

# Documentación Técnica: Sistema de Gestión de Usuarios

Autor: Jhon Sebastian Bulla Rojas

## Introducción

El Sistema de Gestión de Usuarios es una aplicación web desarrollada utilizando ASP.NET MVC y ASP.NET Web API. Su propósito principal es gestionar usuarios, permitiendo el registro, edición, eliminación y visualización de la información de los mismos. Esta documentación técnica proporciona una visión detallada de los aspectos técnicos y de diseño del sistema, así como una guía para comprender su funcionamiento y configuración.

## Objetivo del Proyecto

El objetivo principal del proyecto es desarrollar un sistema robusto y escalable que permita a los usuarios registrarse en la plataforma, editar su información personal y ser eliminados en caso necesario. Además, se busca proporcionar una interfaz de programación de aplicaciones (API) REST que permita a otras aplicaciones interactuar con la base de datos de usuarios a través de operaciones CRUD (Crear, Leer, Actualizar, Eliminar).

## Requerimientos Funcionales

El desarrollo del Sistema de Gestión de Usuarios incluye la implementación de las siguientes funcionalidades:

- Registro de Usuarios:** Desarrollar una página de registro que permita a los usuarios introducir su nombre, correo electrónico y contraseña para crear una cuenta en el sistema.
- Listado de Usuarios:** Implementar una página que muestre un listado completo de todos los usuarios registrados en el sistema.
- Edición de Usuarios:** Permitir a los usuarios editar su información personal, incluyendo su nombre, correo electrónico y contraseña.
- Eliminación de Usuarios:** Implementar la funcionalidad para eliminar usuarios de forma permanente del sistema.
- API REST:** Crear una API REST que proporcione endpoints para realizar operaciones CRUD en la base de datos de usuarios, permitiendo a otras aplicaciones interactuar con el sistema de forma programática.

## Requerimientos Técnicos

El desarrollo del Sistema de Gestión de Usuarios se basa en los siguientes requerimientos técnicos:

1. **ASP.NET MVC:** Utilización del framework ASP.NET MVC para la implementación de las páginas web del sistema.
2. **ASP.NET Web API:** Implementación de una API REST utilizando ASP.NET Web API para permitir la interacción con la base de datos de usuarios.
3. **SQL Server y ORM:** Utilización de SQL Server para el almacenamiento de la información de los usuarios en base de datos y la configuración de un ORM (Object Relational Mapper) para facilitar las operaciones de persistencia de datos.
4. **Patrones de Diseño:** Aplicación de al menos un patrón de diseño en la implementación del sistema para mejorar su mantenibilidad y escalabilidad.

## Decisiones de diseño

El proyecto se desarrolló en base a la siguiente configuración:

**Framework:** El proyecto se desarrolló utilizando el framework ASP.NET MVC en la versión .NET 8.0 como el framework principal.

**Motor de Base de Datos:** El proyecto utiliza SQL Server 2022 como el motor de base de datos para almacenar la información de los usuarios.

**ORM:** El proyecto utiliza como ORM (Object Relational Mapper) a Entity Framework Core para el mapeo y la interacción del proyecto con la base de datos SQL Server y realizar las operaciones CRUD en la tabla de usuarios. Este se seleccionó por su facilidad de uso y su compatibilidad con el Framework .NET.

- **Versión Específica:** Entity Framework Core 8.0.4

**Paquetes NuGet adicionales:** Además del paquete de Entity Framework Core, se utilizó un paquete adicional relacionado a Entity Framework.

- **Microsoft.EntityFrameworkCore.SqlServer 8.0.4:** Esta es una biblioteca proporciona el proveedor de SQL Server para Entity Framework Core, la cual permite la comunicación con la base de datos SQL Server.

**Arquitectura:** Para el desarrollo del Sistema de Gestión de Usuarios se implementó el patrón de arquitectura Modelo-Vista-Controlador (MVC). Este patrón permite separar las responsabilidades de la aplicación en tres componentes o capas principales:

- **Modelo (Model):** Es la capa encargada de representar los datos y la lógica del negocio, en esta capa se definen las clases que representan las entidades y las operaciones que se pueden realizar sobre ellas.

- **Vista (View):** Es la capa encargada de mostrar la interfaz de usuario al usuario final, en esta capa se definen las páginas web, las plantillas HTML y los elementos de la interfaz gráfica que el usuario puede interactuar.
- **Controlador (Controller):** Es la capa encargada de gestionar las solicitudes del usuario y coordinar la interacción entre el Modelo y la Vista. En esta capa se definen los métodos que responden a las solicitudes HTTP, procesan la información recibida, interactúan con el Modelo y seleccionan la Vista adecuada para mostrar al usuario.

**Patrones de Diseño:** Dejando de lado el patrón de arquitectura MVC, dentro del proyecto se implementaron los siguientes patrones de diseño con el fin de mejorar la eficiencia y eficacia del mismo:

- **Repositorio (Repository):** Este patrón se utiliza para separar la lógica de negocio de la lógica de acceso a datos en la aplicación, de modo que actúa como una capa intermedia entre la lógica de la aplicación y el almacenamiento de datos. Este patrón genera una abstracción del acceso a datos lo cual facilita la interacción a través de los repositorios con métodos mas simples.
- **DTO (Data Transfer Object):** Este patrón se utiliza para transferir datos entre las capas de la aplicación de manera estructurada y eficiente, funcionando de la misma manera entre el servidor y el cliente para el caso del API REST, también permite separar o desacoplar las capas de la aplicación de modo que los cambios que se realicen en la estructura de las entidades u objetos de dominio no afectaran la capa ni la lógica de presentación. Además, este patrón permite controlar que datos se exponen y como se expondrán en la capa de presentación, proporcionando una mayor seguridad y control sobre los mismos, también ayudan a reducir la carga en el servidor al evitar el procesamiento de datos innecesarios que podrían estar presentes en los objetos de dominio.
- **Inyección de dependencias:** Este patrón se utiliza para lograr un acoplamiento más bajo entre las clases y facilitar el modularidad, la reutilización y las pruebas unitarias en el proyecto. Este patrón permite desacoplar los componentes eliminando las dependencias directas entre componentes facilitando las modificaciones y reutilización de los componentes individualmente, lo que hace que estas sean más flexibles y fáciles de mantener.

**API REST:** Se desarrolló una API REST utilizando el framework ASP.NET Web API para realizar operaciones CRUD en la base de datos de usuarios.

**Reutilización de código:** El proyecto de Web API se desarrolló utilizando el proyecto MVC como referencia y base de código existente en la misma solución, de modo que se aprovecharon las funcionalidades y estructuras existentes del proyecto MVC para agilizar el desarrollo del proyecto Web API, de modo que, se reutilizaron ciertos componentes, como modelos de datos, lógica de negocio y validaciones (Modelo, Repositorios, Interfaces, contexto de conexión a base de datos), del proyecto MVC en el proyecto Web API para mantener consistencia y evitar duplicación de código. Esto con el fin de no duplicar código que puede ser reutilizado fácilmente dentro del proyecto de Web API.

## Creación de la base de datos

Para la creación de la Base de datos se deben ejecutar los siguientes scripts en SQL Server.

### Base de datos: UsersDB

```
USE [master]
GO

CREATE DATABASE [UsersDB]
    CONTAINMENT = NONE
    ON PRIMARY
    ( NAME = N'UsersDB',
      FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\UsersDB.mdf',
      SIZE = 8192KB,
      MAXSIZE = UNLIMITED,
      FILEGROWTH = 65536KB)
    LOG ON
    ( NAME = N'UsersDB_log',
      FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\UsersDB_log.ldf' ,
      SIZE = 8192KB ,
      MAXSIZE = 2048GB ,
      FILEGROWTH = 65536KB )
    WITH CATALOG_COLLATION = DATABASE_DEFAULT, LEDGER = OFF
GO
```

### Tabla: Users

```
USE [UsersDB]
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Users](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Name] [varchar](100) NULL,
    [Email] [varchar](100) NULL,
    [Password] [varchar](500) NULL,
    CONSTRAINT [PK_Users] PRIMARY KEY CLUSTERED
    (
        [Id] ASC
    )WITH (PAD_INDEX = OFF,
    STATISTICS_NORECOMPUTE = OFF,
    IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON,
    ALLOW_PAGE_LOCKS = ON,
    OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

## Configuración del ORM (Entity Framework)

Para el Sistema de Gestión de Usuarios se utilizó Entity Framework Core 8.0.4 como ORM para la interacción con la base de datos utilizando la siguiente configuración:

**Cadena de conexión:** Puesto que Entity Framework utiliza una cadena de conexión para establecer la conexión con la base de datos. Se especifica la cadena de conexión en el archivo de configuración `appsettings.json` tanto del proyecto MVC como del proyecto Web API.

```
{
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "Connection": "Server=localhost; Database=UsersDB; Integrated Security=True; TrustServerCertificate=True;"
  }
}
```

**DbContext:** Para la configuración de Entity Framework se definió la clase `ModelContext.cs` que hereda de `DbContext` y define los conjuntos de entidades y las configuraciones de mapeo para la base de datos. `DbContext` que representa el contexto de base de datos y contiene el `DbSet` para la entidad `Users`, la cual es la que se quiere mapear de las tablas de la base de datos. Y un `DbSet` para la entidad `UserDto`.

```
using Microsoft.EntityFrameworkCore;
using SistemaGestionUsuarios.Models;
using SistemaGestionUsuarios.Models.DTO;

namespace SistemaGestionUsuarios.Data
{
    /// <summary> Contexto de base de datos para el sistema de gestión de usuarios.
    6 referencias
    public class ModelContext : DbContext
    {
        /// <summary> Constructor de la clase ModelContext.
        0 referencias
        public ModelContext(DbContextOptions<ModelContext> options) : base(options)
        {
        }

        /// <summary> Conjunto de entidades de usuarios en la base de datos.
        6 referencias
        public virtual DbSet<User> Users { get; set; }

        /// <summary> Representa el conjunto de entidades de tipo UserDto en el contexto ...
        0 referencias
        public DbSet<SistemaGestionUsuarios.Models.DTO.UserDto> UserDto { get; set; } = default!;
    }
}
```

**Configuración de DbContext:** Desde el archivo `Program.cs`, se configuró el `DbContext` para que pueda ser inyectado automáticamente en otras partes de tu aplicación que lo necesiten, como en tus controladores o clases de repositorio. Esto se realizó a través de la función `AddDbContext`, donde también se relaciona con la cadena de conexión proporcionada.

```
//Configuración de la cadena de conexión
builder.Services.AddDbContext<ModelContext>(options => options.UseSqlServer(builder.Configuration.GetConnectionString("Connection")));
```

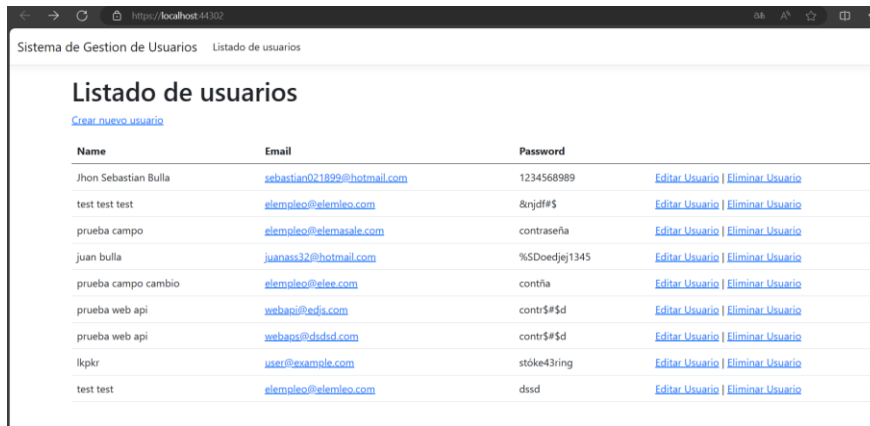
## Guía de uso y Pruebas funcionales

A continuación, se presentan la guía para el funcionamiento y evidencia de pruebas funcionales de los métodos de los proyectos MVC y Web API.

### MVC

#### 1. Listado de Usuarios

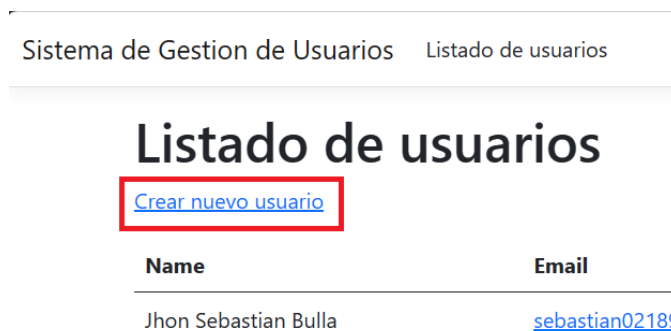
- Este se muestra por defecto como la página de inicio al ejecutar la aplicación.



Name	Email	Password	
Jhon Sebastian Bulla	<a href="mailto:sebastian021899@hotmail.com">sebastian021899@hotmail.com</a>	1234568989	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
test test test	<a href="mailto:ejemplo@ejemplo.com">ejemplo@ejemplo.com</a>	&njdf#\$	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba campo	<a href="mailto:ejemplo@ejemasa.com">ejemplo@ejemasa.com</a>	contraseña	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
juan bulla	<a href="mailto:juanas32@hotmail.com">juanas32@hotmail.com</a>	%SDoejdje1345	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba campo cambio	<a href="mailto:ejemplo@ejee.com">ejemplo@ejee.com</a>	contra	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba web api	<a href="mailto:webapi@edjs.com">webapi@edjs.com</a>	contr\$#\$d	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba web api	<a href="mailto:webapi@dudsd.com">webapi@dudsd.com</a>	contr\$#\$d	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
lkpr	<a href="mailto:user@example.com">user@example.com</a>	stoke43ring	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
test test	<a href="mailto:ejemplo@ejemplo.com">ejemplo@ejemplo.com</a>	dssd	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>

#### 2. Registro de Usuarios

- Desde la página de Inicio/Listado de usuarios seleccionar la opción “**Crear nuevo usuario**” en la parte superior izquierda bajo el título “**Listado de usuarios**”



- Completa los campos requeridos, como nombre, correo electrónico y contraseña.

Sistema de Gestion de Usuarios Listado de usuarios

## Crear Usuario

### Usuario

Nombre  
usuario nuevo prueba

Correo  
correoprueba@yopmail.com

Contraseña  
PasswordPrueba

[Crear usuario](#) | [Regresar al inicio](#)

- Haz clic en el botón Crear usuario para guardar el nuevo usuario.

Sistema de Gestion de Usuarios Listado de usuarios

## Crear Usuario

### Usuario

Nombre  
usuario nuevo prueba

Correo  
correoprueba@yopmail.com

Contraseña  
PasswordPrueba

[Crear usuario](#) | [Regresar al inicio](#)

- El paso anterior retornará a la página de inicio en donde se podrá validar el nuevo usuario al final de la lista.

test test	<a href="#">empleo@elemleo.com</a>	dssd	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
usuario nuevo prueba	<a href="#">correoprueba@yopmail.com</a>	PasswordPrueba	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>

### 3. Edición de Usuarios

- Desde la página de Inicio/Listado de usuarios seleccionar la opción “Editar Usuario” en la parte derecha frente al usuario que se desea editar.

Listado de usuarios

[Crear nuevo usuario](#)

Name	Email	Password	
Jhon Sebastian Bulla	<a href="#">sebastian021899@hotmail.com</a>	1234568989	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
test test test	<a href="#">elempleo@elemleo.com</a>	&njd#f\$	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba campo	<a href="#">elempleo@elemasale.com</a>	contraseña	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
juan bulla	<a href="#">juanass32@hotmail.com</a>	%SDoejdeJ1345	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba campo cambio	<a href="#">elempleo@elee.com</a>	contha	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba web api	<a href="#">webapi@edjs.com</a>	contr\$#d	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba web api	<a href="#">webaps@dssdsd.com</a>	contr\$#d	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
lkpr	<a href="#">user@example.com</a>	stoke43ring	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
test test	<a href="#">elempleo@elemleo.com</a>	dssd	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
usuario nuevo prueba	<a href="#">correoprueba@yopmail.com</a>	PasswordPrueba	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>

- Edita el o los campos que se desean modificar.

Sistema de Gestion de Usuarios    [Listado de usuarios](#)

## Editar Usuario

Usuario

Nombre

usuario nuevo prueba

Correo

correopruebamodificado@yopmail.com

Contraseña

PasswordPrueba123

Guardar cambios

[Regresar al inicio](#)

- Haz clic en el botón de “Guardar cambios”.

Sistema de Gestion de Usuarios    [Listado de usuarios](#)

## Editar Usuario

Usuario

Nombre

usuario nuevo prueba

Correo

correopruebamodificado@yopmail.com

Contraseña

PasswordPrueba123

Guardar cambios

[Regresar al inicio](#)

- El paso anterior retornará a la página de inicio en donde se podrá validar el ajuste del usuario seleccionado.



Listado de usuarios

[Crear nuevo usuario](#)

Name	Email	Password	
Jhon Sebastian Bulla	<a href="#">sebastian021899@hotmail.com</a>	1234568989	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
test test test	<a href="#">elempleo@elemleo.com</a>	&njdf#\$	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba campo	<a href="#">elempleo@elemasale.com</a>	contraseña	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
juan bulla	<a href="#">juanass32@hotmail.com</a>	%SDoedjej1345	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba campo cambio	<a href="#">elempleo@elee.com</a>	contña	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba web api	<a href="#">webapi@edjs.com</a>	contr\$#\$d	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba web api	<a href="#">webapi@dstdsd.com</a>	contr\$#\$d	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
lkpkr	<a href="#">user@example.com</a>	støke43ring	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
test test	<a href="#">elempleo@elemleo.com</a>	dssd	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
usuario nuevo prueba	<a href="#">correopruebamodificado@yopmail.com</a>	PasswordPrueba123	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>

4. Eliminación de Usuarios

- Desde la página de Inicio/Listado de usuarios seleccionar la opción “Eliminar Usuario” en la parte derecha frente al usuario que se desea eliminar.

Listado de usuarios

[Crear nuevo usuario](#)

Name	Email	Password	
Jhon Sebastian Bulla	<a href="#">sebastian021899@hotmail.com</a>	1234568989	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
test test test	<a href="#">elempleo@elemleo.com</a>	&njdf#\$	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba campo	<a href="#">elempleo@elemasale.com</a>	contraseña	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
juan bulla	<a href="#">juanass32@hotmail.com</a>	%SDoedjej1345	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba campo cambio	<a href="#">elempleo@elee.com</a>	contña	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba web api	<a href="#">webapi@edjs.com</a>	contr\$#\$d	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba web api	<a href="#">webapi@dstdsd.com</a>	contr\$#\$d	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
lkpkr	<a href="#">user@example.com</a>	støke43ring	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
test test	<a href="#">elempleo@elemleo.com</a>	dssd	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
usuario nuevo prueba	<a href="#">correopruebamodificado@yopmail.com</a>	PasswordPrueba123	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
usuario prueba creacion	<a href="#">correopruebacreacion@yopmail.com</a>	contraseña2024	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>

- Haz clic en el botón de “Eliminar usuario”.

Sistema de Gestion de Usuarios    Listado de usuarios

Eliminar Usuario

Esta seguro que desea eliminar al usuario?  
Usuario

Nombre                    usuario nuevo prueba  
Correo                    [correopruebamodificado@yopmail.com](#)  
Contraeña                PasswordPrueba123

[Eliminar usuario](#)    [Regresar al inicio](#)

- El paso anterior retornará a la página de inicio en donde se podrá validar la eliminación del usuario seleccionado.

### Listado de usuarios

[Crear nuevo usuario](#)

Name	Email	Password	
Jhon Sebastian Bulla	sebastian021899@hotmail.com	1234568989	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
test test test	empleo@empleo.com	&njdf#\$	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba campo	empleo@emasale.com	contraseña	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
juan bulla	juanass32@hotmail.com	%SDoeDje1345	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba campo cambio	empleo@alee.com	conña	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba web api	webapi@ads.com	contr\$#5d	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
prueba web api	webapi@dsd.com	contr\$#5d	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
lpgkr	user@example.com	stoke43ring	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
test test	empleo@empleo.com	dsd	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>
usuario prueba creacion	consoquebacreacion@yoomail.com	contraseña2024	<a href="#">Editar Usuario</a>   <a href="#">Eliminar Usuario</a>

## Web API

### 1. Obtener todos los usuarios

- **Descripción:** Este método obtiene todos los usuarios registrados en el sistema.
- **Método HTTP:** GET
- **Ruta:** /User
- **Parámetros de consulta:** Ninguno
- **Respuestas:**
  - 200 OK: Retorna una lista de usuarios en formato JSON.
  - 500 Internal Server Error: Se produce un error al intentar obtener los usuarios.
- **Ejemplo de solicitud:**

```
curl -X 'GET' \
  'https://localhost:7164/User' \
  -H 'accept: */*'
```

- **Ejemplo de respuesta (200 OK):**

```
[
  {
    "id": 1,
    "name": "Jhon Sebastian Bulla",
    "email": "sebastian021899@hotmail.com",
    "password": "1234568989"
  },
  {
    "id": 7,
    "name": "test test test",
    "email": "empleo@empleo.com",
    "password": "&njdf#$"
  },
  {
    "id": 11,
    "name": "prueba campo ",
    "email": "empleo@emasale.com",
    "password": "contraseña"
  }
]
```

### 2. Obtener usuario por Id

- **Descripción:** Este método obtiene un usuario específico por su Id.
- **Método HTTP:** GET
- **Ruta:** /User/{Id}

- **Parámetros de consulta:** Id (int): El Id del usuario que se desea obtener.
- **Respuestas:**
  - 200 OK: Retorna la información del usuario correspondiente al Id en formato JSON.
  - 404 Not Found: No se encontró ningún usuario por el Id especificado.
  - 500 Internal Server Error: Se produce un error al intentar obtener al usuario.
- **Ejemplo de solicitud:**

```
curl -X 'GET' \
  'https://localhost:7164/User/1' \
  -H 'accept: */*'
```

- **Ejemplo de respuesta (200 OK):**

```
{
  "id": 1,
  "name": "Jhon Sebastian Bulla",
  "email": "sebastian021899@hotmail.com",
  "password": "1234568989"
}
```

- **Ejemplo de respuesta (404 Not Found):**

```
No se encontró el usuario: 2
```

### 3. Crear nuevo usuario

- **Descripción:** Este método inserta un nuevo usuario.
- **Método HTTP:** POST
- **Ruta:** /User/
- **Parámetros de consulta:**
  - name (string): Nombre del usuario nuevo a insertar
  - email (email): Email del usuario nuevo a insertar
  - password (string): Contraseña del usuario nuevo a insertar
- **Respuestas:**
  - 200 OK: Retorna mensaje “¡El usuario se creó exitosamente!”.
  - 500 Internal Server Error: Se produce un error al intentar insertar al usuario. Retorna mensaje “Error al crear el usuario.”
- **Ejemplo de solicitud:**

```
curl -X 'POST' \
  'https://localhost:7164/User' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 0,
    "name": "usuario web api",
    "email": "userapi@example.com",
    "password": "$#contraapi123$"
  }'
```

- **Ejemplo de respuesta (200 OK):**

```
¡El usuario se creó exitosamente!
```

- **Validación de creación del nuevo usuario (método para obtener usuario):**

```
{
  "id": 21,
  "name": "usuario web api",
  "email": "userapi@example.com",
  "password": "$#contraapi123$"
}
```

#### 4. Editar usuario

- **Descripción:** Este método edita la información de un usuario específico por su Id.
- **Método HTTP:** PUT
- **Ruta:** /User/{Id}
- **Parámetros de consulta:**
  - id (int): Id del usuario a editar
  - name (string): Nombre del usuario a editar
  - email (email): Email del usuario a editar
  - password (string): Contraseña del usuario a editar
- **Respuestas:**
  - 200 OK: Retorna mensaje "¡El usuario se actualizó exitosamente!".
  - 404 Not Found: No se encontró ningún usuario por el Id especificado. Retorna mensaje "No se encontró el usuario + id"
  - 500 Internal Server Error: Se produce un error al intentar insertar al usuario. Retorna mensaje "Error al actualizar el usuario."
- **Ejemplo de solicitud:**

```
curl -X 'PUT' \
  'https://localhost:7164/User/21' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 21,
    "name": "usuario web api editado",
    "email": "userapiedit@example.com",
    "password": "$#contraapi123$2"
  }'
```

- **Ejemplo de respuesta (200 OK):**

```
¡El usuario se actualizó exitosamente!
```

- **Ejemplo de respuesta (404 Not Found):**

```
No se encontró el usuario: 2
```

- **Validación de actualización del usuario (método para obtener usuario):**

```
{
  "id": 21,
  "name": "usuario web api editado",
  "email": "userapiedit@example.com",
  "password": "$#contraapi123$2"
}
```

#### 5. Eliminar usuario

- **Descripción:** Este método elimina un usuario específico por su Id.

- **Método HTTP:** DELETE
- **Ruta:** /User/{Id}
- **Parámetros de consulta:**
  - id (int): Id del usuario a eliminar
- **Respuestas:**
  - 200 OK: Retorna mensaje "¡El usuario se eliminó exitosamente!".
  - 404 Not Found: No se encontró ningún usuario por el Id especificado. Retorna mensaje "No se encontró el usuario + id"
  - 500 Internal Server Error: Se produce un error al intentar eliminar al usuario. Retorna mensaje "Error al eliminar al usuario."
- **Ejemplo de solicitud:**

```
curl -X 'DELETE' \
'https://localhost:7164/User/18' \
-H 'accept: */*'
```

- **Ejemplo de respuesta (200 OK):**

```
¡El usuario se eliminó exitosamente!
```

- **Ejemplo de respuesta (404 Not Found):**

```
No se encontró el usuario: 18
```

- **Validación de eliminación del usuario (método para obtener usuario):**

```
{
  "id": 17,
  "name": "lkpkr",
  "email": "user@example.com",
  "password": "stóke43ring"
},
{
  "id": 20,
  "name": "usuario prueba creacion",
  "email": "correopruebacreacion@yopmail.com",
  "password": "contraseña2024"
},
{
  "id": 21,
  "name": "usuario web api editado",
  "email": "userapiedit@example.com",
  "password": "$#contraapi123$2"
}
```