

Escuela de Formación de Tecnólogos

Programación Orientada a Objetos

Título

SIVET – Sistema Veterinario

Integrantes

Odaliz Balseca

Andrés Panchi

Carrera:

Tecnología Superior en Desarrollo de Software

Docente:

Ing. Yadira Franco

Fecha:

03 de agosto del 2025

Índice

Introducción	3
Objetivos	4
Objetivos Específicos	4
Justificación.....	5
Alcance y requerimientos – roles	5
Alcance Funcional	5
INCLUYE (Dentro del Alcance):	5
NO INCLUYE (Fuera del Alcance):.....	6
ROLES DEL SISTEMA	7
REQUERIMIENTOS FUNCIONALES	7
REQUERIMIENTOS NO FUNCIONALES	8
REQUERIMIENTOS TECNOLÓGICOS.....	8
Diseño de Interfaz (Mockup)	9
Pantalla Principal (Usuario Externo).....	9
Modelado de Base de Datos	12
Organización del Código	21
Ejecución Completa del Proyecto.....	22
Flujo de Ejecución.....	22
Casos de Uso Principales	22
Explicación del Desarrollo	22
Trabajo Colaborativo en GitHub.....	30
Conclusiones	30
Recomendaciones y Trabajos Futuros.....	31

Introducción

El proyecto SIVET (Sistema Integral de Gestión Veterinaria) surge como respuesta a esta necesidad, desarrollando una solución tecnológica integral que permite la digitalización y automatización de los procesos administrativos y clínicos de una clínica veterinaria. Este sistema ha sido diseñado aplicando los principios fundamentales de la Programación Orientada a Objetos (POO), lo que garantiza un código mantenible, escalable y reutilizable.

La implementación de SIVET representa un caso de estudio práctico que demuestra la aplicación efectiva de conceptos avanzados de programación en Java, incluyendo el uso de patrones de diseño, arquitectura MVC (Modelo-Vista-Controlador), conexión a bases de datos NoSQL en la nube, y desarrollo de interfaces gráficas profesionales mediante Swing.

El sistema contempla diferentes niveles de acceso según el rol del usuario: desde usuarios externos que pueden solicitar citas sin necesidad de registro, hasta administradores con capacidades completas de gestión y generación de reportes. Esta segmentación permite un control granular de las funcionalidades y garantiza la seguridad de la información sensible.

Además, SIVET incorpora tecnologías modernas como MongoDB Atlas para el almacenamiento en la nube, lo que proporciona ventajas significativas en términos de escalabilidad, disponibilidad y seguridad de datos. La integración con GitHub para control de versiones asegura un desarrollo ordenado y colaborativo.

Este proyecto no solo cumple con los objetivos académicos de demostrar competencias en programación orientada a objetos, sino que también ofrece una solución real y funcional que puede ser implementada en entornos veterinarios.

Objetivos

Desarrollar un sistema de escritorio utilizando Java aplicando los principios de Programación Orientada a Objetos, que permita gestionar el agendamiento de citas veterinarias, registro de mascotas, control de usuarios y generación de reportes, con conexión a base de datos MySQL en la nube.

Objetivos Específicos

- Implementar clases y subclases aplicando herencia, encapsulamiento, polimorfismo y abstracción para la gestión veterinaria.
- Conectar el sistema a una base de datos remota MySQL desplegada en la nube.
- Crear un diseño visual profesional y funcional mediante Java Swing con navegación intuitiva.
- Organizar el código utilizando paquetes siguiendo el patrón MVC (Modelo-Vista-Controlador).
- Implementar sistema de autenticación con roles diferenciados (Administrador y Asistente).
- Generar reportes estadísticos y control completo de citas médicas con validaciones robustas.
- Aplicar principios SOLID y buenas prácticas de programación orientada a objetos.

Justificación

Este proyecto busca reforzar los conocimientos de POO mediante un caso práctico real que simula un sistema de gestión veterinaria. La implementación utiliza principios de diseño limpio, arquitectura MVC, conexión a bases de datos remotas y código completamente documentado. El sistema SIVET aborda una necesidad real en clínicas veterinarias pequeñas y medianas, proporcionando una solución integral para la gestión de citas, historiales clínicos y administración de usuarios.

La elección de un sistema veterinario permite demostrar conceptos avanzados de POO como:

- **Herencia:** Jerarquía de usuarios (Usuario → Administrador/Asistente)
- **Encapsulamiento:** Protección de datos sensibles como contraseñas y información médica
- **Polimorfismo:** Diferentes comportamientos según el rol del usuario
- **Abstracción:** Interfaces claras para operaciones CRUD

Alcance y requerimientos – roles

Alcance Funcional

INCLUYE (Dentro del Alcance):

Gestión de Usuarios y Autenticación

- Sistema de login con validación de credenciales
- Diferenciación de roles: Administrador, Asistente Veterinario y Usuario Externo
- Control de acceso basado en permisos por rol
- Sesión segura durante el uso del sistema

Módulo de Gestión de Propietarios

- Registro completo de propietarios con datos personales
- Actualización y consulta de información de propietarios
- Validación de datos de contacto (teléfono, email)
- Historial de propietarios registrados

Módulo de Gestión de Mascotas

- Registro detallado de mascotas (nombre, especie, raza, edad, peso)
- Asociación de mascotas con sus respectivos propietarios
- Actualización de información de mascotas
- Consulta de datos completos de pacientes

Sistema de Citas Médicas

- Agendamiento de citas para usuarios registrados
- Formulario de solicitud de citas para usuarios externos
- Gestión de estados de citas (pendiente, atendida, cancelada)
- Registro de motivos de consulta y observaciones

Seguimiento Clínico

- Registro de observaciones médicas
- Historial clínico completo por mascota
- Seguimiento de tratamientos
- Evolución del estado de salud

Gestión de Inventario

- Control de stock de medicamentos e insumos
- Alertas de productos con bajo inventario
- Registro de fechas de vencimiento
- Categorización de productos

Sistema de Reportes

- Reportes de atenciones realizadas
- Estadísticas de citas por período
- Control de vacunaciones aplicadas
- Reportes de inventario y stock

Interfaces de Usuario

- Pantalla principal para usuarios externos
- Panel administrativo para gestión completa
- Panel especializado para asistentes veterinarios
- Formularios intuitivos para cada funcionalidad

NO INCLUYE (Fuera del Alcance):

Funcionalidades Avanzadas

- Sistema de facturación y cobros
- Integración con sistemas de pago
- Aplicación móvil (solo escritorio)
- Versión web responsive
- Integración con APIs externas
- Sistema de notificaciones automáticas (SMS/Email)

ROLES DEL SISTEMA

Usuario Externo (No autenticado)

- **Acceso:** PantallaPrincipal.java
- **Funcionalidades:**
 - Visualizar servicios veterinarios disponibles
 - Llenar formulario de solicitud de cita con datos personales y motivo
 - Las solicitudes se guardan directamente en MongoDB

Asistente Veterinario

- **Acceso:** LoginForm.java → VentanaAsistente.java
- **Funcionalidades:**
 - Registrar nuevos propietarios (RegistroPropietarioForm.java)
 - Registrar mascotas con datos completos (RegistroMascotaForm.java)
 - Agendar citas para atención médica (AgendarCitaForm.java)
 - Registrar vacunaciones aplicadas (VacunacionForm.java)
 - Mantener seguimiento clínico de pacientes (SeguimientoClinicoForm.java)

Administrador

- **Acceso:** LoginForm.java → VentanaAdministrador.java
- **Funcionalidades:**
 - Generar reportes de atenciones (ReporteAtencionesPanel.java)
 - Controlar stock de medicamentos e insumos (StockPanel.java)
 - Acceso a estadísticas para toma de decisiones

REQUERIMIENTOS FUNCIONALES

ID	Rol	Requerimiento
RR01	Admin/Asistente	Iniciar sesión con usuario y contraseña para acceder según rol asignado
RR02	Administrador	Generar reportes de atención, citas y vacunas para estadísticas y toma de decisiones
RR03	Administrador	Verificar stock de medicamentos e insumos para asegurar disponibilidad
RR04	Asistente	Registrar nuevos propietarios para asociarlos con sus mascotas
RR05	Asistente	Registrar mascotas (nombre, especie, raza, edad, peso) para control clínico
RR06	Asistente	Agendar citas organizando la atención médica según disponibilidad
RR07	Asistente	Registrar vacunación para mantener calendario de vacunas actualizado

RR08	Asistente	Registrar observaciones de seguimiento para historial médico completo
RR09	Usuario Externo	Llenar formulario de cita sin registro previo
RR10	Usuario Externo	Visualizar servicios veterinarios disponibles

REQUERIMIENTOS NO FUNCIONALES

ID	Requerimiento
RR01	Diseño responsivo y profesional en ventanas Swing
RR02	Conexión estable a base de datos MongoDB remota
RR03	Código limpio, comentado y mantenible
RR04	Separación clara por paquetes y responsabilidades
RR05	Validaciones visuales en formularios
RR06	Sistema de autenticación seguro

REQUERIMIENTOS TECNOLÓGICOS

- **Lenguaje:** Java JDK 17+
- **IDE:** IntelliJ IDEA
- **Base de Datos:** MongoDB en la nube
- **Librerías:** MongoDB Driver para Java, Swing para GUI
- **Control de Versiones:** GitHub
- **Patrón de Arquitectura:** MVC (Model-View-Controller)

Diseño de Interfaz (Mockup)

The mockup shows a web interface for 'VETERINARIA SIVET'. The header includes a logo with a horse and stethoscope, and navigation links: 'Agendar', 'Login', 'Servicios', and 'Contacto'. The main content area is titled 'BIENVENIDO AGENDAR CITAS' and features a form for booking an appointment. The form includes fields for 'Nombre del Propietario:', 'Nombre de la mascota:', 'Fecha de la cita:', 'Hora:', and 'Motivo:'. A date picker is visible for the 'Fecha de la cita:' field, showing a calendar grid. At the bottom of the form are two buttons: 'Limpiar' and 'Agendar Cita'. The footer section, titled 'REDES SOCIALES', contains links to 'GMAIL', 'FACEBOOK', and 'INSTAGRAM' with their respective icons.

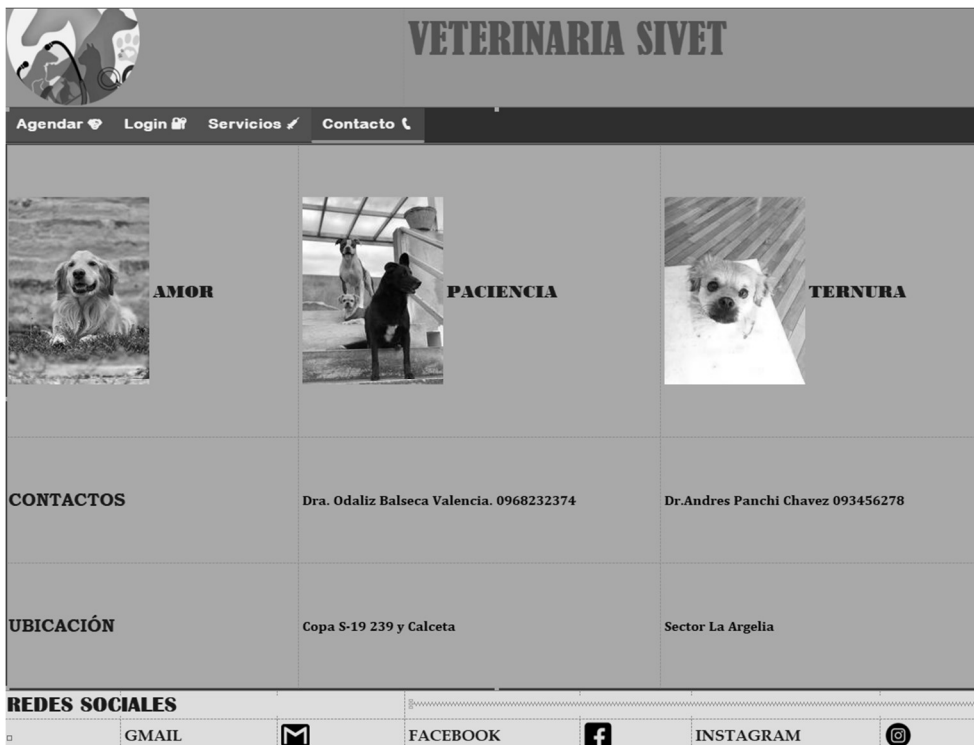
Pantalla Principal (Usuario Externo)

- Header con información de la clínica
- Sección de servicios veterinarios
- Formulario de solicitud de cita
- Información de contacto

The mockup shows the main login screen for 'VETERINARIA SIVET'. The header is identical to the previous mockup, with the logo and navigation links. The main content area features a large, stylized illustration of a cow's face. Below the illustration, there is a login form with the following elements: a 'Perfil de Usuario' dropdown menu currently set to 'Administrador', a 'Contraseña' input field, and an 'Iniciar Sesión' button. The footer section, titled 'REDES SOCIALES', contains links to 'GMAIL', 'FACEBOOK', and 'INSTAGRAM' with their respective icons.

Formulario de Login

- Campos de usuario y contraseña
- Validación de credenciales
- Redirección según rol



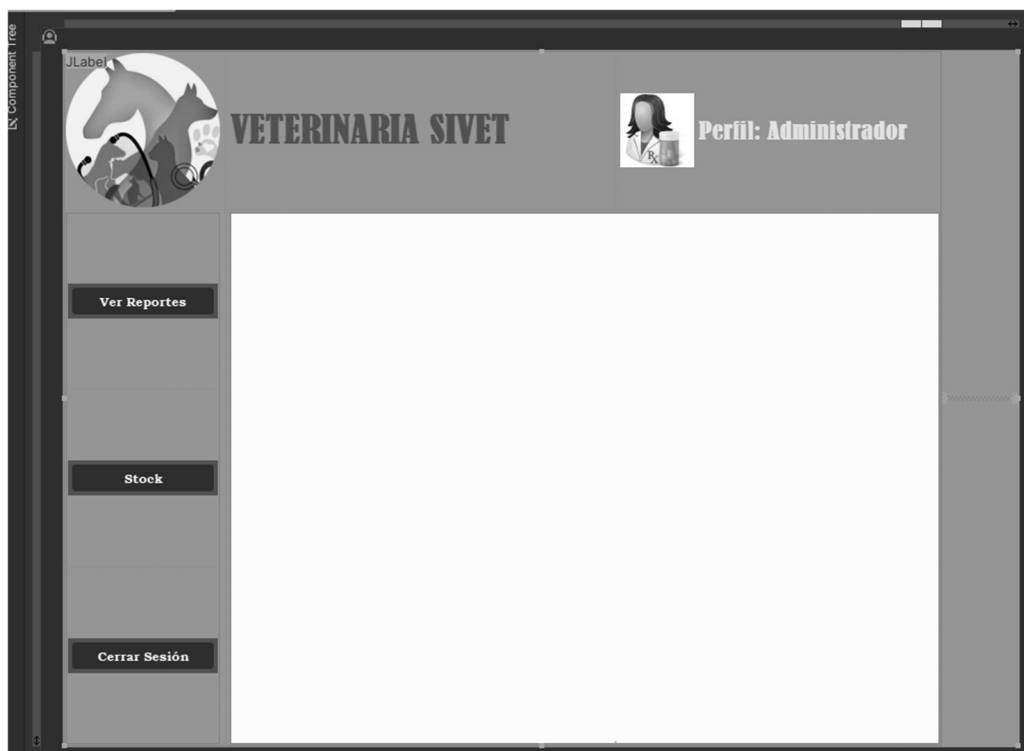
Panel de Asistente

- Menú superior: Registrar Cita | Ver Citas | Eliminar Citas | Ver Mascota | Cerrar Sesión
- Formularios dinámicos según sección seleccionada
- Tablas para visualización de registros



Panel de Administrador

- Menú superior: Ver Reportes | Stock | Cerrar Sesión
- Panel dinámico para visualización de datos
- Gráficos y estadísticas



Modelado de Base de Datos

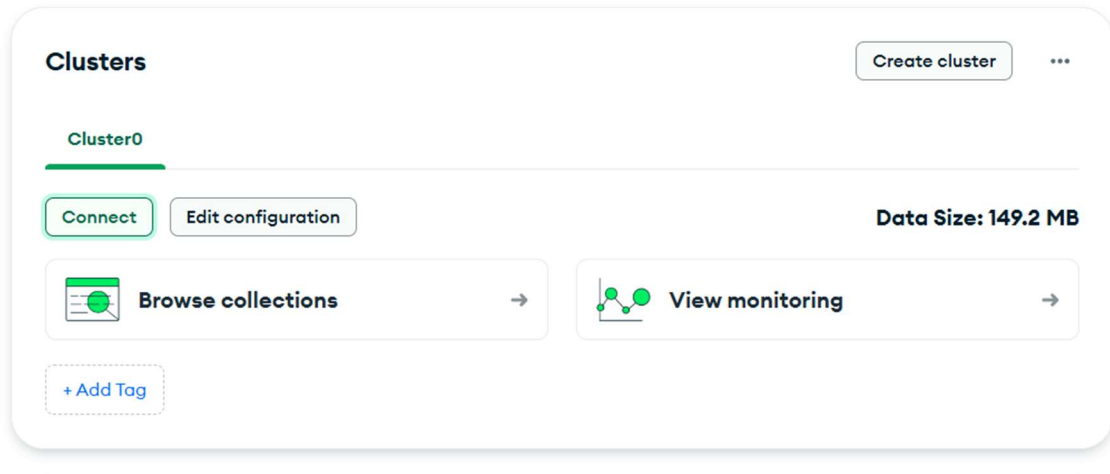
El paquete dao (Data Access Object) es una de las piezas fundamentales del sistema, ya que contiene las clases encargadas de interactuar directamente con la base de datos MongoDB. Todos los registros que se ingresan o consultan a través de la interfaz Java Swing son gestionados a través de este paquete, lo que permite una separación clara entre la lógica de la aplicación y el acceso a los datos.

Este paquete está compuesto por seis clases principales:

Conexion.java: Es la clase clave para establecer la conexión entre el sistema y MongoDB, ya sea en un entorno local o remoto como MongoDB Atlas. Define la instancia única de conexión que se reutiliza a lo largo de todo el proyecto, garantizando eficiencia y seguridad en el acceso a la base de datos.

Para establecer la conexión nos dirigimos a mongo Atlas, en la sección de clusters y connect

Overview



Connect to Cluster0



Connect to your application



Drivers

Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)



Access your data through tools



Compass

Explore, modify, and visualize your data with MongoDB's GUI



Shell

Quickly add & update data using MongoDB's Javascript command-line interface



MongoDB for VS Code

Work with your data in MongoDB directly from your VS Code environment



Atlas SQL

Easily connect SQL tools to Atlas for data analysis and visualization



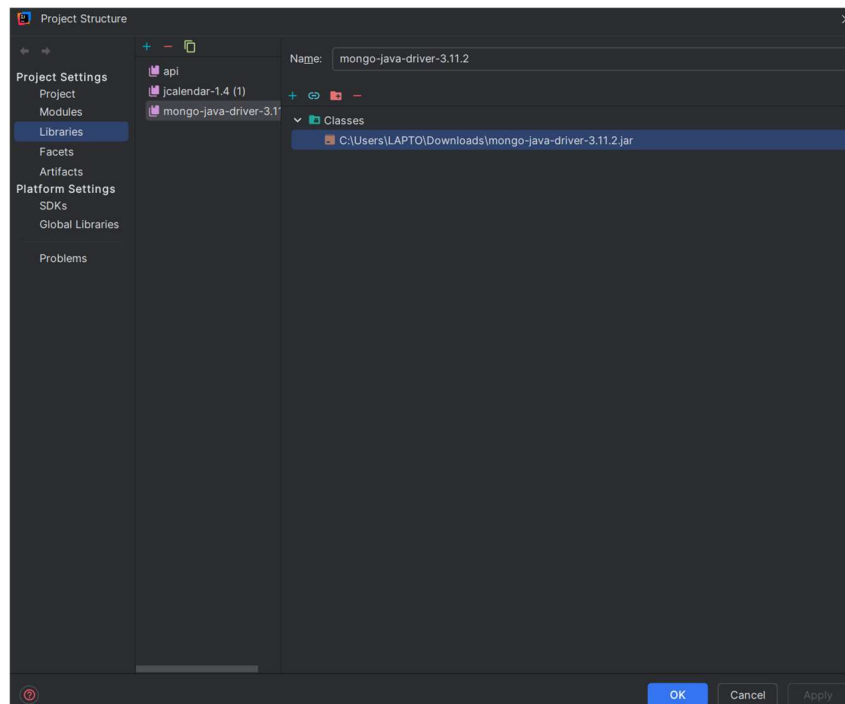
En el punto 1. Seleccionamos el lenguaje en el que estamos trabajando, en este caso en java.

En el punto 2. Debemos agregar la librería de mongo-java-driver para poder establecer la conexión sin problemas.

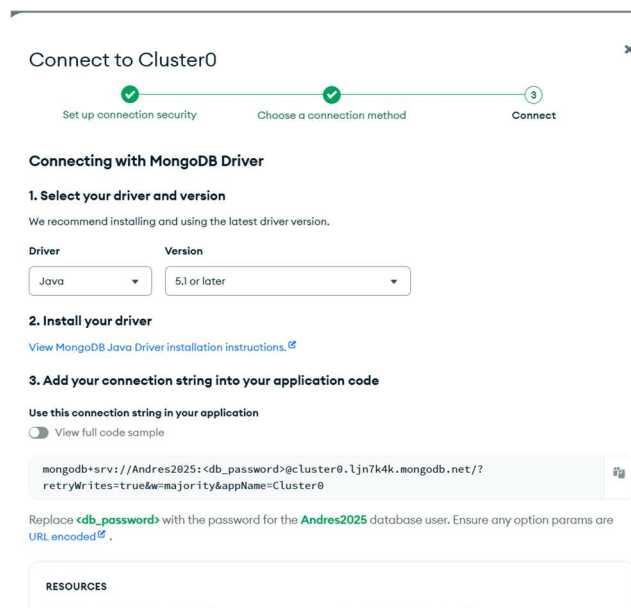


mongo-java-driver-3.11.2

En nuestro IDE, dentro de Project Structure agregamos el archivo.jar y le damos en aplicar.



Finalmente, en el punto 3. Copiamos la Url que nos va a permitir hacer la conexión y cambiamos el campo de `<db_password>` por nuestra contraseña de Mongo Atlas.



La conexión se estableció con el siguiente comando:

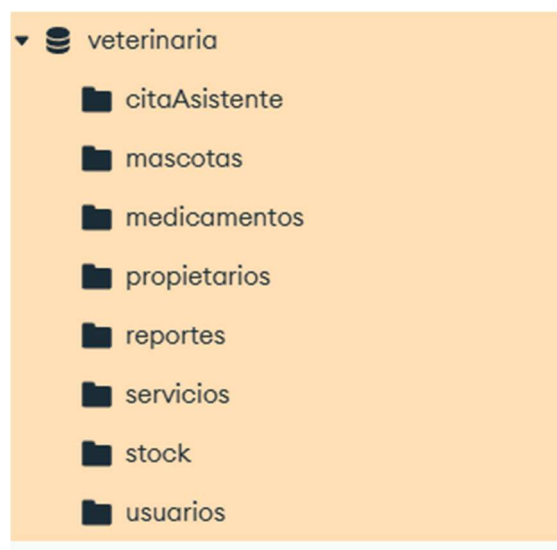
```
1 package dao;
2
3 import com.mongodb.client.MongoClient;
4 import com.mongodb.client.MongoClients;
5 import com.mongodb.client.MongoDatabase;
6
7 public class Conexion { 9 usages Ivanp2003 +1*
8
9     private static Conexion instancia; 3 usages
10     private MongoClient mongoClient; 2 usages
11     private MongoDatabase baseDatos; 2 usages
12
13     private Conexion() { 1 usage Ivanp2003 *
14         String uri = "mongodb+srv://Andres2025:1234@cluster0.ljn7k4k.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0";
15
16         mongoClient = MongoClients.create(uri);
17         baseDatos = mongoClient.getDatabase("veterinaria");
18     }
19
20     public static Conexion getInstancia() { 6 usages Ivanp2003
21         if (instancia == null) {
22             instancia = new Conexion();
23         }
24         return instancia;
25     }
26
27     public MongoDatabase getBaseDatos() { 6 usages Ivanp2003
28         return baseDatos;
29     }
30 }
```

Dentro del método Conexion, se especifica la URI que permite la conexión con MongoDB Atlas:

```
private Conexion() {
    String uri =
"mongodb+srv://Andres2025:1234@cluster0.ljn7k4k.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0";

    mongoClient = MongoClients.create(uri);
    baseDatos = mongoClient.getDatabase("veterinaria");
}
```

Y si queremos hacerlo de manera local, debemos tener creada la base de datos y las colecciones.

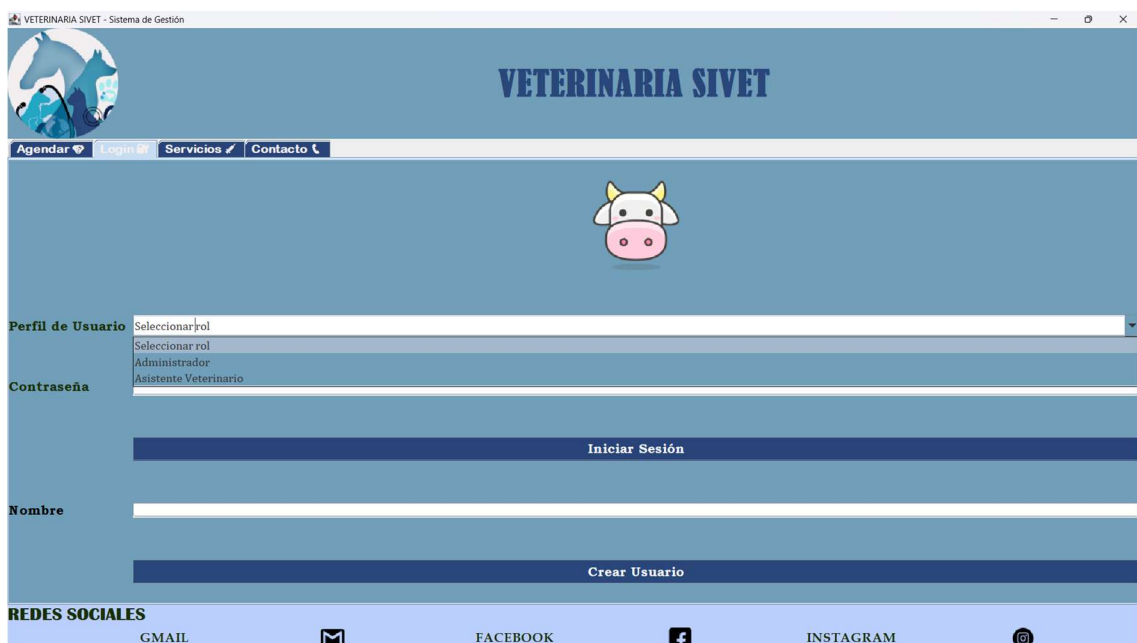


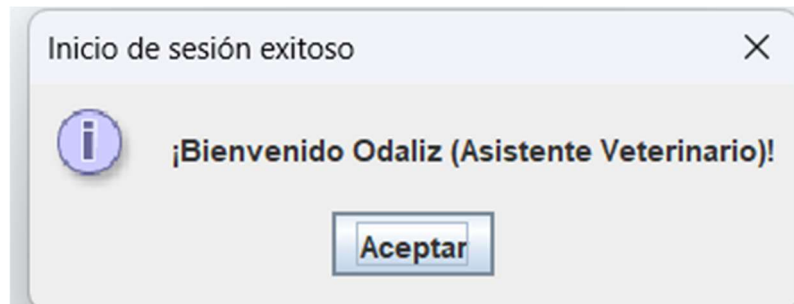
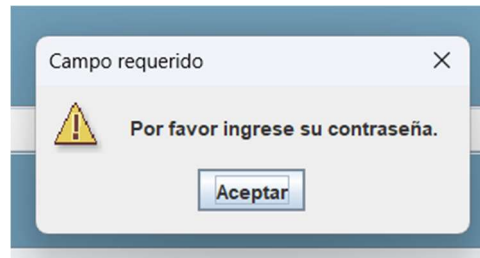
Y en la clase Conexión cambia por localhost y nombre de la base de datos a utilizar.

```

Main.java  Conexion.java x
1 package dao;
2
3 import com.mongodb.MongoClient;
4 import com.mongodb.client.MongoDatabase;
5
6 public class Conexion { 9 usages  Ivanp2003 +1 *
7     private static Conexion instancia; 3 usages
8     private MongoClient mongoClient; 2 usages
9     private MongoDatabase baseDatos; 2 usages
10
11     private Conexion() { 1 usage  Ivanp2003 *
12         mongoClient = new MongoClient(host: "localhost", port: 27017);
13         baseDatos = mongoClient.getDatabase(databaseName: "veterinaria");
14     }
15
16     public static Conexion getInstancia() { 6 usages  Ivanp2003
17         if (instancia == null) {
18             instancia = new Conexion();
19         }
20         return instancia;
21     }
22
23     public MongoDatabase getBaseDatos() { 6 usages  Ivanp2003
24         return baseDatos;
25     }
26 }
```

UsuarioDAO.java: Gestiona la autenticación y creación de usuarios. Permite registrar un máximo de dos administradores y hasta diez asistentes veterinarios, validando que no se repitan los nombres de usuario. Además, muestra un mensaje de bienvenida personalizado al iniciar sesión. Esta clase actúa como el filtro de acceso al sistema.





CitaDAO.java: Se encarga del almacenamiento, consulta y eliminación de las citas médicas. Su funcionalidad se vincula especialmente con el rol del asistente veterinario en la clase MenuAsistente. Las citas pueden eliminarse individualmente, y todos los registros se almacenan en la colección citaAsistente de MongoDB.

A form titled 'Registrar Nueva Cita' with a close button. It contains five input fields: 'Propietario:', 'Mascota:', 'Fecha (dd/mm/yyyy):', 'Hora:', and 'Motivo:'. At the bottom, there are two buttons: 'Aceptar' and 'Cancelar'.

CITAS AGENDADAS EN EL SISTEMA						
ID	Propietario	Mascota	Fecha	Hora	Motivo	Estado
6891679d...	Maria	coco	04/08/2025	11:00	Consulta	PENDIENTE
689167f1...	Camila	Max	04/08/2025	13:00	Vacunas	PENDIENTE
68917b32...	Diego	pepe	04/08/2025	16:00	Atención urgente	PENDIENTE

Eliminar Cita

?

Seleccione la cita a eliminar:
ID:68917b32d9928e08e8a831e7 - Diego (pepe) - 04/08/2025

Aceptar

Cancelar

Confirmar eliminación

?

¿Está seguro de eliminar esta cita?

Cita #68917b32d9928e08e8a831e7
Propietario: Diego
Mascota: pepe
Fecha: 04/08/2025
Hora: 16:00
Motivo: Atención urgente
Estado: PENDIENTE

Sí

No

MascotaDAO.java: Se utiliza al presionar el botón "Ver Mascotas" desde el panel del asistente. Permite editar campos como especie, raza y sexo de las mascotas que fueron registradas previamente al agendar una cita. Esta clase facilita la gestión de los datos veterinarios básicos de cada mascota.

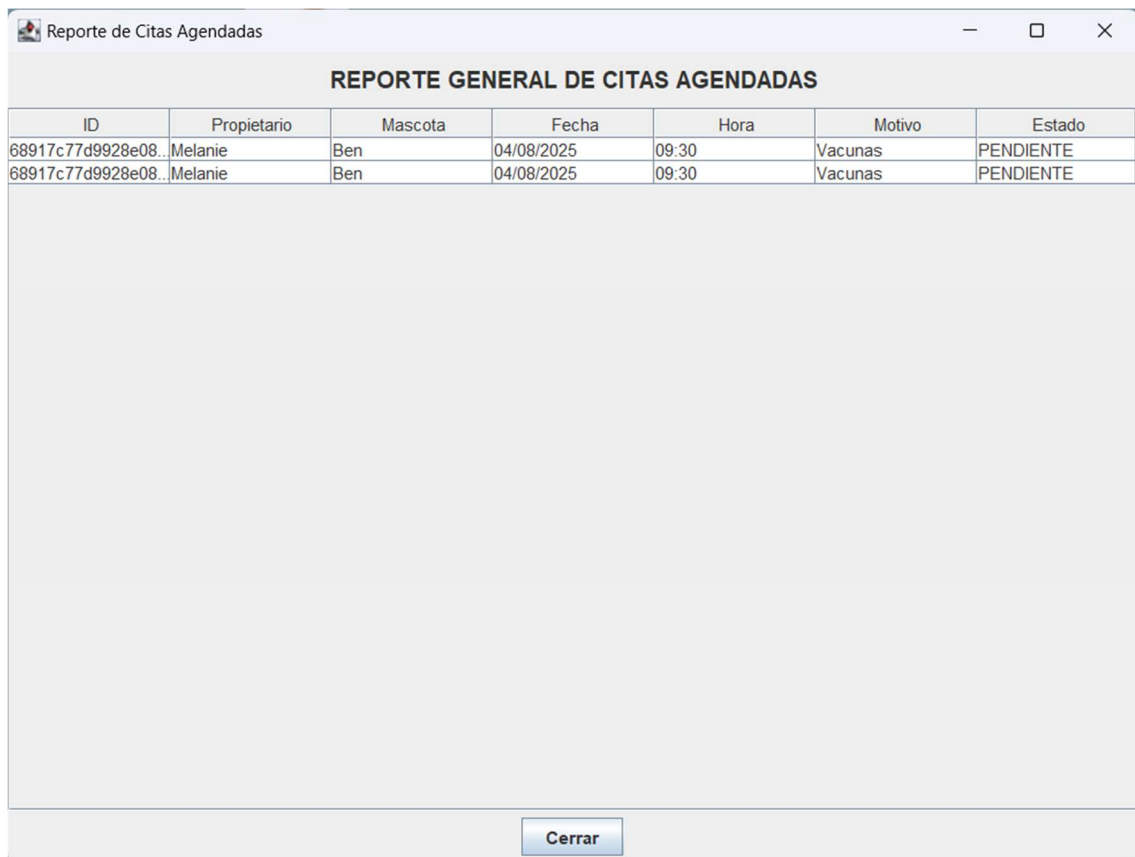
Ver y Editar Mascotas

Haz doble clic en una celda para editar especie, raza o sexo.

ID	Nombre	Especie	Raza	Sexo	Propietario
689167f198f751170080c44f	Max	perro	chihuahua		Camila
68917b33d9928e08e8a831	pepe	gato	siames		Diego

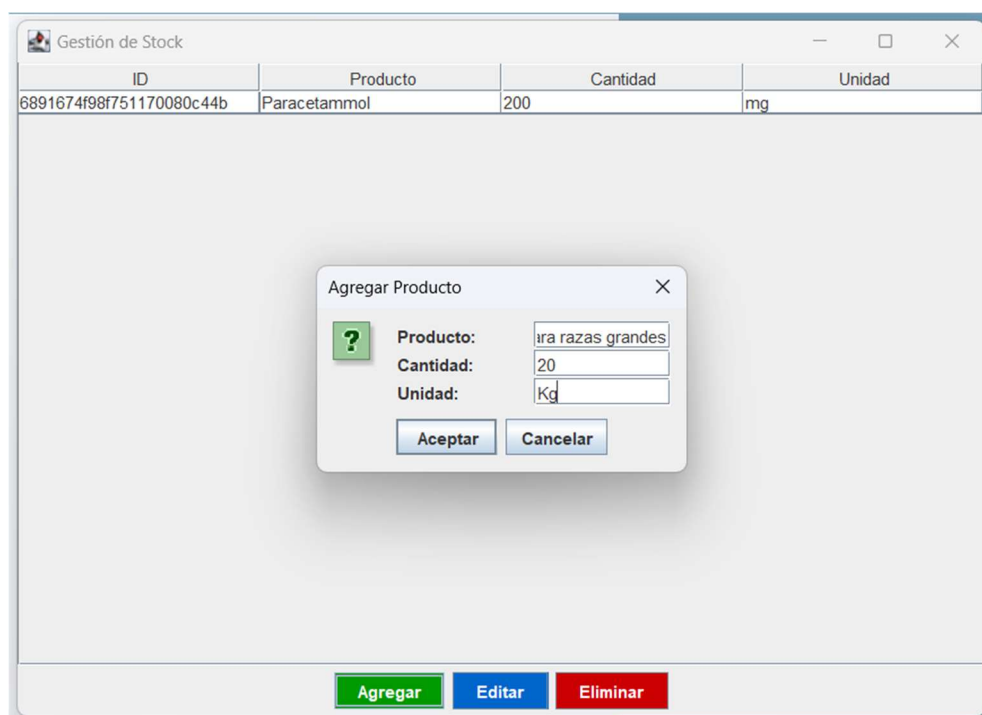
Actualizar

ReporteDAO.java: Permite almacenar y consultar los reportes generados en el sistema. Cuando un administrador pulsa el botón "Ver Reportes" en la interfaz MenuAdmin, esta clase guarda los datos en la colección reportes, permitiendo generar un historial de citas agendadas y otros informes relevantes.



ID	Propietario	Mascota	Fecha	Hora	Motivo	Estado
68917c77d9928e08...	Melanie	Ben	04/08/2025	09:30	Vacunas	PENDIENTE
68917c77d9928e08...	Melanie	Ben	04/08/2025	09:30	Vacunas	PENDIENTE

StockDAO.java: Exclusiva para el administrador, esta clase gestiona el CRUD (crear, leer, actualizar y eliminar) de artículos o insumos veterinarios, como medicamentos, productos de higiene o suministros médicos. El administrador puede controlar el inventario a través del panel MenuAdmin, garantizando que solo él tenga acceso a la edición del stock.



ID	Producto	Cantidad	Unidad
6891674f98f751170080c44b	Paracetamol	200	mg

Agregar Producto

? Producto: ira razas grandes

Cantidad: 20

Unidad: Kg

Aceptar Cancelar


Agregar Editar Eliminar

StockDAO.java. Exclusiva para el administrador, esta clase gestiona el CRUD (crear, leer,

ID	Producto	Cantidad	Unidad
6891674f98f751170080c44b	Paracetammol	200	mg
68917cc1d9928e08e8a831eb	Croquetas para razas grandes	20	Kg

ID	Producto	Cantidad	Unidad
6891674f98f751170080c44b	Paracetammol	200	mg
68917cc1d9928e08e8a831eb	Croquetas para razas grandes	20	Kg

Confirmar

 ¿Seguro que desea eliminar este producto?

Si

No

Agregar

Editar

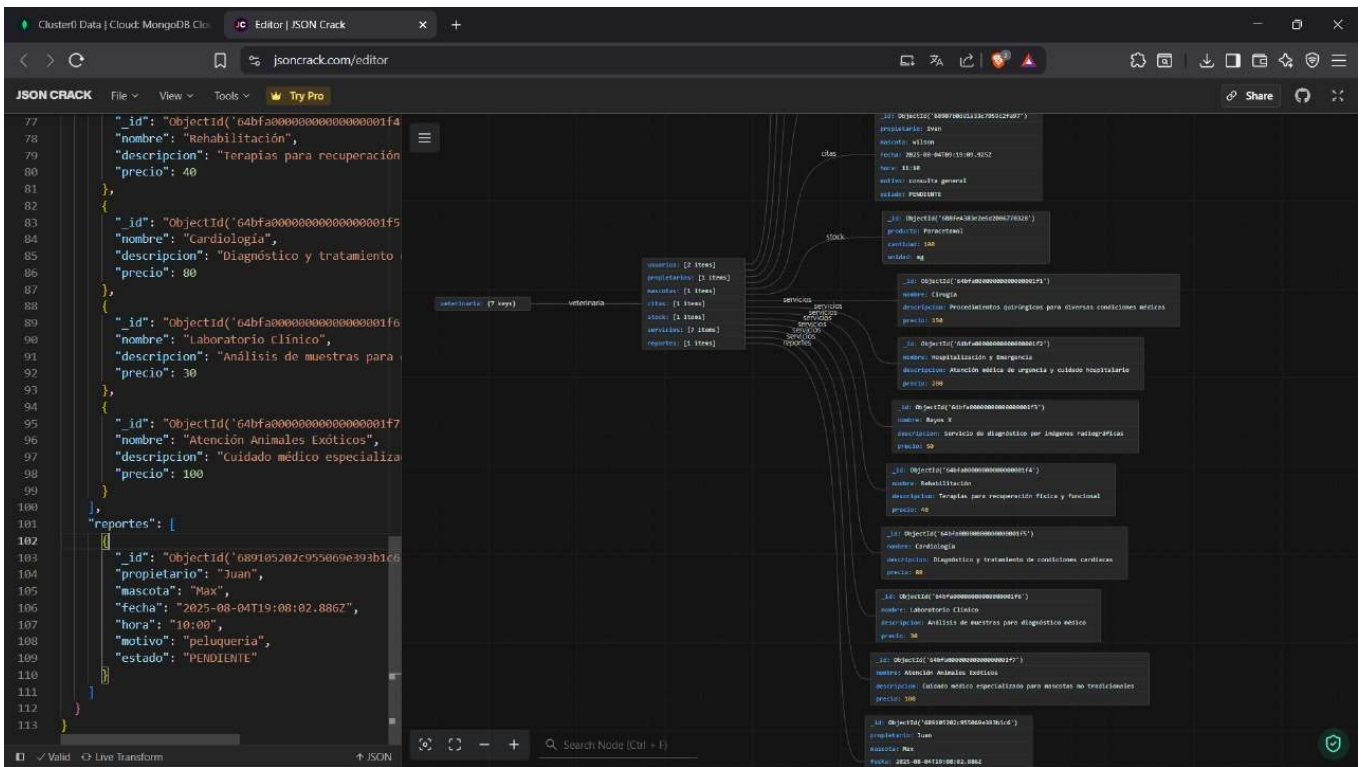
Eliminar

ID	Producto	Cantidad	Unidad
6891674f98f751170080c44b	Paracetammol	200	mg

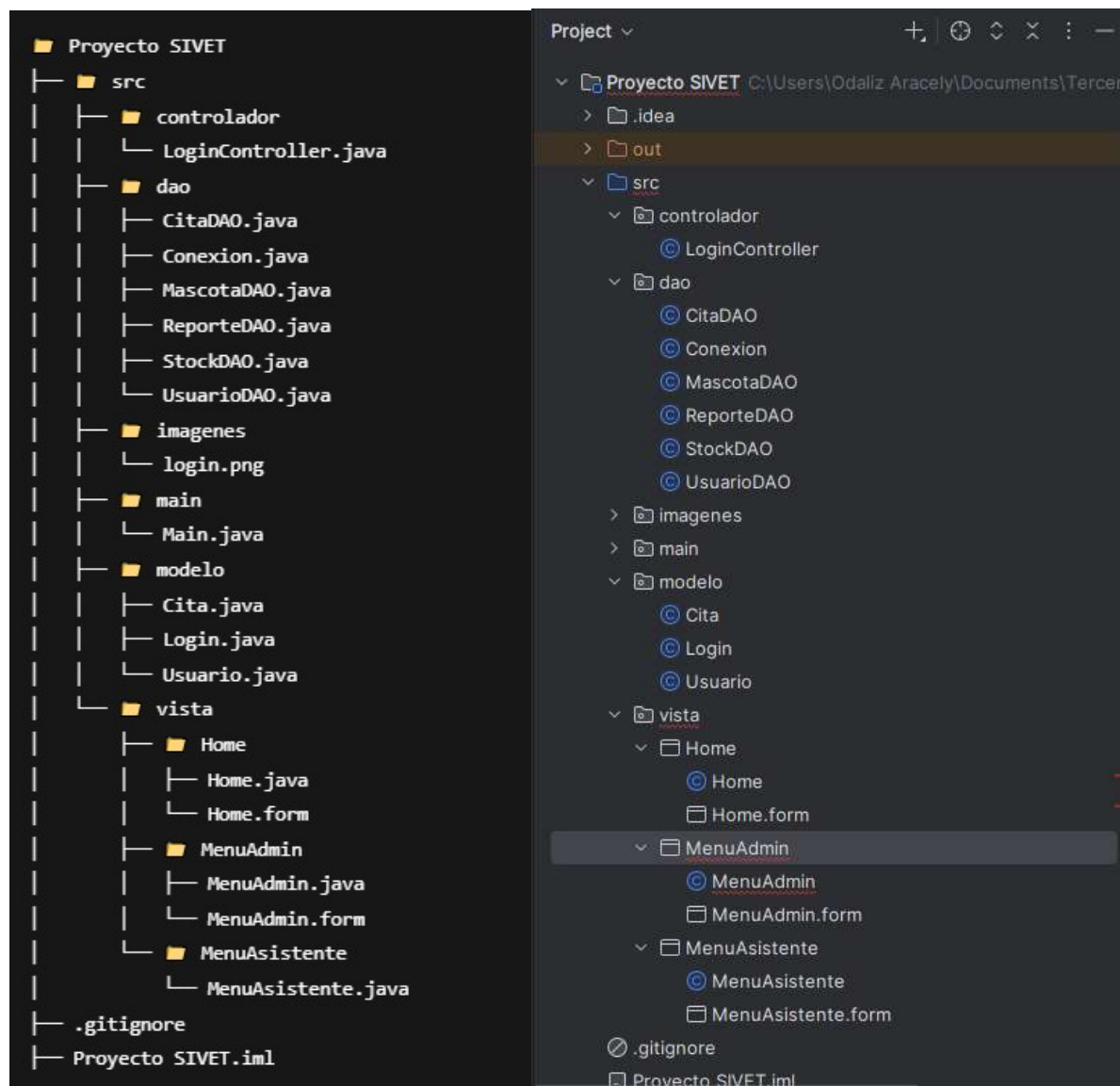
Agregar

Editar

Eliminar



Organización del Código



Ejecución Completa del Proyecto

Flujo de Ejecución

1. **Inicio del sistema:** Clase Home.java con método main()
2. **Inicialización de BD:** Verificación de conexión y creación de tablas
3. **Interfaz principal:** Pantalla de bienvenida con pestañas de navegación
4. **Autenticación:** Sistema de login con validación de credenciales
5. **Panel administrativo:** Acceso diferenciado según rol de usuario
6. **Operaciones CRUD:** Gestión completa de citas médicas
7. **Reportes:** Generación de estadísticas y listados
8. **Cierre seguro:** Desconexión de BD y cierre de sesión

Casos de Uso Principales

CU001: Agendar Cita

- Actor: Usuario público
- Flujo: Completar formulario → Validar datos → Guardar en BD → Mostrar confirmación

CU002: Iniciar Sesión

- Actor: Administrador/Asistente
- Flujo: Seleccionar rol → Ingresar contraseña → Validar credenciales → Abrir panel

CU003: Gestionar Citas

- Actor: Administrador/Asistente
- Flujo: Ver lista → Seleccionar acción → Confirmar operación → Actualizar vista

Explicación del Desarrollo

Encapsulamiento

- Todos los atributos de las clases modelo son privados
- Acceso controlado mediante métodos getter y setter
- Validaciones en métodos set para integridad de datos

```
public class Cita {  
    private int id;           // Atributos privados  
    private String nombrePropietario;  
    private LocalDate fechaCita;  
  
    public int getId() {      // Métodos públicos de acceso  
        return id;  
    }  
  
    public void setId(int id) {  
        this.id = id;  
    }  
}
```

Herencia

- Clase base EntidadBase con atributos comunes (id, fechaCreacion)
- Clases especializadas heredan funcionalidades básicas
- Polimorfismo en métodos de validación y toString()

```
public abstract class Usuario {           // Clase base
    protected String username;
    protected String password;

    public abstract void mostrarMenu(); // Método abstracto
}

public class Administrador extends Usuario { // Herencia
    @Override
    public void mostrarMenu() {
        // Implementación específica para administrador
    }
}
```

Polimorfismo

- Interfaces CRUD<T> implementadas por todos los controladores
- Métodos sobrecargados para diferentes tipos de búsqueda
- Uso de tipos abstractos para flexibilidad

```
public void procesarUsuario(Usuario usuario) {
    usuario.mostrarMenu(); // Comportamiento diferente según tipo
}
```

Abstracción

- Separación clara entre capas de presentación, lógica y datos
- Interfaces bien definidas entre componentes
- Ocultación de complejidad de MongoDB al usuario final

Responsabilidades por Capa:

Modelo

- Representa entidades del dominio veterinario
- Atributos privados con validaciones
- Métodos de utilidad específicos del negocio

Vista

- Interfaces gráficas Swing responsivas
- Manejo de eventos de usuario
- Validaciones visuales y feedback inmediato

Controlador

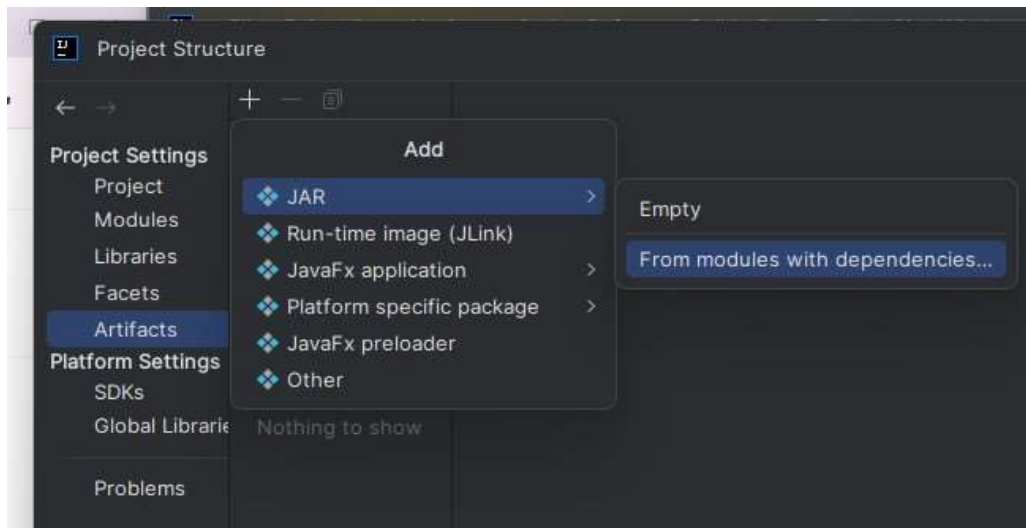
- Lógica de negocio y reglas de validación
- Comunicación con base de datos MongoDB
- Procesamiento de datos entre vista y modelo

Conexión

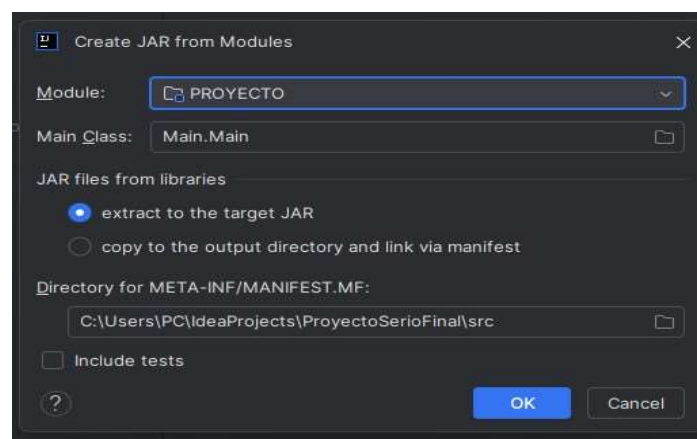
- Gestión de conexión a MongoDB Atlas
- Pool de conexiones para optimización
- Manejo de errores de conectividad

Exe

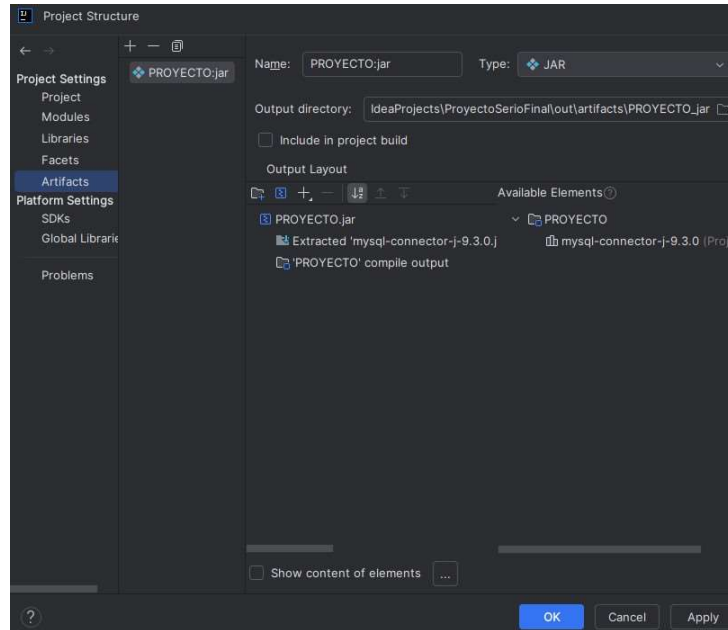
- En project structure ingresamos a artifacts



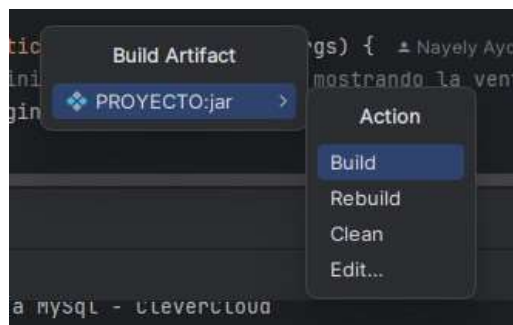
- Luego seleccionamos la clase principal



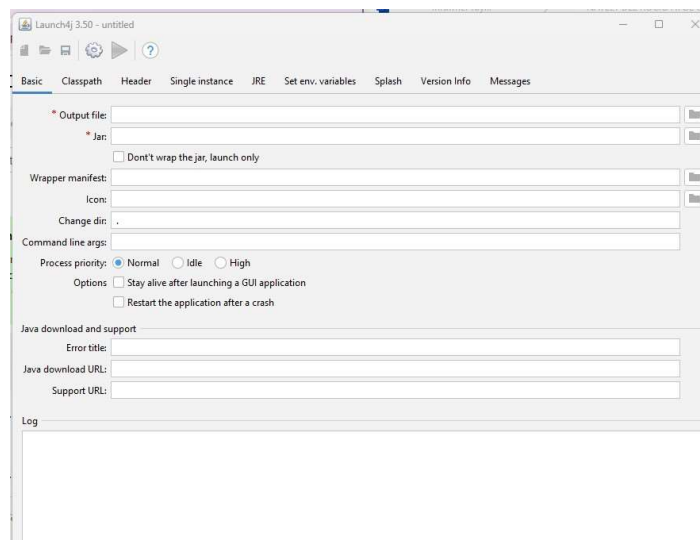
- Presionamos OK en las ventanas emergentes.



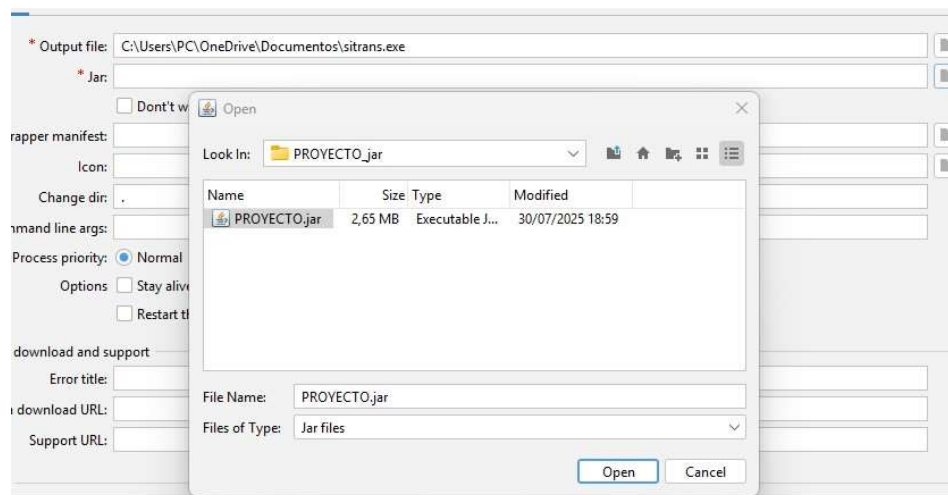
- Luego en Built – Built artifacts



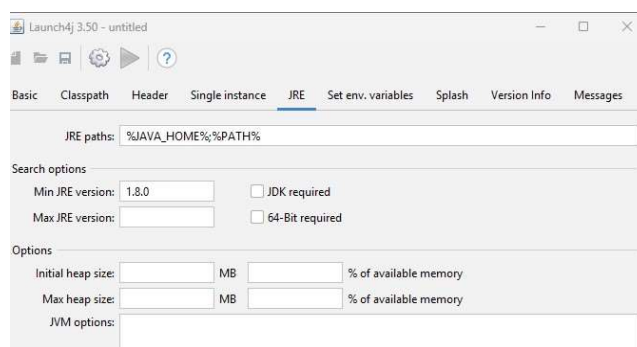
- Luego en Launch4J llenamos los campos solicitados



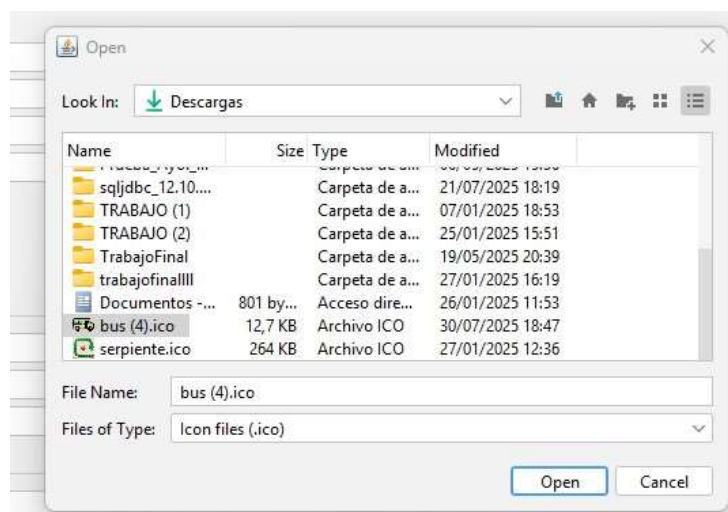
- Seleccionamos el .jar



- Llenamos estos campos dado que



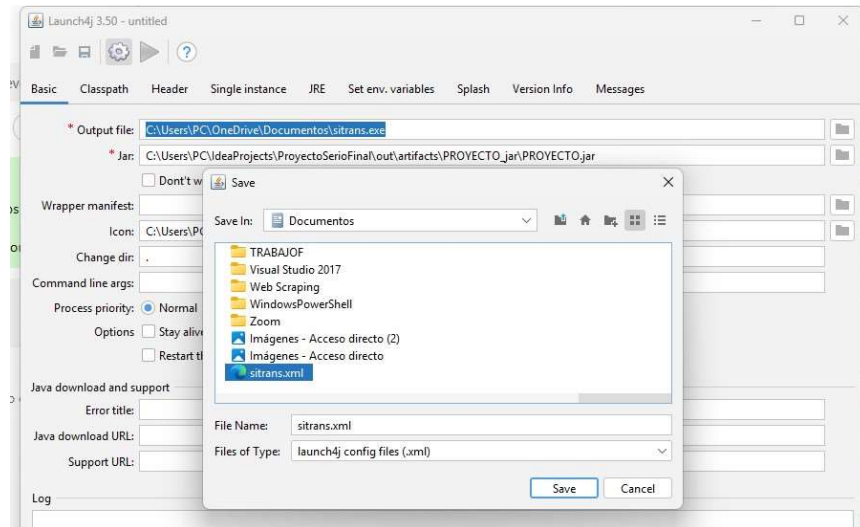
- Seleccionamos un .ico



- Presionamos la opción para construir el sistema



- Seleccionamos un archivo para guardar.



- Presionamos en

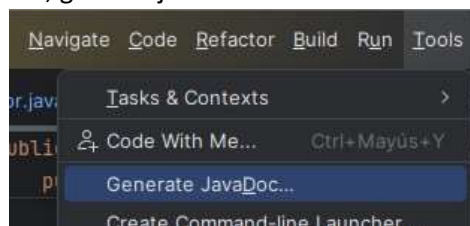


- Y ya tenemos un .exe

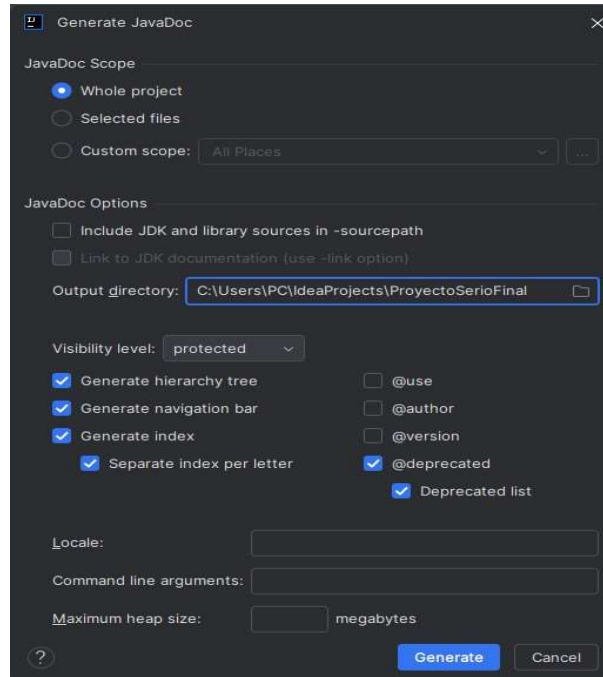


Java Doc

- En tolos, seleccionamos, generar java doc

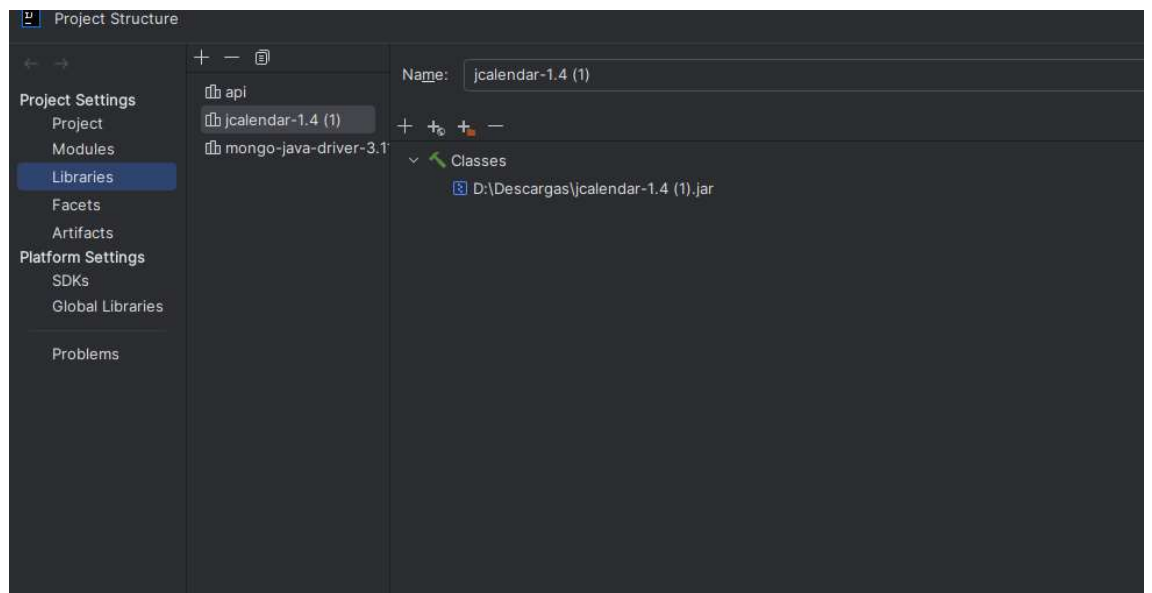


- Seleccionamos el proyecto para generar el java doc



- Se genera el Java doc

- Overview



OVERVIEW

PACKAGE

CLASS

TREE

INDEX

HELP

PACKAGE: DESCRIPTION | RELATED PACKAGES | CLASSES AND INTERFACES

Package controlador

package controlador

Classes

Class	Description
LoginController	Controlador para gestionar la lógica de autenticación y login del sistema.

←

↺

📄 Archivo

C:/Users/Odaliz%20Aracely/Documents/Tercer%20Semestre/POO/JAVA%20DOC%20PROYECTO%20FINAL/dao/package-summary.html

OVERVIEW

PACKAGE

CLASS

TREE

INDEX

HELP

PACKAGE: DESCRIPTION | RELATED PACKAGES | CLASSES AND INTERFACES

Package dao

package dao

Classes

Class	Description
CitaDAO	
Conexion	Clase para gestionar la conexión a la base de datos MongoDB Atlas.
MascotaDAO	Clase DAO para gestionar las operaciones CRUD de mascotas.
ReporteDAO	Clase DAO para generar reportes y estadísticas del sistema veterinario.
StockDAO	Acceso a datos del inventario (stock) de insumos veterinarios.
UsuarioDAO	Acceso a datos para la entidad Usuario.

Package main

package main

Classes

Class	Description
Main	Clase principal del Sistema Integral de Gestión Veterinaria (SIVET).

←

↺

📄 Archivo

C:/Users/Odaliz%20Aracely/Documents/Tercer%20Semestre/POO/JAVA%20DOC%20PROYECTO%20FINAL/modelo/package-

OVERVIEW

PACKAGE

CLASS

TREE

INDEX

HELP

PACKAGE: DESCRIPTION | RELATED PACKAGES | CLASSES AND INTERFACES

Package modelo

package modelo

Classes

Class	Description
Cita	Clase modelo que representa una cita médica veterinaria.
Usuario	Representa un usuario registrado en el sistema.

Trabajo Colaborativo en GitHub

Información del Repositorio

GitHub: <https://github.com/Odaliz2105/Proyecto-SIVET.git>

Link de la Presentación. –

Link Java Doc. -

<file:///C:/Users/Odaliz%20Aracely/Documents/Tercer%20Semestre/POO/JAVA%20DOC%20PROYECTO%20FINAL/vista/package-summary.html>

<https://drive.google.com/drive/folders/1NoxHJwOyxuDPyL47GzvuxK4Q0zMhrr4g?usp=sharing>

Link exe. -

<https://drive.google.com/drive/folders/1c42tgnIfpSyqnN7ZrekNVm6KyQpRTmnQ?usp=sharing>

Conclusiones

- El sistema desarrollado permite una gestión eficiente y segura de las funcionalidades del consultorio veterinario, al implementar un control de acceso mediante roles diferenciados (administrador y asistente veterinario), lo que garantiza que cada usuario solo tenga acceso a las funciones pertinentes a su cargo.
- El uso de interfaces gráficas en Java con Swing facilita la interacción con el usuario final, mejorando la experiencia de uso mediante elementos visuales como botones, menús, paneles e íconos personalizados, lo que hace que el sistema sea más intuitivo y funcional.
- El sistema logró implementar exitosamente la carga y visualización de imágenes con redondeo y escalado en Java, mejorando la presentación visual de los elementos gráficos dentro de la interfaz de usuario. Esto evidencia un dominio adecuado de manipulación de imágenes y componentes gráficos de Swing.
- La resolución de errores relacionados con rutas de archivos, escalado de imágenes y manejo de excepciones permitió optimizar la estabilidad y portabilidad del sistema, asegurando una experiencia más robusta para el usuario final.

Recomendaciones y Trabajos Futuros

- Se recomienda implementar una base de datos relacional para el almacenamiento persistente de los registros, con mecanismos de autenticación y auditoría para mejorar la seguridad y trazabilidad de las acciones realizadas por cada rol.
- Es aconsejable realizar pruebas de usabilidad con usuarios reales (administrador y asistente veterinario) para detectar posibles mejoras en la interfaz y garantizar que las funcionalidades sean claras, accesibles y cumplan con los objetivos del consultorio.
- Se recomienda probar el programa en distintas resoluciones y tamaños de imagen para asegurar que el redondeo y escalado funcionen de manera uniforme en todos los casos.
- Utilizar logs o mensajes de error claros en los métodos de procesamiento de imagen, lo cual facilita la depuración y mejora la mantenibilidad del sistema.