



UNIVERSIDAD DE GUAYAQUIL  
Facultad de Ciencias Matemáticas  
Carrera de Software

[www.ug.edu.ec](http://www.ug.edu.ec)  
Guayaquil - Ecuador

# **Manual Técnico De Las Api Y Procedimientos Almacenados**

## ***Grupo 3***

Maximiliano Cabrera Gamboa

Odalis Vera García

Eduardo Vera Romero

## ***Curso y paralelo***

SOF-S-MA-7-1

## ***Docente***

Ing. Jorge Luis Charco

# EVIDENCIA DE APROBACION



← Responder « Responder a todos → Reenviar ✉ Archivar trash Eliminar ...

## RE: DAWA 7-1



Jorge Charco Aguirre <jorge.charcoa@ug.edu.ec>

19/5/2023 9:37

Para: ODALIS VERA GARCIA

Aprobado.

Saludos,

Ph.D. Jorge Luis Charco Aguirre  
Docente  
Facultad de Ciencias Matemáticas y Físicas

Enviado desde Correo para Windows

---

**De:** ODALIS VERA GARCIA

**Enviado:** miércoles, 17 de mayo de 2023 17:46

**Para:** Jorge Charco Aguirre

**Asunto:** RE: DAWA 7-1

Estimado Ing.

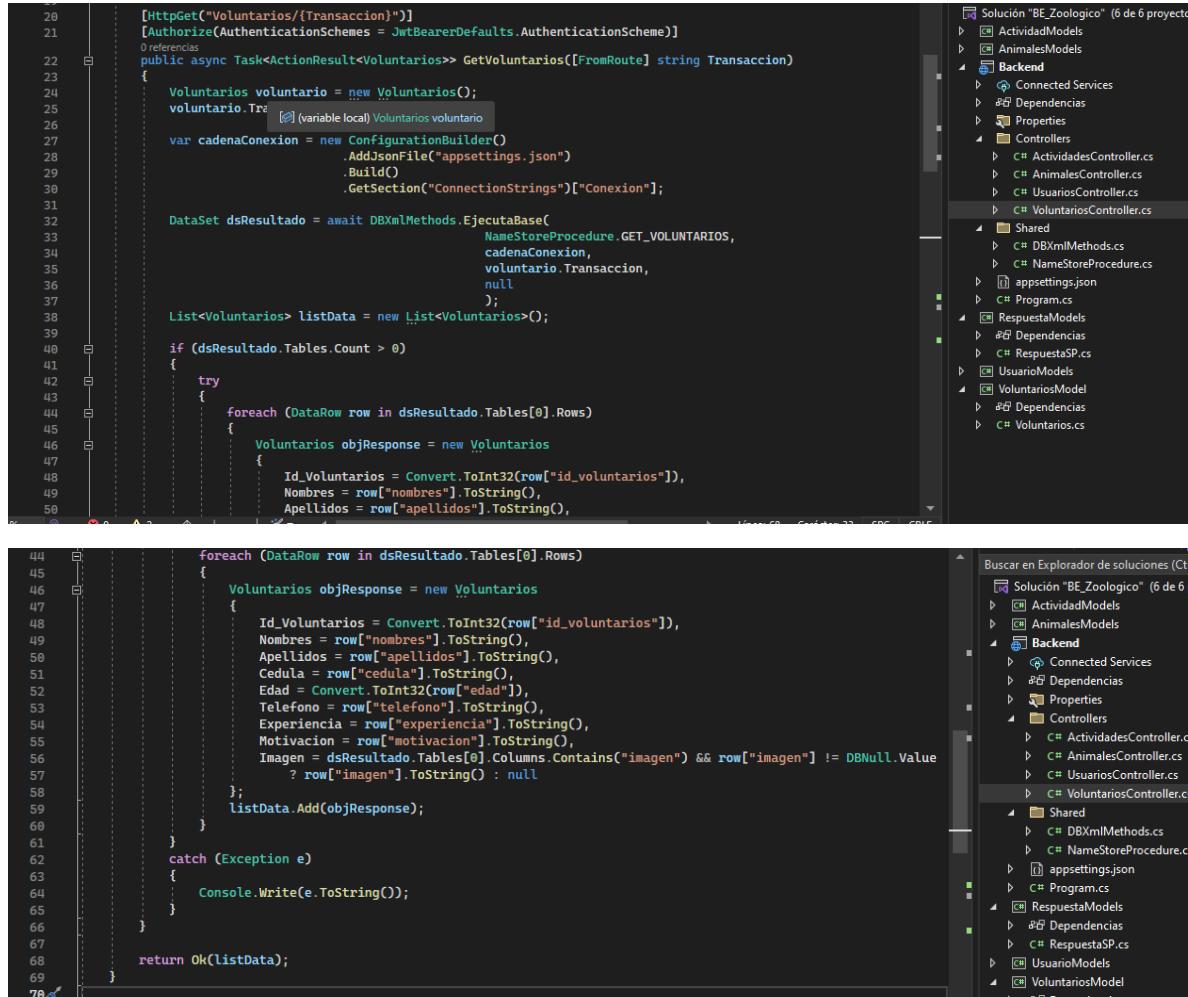
Adjunto el pdf solicitado con la corrección de la propuesta de proyecto.

Quedo al pendiente.

Enviado desde Correo para Windows

## 1. Apis del módulo voluntarios

- Api para obtener los voluntarios registrados en nuestra base de datos



```
20 [HttpGet("Voluntarios/{Transaccion}")]
21 [Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
22 public async Task<ActionResult<Voluntarios>> GetVoluntarios([FromRoute] string Transaccion)
23 {
24     Voluntarios voluntario = new Voluntarios();
25     voluntario.Trx = (variable local) Voluntarios voluntario
26     var cadenaConexion = new ConfigurationBuilder()
27         .AddJsonFile("appsettings.json")
28         .Build()
29         .GetSection("ConnectionString")["Conexion"];
30
31     DataSet dsResultado = await DBXmlMethods.EjecutaBase(
32         NameStoreProcedure.GET_VOLUNTARIOS,
33         cadenaConexion,
34         voluntario.Transaccion,
35         null
36         );
37
38     List<Voluntarios> listaData = new List<Voluntarios>();
39
40     if (dsResultado.Tables.Count > 0)
41     {
42         try
43         {
44             foreach (DataRow row in dsResultado.Tables[0].Rows)
45             {
46                 Voluntarios objResponse = new Voluntarios
47                 {
48                     Id_Voluntarios = Convert.ToInt32(row["id_voluntarios"]),
49                     Nombres = row["nombres"].ToString(),
50                     Apellidos = row["apellidos"].ToString(),
51                     Cedula = row["cedula"].ToString(),
52                     Edad = Convert.ToInt32(row["edad"]),
53                     Telefono = row["telefono"].ToString(),
54                     Experiencia = row["experiencia"].ToString(),
55                     Motivacion = row["motivacion"].ToString(),
56                     Imagen = dsResultado.Tables[0].Columns.Contains("Imagen") && row["Imagen"] != DBNull.Value
57                         ? row["Imagen"].ToString() : null
58                 };
59                 listaData.Add(objResponse);
60             }
61         }
62         catch (Exception e)
63         {
64             Console.WriteLine(e.ToString());
65         }
66     }
67
68     return Ok(listaData);
69 }
70 }
```

The code editor shows two versions of the same API method. The first version is more verbose, using a try-catch block and a foreach loop to iterate through the dataset rows. The second version is a simplified one-liner using LINQ's SelectMany method to achieve the same result. Both versions include annotations for the HTTP GET endpoint and authentication scheme.

The Solution Explorer on the right shows the project structure for 'BE\_Zoologico', which includes 'Backend' and 'Shared' projects, along with various models and controllers.

Este api realiza recibe una transacción, el api se conecta a la base de datos y realiza una consulta a un procedimiento almacenado y este nos devuelve los registros de todos los voluntarios o de las solicitud de voluntarios del sistema como podemos visualizar usamos `HttpGet` para recibir dichos datos.

- Api que permite Registrar, actualizar y eliminar

```
[HttpPost("Voluntarios")]
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
0 referencias
public async Task<ActionResult<RespuestaSP>> PostVoluntarios([FromBody] Voluntarios voluntario)
{
    var cadenaConexion = new ConfigurationBuilder()
        .AddJsonFile("appsettings.json")
        .Build()
        .GetSection("ConnectionStrings")["Conexion"];
    XDocument xmlParam = Shared.DBXmlMethods.GetXml(voluntario);
    DataSet dsResultado = await Shared.DBXmlMethods.EjecutaBase(
        NameStoreProcedure.CRUD_VOLUNTARIOS,
        cadenaConexion,
        voluntario.Transaccion,
        xmlParam.ToString());
    RespuestaSP objRespuesta = new RespuestasSP();
    if (dsResultado.Tables.Count > 0)
    {
        try
        {
            objRespuesta.Respuesta = dsResultado.Tables[0].Rows[0]["Respuesta"].ToString();
            objRespuesta.Leyenda = dsResultado.Tables[0].Rows[0]["Leyenda"].ToString();
        }
        catch (Exception e)
        {
            objRespuesta.Respuesta = "Error";
            objRespuesta.Leyenda = "Lo sentimos ha ocurrido un error";
        }
    }
    return Ok(objRespuesta);
}
```

Este api recibe como parámetro los datos de los voluntarios ya sea para insertar, modificar o eliminar algún registro, así mismo se conecta a la base de datos la cual tiene un procedimiento almacenado que se llama **CRUD\_VOLUNTARIOS** en donde debemos especificar una transacción de lo que quiera realizar el usuario de los ya antes mencionado

- Requisitos

Para poder hacer uso de las api debemos tener una versión de .Net actualizada para este caso las api están siendo compatible con .Net 6.0

- Instalación Y configuración

Debemos realizar la conexión al servicio de la base de datos sql

Debemos instalar los paquetes NuGet que nos servirán para el uso de las api son los siguientes:

```
Install-package Microsoft.AspNetCore.Authentication.JwtBearer;
Install-Package System.Data.SqlClient;
```

- Autenticación y Autorización

Para hacer uso de las api debemos tener una autorización que se realiza mediante token de seguridad, una vez obtenido este token podemos hacer uso de las mismas para que las api tenga dicha seguridad debemos colocar la siguiente línea en nuestro código de .net

```
[HttpGet("Voluntarios/{Transaccion}")]  
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]  
0 referencias  
public async Task<ActionResult<Voluntarios>> GetVoluntarios([FromRoute] string Transaccion)...
```

La línea que debemos poner es la segunda de Authorize.

## 2. Procedimientos almacenados del Módulo Voluntario

- Consultar Voluntarios

```
IF(@iTransaccion = 'CONSULTA_VOLUNTARIOS')  
BEGIN  
    SELECT v.Id_Voluntarios AS id_voluntarios,  
          v.Nombres AS nombres,  
          v.Apellidos AS apellidos,  
          v.Cedula AS cedula,  
          v.Edad AS edad,  
          v.Telefono AS telefono,  
          v.Experiencia AS experiencia,  
          v.Motivacion AS motivacion,  
          v.Imagen AS imagen  
    FROM voluntarios v  
    WHERE v.estado = 1  
    ORDER BY  
        v.id_voluntarios ASC;  
    SET @respuesta = 'OK';  
    SET @leyenda = 'Consulta Exitosa';  
END
```

Este procedimiento nos permite obtener todos los registros de los voluntarios registrados en el sistema

- Consultar Solicitudes

```

IF(@iTransaccion = 'CONSULTA_SOLICITUDES')
BEGIN
    SELECT s.Id                      AS id_voluntarios,
           s.Nombres                  AS nombres,
           s.Apellidos                AS apellidos,
           s.Cedula                   AS cedula,
           s.Edad                     AS edad,
           s.Telefono                 AS telefono,
           s.Experiencia              AS experiencia,
           s.Motivacion               AS motivacion
    FROM   Solicituds s
   WHERE  s.estado = 1
   ORDER BY
          s.id ASC;
   SET @respuesta = 'OK';
   SET @leyenda = 'Consulta Exitosa';
END
END TRY

```

En el sistema tenemos que usuarios que quieran ser parte del zoológico postulen para ser voluntarios del mismo, este procedimiento nos permite visualizar dichas solicitudes.

- Agregar Voluntarios

```

IF(@iTransaccion = 'AGREGAR_VOLUNTARIO')
BEGIN
    SET @Nombres      = (SELECT @iXml.value('/Voluntarios/Nombres')[1],           'VARCHAR(50)')
    SET @Apellidos   = (SELECT @iXml.value('/Voluntarios/Apellidos')[1],           'VARCHAR(50)')
    SET @Cedula      = (SELECT @iXml.value('/Voluntarios/Cedula')[1],            'VARCHAR(10)')
    SET @Edad         = (SELECT @iXml.value('/Voluntarios/Edad')[1],             'INT')
    SET @Telefono     = (SELECT @iXml.value('/Voluntarios/Telefono')[1],           'VARCHAR(10)')
    SET @Experiencia = (SELECT @iXml.value('/Voluntarios/Experiencia')[1],        'VARCHAR(200)')
    SET @Motivacion   = (SELECT @iXml.value('/Voluntarios/Motivacion')[1],        'VARCHAR(200)')
    SET @Imagen       = (SELECT @iXml.value('/Voluntarios/Imagen')[1],           'VARCHAR(100)')

    INSERT INTO voluntarios(Nombres, Apellidos, Cedula, Edad, Telefono, Experiencia, Motivacion, Imagen)
    VALUES (@Nombres, @Apellidos, @Cedula, @Edad, @Telefono, @Experiencia, @Motivacion, @Imagen)

    SET @respuesta = 'Enhорабуена';
    SET @leyenda = 'Voluntario Agregado';
END

```

Este procedimiento nos permite agregar nuevos registros de voluntarios en el sistema

- Actualizar Voluntarios

```

IF(@iTransaccion = 'ACTUALIZAR_VOLUNTARIO')
BEGIN
    SET @Id_Voluntarios      =  (SELECT @iXml.value('/(Voluntarios/Id_Voluntarios)[1]', 'INT'))
    SET @Nombres               =  (SELECT @iXml.value('/(Voluntarios/Nombres)[1]', 'VARCHAR(50)'))
    SET @Apellidos              =  (SELECT @iXml.value('/(Voluntarios/Apellidos)[1]', 'VARCHAR(50)'))
    SET @Cedula                  =  (SELECT @iXml.value('/(Voluntarios/Cedula)[1]', 'VARCHAR(10)'))
    SET @Edad                      =  (SELECT @iXml.value('/(Voluntarios/Edad)[1]', 'INT'))
    SET @Telefono                 =  (SELECT @iXml.value('/(Voluntarios/Telefono)[1]', 'VARCHAR(10)'))
    SET @Experiencia                =  (SELECT @iXml.value('/(Voluntarios/Experiencia)[1]', 'VARCHAR(200)'))
    SET @Motivacion                  =  (SELECT @iXml.value('/(Voluntarios/Motivacion)[1]', 'VARCHAR(200)'))
    SET @Imagen                     =  (SELECT @iXml.value('/(Voluntarios/Imagen)[1]', 'VARCHAR(100)'))

    UPDATE voluntarios SET
        Nombres          =  @Nombres,
        Apellidos         =  @Apellidos,
        Cedula            =  @Cedula,
        Edad              =  @Edad,
        Telefono           =  @Telefono,
        Experiencia        =  @Experiencia,
        Motivacion         =  @Motivacion,
        Imagen             =  @Imagen
    WHERE
        Id_Voluntarios   =  @Id_Voluntarios

    SET @respuesta = 'Enhorabuena';
    SET @leyenda = 'Voluntario Actualizado';
END

```

Este procedimiento nos permite actualizar los registros mediante el id el cual no es modificable, cabe recordar que todos los campos son obligatorios.

- Eliminar Voluntarios

```

IF(@iTransaccion = 'ELIMINAR_VOLUNTARIO')
BEGIN
    SET @Id_Voluntarios =  (SELECT @iXml.value('/(Voluntarios/Id_Voluntarios)[1]', 'INT'))

    UPDATE voluntarios     SET estado = 0
    WHERE Id_Voluntarios  =  @Id_Voluntarios

    SET @respuesta = 'Enhorabuena';
    SET @leyenda = 'Voluntario eliminado';
END

```

Nos permite eliminar voluntarios mediante el Id del registro.

- Agregar Solicitud

```

IF(@iTransaccion = 'AGREGAR_SOLICITUD')
BEGIN
    SET @Nombres      =  (SELECT @iXml.value('/(Voluntarios/Nombres)[1]', 'VARCHAR(50)'))
    SET @Apellidos     =  (SELECT @iXml.value('/(Voluntarios/Apellidos)[1]', 'VARCHAR(50)'))
    SET @Cedula        =  (SELECT @iXml.value('/(Voluntarios/Cedula)[1]', 'VARCHAR(10)'))
    SET @Edad           =  (SELECT @iXml.value('/(Voluntarios/Edad)[1]', 'INT'))
    SET @Telefono       =  (SELECT @iXml.value('/(Voluntarios/Telefono)[1]', 'VARCHAR(10)'))
    SET @Experiencia     =  (SELECT @iXml.value('/(Voluntarios/Experiencia)[1]', 'VARCHAR(200)'))
    SET @Motivacion      =  (SELECT @iXml.value('/(Voluntarios/Motivacion)[1]', 'VARCHAR(200)'))

    INSERT INTO Solicitudes(Nombres, Apellidos, Cedula, Edad, Telefono, Experiencia, Motivacion)
    VALUES (@Nombres, @Apellidos, @Cedula, @Edad, @Telefono, @Experiencia, @Motivacion)

    SET @respuesta = 'Enhorabuena';
    SET @leyenda = 'Solicitud enviada';
END

```

Nos permite agregar las solicitudes al sistema

- Actualizar Voluntarios

```
IF(@iTransaccion = 'ACTUALIZAR_SOLICITUD')
BEGIN
    SET @Id_Voluntarios      = (SELECT @iXml.value('/Voluntarios/Id_Voluntarios')[1],           'INT'))
    SET @Nombres              = (SELECT @iXml.value('/Voluntarios/Nombres')[1],                 'VARCHAR(50)'))
    SET @Apellidos             = (SELECT @iXml.value('/Voluntarios/Apellidos')[1],                'VARCHAR(50)'))
    SET @Cedula               = (SELECT @iXml.value('/Voluntarios/Cedula')[1],                  'VARCHAR(10)'))
    SET @Edad                  = (SELECT @iXml.value('/Voluntarios/Edad')[1],                   'INT'))
    SET @Telefono              = (SELECT @iXml.value('/Voluntarios/Telefono')[1],                 'VARCHAR(10)'))
    SET @Experiencia           = (SELECT @iXml.value('/Voluntarios/Experiencia')[1],            'VARCHAR(200)'))
    SET @Motivacion             = (SELECT @iXml.value('/Voluntarios/Motivacion')[1],            'VARCHAR(200)'))

    UPDATE Solicitudes SET
        Nombres          = @Nombres,
        Apellidos         = @Apellidos,
        Cedula            = @Cedula,
        Edad              = @Edad,
        Telefono          = @Telefono,
        Experiencia       = @Experiencia,
        Motivacion        = @Motivacion
    WHERE
        Id    = @Id_Voluntarios

    SET @respuesta = 'Enhorabuena';
    SET @leyenda = 'Voluntario Actualizado';
END
```

- Aceptar o Rechazar Solicitud

```
IF(@iTransaccion = 'ACEPTAR_SOLICITUD')
BEGIN
    SET @Id_Voluntarios      = (SELECT @iXml.value('/Voluntarios/Id_Voluntarios')[1], 'INT'))

    UPDATE Solicitudes
    SET estado = 0
    WHERE Id     = @Id_Voluntarios

    SET @respuesta = 'Enhorabuena';
    SET @leyenda = 'Solicitud aceptada';
END

IF(@iTransaccion = 'RECHAZAR_SOLICITUD')
BEGIN
    SET @Id_Voluntarios      = (SELECT @iXml.value('/Voluntarios/Id_Voluntarios')[1], 'INT'))

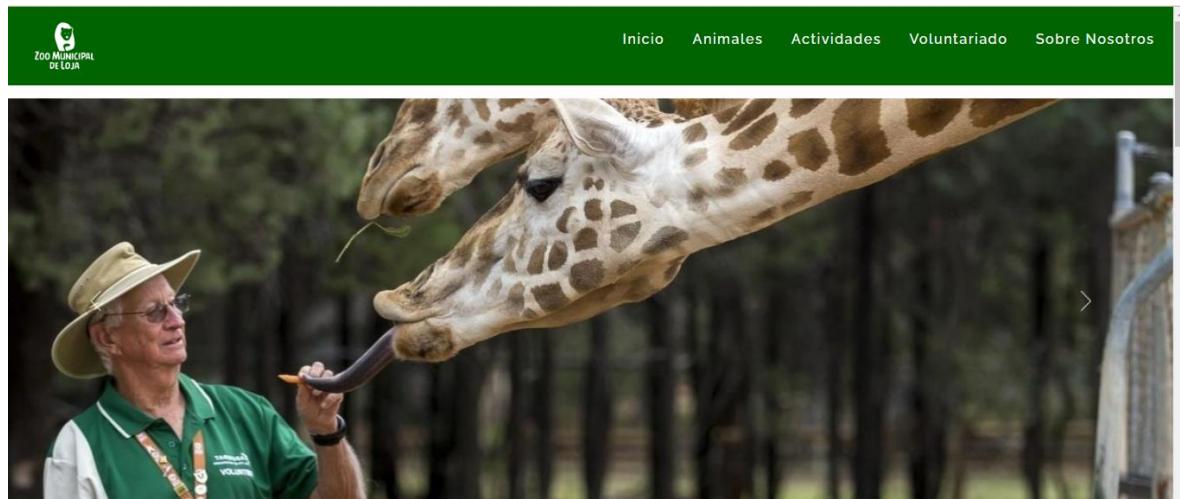
    UPDATE Solicitudes
    SET estado = 0
    WHERE Id     = @Id_Voluntarios

    SET @respuesta = 'Enhorabuena';
    SET @leyenda = 'Solicitud rechazada';
END
```

Una vez corroborados los datos de la solicitud se procede a aceptar a la persona o rechazar dicha solicitud para ello tenemos estos procedimientos que nos permite hacer ambas cosas.

### 3. Ventanas del módulo Voluntario

- Ventana principal del modulo



ZOO MUNICIPAL  
DE LOJA

Inicio Animales Actividades Voluntariado Sobre Nosotros

¡Únete a nuestra familia de voluntarios v sé parte del cambio!

Buscar 10:19 19/08/2023

**Descubre una experiencia inolvidable**

¡Únete a nuestro apasionado equipo de voluntarios y sé parte del esfuerzo por proteger y cuidar de la vida silvestre en nuestro zoológico! Si eres un amante de los animales y anhelas marcar la diferencia en su bienestar, te invitamos a formar parte de nuestra misión de conservación. Como voluntario, tendrás la oportunidad única de interactuar directamente con diversas especies, participar en su alimentación y cuidado diario, y contribuir a la educación y sensibilización de nuestros visitantes. Tu dedicación y entusiasmo serán invaluables para el bienestar de nuestros residentes peludos, emplumados y escamosos. ¡Únete a nosotros y marca una diferencia real en el mundo animal!



**¿Sueñas con trabajar rodeado de animales fascinantes y contribuir a su protección y bienestar?**

¡Entonces, ser voluntario en nuestro zoológico es la oportunidad perfecta para ti! Al unirte a nuestro equipo, te sumergirás en un entorno estimulante y enriquecedor, donde aprenderás de cerca sobre el cuidado y conservación de diversas especies. Tu labor como voluntario abarcará desde participar en actividades de enriquecimiento animal hasta brindar apoyo en programas de reproducción y rehabilitación. Además, formarás parte de una comunidad apasionada de amantes de la naturaleza que comparten tu compromiso y entusiasmo. No importa tu experiencia previa, lo más importante es tu amor por los animales y tu deseo de marcar la diferencia. ¡Únete a nuestro equipo de voluntarios y vive una experiencia única en el cuidado y protección de la vida

Buscar 10:19 19/08/2023

## Conoce a nuestro equipo de voluntarios



María



Angela



Derian



Lady



Erick



Luis



Laura



Luis



Laura

Buscar 10:19  
19/08/2023

"¡Únete a nuestra familia de voluntarios y sé parte del cambio! Tu compromiso y pasión pueden marcar la diferencia en la vida de los animales y en la conservación de nuestro entorno natural."

[¡Inscríbete!](#)

  
**Zoo Municipal de Loja**  
Av. 8 de Diciembre, Loja/Loja-Ecuador  
Horarios:Lunes-Cerrado  
Martes a Domingo gam-spm

La Municipalidad de Loja creó el Zoológico Municipal de Loja en el año 2004 para brindar un espacio de aprendizaje y entretenimiento a personas de todas las edades interesadas en conocer las especies animales del país. Este lugar se encuentra en la ciudad de Loja y es ideal para visitar debido a su belleza natural.

**Personal Autorizado**

- [Animales](#)
- [Voluntariado](#)
- [Actividades](#)

Buscar 10:20  
19/08/2023

- Ventana de solicitud para ser voluntario

**Registro de solicitud de Voluntarios**

Nombres *	Apellidos *
Cedula : *	Edad: *
Telefono : *	Experiencia : *
motivacion : *	<b>Regresar</b>
	<b>Enviar</b>

Zoológico Municipal de Loja en el año 2004 para brindar un espacio de aprendizaje y entretenimiento a personas de todas las edades interesadas en conocer las especies

**Personal Autorizado**

- Animales

10:21  
19/08/2023

- Ventana de administrador del módulo voluntario

**Gestionar Voluntarios**

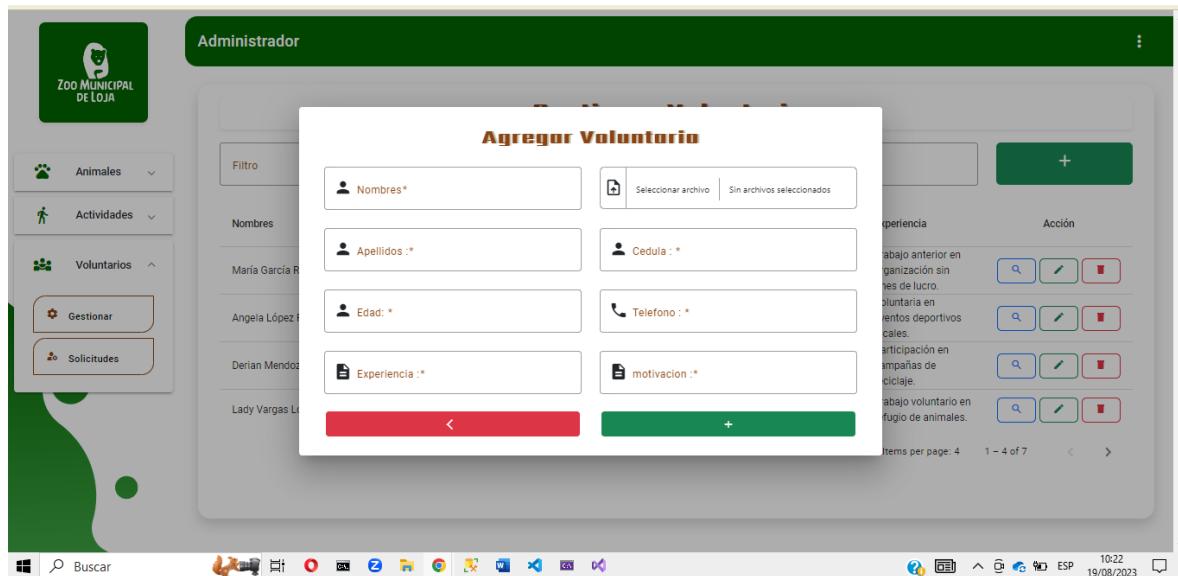
Nombres	Cedula	Edad	Telefono	Experiencia	Acción
Maria Garcia Rodriguez	1712345678	28	0987654321	Trabajo anterior en organización sin fines de lucro.	
Angela Lopez Perez	1723456789	22	0998765432	Voluntaria en eventos deportivos locales.	
Derian Mendoza Gonzalez	1756789012	35	0987123456	Participación en campañas de reciclaje.	
Lady Vargas Lopez	1789012345	29	0976543210	Trabajo voluntario en refugio de animales.	

Items per page: 4    1 ~ 4 of 7

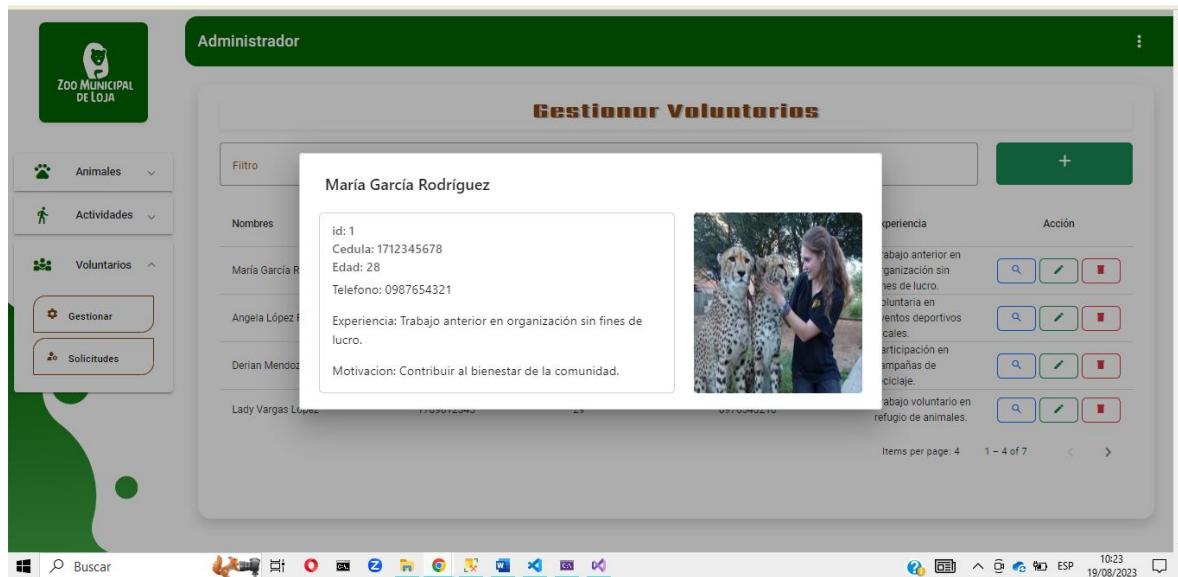
Animales    Actividades    Voluntarios    Gestión    Solicitud

10:22  
19/08/2023

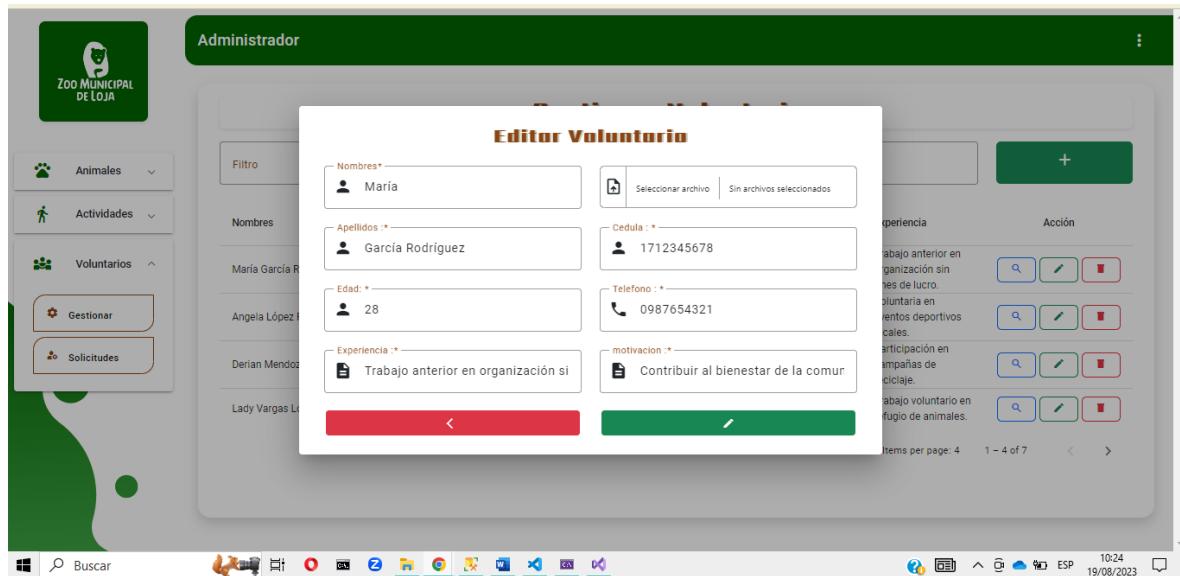
- Ventana para agregar nuevo voluntario



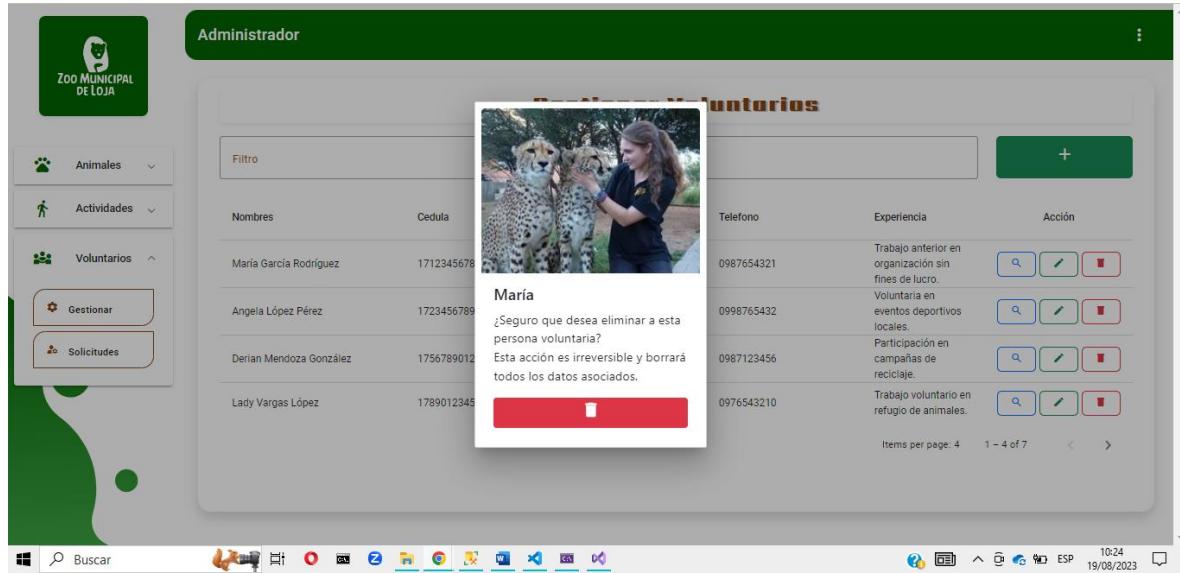
- Ventana para poder ver la información del registro



- Ventana para actualizar registros de voluntarios



- Ventana para eliminar registro de voluntarios



- Ventana de gestión de solicitudes

The screenshot shows a web-based application interface titled "Administrador". The main title is "Gestionar Solicitud de Voluntarios". On the left, there is a sidebar with a logo for "ZOO MUNICIPAL DE LOJA" and navigation links for "Animales", "Actividades", "Voluntarios", "Gestionar", and "Solicitudes". The "Solicitudes" link is highlighted. The main content area displays a table with columns: Nombres, Apellidos, Cedula, Edad, telefono, and Acción. The table contains 5 rows of data. Each row has a set of icons for search, edit, and delete operations. At the bottom right of the table, there is a pagination message: "Items per page: 4 1 - 4 of 7". The status bar at the bottom shows the date and time: "10:25 19/08/2023".

- Ventana para rechazar solicitudes

This screenshot is similar to the previous one, showing the "Gestionar Solicitud de Voluntarios" window. However, a modal dialog box is overlaid on the table. The dialog box contains the text: "¿Seguro que desea Rechazar la solicitud de María? Esta acción es irreversible y borrará todos los datos asociados." Below the text is a large red button with a white trash icon. The rest of the table and interface elements are visible in the background.

- Ventana para aceptar solicitudes

The screenshot shows a web application interface for managing volunteer requests. On the left, there's a sidebar with a logo for 'ZOO MUNICIPAL DE LOJA' and navigation links for 'Animales', 'Actividades', 'Voluntarios' (selected), 'Gestionar', and 'Solicitudes'. The main content area has a title 'Gestionar Solicitud de Voluntarios'. Below it is a table with columns: Nombres, Apellidos, Edad, telefono, and Acción. A modal dialog box is overlaid on the table, containing the message: '¿Seguro que desea aceptar la solicitud de Lady?' (Are you sure you want to accept the request from Lady?) and a note: 'Esta acción es irreversible y borrará todos los datos asociados.' (This action is irreversible and will delete all associated data). At the bottom of the modal is a green button labeled 'ACEPtar' (Accept).

Nombres	Apellidos	Edad	telefono	Acción
Maria	Garcia Rodriguez	28	0987654321	
Angela	Lopez Perez	22	0998765432	
Derian	Mendoza Gonzalez	35	0987123456	
Lady	Vargas Lopez	29	0976543210	

- Ventana para ver la información de la solicitud

This screenshot shows the same application interface as the previous one, but with a different modal dialog. The modal displays detailed information about a specific volunteer request: 'Detalles de la Solicitud'. It lists the following details: Nombres: Maria, apellido: Garcia Rodriguez, cedula: 1712345678, Edad: 28, Telefono: 0987654321, Experiencia: Trabajo anterior en organización sin fines de lucro, and Motivacion: Contribuir al bienestar de la comunidad.

## 4. Apis Modulo Actividades

- Api para las actividades mostradas al usuario.

```
[HttpGet("Actividad/{Transaccion}")]
0 referencias
public async Task<ActionResult<Actividad>> GetActividades([FromRoute] string Transaccion)
{
    Actividad actividad     = new Actividad();
    actividad.Transaccion  = Transaccion;

    var cadenaConexion = new ConfigurationBuilder()
        .AddJsonFile("appsettings.json")
        .Build()
        .GetSection("ConnectionStrings")["Conexion"];

    DataSet dsResultado = await DBXMLMethods.EjecutaBase(
        NameStoreProcedure.GET_ACTIVIDADES,
        cadenaConexion,
        actividad.Transaccion,
        null
    );
    List<Actividad> listaData = new List<Actividad>();

    if (dsResultado.Tables.Count > 0)
    {
        try
        {
            foreach (DataRow row in dsResultado.Tables[0].Rows)
            {
                Actividad objResponse = new Actividad
                {
                    ActividadInformacion =
                    {
                        Id_ActividadInformacion = Convert.ToInt32(row["id_actividad"]),
                        Horario =
                        {
                            Id_Horario = Convert.ToInt32(row["id_hora"]),
                            Hora = row["hora"].ToString()
                        },
                        CantidadPersonas = Convert.ToInt32(row["cantidadPersonas"]),
                        CantidadGuías = Convert.ToInt32(row["cantidadGuías"]),
                        Precio = Convert.ToDecimal(row["precio"]),
                        Descripción = row["descripción"].ToString()
                    },
                    Nombre = row["nombre"].ToString(),
                    Tiempo = row["tiempo"].ToString(),
                    Imagen = row["imagen"].ToString(),
                };
                listaData.Add(objResponse);
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.ToString());
        }
    }

    return Ok(listaData);
}
```

El api recibe una transacción, se conecta a la base de datos y realiza una consulta al procedimiento almacenado el cual nos devuelve los registros de las actividades.

- Api para las actividades personalizadas.

```
[HttpGet("Personalizado/{Transaccion}")]
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
0 referencias
public async Task<ActionResult<Personalizado>> GetPersonalizados([FromRoute] string Transaccion)
{
    Personalizado personalizado      = new Personalizado();
    personalizado.Transaccion       = Transaccion;

    var cadenaConexion = new ConfigurationBuilder()
        .AddJsonFile("appsettings.json")
        .Build()
        .GetSection("ConnectionStrings")["Conexion"];

    DataSet dsResultado = await DBXmlMethods.EjecutaBase(
        NameStoreProcedure.GET_ACTIVIDADES,
        cadenaConexion,
        personalizado.Transaccion,
        null
    );
    List<Personalizado> listaData = new List<Personalizado>();

    if (dsResultado.Tables.Count > 0)
    {
        try
        {
            foreach (DataRow row in dsResultado.Tables[0].Rows)
            {
                Personalizado objResponse      = new Personalizado()
                {
                    ActividadInformacion
                    {
                        Id_ActividadInformacion = Convert.ToInt32(row["id_personalizado"]),
                        Horario
                        {
                            Id_Horario           = Convert.ToInt32(row["id_hora"]),
                            Hora                 = row["hora"].ToString()
                        },
                        CantidadPersonas       = Convert.ToInt32(row["cantidadPersonas"]),
                        CantidadGuias         = Convert.ToInt32(row["cantidadGuias"]),
                        Precio                = Convert.ToDecimal(row["precio"]),
                        Descripcion          = row["descripcion"].ToString()
                    },
                    NombreUsuario          = row["nombreUsuario"].ToString(),
                    Telefono               = row["telefono"].ToString(),
                    Fecha                 = Convert.ToDateTime(row["fecha"].ToString())
                };
                listaData.Add(objResponse);
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.ToString());
        }
    }

    return Ok(listaData);
}
```

El api recibe una transacción, se conecta a la base de datos y realiza una consulta al procedimiento almacenado el cual nos devuelve los registros de las actividades personalizadas que han sido ingresadas por los usuarios.

- Api para los horarios.

```
[HttpGet("Horario/{Transaccion}")]
0 referencias
public async Task<ActionResult<Horario>> GetHorarios([FromRoute] string Transaccion)
{
    Horario horario      = new Horario();
    horario.Transaccion = Transaccion;

    var cadenaConexion   = new ConfigurationBuilder()
        .AddJsonFile("appsettings.json")
        .Build()
        .GetSection("ConnectionStrings")["Conexion"];

    DataSet dsResultado   = await DBXmlMethods.EjecutaBase(
        NameStoreProcedure.GET_ACTIVIDADES,
        cadenaConexion,
        horario.Transaccion,
        null
    );
    List<Horario> listaData = new List<Horario>();

    if (dsResultado.Tables.Count > 0)
    {
        try
        {
            foreach (DataRow row in dsResultado.Tables[0].Rows)
            {
                Horario objResponse = new Horario
                {
                    Id_Horario = Convert.ToInt32(row["id_horario"]),
                    Hora       = row["hora"].ToString()
                };
                listaData.Add(objResponse);
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.ToString());
        }
    }
    return Ok(listaData);
}
```

Esta Api permite consultar los horarios registrados, se conecta a la base de datos y ejecuta la consulta para obtener los horarios asociados a la transacción.

- Api para las actividades.

```
[HttpPost("Actividad")]
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
0 referencias
public async Task<ActionResult<RespuestaSP>> PostActividades([FromBody] Actividad actividad)
{
    var cadenaConexion = new ConfigurationBuilder()
        .AddJsonFile("appsettings.json")
        .Build()
        .GetSection("ConnectionStrings")["Conexion"];
    XDocument xmlDocParam = Shared.DBXmlMethods.GetXml(actividad);
    DataSet dsResultado = await Shared.DBXmlMethods.EjecutaBase(
        NameStoreProcedure.CRUD_ACTIVIDADES,
        cadenaConexion,
        actividad.Transaccion,
        xmlDocParam.ToString());

    RespuestaSP objRespuesta = new RespuestaSP();

    if (dsResultado.Tables.Count > 0)
    {
        try
        {
            objRespuesta.Respuesta     = dsResultado.Tables[0][ "Respuesta" ].ToString();
            objRespuesta.Leyenda       = dsResultado.Tables[0][ "Leyenda" ].ToString();
        }
        catch (Exception e)
        {
            objRespuesta.Respuesta = "Error";
            objRespuesta.Leyenda = "Lo sentimos ha ocurrido un error";
        }
    }
    return Ok(objRespuesta);
}
```

Esta Api permite crear o actualizar una actividad al interactuar con la base de datos. Requiere autenticación mediante token.

- Api para las actividades personalizadas

```
[HttpPost("Personalizado")]
0 referencias
public async Task<ActionResult<RespuestaSP>> PostPersonalizado([FromBody] Personalizado personalizado)
{
    var cadenaConexion = new ConfigurationBuilder()
        .AddJsonFile("appsettings.json")
        .Build()
        .GetSection("ConnectionString")["Conexion"];
    XDocument xmlDocParam = Shared.DBXmlMethods.GetXml(personalizado);
    DataSet dsResultado = await Shared.DBXmlMethods.EjecutaBase(
        NameStoreProcedure.CRUD_PERSONALIZADO,
        cadenaConexion,
        personalizado.Transaccion,
        xmlDocParam.ToString());
    RespuestaSP objRespuesta = new RespuestaSP();
    if (dsResultado.Tables.Count > 0)
    {
        try
        {
            objRespuesta.Respuesta = dsResultado.Tables[0].Rows[0]["Respuesta"].ToString();
            objRespuesta.Leyenda = dsResultado.Tables[0].Rows[0]["Leyenda"].ToString();
        }
        catch (Exception e)
        {
            objRespuesta.Respuesta = "Error";
            objRespuesta.Leyenda = "Lo sentimos ha ocurrido un error";
        }
    }
    return Ok(objRespuesta);
}
```

Esta Api permite crear o actualizar entradas personalizadas en nuestra base de datos.

- Api administradores

```
[HttpPost("Personalizado/Admin")]
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
0 referencias
public async Task<ActionResult<RespuestaSP>> PostAdminPersonalizado([FromBody] Personalizado personalizado)
{
    var cadenaConexion = new ConfigurationBuilder()
        .AddJsonFile("appsettings.json")
        .Build()
        .GetSection("ConnectionString")["Conexion"];
    XDocument xmlDocParam = Shared.DBXmlMethods.GetXml(personalizado);
    DataSet dsResultado = await Shared.DBXmlMethods.EjecutaBase(
        NameStoreProcedure.CRUD_PERSONALIZADO,
        cadenaConexion,
        personalizado.Transaccion,
        xmlDocParam.ToString());
    RespuestaSP objRespuesta = new RespuestaSP();
    if (dsResultado.Tables.Count > 0)
    {
        try
        {
            objRespuesta.Respuesta = dsResultado.Tables[0].Rows[0]["Respuesta"].ToString();
            objRespuesta.Leyenda = dsResultado.Tables[0].Rows[0]["Leyenda"].ToString();
        }
        catch (Exception e)
        {
            objRespuesta.Respuesta = "Error";
            objRespuesta.Leyenda = "Lo sentimos ha ocurrido un error";
        }
    }
    return Ok(objRespuesta);
}
```

Esta Api, protegida por autenticación, permite a los administradores crear o actualizar entradas personalizadas en la base de datos.

## 5. Procedimientos almacenados Módulo Actividades

### Store Procedure GET\_SP\_ACTIVIDADES

```
--  
ALTER PROCEDURE [dbo].[GET_SP_ACTIVIDADES] (@iTransaccion AS VARCHAR(50)) AS  
BEGIN  
    DECLARE @respuesta AS VARCHAR(50)  
    DECLARE @leyenda AS VARCHAR(50)
```

### CONSULTA\_ACTIVIDADES

```
IF(@iTransaccion = 'CONSULTA_ACTIVIDADES')  
BEGIN  
    SELECT a.id_actividad_informacion      AS id_actividad,  
          a.nombre                      AS nombre,  
          h.id_horario                  AS id_hora,  
          h.hora                        AS hora,  
          ai.cantidadPersonas           AS cantidadPersonas,  
          ai.cantidadGuías              AS cantidadGuías,  
          a.tiempo                       AS tiempo,  
          ai.precio                      AS precio,  
          ai.descripcion                AS descripcion,  
          a.imagen                      AS imagen  
    FROM actividad a  
    JOIN actividadCabecera ai ON a.id_actividad_informacion = ai.id_actividad_informacion  
    JOIN horario h ON ai.id_horario = h.id_horario  
    WHERE a.estado = 1  
    ORDER BY  
          a.id_actividad DESC;  
    SET @respuesta = 'OK';  
    SET @leyenda = 'Consulta Exitosa';  
END
```

### CONSULTA\_PERSONALIZADOS

```
IF(@iTransaccion = 'CONSULTA_PERSONALIZADOS')  
BEGIN  
    SELECT p.id_actividad_informacion      AS id_personalizado,  
          p.nombreUsuario                 AS nombreUsuario,  
          p.telefono                     AS telefono,  
          h.id_horario                  AS id_hora,  
          h.hora                        AS hora,  
          ai.cantidadPersonas           AS cantidadPersonas,  
          ai.cantidadGuías              AS cantidadGuías,  
          p.fecha                        AS fecha,  
          ai.precio                      AS precio,  
          ai.descripcion                AS descripcion,  
          ai.estado                      AS estado  
    FROM personalizado p  
    JOIN actividadCabecera ai ON p.id_actividad_informacion = ai.id_actividad_informacion  
    JOIN horario h ON ai.id_horario = h.id_horario  
    WHERE p.estado = 1  
    ORDER BY  
          p.id_personalizado DESC;  
  
    SET @respuesta = 'OK';  
    SET @leyenda = 'Consulta Exitosa';  
END
```

## **CONSULTA\_HORARIO**

```
IF(@iTransaccion = 'CONSULTA_HORARIO')
BEGIN
    SELECT id_horario, hora FROM Horario WHERE estado = 1;

    SET @respuesta = 'OK';
    SET @leyenda = 'Consulta Exitosa';
END
```

## **Store Procedure SET\_SP\_ACTIVIDADES**

```
ALTER PROCEDURE [dbo].[SET_SP_ACTIVIDADES] (@iTransaccion AS VARCHAR(50), @iXml AS XML) AS
BEGIN

    SET NOCOUNT ON;
    DECLARE @respuesta AS VARCHAR(50)
    DECLARE @leyenda AS VARCHAR(50)

    DECLARE @id_actividad_informacion      AS INT
    DECLARE @id_horario                   AS INT
    DECLARE @cantidadPersonas            AS INT
    DECLARE @cantidadGuias              AS INT
    DECLARE @precio                      AS DECIMAL(8,2)
    DECLARE @descripcion                AS VARCHAR(500)
    DECLARE @nombre                      AS VARCHAR(250)
    DECLARE @tiempo                      AS VARCHAR(100)
    DECLARE @imagen                     AS VARCHAR(100)

    BEGIN TRY
```

## **AGREGAR\_ACTIVIDAD**

```
BEGIN TRANSACTION TRX_DATOS

IF(@iTransaccion = 'AGREGAR_ACTIVIDAD')
BEGIN
    SET @id_horario      = (SELECT @iXml.value('/Actividad/ActividadInformacion/Horario/Id_Horario')[1],           'INT')
    SET @cantidadPersonas = (SELECT @iXml.value('/Actividad/ActividadInformacion/CantidadPersonas')[1],           'INT')
    SET @cantidadGuias   = (SELECT @iXml.value('/Actividad/ActividadInformacion/CantidadGuias')[1],           'INT')
    SET @precio          = (SELECT @iXml.value('/Actividad/ActividadInformacion/Precio')[1],           'DECIMAL(8,2)')
    SET @descripcion     = (SELECT @iXml.value('/Actividad/ActividadInformacion/Descripcion')[1],           'VARCHAR(500)')
    SET @nombre           = (SELECT @iXml.value('/Actividad/Nombre')[1],           'VARCHAR(250)')
    SET @tiempo           = (SELECT @iXml.value('/Actividad/Tiempo')[1],           'VARCHAR(100)')
    SET @imagen           = (SELECT @iXml.value('/Actividad/Imagen')[1],           'VARCHAR(100)')

    INSERT INTO actividadCabecera(id_horario, cantidadPersonas, cantidadGuias, precio, descripcion)
    VALUES (@id_horario, @cantidadPersonas, @cantidadGuias, @precio, @descripcion)

    DECLARE @id INT;
    SET @id = SCOPE_IDENTITY();

    INSERT INTO actividad(id_actividad_informacion, nombre, tiempo, imagen)
    VALUES (@id, @nombre, @tiempo, @imagen)

    SET @respuesta = 'Enhorabuena';
    SET @leyenda = 'Actividad Agregada';
END
```

## ACTUALIZAR\_ACTVIDAD

```
IF(@iTransaccion = 'ACTUALIZAR_ACTVIDAD')
BEGIN
    SET @id_actividad_informacion = (SELECT @iXml.value('/Actividad/ActividadInformacion/Id_ActividadInformacion')[1], 'INT')
    SET @id_horario = (SELECT @iXml.value('/Actividad/ActividadInformacion/Horario/Id_Horario')[1], 'INT')
    SET @cantidadPersonas = (SELECT @iXml.value('/Actividad/ActividadInformacion/CantidadPersonas')[1], 'INT')
    SET @cantidadGuías = (SELECT @iXml.value('/Actividad/ActividadInformacion/CantidadGuías')[1], 'INT')
    SET @precio = (SELECT @iXml.value('/Actividad/ActividadInformacion/Precio')[1], 'DECIMAL(8,2)')
    SET @descripcion = (SELECT @iXml.value('/Actividad/ActividadInformacion/Descripcion')[1], 'VARCHAR(500)')
    SET @nombre = (SELECT @iXml.value('/Actividad/Nombre')[1], 'VARCHAR(250)')
    SET @tiempo = (SELECT @iXml.value('/Actividad/Tiempo')[1], 'VARCHAR(100)')
    SET @imagen = (SELECT @iXml.value('/Actividad/Imagen')[1], 'VARCHAR(100)')

    UPDATE actividadCabecera SET
        id_horario = @id_horario,
        cantidadPersonas = @cantidadPersonas,
        cantidadGuías = @cantidadGuías,
        precio = @precio,
        descripción = @descripcion
    WHERE
        id_actividad_informacion = @id_actividad_informacion

    UPDATE actividad SET
        nombre = @nombre,
        tiempo = @tiempo,
        imagen = @imagen
    WHERE
        id_actividad_informacion = @id_actividad_informacion

    SET @respuesta = 'Enhorabuena';
    SET @leyenda = 'Actividad Actualizada';
END
```

## ELIMINAR\_ACTVIDAD

```
IF(@iTransaccion = 'ELIMINAR_ACTVIDAD')
BEGIN
    SET @id_actividad_informacion = (SELECT @iXml.value('/Actividad/ActividadInformacion/Id_ActividadInformacion')[1], 'INT')

    UPDATE actividadCabecera SET estado = 0
    WHERE id_actividad_informacion = @id_actividad_informacion
    UPDATE actividad SET estado = 0
    WHERE id_actividad_informacion = @id_actividad_informacion

    SET @respuesta = 'Enhorabuena';
    SET @leyenda = 'Actividad eliminada';
END

if @@TRANCOUNT > 0
BEGIN
    COMMIT TRANSACTION TRX_DATOS;
END

END TRY

BEGIN CATCH
    if @@TRANCOUNT > 0
    BEGIN
        ROLLBACK TRANSACTION TRX_DATOS;
    END
    SET @respuesta = 'ERROR';
    SET @leyenda = 'Inconvenientes para realizar la transaccion: ';
END CATCH
```

## Store Procedure SET\_SP\_PERSONALIZADOS

```
ALTER PROCEDURE [dbo].[SET_SP_PERSONALIZADOS] (@iTransaccion AS VARCHAR(50), @iXml AS XML) AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @respuesta AS VARCHAR(50)
    DECLARE @leyenda AS VARCHAR(50)

    DECLARE @id_actividad_informacion      AS INT
    DECLARE @id_horario                   AS INT
    DECLARE @cantidadPersonas            AS INT
    DECLARE @cantidadGuías              AS INT
    DECLARE @precio                      AS DECIMAL(8,2)
    DECLARE @descripcion                AS VARCHAR(500)
    DECLARE @nombreUsuario               AS VARCHAR(250)
    DECLARE @telefono                    AS VARCHAR(100)
    DECLARE @fecha                       AS DATE

    BEGIN TRY
```

## AGREGAR\_PERSONALIZADO

```
BEGIN TRANSACTION TRX_DATOS

IF(@iTransaccion = 'AGREGAR_PERSONALIZADO')
BEGIN
    SET @id_horario      = (SELECT @iXml.value('/Personalizado/ActividadInformacion/Horario/Id_Horario')[1], 'INT'))
    SET @cantidadPersonas = (SELECT @iXml.value('/Personalizado/ActividadInformacion/CantidadPersonas')[1], 'INT'))
    SET @cantidadGuías   = (SELECT @iXml.value('/Personalizado/ActividadInformacion/CantidadGuías')[1], 'INT'))
    SET @precio           = (SELECT @iXml.value('/Personalizado/ActividadInformacion/Precio')[1], 'DECIMAL(8,2))')
    SET @descripcion     = (SELECT @iXml.value('/Personalizado/ActividadInformacion/Descripcion')[1], 'VARCHAR(500))')
    SET @nombreUsuario    = (SELECT @iXml.value('/Personalizado/NombreUsuario')[1], 'VARCHAR(250))')
    SET @telefono         = (SELECT @iXml.value('/Personalizado/Telefono')[1], 'VARCHAR(100))')
    SET @fecha             = (SELECT @iXml.value('/Personalizado/Fecha')[1], 'DATE'))

    INSERT INTO actividadCabecera(id_horario, cantidadPersonas, cantidadGuías, precio, descripcion)
    VALUES (@id_horario, @cantidadPersonas, @cantidadGuías, @precio, @descripcion)

    DECLARE @id INT;
    SET @id = SCOPE_IDENTITY();

    INSERT INTO personalizado(id_actividad_informacion, nombreUsuario, telefono, fecha)
    VALUES (@id, @nombreUsuario, @telefono, @fecha)

    SET @respuesta = 'Enhorabuena';
    SET @leyenda = 'Tu valor a cancelar es ' + CONVERT(VARCHAR(10), @precio);
END
```

## ACTUALIZAR\_PERSONALIZADO

```
IF(@iTransaccion = 'ACTUALIZAR_PERSONALIZADO')
BEGIN
    SET @id_actividad_informacion = (SELECT @iXml.value('/Personalizado/ActividadInformacion/Id_ActividadInformacion')[1], 'INT'))
    SET @id_horario              = (SELECT @iXml.value('/Personalizado/ActividadInformacion/Horario/Id_Horario')[1], 'INT'))
    SET @cantidadPersonas        = (SELECT @iXml.value('/Personalizado/ActividadInformacion/CantidadPersonas')[1], 'INT'))
    SET @cantidadGuías          = (SELECT @iXml.value('/Personalizado/ActividadInformacion/CantidadGuías')[1], 'INT'))
    SET @precio                  = (SELECT @iXml.value('/Personalizado/ActividadInformacion/Precio')[1], 'DECIMAL(8,2))')
    SET @descripcion            = (SELECT @iXml.value('/Personalizado/ActividadInformacion/Descripcion')[1], 'VARCHAR(500))')
    SET @nombreUsuario           = (SELECT @iXml.value('/Personalizado/NombreUsuario')[1], 'VARCHAR(250))')
    SET @telefono                = (SELECT @iXml.value('/Personalizado/Telefono')[1], 'VARCHAR(100))')
    SET @fecha                   = (SELECT @iXml.value('/Personalizado/Fecha')[1], 'DATE'))

    UPDATE actividadCabecera SET
        id_horario      = @id_horario,
        cantidadPersonas = @cantidadPersonas,
        cantidadGuías   = @cantidadGuías,
        precio           = @precio,
        descripcion     = @descripcion
    WHERE
        id_actividad_informacion = @id_actividad_informacion

    UPDATE personalizado SET
        nombreUsuario    = @nombreUsuario,
        telefono         = @telefono,
        fecha             = @fecha
    WHERE
        id_actividad_informacion = @id_actividad_informacion

    SET @respuesta = 'Enhorabuena';
    SET @leyenda = 'Actividad Personalizada Actualizada';
END
```

## ELIMINAR\_PERSONALIZADO

```
IF(@iTransaccion = 'ELIMINAR_PERSONALIZADO')
BEGIN
    SET @id_actividad_informacion = (SELECT @iXml.value('(/Personalizado/ActividadInformacion/Id_ActividadInformacion)[1]', 'INT'))
    UPDATE actividadCabecera SET estado = 0
    WHERE id_actividad_informacion = @id_actividad_informacion
    UPDATE personalizado SET estado = 0
    WHERE id_actividad_informacion = @id_actividad_informacion

    SET @respuesta = 'Enhorabuena';
    SET @leyenda = 'Actividad Personalizada eliminada';
END

if @@TRANCOUNT > 0
BEGIN
    COMMIT TRANSACTION TRX_DATOS;
END

END TRY
BEGIN CATCH
if @@TRANCOUNT > 0
BEGIN
    ROLLBACK TRANSACTION TRX_DATOS;
END
SET @respuesta = 'ERROR';
SET @leyenda = 'Inconvenientes para realizar la transaccion';
END CATCH
```

## 6. Ventanas Modulo Actividades

- Ventana principal Actividades

The screenshot shows the homepage of the Zoo Municipal de Loja. At the top, there's a green header bar with the logo 'ZOO MUNICIPAL DE LOJA' and a navigation menu with links to 'Inicio', 'Animales', 'Actividades', 'Voluntariado', and 'Sobre Nosotros'. Below the header is a large banner featuring a photograph of several animals (a tiger, an elephant, and a giraffe) and visitors. The banner has text overlay: 'Zoo Municipal de Loja' and 'Disfruta de la diversión de aprender sobre la vida salvaje y experimentar de primera mano junto a tu familia o diviértete jugando con amigos.' Below the banner, there's an address: 'Av. 8 de Diciembre, Loja/Loja-Ecuador'. The main content area contains two columns of text. The left column says: '¡Descubre la magia de la vida salvaje en nuestro zoológico! Disfruta de un día inolvidable rodeado de una diversidad de animales fascinantes, desde leones hasta pingüinos. Ofrecemos actividades'. The right column says: 'personalizadas para todas las edades, como paseos educativos, alimentación de animales y charlas interactivas. Ya sea que nos visites de día o por la tarde, encontrarás experiencias emocionantes y educativas. Únete a nosotros y vive momentos inolvidables con tu familia en un ambiente seguro y entretenido. ¡Te esperamos en nuestro zoológico!'.

- Actividades Matutinas



**ACTIVIDADES MATUTINAS**

Experiencia Con Aguilas	Hora: 11:00 A.M
<b>Precio:</b> \$20 Guias: 3   Tiempo: 1 Hora Cantidad maxima de personas: 10 <small>Te invitamos a vivir una experiencia única, podrás acercarte a estas increíbles aves y descubrir su grandeza de cerca.</small>	
	
Cena Temática Africana	Hora: 11:00 A.M
	
Sesión De Fotos Con Animales	Hora: 11:00 A.M
	

- Actividades Vespertinas



**ACTIVIDADES VESPERTINAS**

Paseo En Bicicleta Por El Parque	Hora: 16:00 P.M
	
<b>Precio:</b> \$8 Guias: 2   Tiempo: 1 Hora 30 minutos Cantidad máxima de personas: 12 <small>Recorre nuestro extenso parque en bicicleta y disfruta de la naturaleza mientras haces ejercicio.</small>	
	
Exhibición De Delfines	Hora: 15:00 P.M
	
Safari En Jeep Por La Reserva	Hora: 14:00 P.M
	

Embárcate en un emocionante safari en jeep por la reserva, disfruta de una exhibición de delfines, haz un paseo en bicicleta por el parque y ten un encuentro especial con los lemurres, no te pierdas una deliciosa cena temática africana, esto y mucho mas. ¡Las tardes en nuestro zoológico están llenas de diversión y aventura!



- Actividades Personalizadas



**PERSONALIZA TU ACTIVIDAD**

En nuestro zoológico, te ofrecemos la oportunidad de personalizar tus actividades y crear una experiencia única.

<input type="text"/> Nombre*	<input type="text"/> Teléfono*	<input type="text"/> Fecha *
<input type="text"/> Hora*	<input type="text"/> N° Personas*	<input type="text"/> N° Guias*
<input type="text"/> Descripción*		
PERSONALIZAR ACTIVIDAD		

Aviso importante: Se le otorga un plazo de 72 horas para dirigirse a las oficinas del Zoológico Municipal de Loja con el código único generado para cancelar su actividad personalizada. Transcurrido este tiempo, su actividad personalizada habrá expirado.

- Gestión de actividades

Nombre	N° Personas	Precio	Hora	Time	Acción
Experiencia con águilas	10	20	11:00 A.M	1 Hora	
Cena temática africana	21	35	11:00 A.M	2 Horas	
Paseo en bicicleta por el parque	12	8	16:00 P.M	1 Hora 30 minutos	
Exhibición de delfines	40	12	15:00 P.M	1 Hora	

- Visualizar actividades

- Editar actividades

**Gestor de Actividades**

### Editor de Actividad

**Nombre \***  
 Experiencia con águilas  Seleccionar archivo | Sin archivos seleccionados

**Nº Personas \***  
 10 **Nº Guías \***  
 3

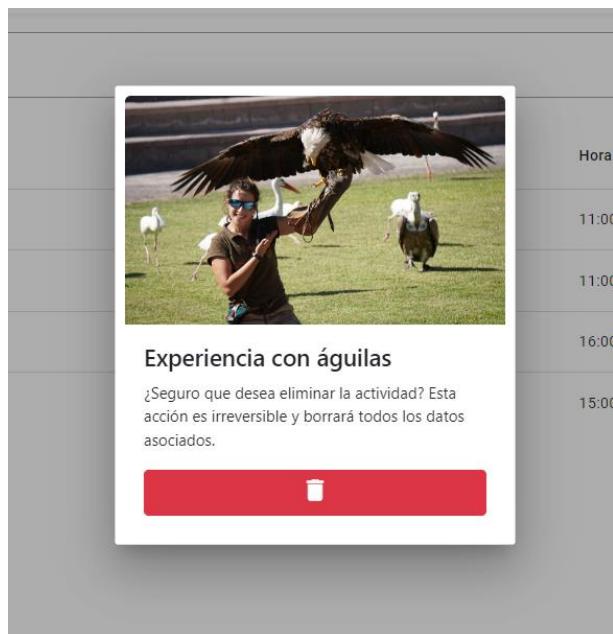
**Hora \*** 11:00 A.M **Tiempo \*** 1 Hora

**Descripción \***  
 Te invitamos a vivir una experiencia única, podrás acercarte a estas increíbles aves y descubrir su grandeza de cerca.

**Precio \*** 20

[Cancelar](#) [Guardar](#)

- Eliminar Actividades



- Gestión Actividades Personalizadas

Nombre	N° Personas	Hora	Fecha	Acción
Prueba	5	12:00 A.M.	2023-08-25	
Visita en el pantano	3	14:00 P.M.	2023-07-27	
Laura García	3	15:00 P.M.	2023-09-14	

Items per page: 4 1 - 3 of 3 < >

- Visualizar Actividades Personalizadas

**Detalles de la Reserva**

**Nombre de Usuario:** Prueba  
**Teléfono:** 0989898989  
**Cantidad de Personas:** 5  
**Cantidad de Guías:** 2  
**Precio:** \$75  
**Hora:** 12:00 A.M.  
**Fecha:** 2023-08-25  
**Descripción:** Ninguna por ahora

- Editar Actividades Personalizadas

**EDITAR ACTIVIDAD PERSONALIZADA**

Nombre\*

Teléfono\*

Fecha \*

Hora

N° Personas\*

N° Guías\*

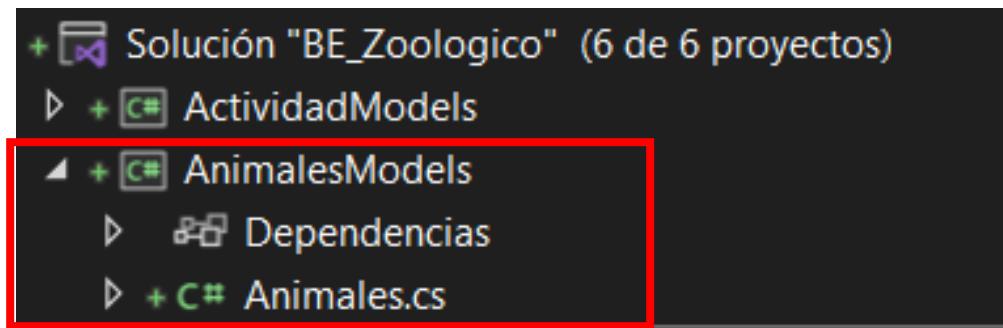
Descripción\*

- Borrar Actividades Personalizadas



## 7. Modulo Animales – API 's

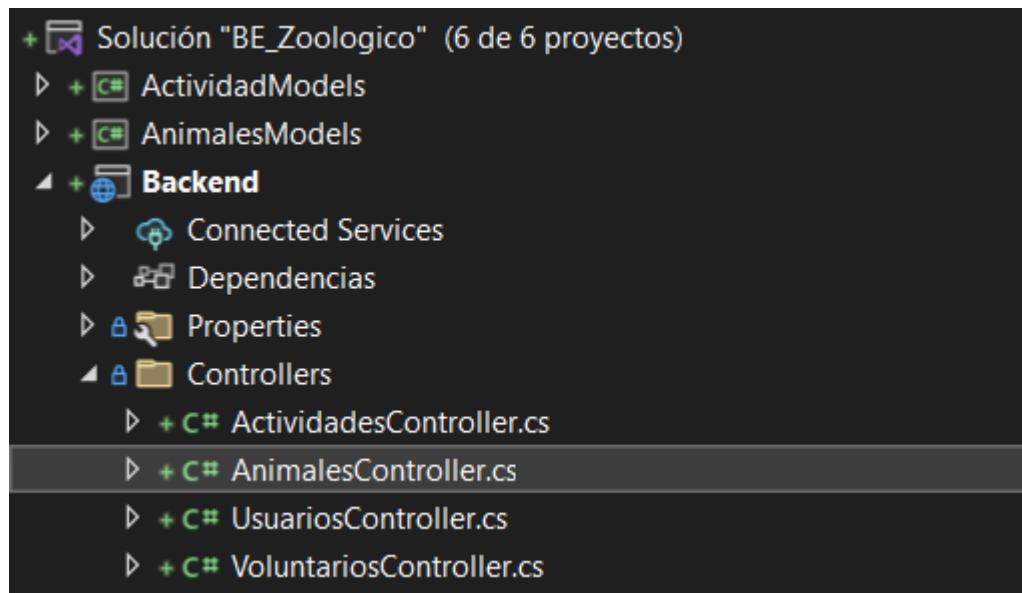
En la solución del proyecto llamada “**BE\_Zoologico**” se aloja una biblioteca de clases llamada “**AnimalesModels**” donde está ubicada la clase “**Animales.cs**” en la cual están los diferentes atributos necesarios.



### Atributos Clase Animales.cs:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace AnimalesModels
8  {
9
10     public class Animales
11     {
12
13         public int Id_animales { get; set; }
14
15         public string? Nombre { get; set; }
16
17         public int? Edad { get; set; }
18
19         public string? Especie { get; set; }
20
21         public string? Genero { get; set; }
22
23         public string? Origen { get; set; }
24
25         public string? Habitat { get; set; }
26
27         public string? Observaciones { get; set; }
28
29         public bool Estado { get; set; }
30
31         public string? Imagen { get; set; }
32
33         public string? Transaccion { get; set; }
34
35     }
36 }
```

La ubicación de las APIs del módulo Animales se encuentran ubicadas en el proyecto “Backend” dentro del controlador “AnimalesController.cs”.



En este controlador se encuentran 2 APIs, una con el método **GET** la cual va a devolver todos los animales que se encuentran en la tabla **dbo.Animales** en la base de datos del proyecto.

## API GetAnimales :

```
public class AnimalesController : Controller
{
    [HttpGet("Animales/{Transaccion}")]
    0 referencias
    public async Task<ActionResult<Animales>> GetAnimales([FromRoute] string Transaccion)
    {
        Animales animal = new Animales();
        animal.Transaccion = Transaccion;

        var cadenaConexion = new ConfigurationBuilder()
            .AddJsonFile("appsettings.json")
            .Build()
            .GetSection("ConnectionStrings")["Conexion"];

        DataSet dsResultado = await DBXmlMethods.EjecutaBase(
            NameStoreProcedure.GET_ANIMALES,
            cadenaConexion,
            animal.Transaccion,
            null
        );
        List<Animales> listData = new List<Animales>();

        if (dsResultado.Tables.Count > 0)
        {
            try
            {
                foreach (DataRow row in dsResultado.Tables[0].Rows)
                {
                    Animales objResponse = new Animales
                    {
                        Id_animales = Convert.ToInt32(row["id_animales"]),
                        Edad = Convert.ToInt32(row["edad"]),
                        Especie = row["especie"].ToString(),
                        Nombre = row["nombre"].ToString(),
                        Habitat = row["habitat"].ToString(),
                        Origen = row["origen"].ToString(),
                        Genero = row["genero"].ToString(),
                        Observaciones = row["observaciones"].ToString(),
                        Imagen = row["imagen"].ToString(),
                    };
                    listData.Add(objResponse);
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }
        }

        return Ok(listData);
    }
}
```

En la cadena de conexión se puede observar que se le pasa como nombre de Store Procedure “**GET\_ANIMALES**” esta constante se ubica en la carpeta **Shared** , en la clase **“NameStoreProcedure.cs”** en esta clase se ubican los diferentes nombres de los procedimientos almacenados alojados en la BD.

```

1  namespace Backend.Shared
2  {
3      10 referencias
4      public class NameStoreProcedure
5      {
6          //Actividades
7          public const string GET_ACTIVIDADES = "GET_SP_ACTIVIDADES";
8          public const string CRUD_ACTIVIDADES = "SET_SP_ACTIVIDADES";
9          public const string CRUD_PERSONALIZADO = "SET_SP_PERSONALIZADOS";
10
11         //Animales
12         public const string GET_ANIMALES = "GET_SP_ANIMALES";
13         public const string CRUD_ANIMALES = "SET_SP_ANIMALES";
14
15         //Voluntarios
16         public const string GET_VOLUNTARIOS = "GET_SP_VOLUNTARIOS";
17         public const string CRUD_VOLUNTARIOS = "SET_SP_VOLUNTARIOS";
18
19         //Usuarios
20
21         public const string GET_USUARIOS = "GET_SP_USUARIOS";
22
23     }
24 }
25
26

```

Para la tabla **dbo.Animales** existen creados 2 SP , llamados : “**GET\_SP\_ANIMALES**” & “**SET\_SP\_ANIMALES**”. El SP “**GET\_SP\_ANIMALES**” es el siguiente :

```

GO
ALTER PROCEDURE [dbo].[GET_SP_ANIMALES] (@iTransaccion AS VARCHAR(50)) AS
BEGIN
    DECLARE @respuesta AS VARCHAR(50)
    DECLARE @leyenda AS VARCHAR(50)

    BEGIN TRY
        IF (@iTransaccion = 'CONSULTA_ANIMALES')
            BEGIN
                SELECT a.id_animales AS id_animales,
                       a.nombre AS nombre,
                       a.edad AS edad,
                       a.especie AS especie,
                       a.genero AS genero,
                       a.origen AS origen,
                       a.habitat AS habitat,
                       a.observaciones AS observaciones,
                       a.imagen AS imagen
                FROM Animales a
                WHERE a.estado = 1
                ORDER BY a.id_animales DESC;
                SET @respuesta = 'OK';
                SET @leyenda = 'Consulta Exitosa';
            END
    END TRY
    BEGIN CATCH
        SET @respuesta = 'ERROR';
        SET @leyenda = CONCAT(' ', ERROR_MESSAGE());
    END CATCH
    SELECT @respuesta AS respuesta, @leyenda AS leyenda;
END

```

En este SP se encuentra la transacción llamada “**CONSULTA\_ANIMALES**” que devolverá a todos los animales dentro de la tabla **dbo.Animales**.

#### API PostAnimales :

En esta API se realizarán las diferentes transacciones CRUD en la tabla dbo.Animales, esta API como se evidencia en la imagen se necesita autorización para su uso . Además, todas estas transacciones que se realizan a través de esta API están ubicadas en el SP llamado “**SET\_SP\_ANIMALES**”. En la clase **NameStoreProcedure** se encuentra una constante llamada “**CRUD\_ANIMALES**” la cual contiene el nombre del SP .

```
[HttpPost("Animales")]
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
0 referencias
public async Task<ActionResult<RespuestaSP>> PostAnimales([FromBody] Animales animal)
{
    var cadenaConexion = new ConfigurationBuilder()
        .AddJsonFile("appsettings.json")
        .Build()
        .GetSection("ConnectionStrings")["Conexion"];
    XDocument xmlDocParam = Shared.DBXmlMethods.GetXml(animal);
    DataSet dsResultado = await Shared.DBXmlMethods.EjecutaBase(
        NameStoreProcedure.CRUD_ANIMALES,
        cadenaConexion,
        animal.Transaccion,
        xmlDocParam.ToString());
    RespuestaSP objRespuesta = new RespuestaSP();
    if (dsResultado.Tables.Count > 0)
    {
        try
        {
            objRespuesta.Respuesta = dsResultado.Tables[0].Rows[0]["Respuesta"].ToString();
            objRespuesta.Leyenda = dsResultado.Tables[0].Rows[0]["Leyenda"].ToString();
        }
        catch (Exception e)
        {
            objRespuesta.Respuesta = "Error";
            objRespuesta.Leyenda = "Lo sentimos ha ocurrido un error";
        }
    }
    return Ok(objRespuesta);
}
```

## SET\_SP\_ANIMALES – Transacción “AGREGAR\_ANIMAL” :

```
BEGIN TRY

BEGIN TRANSACTION TRX_DATOS

IF(@iTransaccion = 'AGREGAR_ANIMAL')
    BEGIN

        SET @nombre = (SELECT @iXml.value('/(Animales/Nombre)[1]', 'VARCHAR(100)'))
        SET @edad = (SELECT @iXml.value('/(Animales/Edad)[1]', 'INT'))
        SET @origen = (SELECT @iXml.value('/(Animales/Origen)[1]', 'VARCHAR(200)'))
        SET @especie = (SELECT @iXml.value('/(Animales/Especie)[1]', 'VARCHAR(100)'))
        SET @genero = (SELECT @iXml.value('/(Animales/Genero)[1]', 'VARCHAR(100)'))
        SET @habitat = (SELECT @iXml.value('/(Animales/Habitat)[1]', 'VARCHAR(200)'))
        SET @observaciones = (SELECT @iXml.value('/(Animales/Observaciones)[1]', 'VARCHAR(200)'))
        SET @imagen = (SELECT @iXml.value('/(Animales/Imagen)[1]', 'VARCHAR(100)'))

        INSERT INTO Animales (nombre, edad, origen, especie, genero, habitat, observaciones, imagen)
        VALUES (@nombre, @edad, @origen, @especie, @genero, @habitat, @observaciones, @imagen)

        SET @respuesta = 'Enhorabuena';
        SET @leyenda = 'Animal Agregado';
    END

```

## SET\_SP\_ANIMALES – Transacción “ACTUALIZAR\_ANIMAL” :

```
IF(@iTransaccion = 'ACTUALIZAR_ANIMAL')
BEGIN
    SET @id_animales = (SELECT @iXml.value('/(Animales/Id_animales)[1]', 'INT'))
    SET @nombre = (SELECT @iXml.value('/(Animales/Nombre)[1]', 'VARCHAR(200)'))
    SET @edad = (SELECT @iXml.value('/(Animales/Edad)[1]', 'INT'))
    SET @especie = (SELECT @iXml.value('/(Animales/Especie)[1]', 'VARCHAR(200)'))
    SET @genero = (SELECT @iXml.value('/(Animales/Genero)[1]', 'VARCHAR(200)'))
    SET @habitat = (SELECT @iXml.value('/(Animales/Habitat)[1]', 'VARCHAR(250)'))
    SET @observaciones = (SELECT @iXml.value('/(Animales/Observaciones)[1]', 'VARCHAR(250)'))
    SET @imagen = (SELECT @iXml.value('/(Animales/Imagen)[1]', 'VARCHAR(200)'))

    UPDATE Animales SET
        nombre = @nombre,
        edad = @edad,
        especie = @especie,
        genero = @genero,
        habitat = @habitat,
        observaciones = @observaciones,
        imagen = @imagen
    WHERE
        id_animales= @id_animales

    SET @respuesta = 'Enhorabuena';
    SET @leyenda = 'Animal Actualizado';
END
```

## SET\_SP\_ANIMALES – Transacción “ELIMINAR\_ANIMAL” :

```
IF(@iTransaccion = 'ELIMINAR_ANIMAL')
BEGIN
    SET @id_animales =  (SELECT @iXml.value('(/Animales/Id_animales)[1]',      'INT'))
    UPDATE Animales SET estado = 0
    WHERE id_animales = @id_animales

    SET @respuesta = 'Enhorabuena';
    SET @leyenda = 'Animal eliminado';
END
```

## 8. Ventanas Modulo Animales

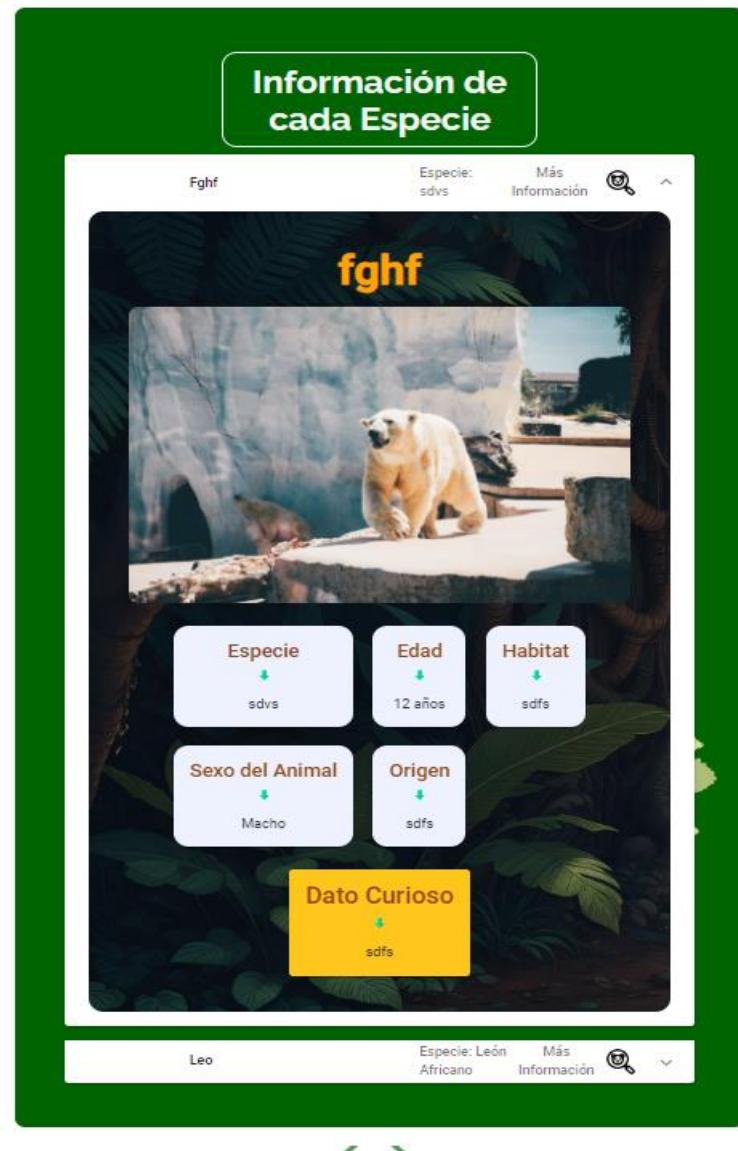
- Ventana Principal Animales



Dale un vistazo a las especies que cuidamos en nuestro Zoológico

¡Bienvenidos a nuestra increíble sección de animales del zoológico! Aquí podrán maravillarse con la diversidad y belleza de las especies que habitan en nuestro santuario animal. Desde majestuosos leones hasta pequeños y curiosos monos, tenemos una amplia variedad de criaturas para que disfruten. Admiren la gracia y elegancia de las jirafas mientras se pasean por su hábitat, o contemplen la destreza de los ágiles tigres mientras saltan de rama en rama. A medida que recorran nuestro zoológico, descubrirán la riqueza de la vida salvaje y aprenderán sobre la importancia de preservar y proteger a estas increíbles especies. ¡Disfruten de esta aventura llena de emociones y descubrimientos en nuestro mundo animal!





- Ventana Gestión Animales

Nombre	Edad	Especie	Sexo del Animal	Origen	Habitat	Observaciones	Acción
fghf	12	sdvs	Macho	sdfa	sdfa	sdfa	
Leo	5	León Africano	Masculino	África	Sabana	Leo es un león majestuoso que ...	

Items per page: 4    1 - 2 of 2    < >

- **Visualización Animales**

Leo  
Especie: León Africano

Edad: 5  
Sexo del Animal: Masculino  
Origen: África  
Habitat: Sabana

Dato Curioso: Leo es un león majestuoso que disfruta tomar el sol por las mañanas.



- **Editar Animales**

### Editor Animal

Nombre \* -

Edad : \* -

Especie :\* -

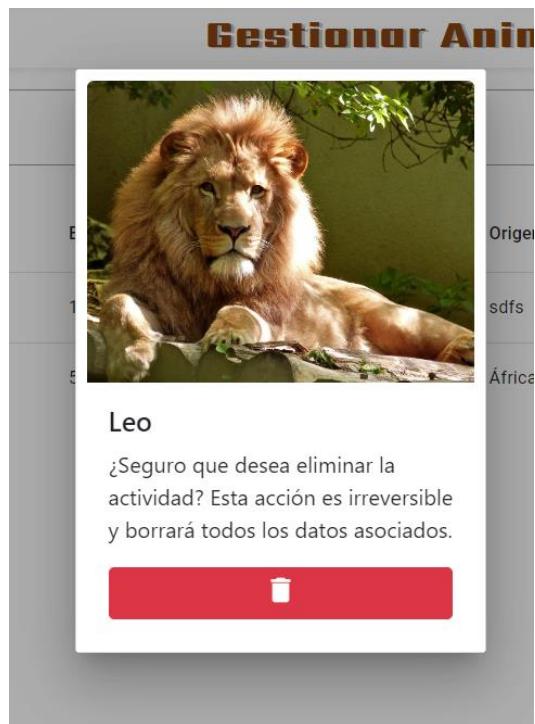
Sexo del Animal:\* -

Origen\* -

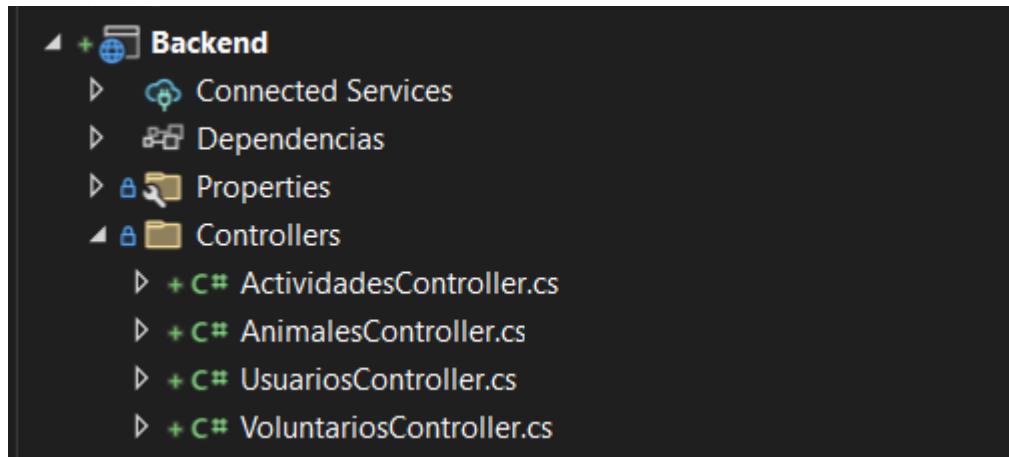
Habitat \* -

Observacion \* -

- Eliminar Animales



## 9. API Usuarios :



Esta API esta alojada en el controlador **UsuariosController** , básicamente esta API es para controlar el acceso al dashboard de la administración . En esta se hace el uso de tokens para la seguridad del aplicativo .

```
[Route("[action]")]
[HttpPost]
0 referencias
public async Task<ActionResult<Usuario>> GetLogin([FromBody] Usuario usuarios)
{
    try
    {
        var cadenaConexion = new ConfigurationBuilder().AddJsonFile("appsettings.json")
            .Build()
            .GetSection("ConnectionStrings")["Conexion"];
        XDocument xmlParam = DBXmlMethods.GetXml(usuarios);

        DataSet resultado = await DBXmlMethods.EjecutaBase(NameStoreProcedure.GET_USUARIOS, cadenaConexion, usuarios.Transaccion, xmlParam.ToString());
        if (resultado.Tables.Count > 0)
        {
            try
            {
                if (resultado.Tables[0].Rows.Count > 0)
                {
                    Usuario usertem = new Usuario();
                    //usertem.Id = Convert.ToInt32(resultado.Tables[0].Rows[0]["Id"]);
                    usertem.Cedula = resultado.Tables[0].Rows[0]["Cedula"].ToString();
                    return Ok(JsonConvert.SerializeObject(CrearToken(usertem)));
                }
                else
                {
                    RespuestaSP objresponse = new RespuestaSP();
                    objresponse.Leyenda = "Error en las credenciales de acceso";
                    objresponse.Respuesta = "Error";
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }
        }
        return Ok();
    }
    catch (Exception e)
    {
        Console.WriteLine("error" + e.Message);
        return StatusCode(500);
    }
}
```

En esta API se hace el uso de la constante ubicada en la clase NameStoreProcedure llamada **GET\_USUARIOS** la cual contiene el nombre del procedimiento almacenado que verificará que las credenciales concuerden y estén alojadas en la tabla **dbo.usuario**. Si el usuario es correcto tendrá los permisos correspondientes mediante el token , para las operaciones CRUD

```

1  namespace Backend.Shared
2  {
3      10 referencias
4      public class NameStoreProcedure
5      {
6          //Actividades
7          public const string GET_ACTIVIDADES = "GET_SP_ACTIVIDADES";
8          public const string CRUD_ACTIVIDADES = "SET_SP_ACTIVIDADES";
9          public const string CRUD_PERSONALIZADO = "SET_SP_PERSONALIZADOS";
10
11         //Animales
12         public const string GET_ANIMALES = "GET_SP_ANIMALES";
13         public const string CRUD_ANIMALES = "SET_SP_ANIMALES";
14
15         //Voluntarios
16         public const string GET_VOLUNTARIOS = "GET_SP_VOLUNTARIOS";
17         public const string CRUD_VOLUNTARIOS = "SET_SP_VOLUNTARIOS";
18
19         //Usuarios
20
21         public const string GET_USUARIOS = "GET_SP_USUARIOS";
22
23
24     }
25
26 }
```

### **GET\_SP\_USUARIOS : Transacción “CONSULTAR\_USUARIO\_LOGIN”**

```

BEGIN
    SET NOCOUNT ON;
    DECLARE @respuesta AS VARCHAR(10);
    DECLARE @leyenda AS VARCHAR(50);
    DECLARE @cedula AS VARCHAR(10);
    DECLARE @clave AS VARCHAR(50);

BEGIN TRY
    IF (@iTransaccion = 'CONSULTAR_USUARIO_LOGIN')
        BEGIN
            SELECT @cedula = DATO_XML.X.value('Cedula [1]', 'VARCHAR(10)');
            @clave = DATO_XML.X.value('Password [1]', 'VARCHAR(50)')
            FROM @iXML.nodes('/Usuario') AS DATO_XML(X)

            SELECT *
            FROM usuario u1
            WHERE u1.cedula = @cedula and u1.password = @clave

            SET @respuesta = 'Bienvenido';
            SET @leyenda = 'Login Exitoso';
        END
END TRY

BEGIN CATCH
    SET @respuesta = 'error';
    SET @leyenda = 'Error al ejecutar el comando en la BD: '+ERROR_MESSAGE();
END CATCH
```

## 10. Diseño Diagrama Entidad Relación

