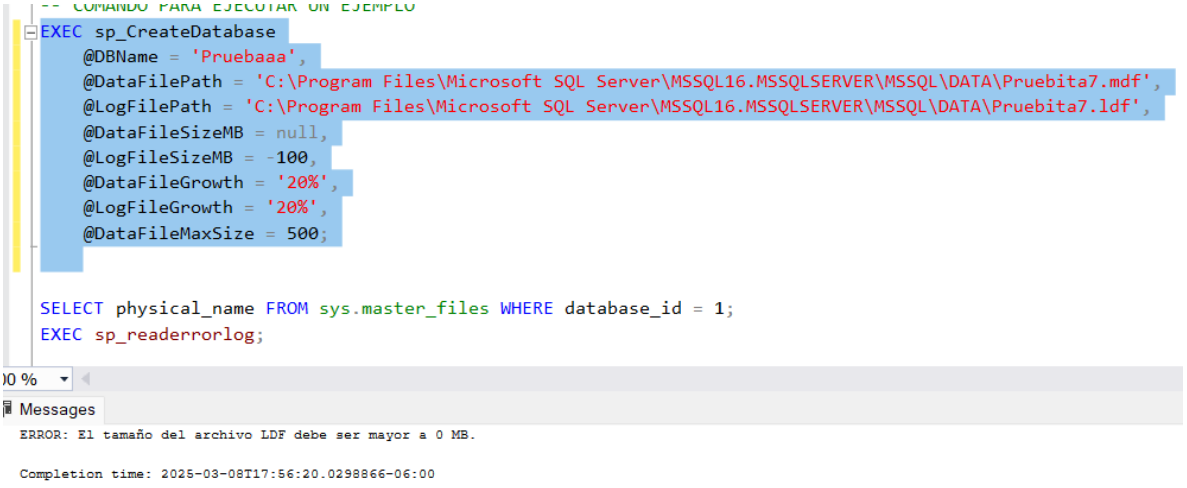
```
-- COMANDO PARA EJECUTAR UN EJEMPLO
EXEC sp_CreateDatabase
    @DBName = 'TestDB',
    @DataFilePath = 'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\TestDB.mdf',
    @LogFilePath = 'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\TestDB.ldf',
    @DataFileSizeMB = 100,
    @LogFileSizeMB = 50,
    @DataFileGrowth = '20%',
    @LogFileGrowth = '20%',
    @DataFileMaxSize = 500;
```

```
)0 %
Messages
ERROR: La base de datos ya existe.

Completion time: 2025-03-08T17:47:11.6841682-06:00
```

- **Prueba con valores nulos**: Se ejecutó el procedimiento con valores nulos en los parámetros requeridos. Resultado: Mensajes de error indicando los valores faltantes.



- **Prueba con tamaños negativos**: Se ingresaron valores negativos para los tamaños de archivos. Resultado: Error validado correctamente.



2. Desarrollo de Aplicación Web

2.1. Descripción del Desarrollo

Se desarrolló una aplicación web utilizando **ASP.NET Core MVC** que permite a los usuarios **crear bases de datos dinámicamente** mediante la ejecución del procedimiento almacenado sp_CreateDatabase. La aplicación proporciona una **interfaz intuitiva** donde el usuario puede ingresar los parámetros de la base de datos y visualizar las bases existentes en el servidor.

2.2. Funcionalidades Implementadas

La aplicación cuenta con las siguientes funcionalidades:

Interacción con el stored procedure:

- Permite enviar los parámetros requeridos al procedimiento almacenado para la creación de la base de datos.

Validación de entradas en el formulario web:

- Se implementaron validaciones en el formulario de entrada para evitar errores de usuario, tales como:
    - **Evitar nombres vacíos o con espacios.**
    - **No permitir valores negativos en los tamaños de archivos.**
    - **Verificar que las rutas de los archivos sean válidas.**
    - **Aceptar solo formatos correctos para el crecimiento de archivos (Ejemplo: '20%').**

Listar bases de datos existentes en el servidor:

- Se implementó una funcionalidad para obtener y mostrar todas las bases de datos creadas por el usuario en el servidor SQL.
- Se evita mostrar bases del sistema (master, tempdb, model, msdb).

Mensajes de confirmación y error:

- Se muestra un **mensaje de éxito** cuando la base de datos es creada correctamente.
- Se muestran **mensajes de error** si se produce una falla durante la creación.

2.3. Tecnologías Utilizadas

- **Backend:** ASP.NET Core MVC
- **Base de Datos:** SQL Server
- **ORM:** Dapper (para ejecutar consultas SQL de manera eficiente)
- **Frontend:** HTML, CSS, Bootstrap
- **Lenguaje:** C#
- **Entorno de Desarrollo:** Visual Studio 2022

2.4. Documentación del Código de la Aplicación Web

2.4.1. Backend - Controlador (DatabaseController.cs)

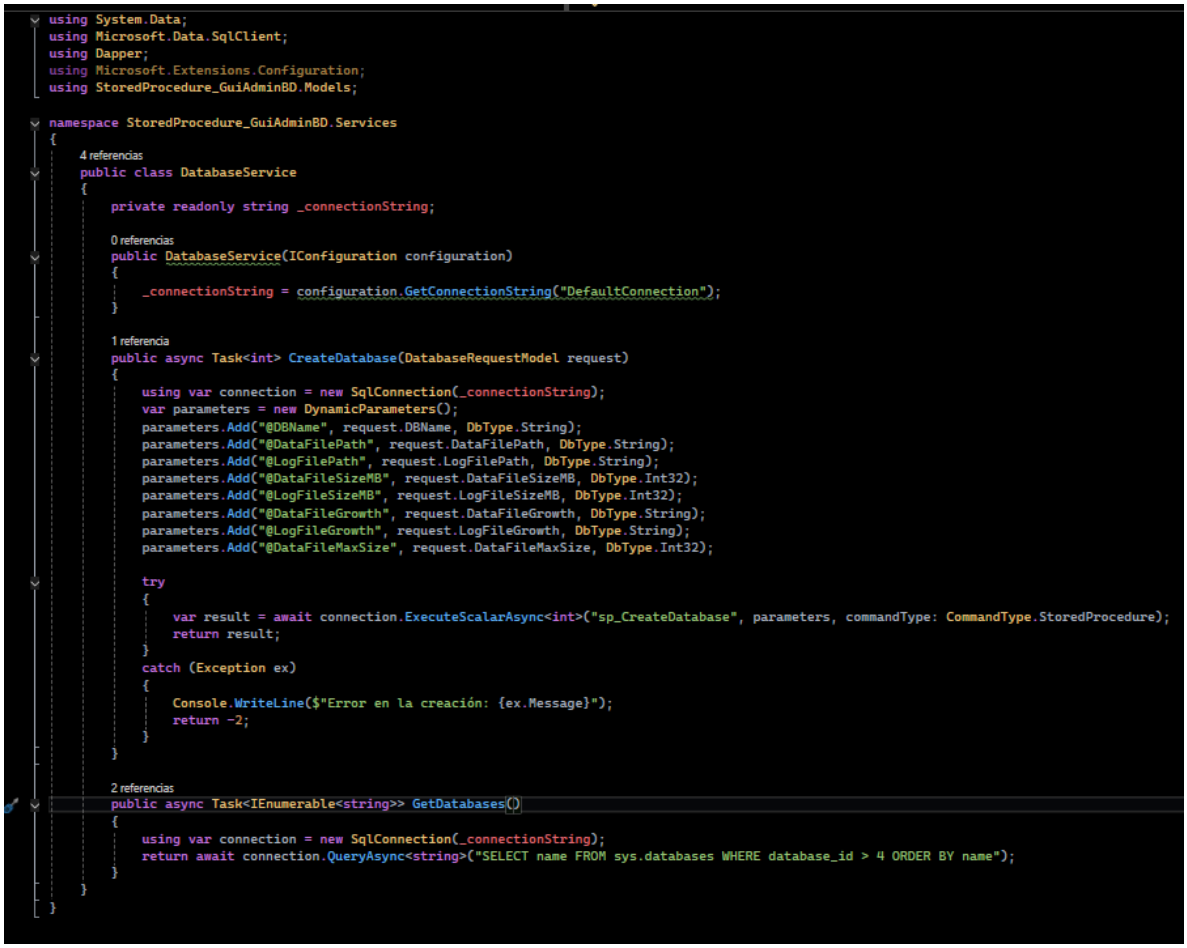
```
namespace StoredProcedure_GuiAdminBD.Controllers
{
    1 referencia
    public class DatabaseController : Controller
    {
        private readonly DatabaseService _databaseService;

        0 referencias
        public DatabaseController(DatabaseService databaseService)
        {
            _databaseService = databaseService;
        }

        0 referencias
        public async Task<IActionResult> Create()
        {
            var databases = await _databaseService.GetDatabases();
            ViewBag.Databases = databases;
            return View();
        }

        [HttpPost]
        0 referencias
        public async Task<IActionResult> Create(DatabaseRequestModel model)
        {
            if (ModelState.IsValid)
            {
                var result = await _databaseService.CreateDatabase(model);

                if (result == 0)
                {
                    ViewBag.Message = "☑ Base de datos creada exitosamente.";
                }
                else if (result == -20)
                {
                    ViewBag.Message = "⚠ Error: La base de datos ya existe.";
                }
                else if (result == -30)
                {
                    ViewBag.Message = "⚠ Error: El nombre de la base de datos no puede estar vacío.";
                }
                else if (result == -40 || result == -41)
                {
                    ViewBag.Message = "⚠ Error: La ruta del archivo MDF o LDF no es válida.";
                }
                else if (result == -50 || result == -51)
                {
                    ViewBag.Message = "⚠ Error: El tamaño de los archivos MDF o LDF debe ser mayor a 0 MB.";
                }
                else if (result == -60 || result == -61)
                {
                    ViewBag.Message = "⚠ Error: El crecimiento de archivos debe ser un porcentaje válido.";
                }
                else
                {
                    ViewBag.Message = "⚠ Error desconocido al crear la base de datos.";
                }
            }

            ViewBag.Databases = await _databaseService.GetDatabases();
            return View(model);
        }
```

Explicación del Código:

- **Create() (GET):** Obtiene la lista de bases de datos existentes llamando al servicio GetDatabases() y las envía a la vista.

- **Create() (POST):** Recibe los datos ingresados en el formulario y llama al método CreateDatabase() del servicio.

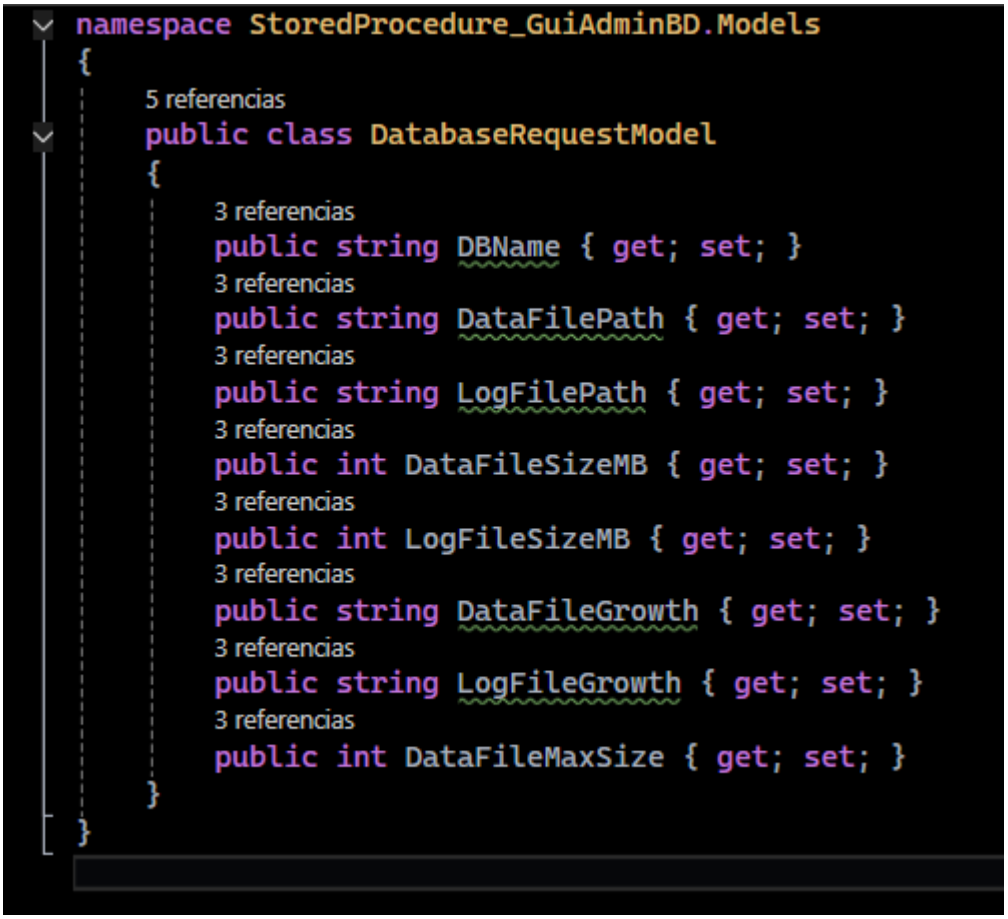
2.4.2. Backend - Servicio (DatabaseService.cs)



```
using System.Data;
using Microsoft.Data.SqlClient;
using Dapper;
using Microsoft.Extensions.Configuration;
using StoredProcedure_GuiAdminBD.Models;

namespace StoredProcedure_GuiAdminBD.Services
{
    4 referencias
    public class DatabaseService
    {
        private readonly string _connectionString;

        0 referencias
        public DatabaseService(IConfiguration configuration)
        {
            _connectionString = configuration.GetConnectionString("DefaultConnection");
        }

        1 referencia
        public async Task<int> CreateDatabase(DatabaseRequestModel request)
        {
            using var connection = new SqlConnection(_connectionString);
            var parameters = new DynamicParameters();
            parameters.Add("@DBName", request.DBName, DbType.String);
            parameters.Add("@DataFilePath", request.DataFilePath, DbType.String);
            parameters.Add("@LogFilePath", request.LogFilePath, DbType.String);
            parameters.Add("@DataFileSizeMB", request.DataFileSizeMB, DbType.Int32);
            parameters.Add("@LogFileSizeMB", request.LogFileSizeMB, DbType.Int32);
            parameters.Add("@DataFileGrowth", request.DataFileGrowth, DbType.String);
            parameters.Add("@LogFileGrowth", request.LogFileGrowth, DbType.String);
            parameters.Add("@DataFileMaxSize", request.DataFileMaxSize, DbType.Int32);

            try
            {
                var result = await connection.ExecuteScalarAsync<int>("sp_CreateDatabase", parameters, commandType: CommandType.StoredProcedure);
                return result;
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error en la creación: {ex.Message}");
                return -2;
            }
        }

        2 referencias
        public async Task<IEnumerable<string>> GetDatabases()
        {
            using var connection = new SqlConnection(_connectionString);
            return await connection.QueryAsync<string>("SELECT name FROM sys.databases WHERE database_id > 4 ORDER BY name");
        }
    }
}
```

Explicación: Este método obtiene todos los parámetros necesarios para ejecutar el stored procedure, de igual forma obtiene la lista de bases de datos existentes en el servidor SQL, excluyendo las bases del sistema.

2.4.3 Backend - Modelo (DatabaseRequestModel.cs)



```
namespace StoredProcedure_GuiAdminBD.Models
{
    5 referencias
    public class DatabaseRequestModel
    {
        3 referencias
        public string DBName { get; set; }
        3 referencias
        public string DataFilePath { get; set; }
        3 referencias
        public string LogFilePath { get; set; }
        3 referencias
        public int DataFileSizeMB { get; set; }
        3 referencias
        public int LogFileSizeMB { get; set; }
        3 referencias
        public string DataFileGrowth { get; set; }
        3 referencias
        public string LogFileGrowth { get; set; }
        3 referencias
        public int DataFileMaxSize { get; set; }
    }
}
```

En el modelo lo que traemos son las variables requeridas del stored procedure con sus respectivos tipos de datos.

2.4.3 Backend - Vistas (Create.cshtml)

```
@model StoredProcedure_GuiAdminBD.Models.DatabaseRequestModel
@{
    ViewData["Title"] = "Crear Base de Datos";
}

<h2>Crear Base de Datos</h2>

@if (ViewBag.Message != null)
{
    <div class="alert @(ViewBag.Message.Contains("⚠") ? "alert-danger" : "alert-success")">
        @ViewBag.Message
    </div>
}

<form method="post" id="databaseForm">
    <div class="form-group">
        <label>Nombre de la Base de Datos</label>
        <input type="text" asp-for="DBName" class="form-control" required />
        <span asp-validation-for="DBName" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label>Ruta del Archivo MDF</label>
        <input type="text" asp-for="DataFilePath" class="form-control" required />
        <span asp-validation-for="DataFilePath" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label>Tamaño del Archivo MDF (MB)</label>
        <input type="number" asp-for="DataFileSizeMB" class="form-control" required min="1" />
        <span asp-validation-for="DataFileSizeMB" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label>Ruta del Archivo LDF</label>
        <input type="text" asp-for="LogFilePath" class="form-control" required />
        <span asp-validation-for="LogFilePath" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label>Tamaño del Archivo LDF (MB)</label>
        <input type="number" asp-for="LogFileSizeMB" class="form-control" required min="1" />
        <span asp-validation-for="LogFileSizeMB" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label>Crecimiento del Archivo MDF (%)</label>
        <input type="text" asp-for="DataFileGrowth" class="form-control" required pattern="^[1-9][0-9]*%$" />
        <span asp-validation-for="DataFileGrowth" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label>Crecimiento del Archivo LDF (%)</label>
        <input type="text" asp-for="LogFileGrowth" class="form-control" required pattern="^[1-9][0-9]*%$" />
        <span asp-validation-for="LogFileGrowth" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label>Tamaño Máximo del Archivo MDF (MB)</label>
        <input type="number" asp-for="DataFileMaxSize" class="form-control" required min="1" />
        <span asp-validation-for="DataFileMaxSize" class="text-danger"></span>
    </div>

    <br />
    <button type="submit" class="btn btn-primary">Crear</button>
    <br /> <br />
</form>
```

En esta vista temenos el formulario con HTML y Bootstrap para introducir los respectivos datos a insertar en el formulario de la creación de la base de datos.

```
<h3>Bases de Datos Existentes</h3>
<ul class="list-group">
    @foreach (var db in ViewBag.Databases as IEnumerable<string>)
    {
        <li class="list-group-item">@db</li>
    }
</ul>
```

De igual forma hacemos llamado a nuestra lista de bases de datos ya insertadas para poder visualizarlas en una lista ordenada debajo de nuestro formulario de creación de base de datos.

3. Dificultades y Soluciones Aplicadas

Durante el desarrollo del proyecto, se presentaron algunos desafíos técnicos y funcionales que fueron resueltos con estrategias adecuadas:

Error en la validación de rutas de archivos MDF y LDF en el procedimiento almacenado

- Inicialmente, se implementó una validación que impedía que el procedimiento aceptara rutas de archivos. Sin embargo, esto causó problemas, ya que se rechazaban rutas que en realidad eran válidas en SQL Server.
- **Solución:** Opté por quitar la validación del stored procedure ya que validaba que esa ruta exista en la computadora pero traía error al tratar de crear la base, entonces mejor la eliminé.

La aplicación web mostraba un error al crear una base de datos, aunque sí se creaba en SQL Server

- Cuando un usuario intentaba crear una base de datos desde la interfaz web, se mostraba un mensaje de error, pero al verificar en SQL Server, la base de datos se había creado correctamente.
- **Solución:** Se ajustó la lógica de manejo de errores en el backend para diferenciar entre errores críticos y advertencias. Ahora, si la base de datos se crea correctamente, el sistema muestra un mensaje de éxito en lugar de un error.