Grupo nro 5

SISTEMA DE GESTION DE ANIMALES EN UN ZOOLOGICO

UNIVERSIDAD “MAYOR DE SAN ANDRES”

FACULTAD DE CIENCIAS PURAS Y NATURALES

PROGRAMACION II - 121

-INTEGRANTES DE GRUPO :

Universitarios :
- Alan Eduardo Saravia Lopez
  C.I. 9088976
- Miguel Alejandro Quisbert Mamani
  C.I. 10064367
- Ariel Alejandro Terrazas Zalles
  C.I. 8318384
- Carlos Andres Mamani Mamani
  C.I. 8439346
- Jhoel Yesid Mamani Condori
  C.I. 12575297
- Norvin Jose Esquivel Zabaleta
  C.I. 10923758
- Kevin German Vega Jimenez
  C.I. 13648557

La Paz - Bolivia

Fecha: 27/01/2025

Introducción

Este documento detalla los pasos y aspectos esenciales para la elaboración de un proyecto completo utilizando los principios de la Programación Orientada a Objetos (POO). Se desarrolla un ejemplo práctico implementado en Java, integrando diagramas UML, constructores, sobrecarga de métodos, herencia, agregación, composición, genericidad, manejo de archivos, interfaces y patrones de diseño.

Objetivo del Proyecto

El objetivo del proyecto final es consolidar los conocimientos adquiridos en la materia de Programación Orientada a Objetos (POO) mediante el diseño, implementación y presentación de un sistema funcional desarrollado en Java. Este proyecto permitirá a los estudiantes demostrar su capacidad para aplicar conceptos clave de POO y técnicas avanzadas en el desarrollo de aplicaciones de escritorio con integración de bases de datos.

Objetivos del Sistema

- **Optimizar el Control de animales en el zoo :** Implementar mecanismos que permitan un seguimiento eficiente de los animales en tiempo real, organizando los animales y separándolos por especies y también implementando un habitad
- Agregar animales nuevos al zoológico

- Desarrollar una base de datos para almacenar la información relevante del zoológico.
- Implementar la gestión de animales utilizando agregación, composición y herencia.
- Aplicar los principios de POO para garantizar un código limpio y mantenible.

Tecnologías Utilizadas

- **Lenguaje de programación**: Java
- **Base de datos**: MySQL

Diseño del Sistema

Modelo de Base de Datos El sistema utiliza una base de datos MySQL para almacenar la información de los animales, cuidadores y visitantes. Las tablas principales incluyen:

- Animales: Contiene información sobre cada animal, como especie, nombre, edad, salud, etc.
- Cuidadores: Almacena datos sobre los cuidadores, como nombre, turno, animales a su cargo, etc.

Definición del Proyecto

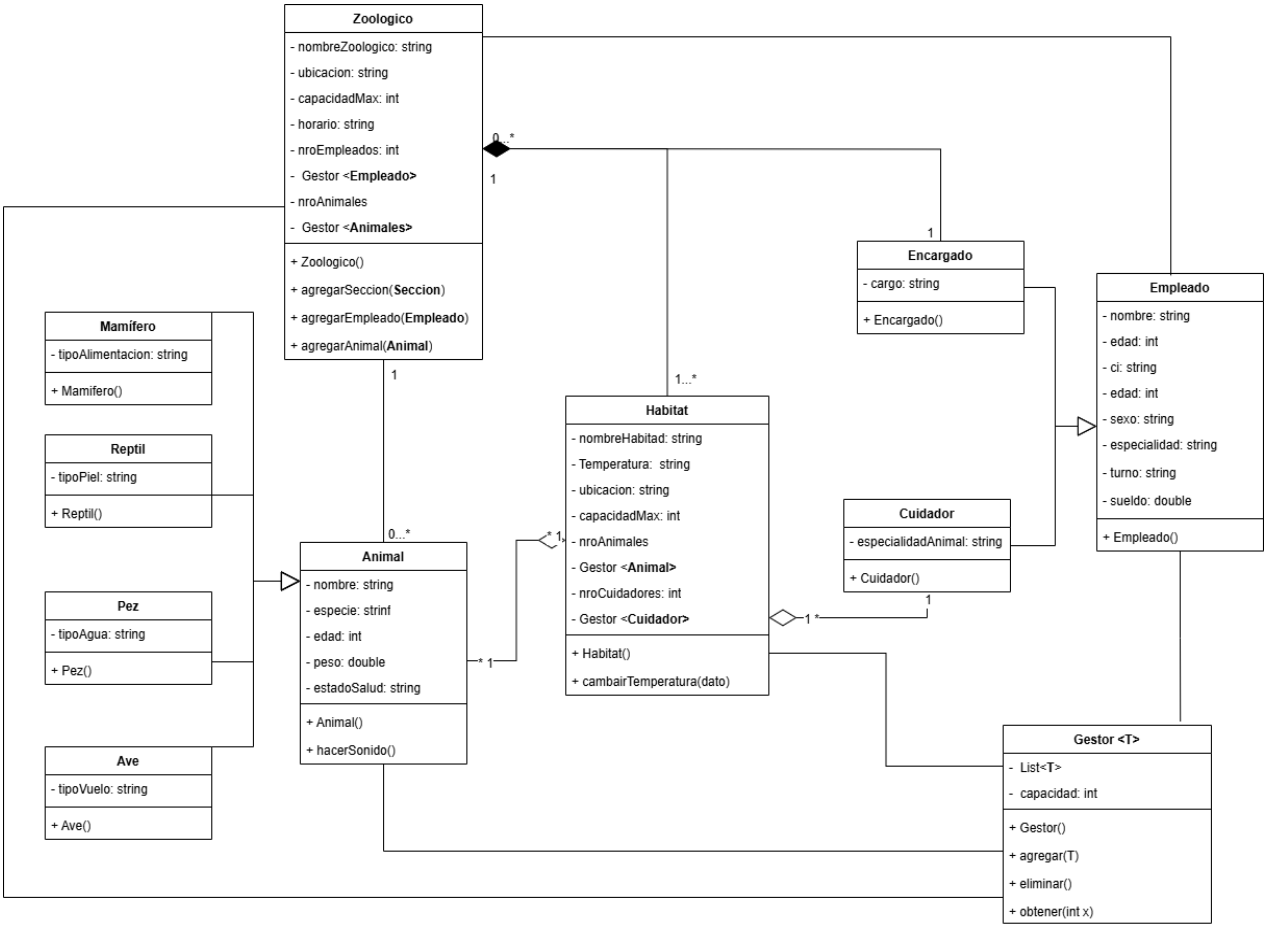
Descripción General

el proyecto consiste en un sistema de gestión de animales para un zoologico, que permita: administrar animales y los habitads a fin de evitar que un habitad se llene y como tal ya no se pueda agregar mas animales.
registrar empleados nuevos (cuidador y encargado) asi como también darles un cargo , sueldo y también el de gestiona el sueldo de dicho empleado .
poder mostrar el animal mas longevo del zoológico asi como también mostrar el animal mas pesado del zoo y también el sistema puede cambiar la temperatura del habitad.

Análisis y Diseño

Diagrama UML

El sistema incluirá los siguientes diagramas UML:



Principios de Diseño

Herencia: Las clases mamífero , reptil , pez y ave heredan de animal aparte de eso se creo otra clase denominada empleado que hereda a encargado y cuidador.

Composición: La clase Préstamo incluye un objeto de tipo Material.

Agregación: Un Usuario puede estar asociado a varios Préstamos.

Genericidad: Uso de clases genéricas para almacenar muestra la lista de empleados y animales.

Interfaces : implementación de una interfaz gestionable (pantalla principal) para operaciones CRUD.

Persistencia: En la base de datos .

Patrones de diseño: Usamos diferentes diseños de patrones como imágenes de fondo .

Implementación en Java

Código Fuente

Clase principal: zoológico

```
public class Zoologico {
   private String nombreZoologico;
   private String ubicacion;
   private int capacidadMax;
   private String horario;
   private int nroSecciones;
   private Gestor<Seccion> secciones;
   private int nroEmpleados;
   private Gestor<Empleado> empleados;
   private int nroAnimales;
   private Gestor<Animal> animales;

   public Zoologico(String nombreZoologico, String ubicacion, int capacidadMax, String
horario) {
      this.nombreZoologico = nombreZoologico;
      this.ubicacion = ubicacion;
      this.capacidadMax = capacidadMax;
      this.horario = horario;
      this.secciones = new Gestor<>(10);
```

```
    this.empleados = new Gestor<>(10);
    this.animales = new Gestor<>(10);
}

// Métodos para agregar secciones, empleados y animales
public void agregarSeccion(Seccion seccion) {
    secciones.agregar(seccion);
    nroSecciones++;
}

public void agregarEmpleado(Empleado empleado) {
    empleados.agregar(empleado);
    nroEmpleados++;
}

public void agregarAnimal(Animal animal) {
    animales.agregar(animal);
    nroAnimales++;
}

// Getters y Setters
public String getNombreZoologico() {
    return nombreZoologico;
}

public void setNombreZoologico(String nombreZoologico) {
    this.nombreZoologico = nombreZoologico;
}

public String getUbicacion() {
    return ubicacion;
}

public void setUbicacion(String ubicacion) {
    this.ubicacion = ubicacion;
}

public int getCapacidadMax() {
    return capacidadMax;
}
```

```
public void setCapacidadMax(int capacidadMax) {
   this.capacidadMax = capacidadMax;
}

public String getHorario() {
   return horario;
}

public void setHorario(String horario) {
   this.horario = horario;
}

public int getNroSecciones() {
   return nroSecciones;
}

public void setNroSecciones(int nroSecciones) {
   this.nroSecciones = nroSecciones;
}
public Gestor<Seccion> getSecciones() {
   return secciones;
}
public void setSecciones(Gestor<Seccion> secciones) {
   this.secciones = secciones;
}
public int getNroEmpleados() {
   return nroEmpleados;
}
public void setNroEmpleados(int nroEmpleados) {
   this.nroEmpleados = nroEmpleados;
}
public Gestor<Empleado> getEmpleados() {
   return empleados;
}
public void setEmpleados(Gestor<Empleado> empleados) {
   this.empleados = empleados;
}
public int getNroAnimales() {
   return nroAnimales;
}
```

```
    public void setNroAnimales(int nroAnimales) {
        this.nroAnimales = nroAnimales;
    }
    public Gestor<Animal> getAnimales() {
        return animales;
    }
    public void setAnimales(Gestor<Animal> animales) {
        this.animales = animales;
    }
}
```

Clase derivada : Animal

```
public abstract class Animal {
    private String nombre;
    private String especie;
    private int edad;
    private double peso;
    private String estadoSalud;

    public Animal(String nombre, String especie, int edad, double peso, String estadoSalud)
{
        this.nombre = nombre;
        this.especie = especie;
        this.edad = edad;
        this.peso = peso;
        this.estadoSalud = estadoSalud;
    }

    // Getters y Setters
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getEspecie() {
        return especie;
    }
    public void setEspecie(String especie) {
        this.especie = especie;
    }
```

```
    public int getEdad() {
        return edad;
    }
    public void setEdad(int edad) {
        this.edad = edad;
    }
    public double getPeso() {
        return peso;
    }
    public void setPeso(double peso) {
        this.peso = peso;
    }
    public String getEstadoSalud() {
        return estadoSalud;
    }
    public void setEstadoSalud(String estadoSalud) {
        this.estadoSalud = estadoSalud;
    }
    // Método abstracto para hacer que cada animal pueda hacer un sonido
    public abstract void hacerSonido();
}
```

Clase Derivada: empleado

```
public class Empleado {
    protected String nombre;
    protected int edad;
    protected String ci;
    protected String sexo;
    protected String especialidad;
    protected String turno;
    protected double sueldo;
    public Empleado(String nombre, int edad, String ci, String sexo, String especialidad,
String turno, double sueldo) {
        this.nombre = nombre;
        this.edad = edad;
        this.ci = ci;
        this.sexo = sexo;
        this.especialidad = especialidad;
        this.turno = turno;
        this.sueldo = sueldo;
    }
```

```
// Getters y Setters
public String getNombre() {
   return nombre;
}
public void setNombre(String nombre) {
   this.nombre = nombre;
}
public int getEdad() {
   return edad;
}
public void setEdad(int edad) {
   this.edad = edad;
}
public String getCi() {
   return ci;
}
public void setCi(String ci) {
   this.ci = ci;
}
public String getSexo() {
   return sexo;
}
public void setSexo(String sexo) {
   this.sexo = sexo;
}
public String getEspecialidad() {
   return especialidad;
}
public void setEspecialidad(String especialidad) {
   this.especialidad = especialidad;
}
public String getTurno() {
   return turno;
}
public void setTurno(String turno) {
   this.turno = turno;
}
public double getSueldo() {
   return sueldo;
}
```

```
    public void setSueldo(double sueldo) {
        this.sueldo = sueldo;
    }
}
```

Clase : habitad

```
public class Habitat {
    private Cuidador cuidador;
    private String nombreHabitat;
    private double temperatura;
    private String ubicacion;
    private int capacidadMax;
    private int nroAnimales;
    private Gestor<Animal> gestorAnimales;
    private int nroCuidadores;
    private Gestor<Cuidador> gestorCuidadores;
    public Habitat(Cuidador cuidador, String nombreHabitat, double temperatura, String
ubicacion, int capacidadMax, int nroAnimales, Gestor<Animal> gestorAnimales, int
nroCuidadores, Gestor<Cuidador> gestorCuidadores) {
        this.cuidador = cuidador;
        this.nombreHabitat = nombreHabitat;
        this.temperatura = temperatura;
        this.ubicacion = ubicacion;
        this.capacidadMax = capacidadMax;
        this.nroAnimales = nroAnimales;
        this.gestorAnimales = gestorAnimales;
        this.nroCuidadores = nroCuidadores;
        this.gestorCuidadores = gestorCuidadores;
    }
    public void cambiarTemperaturaSegunEstacion(String estacion) {
        switch (estacion.toLowerCase()) {
            case "primavera":
                setTemperatura(20);
                break;
            case "invierno":
                setTemperatura(60);
                break;
            case "verano":
                setTemperatura(15);
                break;
```

```
        case "otono":
            setTemperatura(40);
            break;
        default:
            System.out.println("Estación no válida.");
    }
}
// Getters y Setters
public Cuidador getCuidador() {
    return cuidador;
}
public void setCuidador(Cuidador cuidador) {
    this.cuidador = cuidador;
}
public String getNombreHabitat() {
    return nombreHabitat;
}
public void setNombreHabitat(String nombreHabitat) {
    this.nombreHabitat = nombreHabitat;
}
public double getTemperatura() {
    return temperatura;
}
public void setTemperatura(double temperatura) {
    this.temperatura = temperatura;
}
public String getUbicacion() {
    return ubicacion;
}
public void setUbicacion(String ubicacion) {
    this.ubicacion = ubicacion;
}
public int getCapacidadMax() {
    return capacidadMax;
}
public void setCapacidadMax(int capacidadMax) {
    this.capacidadMax = capacidadMax;
}
public int getNroAnimales() {
    return nroAnimales;
}
```

```
    public void setNroAnimales(int nroAnimales) {
        this.nroAnimales = nroAnimales;
    }
    public Gestor<Animal> getGestorAnimales() {
        return gestorAnimales;
    }
    public void setGestorAnimales(Gestor<Animal> gestorAnimales) {
        this.gestorAnimales = gestorAnimales;
    }
    public int getNroCuidadores() {
        return nroCuidadores;
    }
    public void setNroCuidadores(int nroCuidadores) {
        this.nroCuidadores = nroCuidadores;
    }
    public Gestor<Cuidador> getGestorCuidadores() {
        return gestorCuidadores;
    }
    public void setGestorCuidadores(Gestor<Cuidador> gestorCuidadores) {
        this.gestorCuidadores = gestorCuidadores;
    }
}
```

Clase gestor:

```
import java.util.ArrayList;
import java.util.List;

public class Gestor<T> {
    private List<T> lista;
    private int capacidad;

    public Gestor(int capacidad) {
        this.capacidad = capacidad;
        this.lista = new ArrayList<>(capacidad);
    }
    public void agregar(T objeto) {
        if (lista.size() < capacidad) {
            lista.add(objeto);
        } else {
            System.out.println("Capacidad máxima alcanzada.");
        }
```

```
    }
    public void eliminar(T objeto) {
        lista.remove(objeto);
    }
    public T obtener(int index) {
        if (index >= 0 && index < lista.size()) {
            return lista.get(index);
        } else {
            System.out.println("Índice fuera de rango.");
            return null;  // O devolver algún valor por defecto, según lo necesites
        }
    }
    public int size() {
        return lista.size();
    }
    public List<T> getLista() {
        return lista;
    }
    public void setLista(List<T> lista) {
        this.lista = lista;
    }
}
```

Clase heredada : encargado :: empleado

```
public class Encargado extends Empleado {
    private String cargo;

    public Encargado(String nombre, int edad, String ci, String sexo, String especialidad,
String turno, double sueldo, String cargo) {
        super(nombre, edad, ci, sexo, especialidad, turno, sueldo);
        this.cargo = cargo;
    }

    public String getCargo() {
        return cargo;
    }

    public void setCargo(String cargo) {
        this.cargo = cargo;
    }
}
```

Clase heredada : cuidador :: empleado

```
public class Cuidador extends Empleado {
   private String especialidadAnimal;

   public Cuidador(String nombre, int edad, String ci, String sexo, String
especialidadCuidador, String turno, double sueldo, String especialidadAnimal) {
      super(nombre, edad, ci, sexo, especialidadCuidador, turno, sueldo);
      this.especialidadAnimal = especialidadAnimal;
   }

   public String getEspecialidadAnimal() {
      return especialidadAnimal;
   }

   public void setEspecialidadAnimal(String especialidadAnimal) {
      this.especialidadAnimal = especialidadAnimal;
   }
}
```

Clase heredada : mamífero::animal

```
public class Mamifero extends Animal {
   private String tipoAlimentacion;

   public Mamifero(String nombre, String especie, int edad, double peso, String
estadoSalud, String tipoAlimentacion) {
      super(nombre, especie, edad, peso, estadoSalud);
      this.tipoAlimentacion = tipoAlimentacion;
   }

   public String getTipoAlimentacion() {
      return tipoAlimentacion;
   }

   public void setTipoAlimentacion(String tipoAlimentacion) {
      this.tipoAlimentacion = tipoAlimentacion;
   }

   @Override
   public void hacerSonido() {
      System.out.println("El mamífero hace un sonido característico.");
   }
```

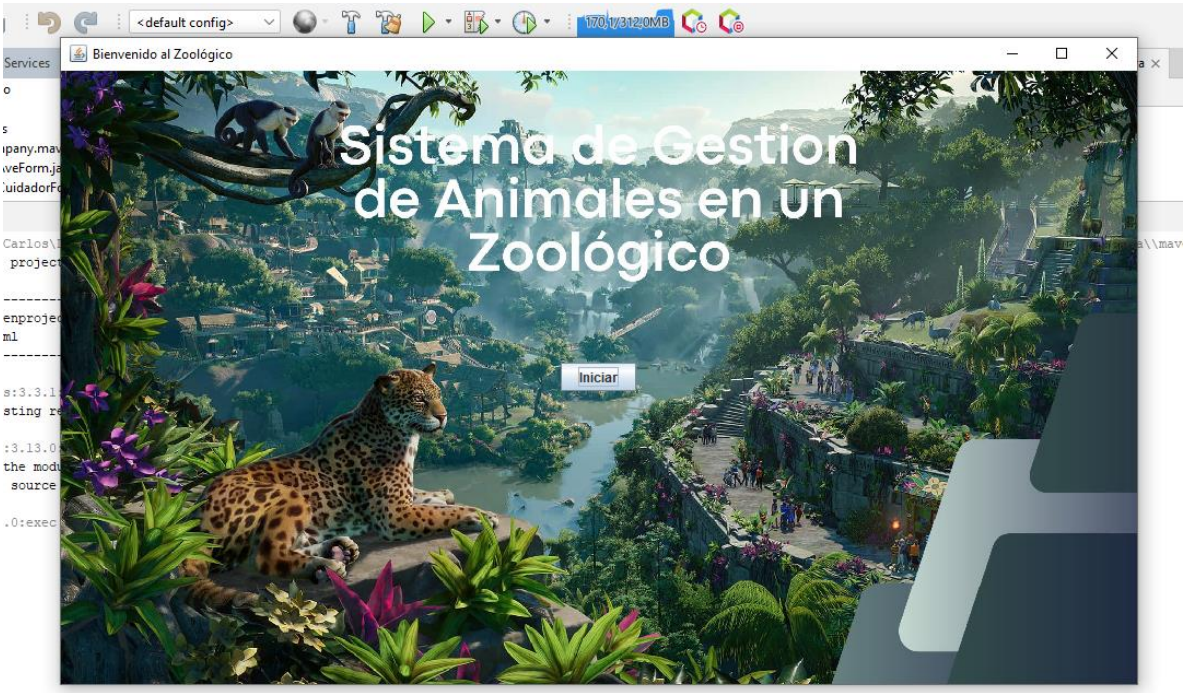
```
}
```
Clase heredada :Reptil::animal
```
public class Reptil extends Animal {
    private String tipoPiel;

    public Reptil(String nombre, String especie, int edad, double peso, String estadoSalud,
String tipoPiel) {
        super(nombre, especie, edad, peso, estadoSalud);
        this.tipoPiel = tipoPiel;
    }

    public String getTipoPiel() {
        return tipoPiel;
    }

    public void setTipoPiel(String tipoPiel) {
        this.tipoPiel = tipoPiel;
    }

    @Override
    public void hacerSonido() {
        System.out.println("El reptil hace un sonido característico.");
    }
}
```
Clase heredada :Pez ::animal
```
public class Pez extends Animal {
    private String tipoAgua;

    public Pez(String nombre, String especie, int edad, double peso, String estadoSalud,
String tipoAgua) {
        super(nombre, especie, edad, peso, estadoSalud);
        this.tipoAgua = tipoAgua;
    }

    public String getTipoAgua() {
        return tipoAgua;
    }

    public void setTipoAgua(String tipoAgua) {
        this.tipoAgua = tipoAgua;
    }
```

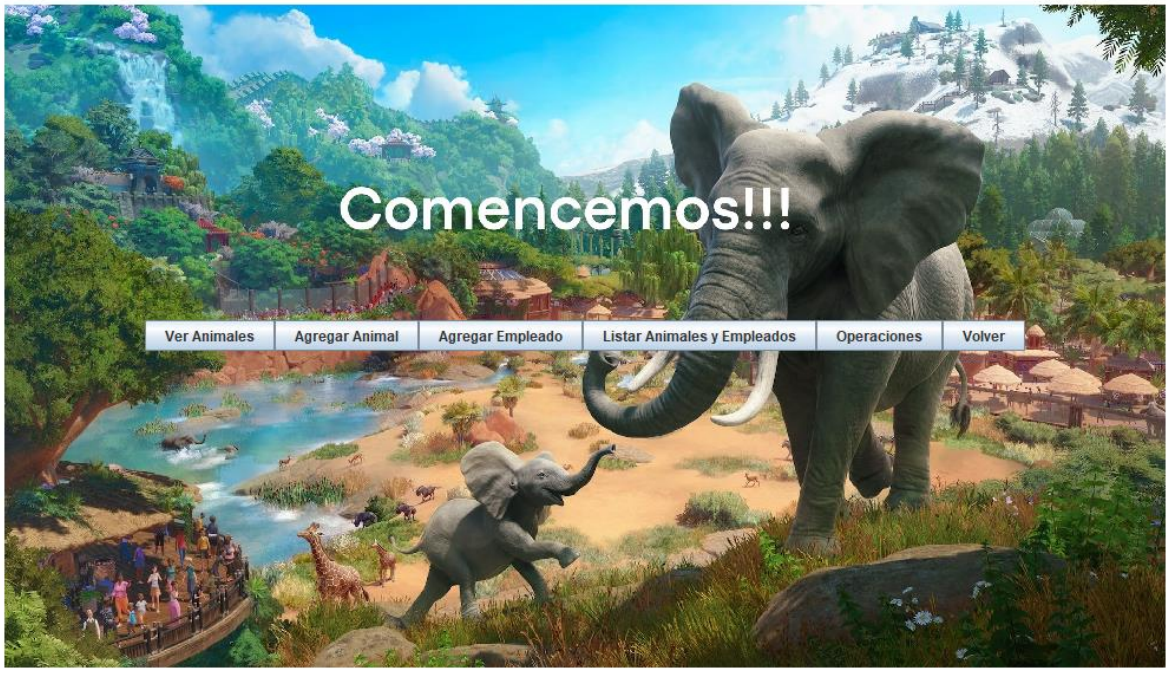
```
    @Override
    public void hacerSonido() {
        System.out.println("El pez hace burbujeas.");
    }
}
```

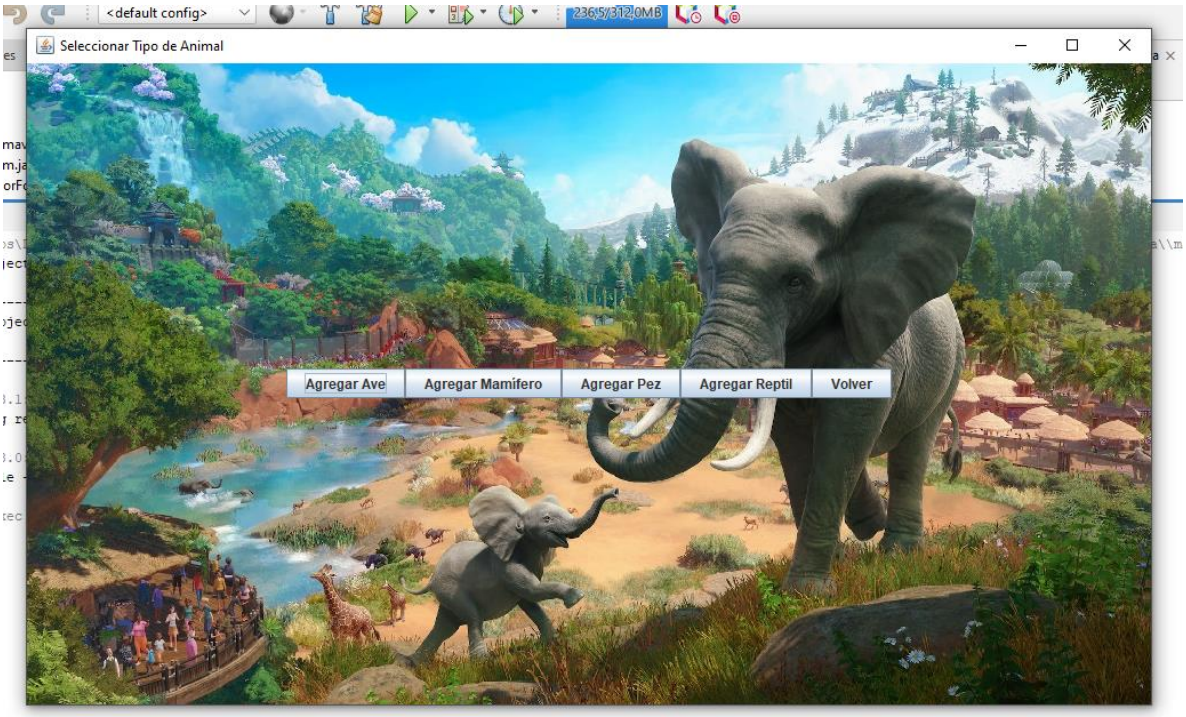
Clase heredada :Ave ::animal

```
public class Ave extends Animal {
    private String tipoVuelo;

    public Ave(String nombre, String especie, int edad, double peso, String estadoSalud,
String tipoVuelo) {
        super(nombre, especie, edad, peso, estadoSalud);
        this.tipoVuelo = tipoVuelo;
    }

    public String getTipoVuelo() {
        return tipoVuelo;
    }

    public void setTipoVuelo(String tipoVuelo) {
        this.tipoVuelo = tipoVuelo;
    }

    @Override
    public void hacerSonido() {
        System.out.println("El ave canta.");
    }
}
```

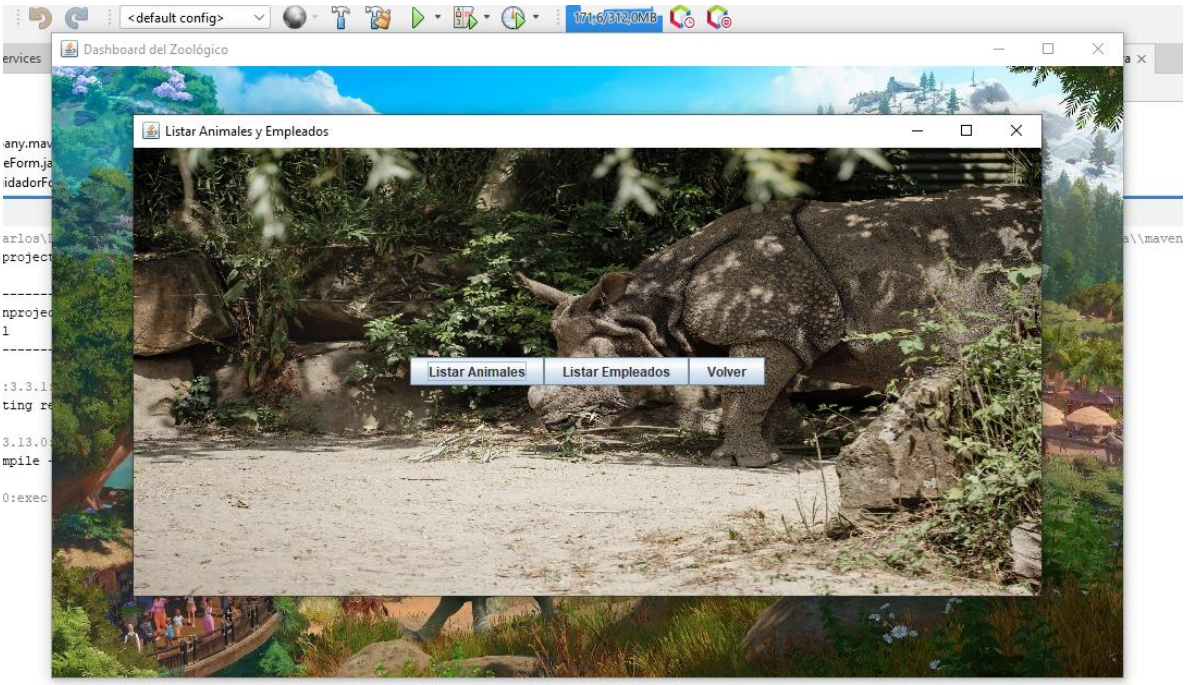
Diseño de Interfaces

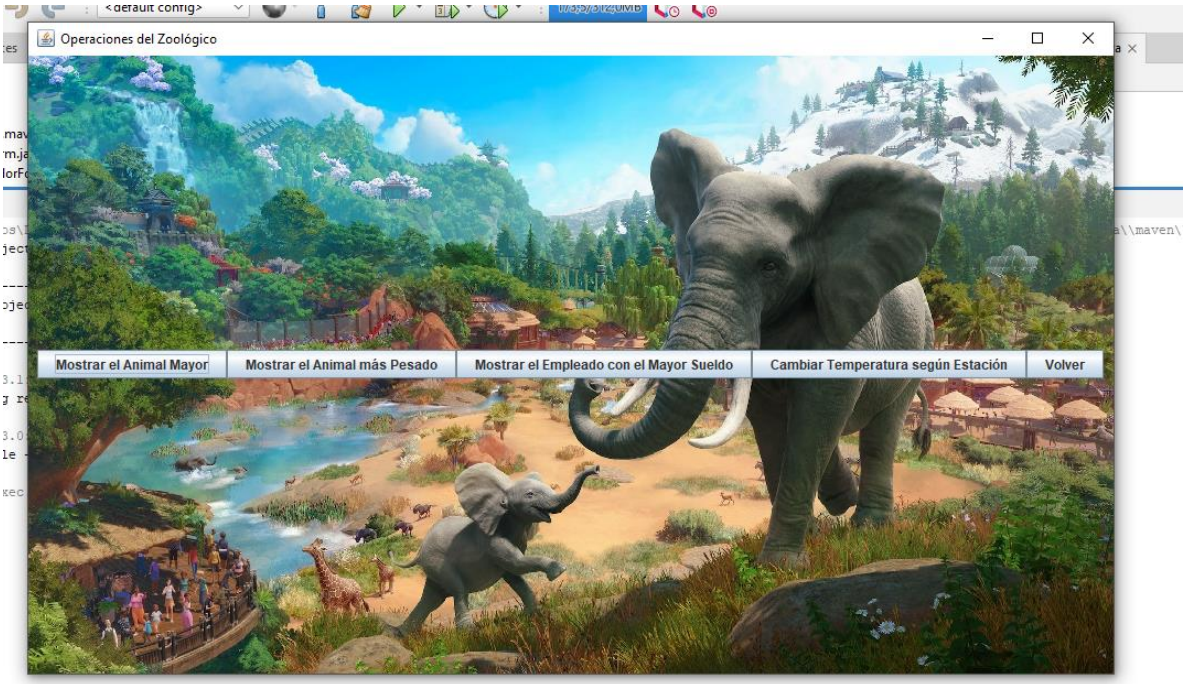





Seleccionar Tipo de Animal
Agregar Ave
Agregar Mamífero
Agregar Pez
Agregar Reptil
Volver


Agregar Cuidador
Agregar Encargado
Volver





Manejo de Archivos

Se implementará la persistencia de datos utilizando conexiones a base de datos MySql

-- phpMyAdmin SQL Dump
-- version 5.2.1

```
-- https://www.phpmyadmin.net/
--
-- Servidor: 127.0.0.1
-- Tiempo de generación: 28-01-2025 a las 05:49:20
-- Versión del servidor: 10.4.32-MariaDB
-- Versión de PHP: 8.0.30

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Base de datos: `zooproy`
--

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `animales`
--

CREATE TABLE `animales` (
  `id` int(11) NOT NULL,
  `nombre` varchar(50) NOT NULL,
  `especie` varchar(50) NOT NULL,
  `edad` int(11) NOT NULL,
  `peso` double NOT NULL,
  `estadoSalud` varchar(50) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Volcado de datos para la tabla `animales`
--

INSERT INTO `animales` (`id`, `nombre`, `especie`, `edad`, `peso`, `estadoSalud`) VALUES
```

```
(1, 'pepillo', 'kakatua', 5, 12, 'bueno'),
(2, 'coco', 'reptil', 21, 120, 'bueno');

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `aves`
--

CREATE TABLE `aves` (
  `id` int(11) NOT NULL,
  `id_animal` int(11) DEFAULT NULL,
  `tipoVuelo` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Volcado de datos para la tabla `aves`
--

INSERT INTO `aves` (`id`, `id_animal`, `tipoVuelo`) VALUES
(1, 1, 'sdawf');

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `cuidadores`
--

CREATE TABLE `cuidadores` (
  `id` int(11) NOT NULL,
  `nombre` varchar(50) NOT NULL,
  `edad` int(11) NOT NULL,
  `ci` varchar(20) NOT NULL,
  `sexo` varchar(10) NOT NULL,
  `especialidad` varchar(50) NOT NULL,
  `turno` varchar(20) NOT NULL,
  `sueldo` double NOT NULL,
  `especialidadAnimal` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
```

```
-- Volcado de datos para la tabla `cuidadores`
--

INSERT INTO `cuidadores` (`id`, `nombre`, `edad`, `ci`, `sexo`, `especialidad`, `turno`,
`sueldo`, `especialidadAnimal`) VALUES
(8, 'simon', 21, '3534', 'masculino', 'veterinaria', 'tarde', 5000, 'mamiferos');

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `empleados`
--

CREATE TABLE `empleados` (
  `id` int(11) NOT NULL,
  `nombre` varchar(50) NOT NULL,
  `edad` int(11) NOT NULL,
  `ci` varchar(20) NOT NULL,
  `sexo` varchar(10) NOT NULL,
  `especialidad` varchar(50) DEFAULT NULL,
  `turno` varchar(20) DEFAULT NULL,
  `sueldo` double NOT NULL,
  `puesto` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Volcado de datos para la tabla `empleados`
--

INSERT INTO `empleados` (`id`, `nombre`, `edad`, `ci`, `sexo`, `especialidad`, `turno`,
`sueldo`, `puesto`) VALUES
(1, 'aasdda', 21, '2345', 'masculino', 'sadaw', 'tarde', 2312, 'Cuidador'),
(2, 'aasdda', 21, '2345', 'masculino', 'sadaw', 'tarde', 2312, 'Cuidador'),
(3, 'asaw', 21, '2311', 'dfaa', 'dawcas', 'asad', 2310, 'Cuidador'),
(4, 'dfsefsdf', 21, '34532', 'gdrgd', 'sdfsef', 'hfhfghg', 3435, 'Cuidador'),
(5, 'aasdda', 21, '2345', 'masculino', 'sadaw', 'tarde', 2312, 'Cuidador'),
(6, 'juanito', 54, '342', 'faa', 'sdasd', 'asdas', 4567, 'Cuidador'),
(7, 'kick', 21, '43323', 'aerasa', 'sdawdaw', 'sdawd', 1234, 'Encargado'),
(8, 'simon', 21, '3534', 'masculino', 'veterinaria', 'tarde', 5000, 'Cuidador'),
(9, 'Aramis', 34, '34231', 'masculino', 'reptiles', 'mañana', 6000, 'Encargado');
```

```
-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `encargados`
--

CREATE TABLE `encargados` (
  `id` int(11) NOT NULL,
  `nombre` varchar(50) NOT NULL,
  `edad` int(11) NOT NULL,
  `ci` varchar(20) NOT NULL,
  `sexo` varchar(10) NOT NULL,
  `especialidad` varchar(50) NOT NULL,
  `turno` varchar(20) NOT NULL,
  `sueldo` double NOT NULL,
  `cargo` varchar(50) NOT NULL,
  `tipoEmpleado` varchar(20) DEFAULT 'Encargado'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Volcado de datos para la tabla `encargados`
--

INSERT INTO `encargados` (`id`, `nombre`, `edad`, `ci`, `sexo`, `especialidad`, `turno`,
`sueldo`, `cargo`, `tipoEmpleado`) VALUES
(9, 'Aramis', 34, '34231', 'masculino', 'reptiles', 'mañana', 6000, 'gerente', 'Encargado');

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `mamiferos`
--

CREATE TABLE `mamiferos` (
  `id` int(11) NOT NULL,
  `id_animal` int(11) DEFAULT NULL,
  `tipoAlimentacion` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

-- --------------------------------------------------------
```

```
--
-- Estructura de tabla para la tabla `peces`
--

CREATE TABLE `peces` (
  `id` int(11) NOT NULL,
  `id_animal` int(11) DEFAULT NULL,
  `tipoAgua` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `reptiles`
--

CREATE TABLE `reptiles` (
  `id` int(11) NOT NULL,
  `id_animal` int(11) DEFAULT NULL,
  `tipoPiel` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Volcado de datos para la tabla `reptiles`
--

INSERT INTO `reptiles` (`id`, `id_animal`, `tipoPiel`) VALUES
(1, 2, 'escamoso');

--
-- Índices para tablas volcadas
--

--
-- Indices de la tabla `animales`
--
ALTER TABLE `animales`
  ADD PRIMARY KEY (`id`);

--
-- Indices de la tabla `aves`
```

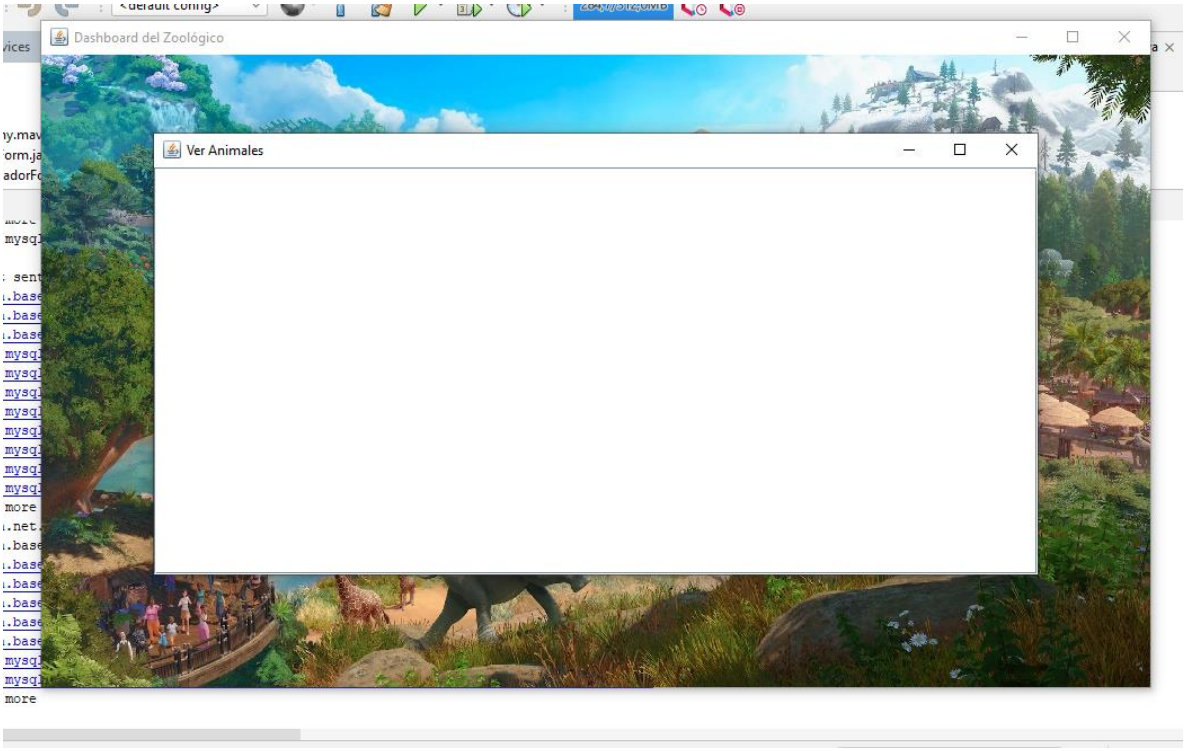
```
--
ALTER TABLE `aves`
  ADD PRIMARY KEY (`id`),
  ADD KEY `id_animal` (`id_animal`);

--
-- Indices de la tabla `cuidadores`
--
ALTER TABLE `cuidadores`
  ADD PRIMARY KEY (`id`);

--
-- Indices de la tabla `empleados`
--
ALTER TABLE `empleados`
  ADD PRIMARY KEY (`id`);

--
-- Indices de la tabla `encargados`
--
ALTER TABLE `encargados`
  ADD PRIMARY KEY (`id`);

--
-- Indices de la tabla `mamiferos`
--
ALTER TABLE `mamiferos`
  ADD PRIMARY KEY (`id`),
  ADD KEY `id_animal` (`id_animal`);

--
-- Indices de la tabla `peces`
--
ALTER TABLE `peces`
  ADD PRIMARY KEY (`id`),
  ADD KEY `id_animal` (`id_animal`);

--
-- Indices de la tabla `reptiles`
--
ALTER TABLE `reptiles`
```

```
  ADD PRIMARY KEY (`id`),
  ADD KEY `id_animal` (`id_animal`);

--
-- AUTO_INCREMENT de las tablas volcadas
--

--
-- AUTO_INCREMENT de la tabla `animales`
--
ALTER TABLE `animales`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

--
-- AUTO_INCREMENT de la tabla `aves`
--
ALTER TABLE `aves`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

--
-- AUTO_INCREMENT de la tabla `cuidadores`
--
ALTER TABLE `cuidadores`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=9;

--
-- AUTO_INCREMENT de la tabla `empleados`
--
ALTER TABLE `empleados`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=10;

--
-- AUTO_INCREMENT de la tabla `encargados`
--
ALTER TABLE `encargados`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=10;

--
-- AUTO_INCREMENT de la tabla `mamiferos`
--
ALTER TABLE `mamiferos`
```

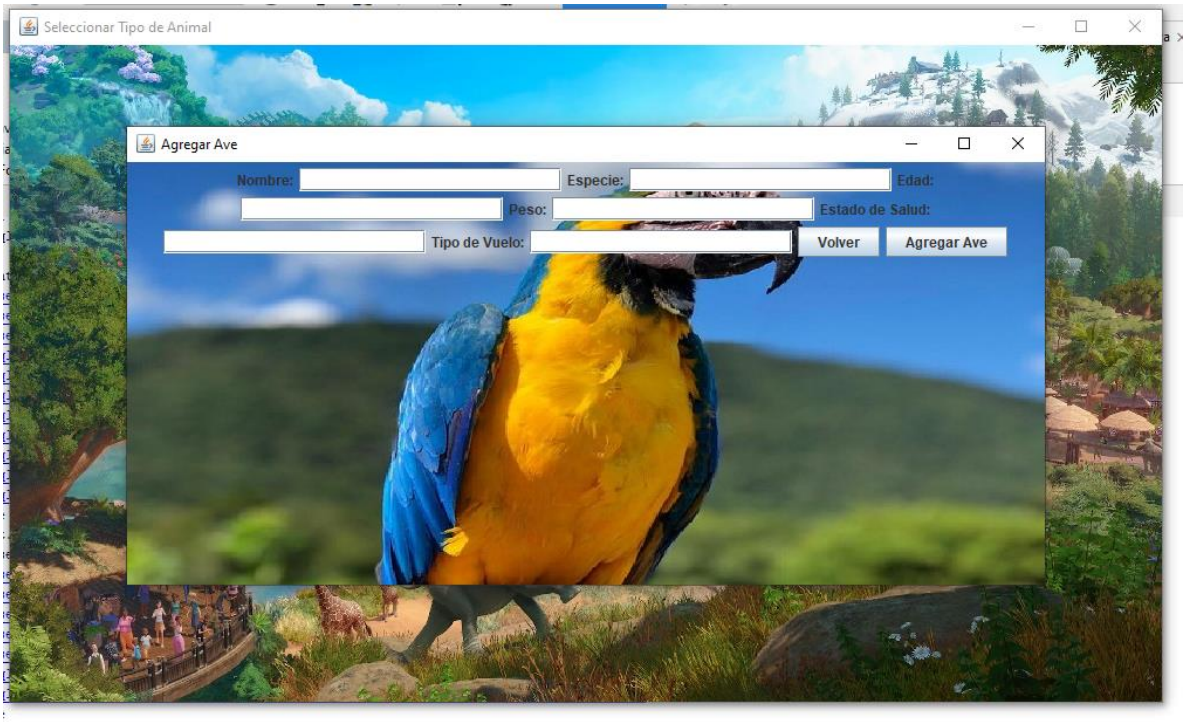
```
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT de la tabla `peces`
--
ALTER TABLE `peces`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT de la tabla `reptiles`
--
ALTER TABLE `reptiles`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

--
-- Restricciones para tablas volcadas
--

--
-- Filtros para la tabla `aves`
--
ALTER TABLE `aves`
  ADD CONSTRAINT `aves_ibfk_1` FOREIGN KEY (`id_animal`) REFERENCES `animales` (`id`);

--
-- Filtros para la tabla `mamiferos`
--
ALTER TABLE `mamiferos`
  ADD CONSTRAINT `mamiferos_ibfk_1` FOREIGN KEY (`id_animal`) REFERENCES `animales`
(`id`);

--
-- Filtros para la tabla `peces`
--
ALTER TABLE `peces`
  ADD CONSTRAINT `peces_ibfk_1` FOREIGN KEY (`id_animal`) REFERENCES `animales`
(`id`);

--
-- Filtros para la tabla `reptiles`
--
```

```
ALTER TABLE `reptiles`
  ADD CONSTRAINT `reptiles_ibfk_1` FOREIGN KEY (`id_animal`) REFERENCES `animales`
(`id`);
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

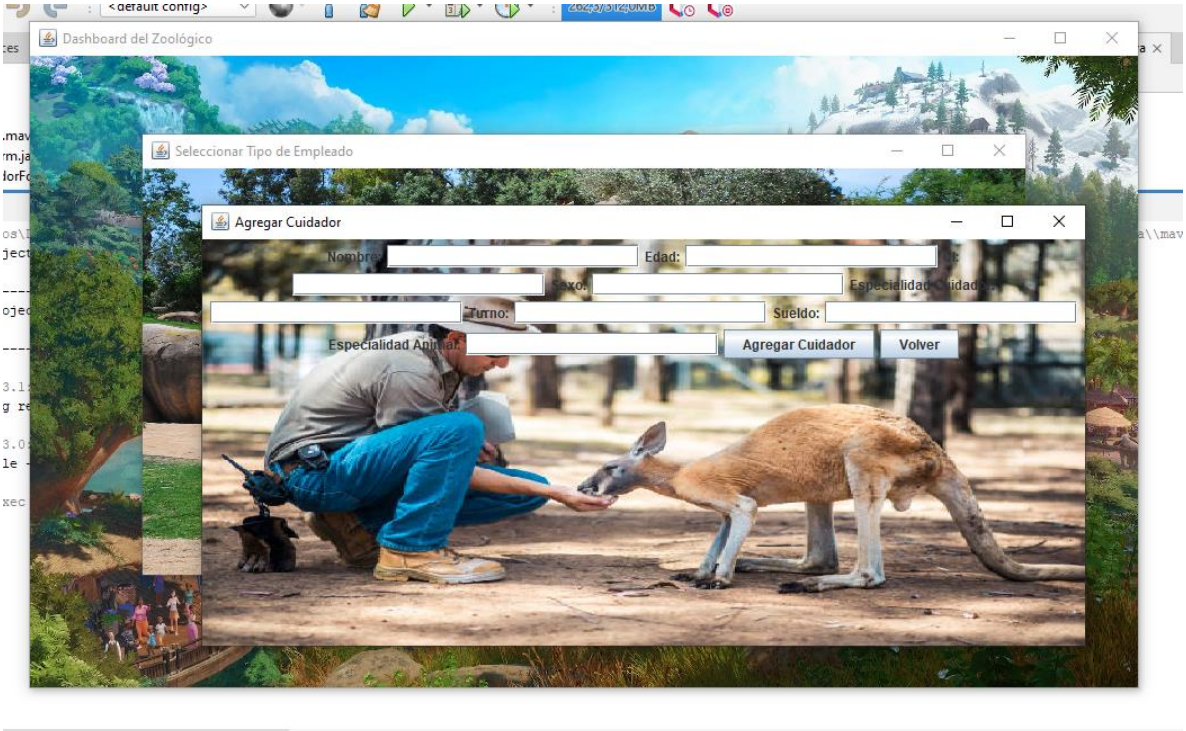
Pruebas del Sistema

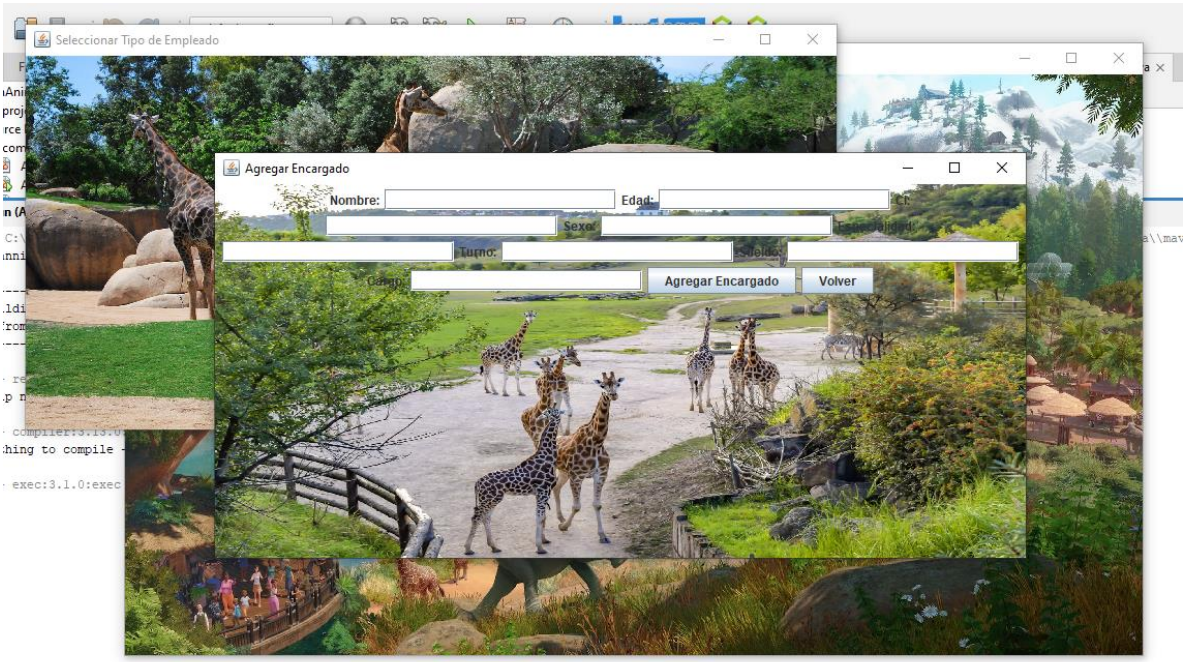


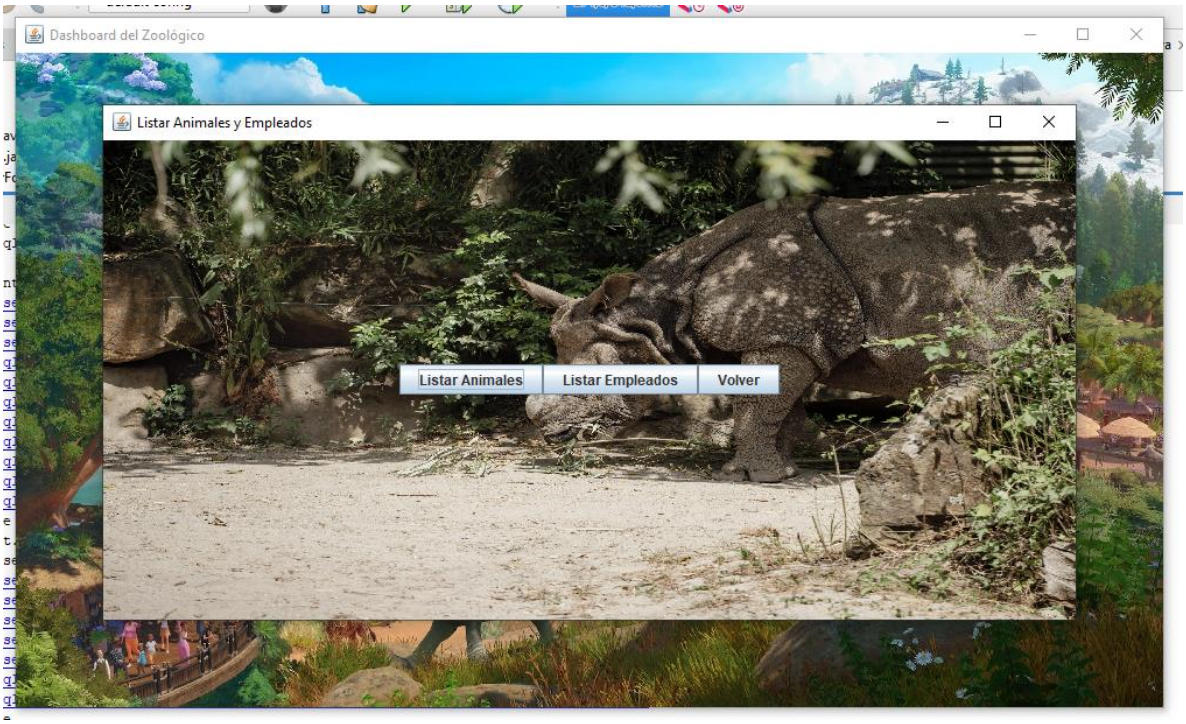
En este caso no hay datos ingresados

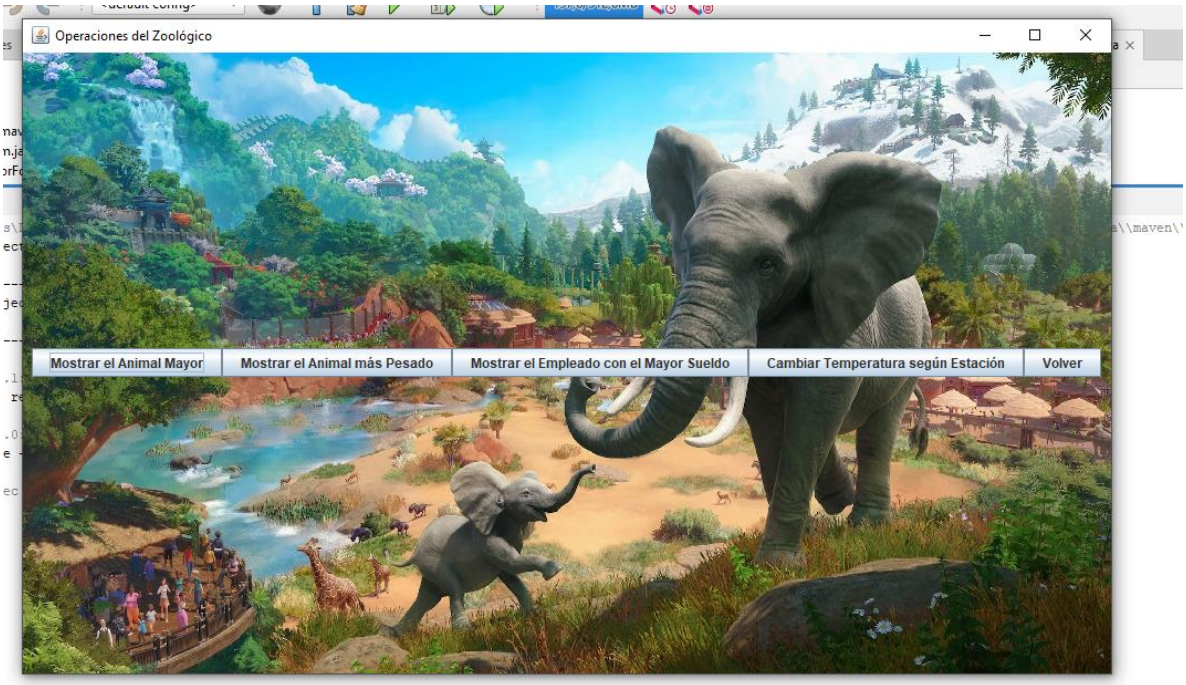

Seleccionar Tipo de Animal
Agregar Ave
Nombre:
Especie:
Edad:
Peso:
Estado de Salud:
Tipo de Vuelo:
Volver
Agregar Ave

Agregar ave


Dashboard del Zoológico
Seleccionar Tipo de Empleado
Agregar Cuidador
Nombre:
Edad:
Turno:
Sueldo:
Especialidad Animal:
Agregar Cuidador
Volver


Seleccionar Tipo de Empleado
Agregar Encargado
Nombre:
Edad:
Sexo:
Turno:
Sueldo:
Cargo:
Agregar Encargado
Volver


Dashboard del Zoológico
Listar Animales y Empleados
Listar Animales
Listar Empleados
Volver



Conclusión

El proyecto desarrollado cubre los aspectos fundamentales y avanzados de la POO, ofreciendo una solución escalable y modular para la gestión de animales de un zoologico. La integración de patrones de diseño mejora la flexibilidad y mantenibilidad del sistema.