

**Guía práctica de Talleres No 2 Proyecto:**

**Manual del Desarrollador**

**Veterinaria Dressrosa**

**Por:**

**Carlos Andres Pardo Salinas**

**Mattias Henao Ricaurte**

**Para:**

**Manolo Pájaro Borrás**

**Politécnico Colombiano Jaime Isaza Cadavid Medellín, Colombia**

**2023**

pág. 1

# Indice

[Introducción 3](#_bookmark0)

[Objetivos 4](#_bookmark1)

[Requisitos del Software "Control Veterinaria". Versión 1.5 5](#_bookmark2)

[Diseño del software del sistema 6](#_bookmark3)

[Desarrollo del aplicativo 7](#_bookmark4)

[PAQUETE controlador 7](#_bookmark5)

* [Clase LAdmin 7](#_bookmark6)
* [Clase LHistoriaClinica](#_bookmark7) 8
* [Clase LMascota](#_bookmark8) 9
* [Clase LMedico 1](#_bookmark9)0
* Clase LVendedor 10
* Clase UserManager 11

[PAQUETE modelo 12](#_bookmark10)

* [Clase HistoriaClinica 12](#_bookmark11)
* [Clase Mascota 13](#_bookmark12)
* [Clase OrdenMedica 1](#_bookmark12)3
* Clase Usuario 14

[PAQUETE vista 14](#_bookmark13)

* [Clase Login 15](#_bookmark14)
* [Clase Main 16](#_bookmark15)
* [Clase Principal 17](#_bookmark16)
* [Clase Registro 18](#_bookmark17)
* Clase RegistroAdmin 19
* Clase RegistroDueño 20
* Clase RegistroMedico 21

[Conclusiones 2](#_bookmark21)2

pág. 2

# Introducción

En este proyecto se desarrolló desde la herramienta de trabajo ECLIPSE, en el cual se busca plasmar la idea de una veterinaria en la que se disponen de diferentes opciones para registrar y mantener un control de las diferentes mascotas que se puedan llegar a intervenir en la veterinaria.

Esto con el fin de poder dar una forma más práctica y sencilla a todos los usuarios que lleguen a necesitar del producto alguna vez, que desde la practicidad de su manejo se hace óptimo y rápido para su uso.

El programa cuenta con una acceso, mediante una serie de paneles gráficos que mediante su funcionalidad hacen posible el ingreso al programa, esto controlado y desarrollado para que el usuario quede registrado y pueda digitar su id y contraseña para acceder y navegar en el producto.

Dentro de la aplicación, se permite registrar dueños de las mismas mascotas, además de mostrar sus facturas e historias clínicas que se le realicen a cada mascota del complejo que permitiría el control óptimo y seguro de cada ítem que tenga relación con la mascota intervenida en la veterinaria.

Toda la aplicación se desarrolló con la idea de aplicar toda la base de programación orientada a objetos y control dinámico de paneles gráficos desde el manejo de archivos planos, que dentro de este programa permiten el almacenamiento de cada dato que se ingrese y consulte así mismo.

pág. 3

# Objetivos:

- Facilitar el acceso rápido a los historiales médicos de los pacientes, además de mejorar la precisión y la integridad de la información del paciente.

-Agilizar los procesos internos de la clínica veterinaria, reduciendo el papeleo manual y mejorando la eficiencia en la programación de citas, facturación y seguimiento de tratamientos.

- Potenciar la calidad de la atención veterinaria al permitir a los profesionales acceder de inmediato a los historiales médicos, posibilitando recomendaciones precisas a los dueños de mascotas.

-Asegurar la confidencialidad de la información del paciente mediante un acceso controlado, cumpliendo con normativas de privacidad, y fortaleciendo la confianza de los clientes.

-Proporcionar a los profesionales veterinarios información clara y organizada sobre el estado de salud de los pacientes, lo que les permitirá tomar decisiones informadas y proporcionar tratamientos más efectivos.

-Optimizar todos los procesos manuales que se ejecutan dentro de una empresa de veterinaria, evitando papeleos y registros a mano, que a veces no se asegura su total efectividad.

pág. 4

# Requisitos del Software " Control Veterinaria". Versión 1.5

1. El proyecto debe estar presentado MVC (modelo, controlador, vista).
2. Debe estar en ventanas modales, o sea que no puede existir una ventana externa o fuera de la ventana principal, deben estar centradas.
3. Las ventanas no se pueden maximizar ni minimizar, sólo pueden cerrarse mediante un botón de comando.
4. Los botones deben tener imágenes y texto.
5. Los cruds deben estar implementado en el 50% de los formularios, por ejemplo si su proyecto tiene 10 formularios y son 10 tablas o archivos de texto 5 deben estar funcionales.

1. A este punto o nivel ya deben estar definidos todas las tablas o archivos de textos que utilizará el sistema.
2. La aplicación debe funcionar 100% con los elementos o formularios asignados. (Debe poder ejecutarse en el equipo del docente).

1. Deben tener como mínimo 2 tablas realizadas en Json e implementadas en cualquier formulario
2. los formularios a los cuales se les está aplicando el crudo deben estar validados.

1. Los manuales tanto del usuario como del programador deben estar alineados con lo que están entregando hasta el momento.

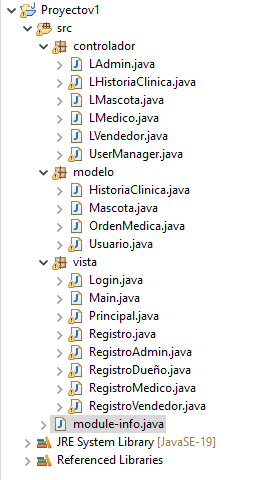
pág. 5

# Desarrollo del software del sistema

El proyecto ha sido desarrollado bajo la lógica MVC (modelo-vista-controlador) el cual hace más práctico y eficiente la programación orientada a objetos.

## Estructura del código

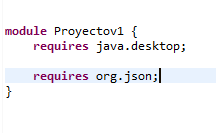
La aplicación cuenta con los 3 paquetes del patrón MVC, además de las clases que integran a cada uno de los paquetes.



**Referencias de los paquetes y librerías:**

Se hace referencia en la parte de abajo en la clase: module-info.java -> Se aplica el llamado a la librería requires java.desktop, que es la que hace posible el llamado a todas las importaciones de JAVA para darle ejecución y desarrollo al programa.Ademas de la función requires org.json; para

Darle función a los archivos Json.



En el apartado de Referenced Libraries, se hace el llamado y la descarga del procesador de JSON para permitir el manejo de archivos y controles de este tipo de herramienta.

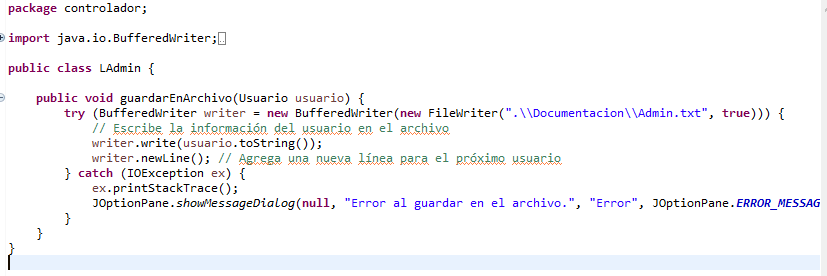
pág. 6

# Desarrollo del aplicativo

## PAQUETE CONTROLADOR

En este paquete se desarrolla toda la lógica operativa del software, este mismo es el que permite manejar toda la información y trasladar de un paquete a otro los recursos necesarios para el funcionamiento del patrón MVC. Esta clase se encarga de recibir y adaptar en el programa todas las acciones que el usuario ejecute y agregue al programa, tales como la creación de nuevos pacientes o la edición de historiales médicos, para a su vez implementar cada lógica y manipulación de datos en el modelo correspondiente. Además, controla que la manipulación del sistema se haga de una forma correcta basándose en los parámetros del sistema mismo. Lo que permite que el controlador se encargue de actualizar la vista para reflejar los cambios realizados en el modelo, garantizando que la interfaz de usuario muestre información actualizada y coherente en todo momento. En conclusión, la clase Controlador desempeña un rol central en la gestión de la interacción del usuario con la aplicación y en la coordinación de la lógica del aplicativo. Para su eficiente manejo este consta de las siguientes clases:

* **Clase LAdmin:** Esta clase se encarga de escribir y adaptar el diligenciamiento de los usuarios administradores en diferentes archivos de texto, para agregarlos a la base del sistema y ser operativos.

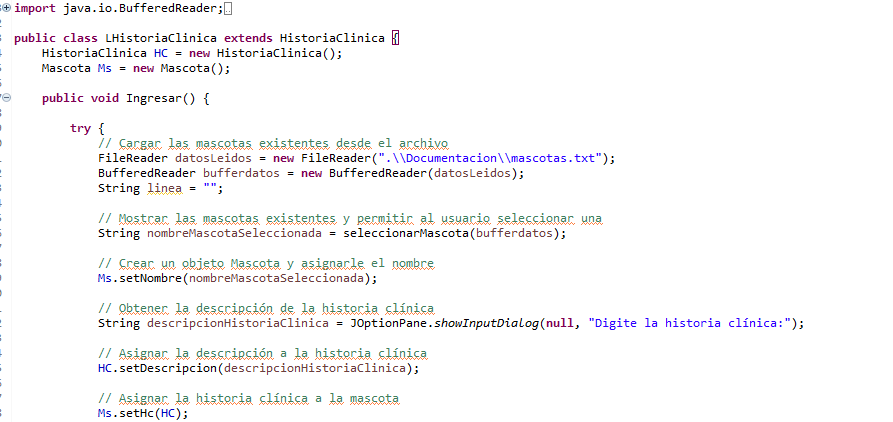


Esta clase al pertenecer a un administrador, este usuario podrá hacer manejo de todo el aplicativo, sin restricciones algunas.

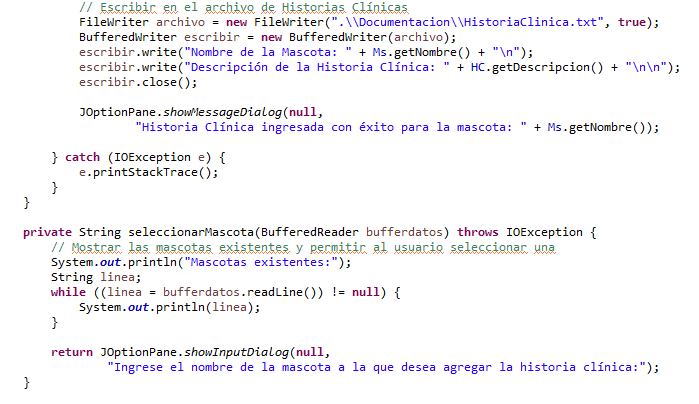
Consta del procedimiento guardarEnArchivo, que hace posible todo este proceso.

pág. 7

* **Clase LHistoriaClinica:** Se encarga de asignar, cargar, crear y disponer de todos los datos que puedan ser asignables a cada mascota, según el procedimiento que se requiera hacer.



Se puede evidenciar como cada llamado, hace un uso para apropiar la historia clínica con todos los datos y descripciones de la mascota, además de que hace parte de heredar de las variables que hacen operativa a esta clase.

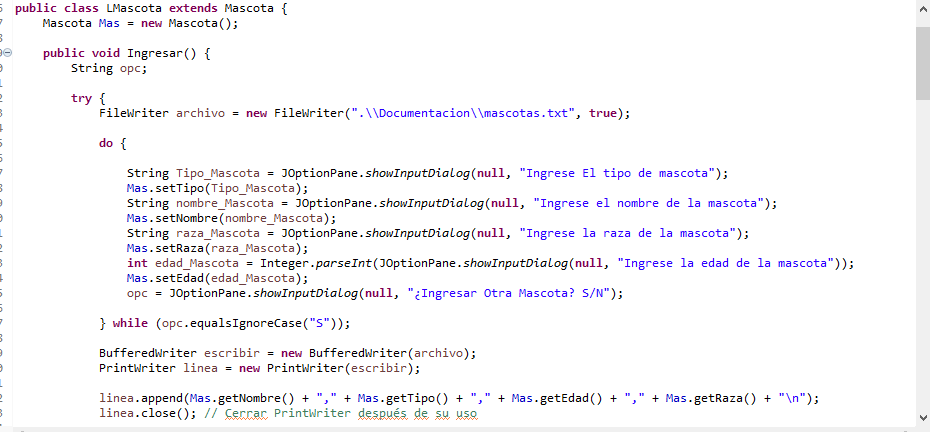


Después de obtener todos los datos de la mascota esta se agrega al sistema y pasa a ser operativa con las distintas funciones creadas para este tipo de acciones.

La mascota pasa a estar registrada en el sistema para después crearle su propia historia clínica.

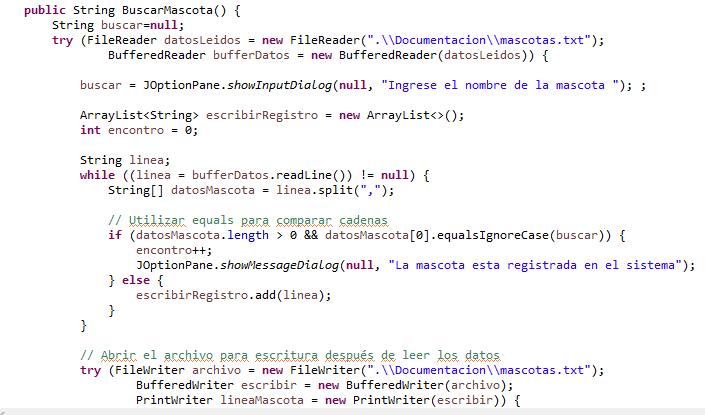
pág. 8

* **Clase LMascota:** En esta clase se ingresan todos los datos acerca de la mascota para que esta misma pueda ser ingresada al sistema.



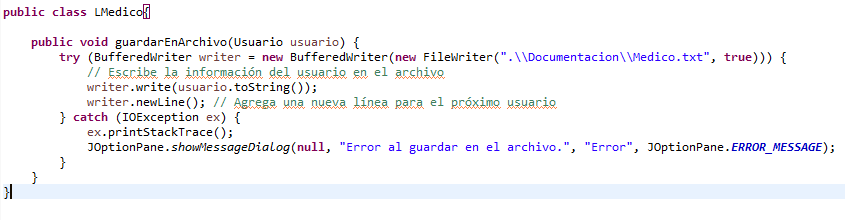
Además en esta clase se incluyen, los métodos que permiten buscar, crear, consultar, eliminar a todas las mascotas que han sido ingresados en el sistema.





pág. 9

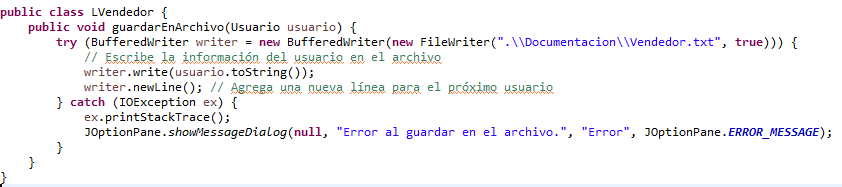
* **Clase LMedico:** Esta clase se encarga de escribir y adaptar el diligenciamiento de los usuarios Médicos en diferentes archivos de texto, para agregarlos a la base del sistema y ser operativos.



Esta clase al pertenecer a un Médico, este usuario podrá hacer manejo de las historias clínicas que se le tengan que referir a cada mascota, para brindarles su debido tratamiento.

Consta del procedimiento guardarEnArchivo, que hace posible todo este proceso.

* **Clase LVendedor:** Esta clase se encarga de escribir y adaptar el diligenciamiento de los usuarios Vendedores en diferentes archivos de texto, para agregarlos a la base del sistema y ser operativos.

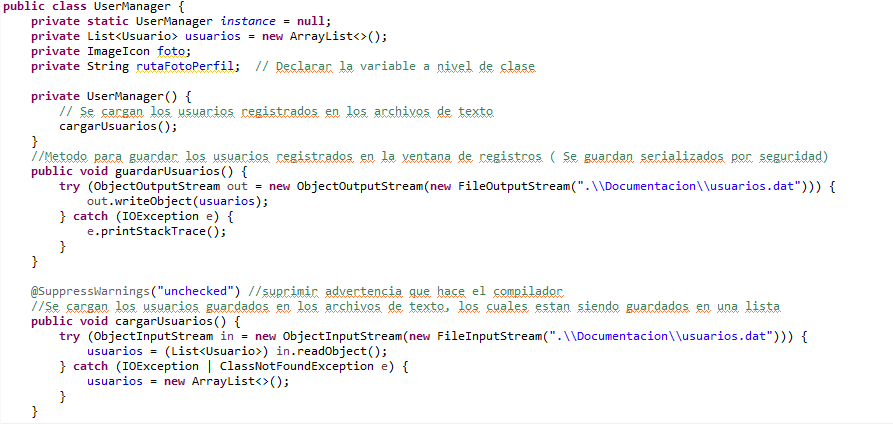
****

Esta clase al pertenecer a un Vendedor, este usuario podrá hacer manejo de todos los productos que se puedan vender en la veterinaria, para ofrecerlos al público mediante el sistema.

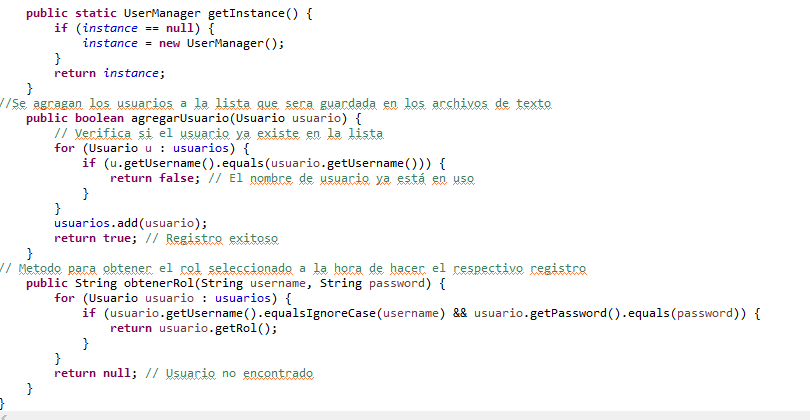
Consta del procedimiento guardarEnArchivo, que hace posible todo este proceso.

pág. 10

* **Clase UserManager:** Esta clase es la encargada de cargar y almacenar cada uno de los usuarios que han sido ingresados en el sistema.



Esta clase es fundamental para el aplicativo, ya que esta misma es aquella que le brinda los permisos a cada usuario según su rol en la veterinaria, esta ayuda para que un médico no pueda vender productos, pero un vendedor tampoco pueda hacer historias clínicas.



Además esta clase permite a cada integrante modificar su perfil y hacerlo operativo dentro de su respectivo rol dentro de la veterinaria.

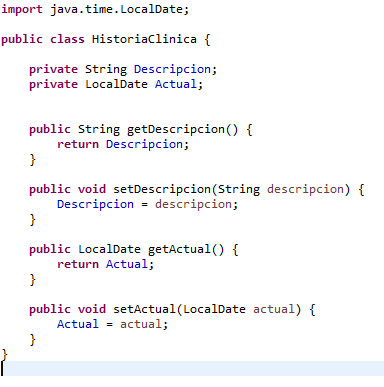
pág. 11

## 

## PAQUETE MODELO

En el paquete Modelo se evidencia la creación de los datos que son utilizados para operar en el paquete constructor y para mostrar en el paquete vista. Esta clase almacena información relevante, como los registros de pacientes, historiales médicos, y detalles de usuarios, y proporciona métodos para acceder, modificar y actualizar estos datos. Además, el Modelo instancia cada uno de los datos en el programa para permitir manejar la programación orientada a objetos dentro de todo el aplicativo. También cumple la función de actualizar los métodos que se ejecutan en el paquete controlador, para así manejar los datos con exactitud y de forma correcta en cada parte del aplicativo. En resumen, el modelo es aquel que da la función para que todo el aplicativo tenga sentido al total de sí mismo, verificando, actualizado e instanciando cada dato que se usa en la aplicación del sistema.

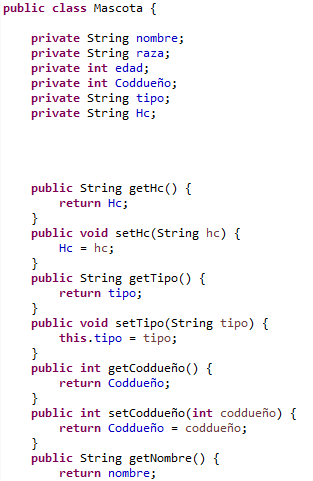
* **Clase HistoriaClinica:** En esta clase se instancia cada uno de los parámetros que necesita la historia clínica para ser ejecutada en cualquier parte del programa.



En esta clase se incluyen los métodos getter y setter de cada variable, que permiten que estos puedan actualizarse y llamarse en el programa.

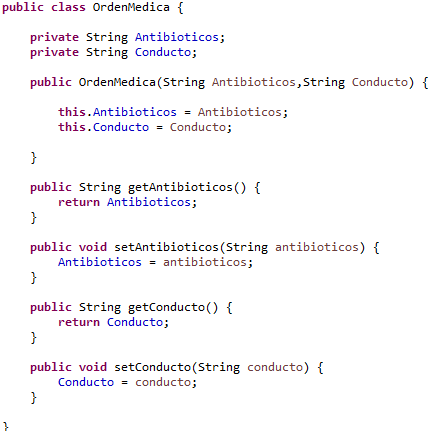
pág. 12

* **Clase Mascota:** En esta clase se declaran todas las variables que se necesitan conocer de la mascota para que esta pueda ser registrada con éxito dentro del sistema.

****

En esta clase se incluyen los métodos getter y setter de cada variable, que permiten que estos puedan actualizarse y llamarse en el programa.

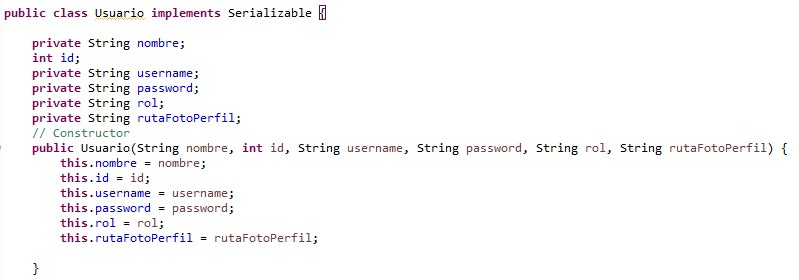
* **Clase OrdenMedica:** En esta clase se hace referencia a los antibióticos y conducto a seguir cuando se atiende a una mascota, esta cuenta con dos variables que se evidencian a la hora de digitar una historia clínica.



En esta clase se incluyen los métodos getter y setter de cada variable, que permiten que estos puedan actualizarse y llamarse en el programa.

pág. 13

* **Clase Usuario:** En esta clase se anuncia cada una de las variables y requisitos que tiene cada uno de los empleados para ser parte de la veterinaria. Estos datos son los que le permiten al sistema asignarle un rol a cada uno y darle función en el aplicativo. Esta clase se implementa serializable para darle más dinamismo y efectividad dentro de las funciones internas del programa.



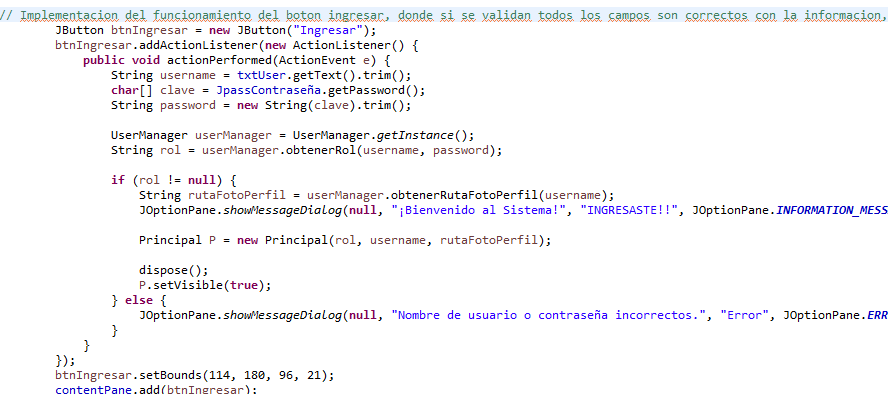
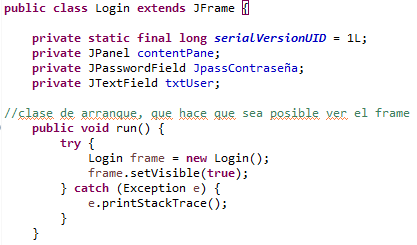
En esta clase se incluyen los métodos getter y setter de cada variable, que permiten que estos puedan actualizarse y llamarse en el programa. Además del método toString, que sirve para diligenciar cada usuario de forma correcta en el archivo de texto.

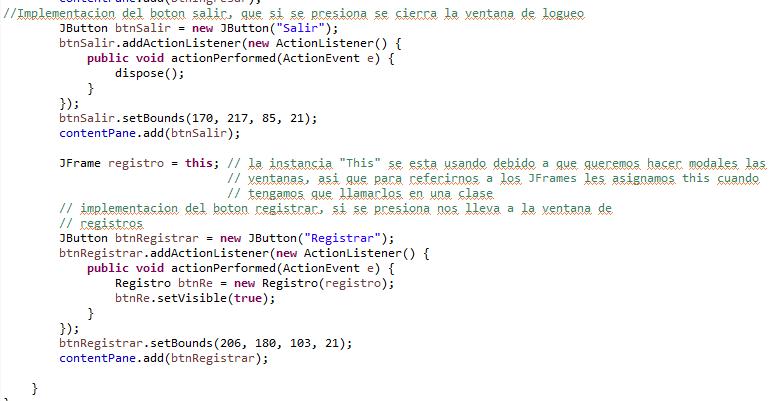
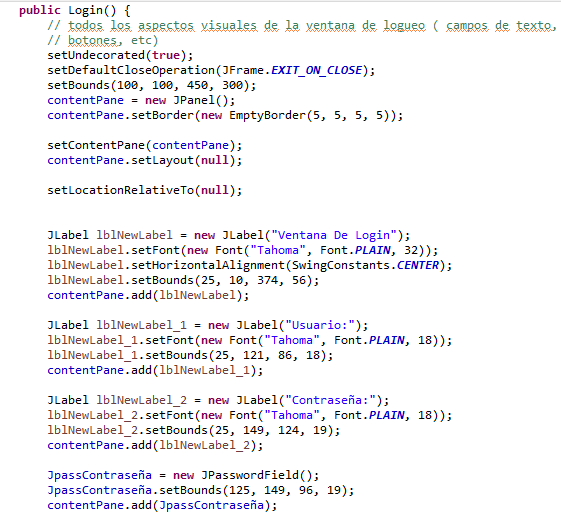
## PAQUETE VISTA

El paquete Vista es responsable de la presentación de la información al usuario y el almacenamiento de cada una de las intervenciones que el usuario realice en el aplicativo. Este paquete se encarga de crear y mostrar la interfaz de usuario, incluyendo ventanas, formularios y elementos gráficos, a través de los cuales los usuarios interactúan con la aplicación. El paquete vista está capacitado gracias a su entorno grafico para recibir acciones como clics, practica con botones, almacenamiento en consola, entre otras capacidades que han sido empleadas junto a los otros paquetes del modelo MVC. Su función es hacer que el usuario pueda interactuar con el aplicativo y desarrollar las diferentes tareas que busca dentro de este mismo, sabiendo que este paquete debe contar con la suficiente practicidad para que le usuario pueda indagar y hacer todos sus proced9ientos de manera práctica y sencilla.

pág. 14

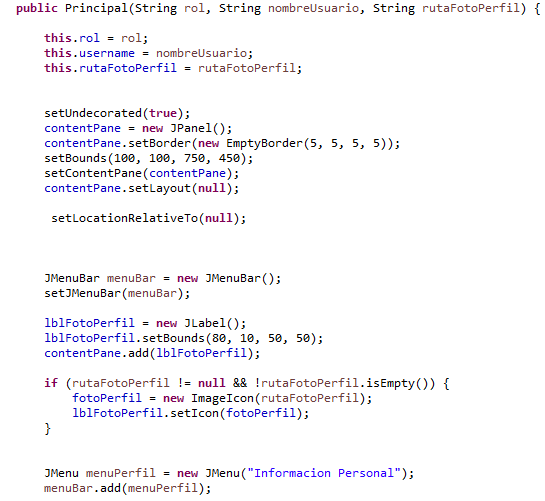
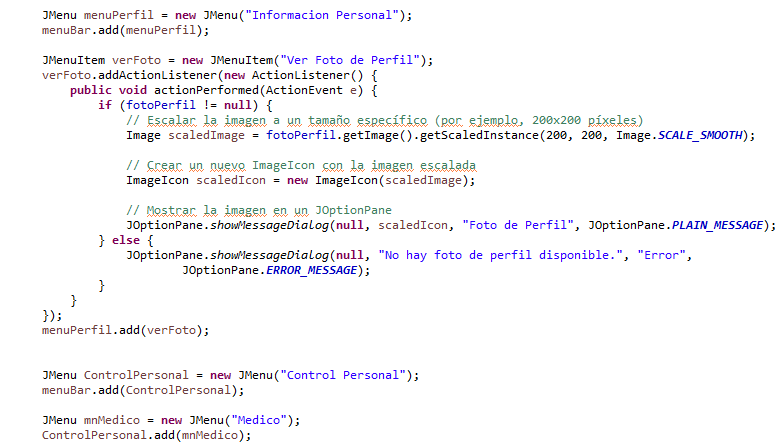
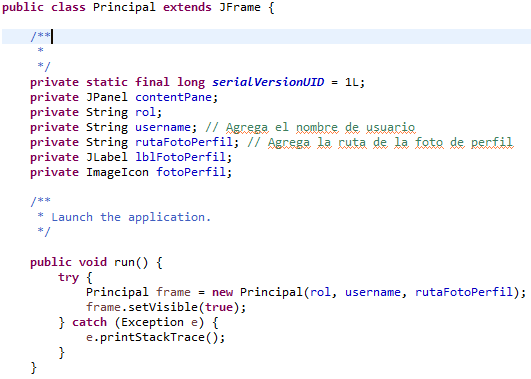
* **Clase Login:** Esta clase trae todo el entorno grafico implementado en el login.





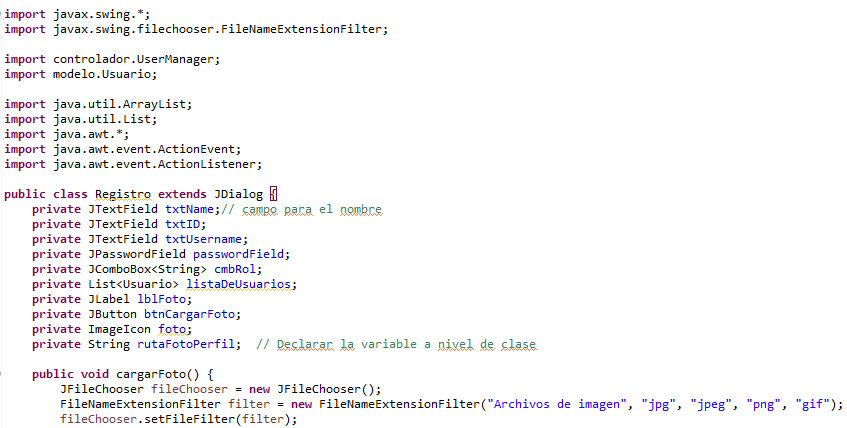
pág. 15

* **Clase Principal:** Esta clase brinda toda la interfaz de la página principal del aplicativo.

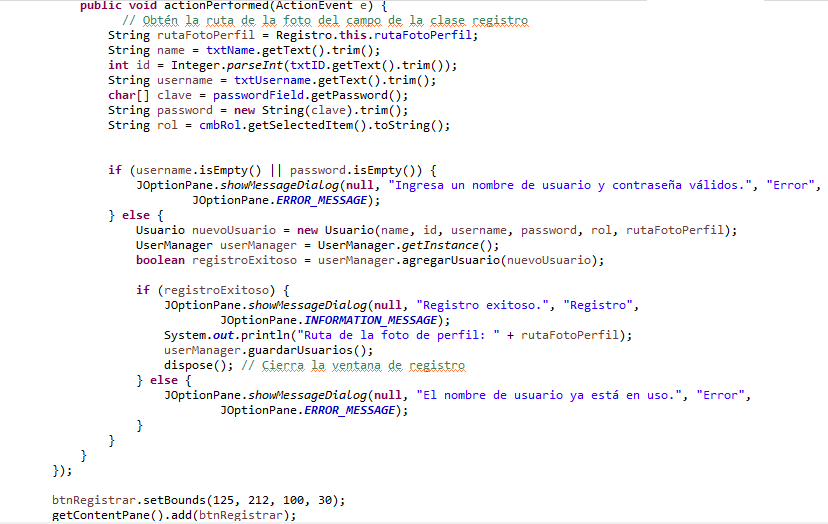


pág. 16

* + **Clase Registro:** Emplea toda la interfaz del registro general en la aplicación



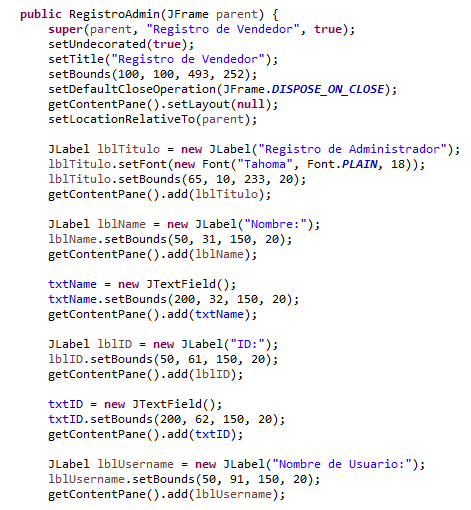


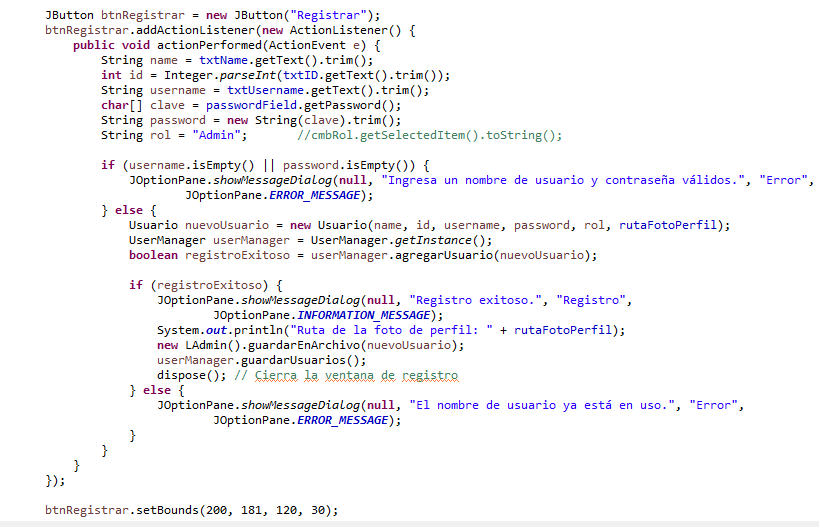


pág. 17

* **Clase RegistroAdmin:** Gestiona la interfaz de registro en el administrador.

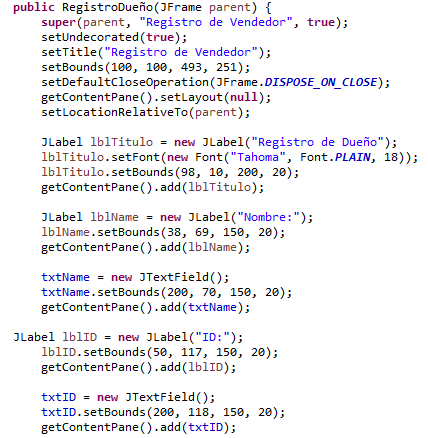




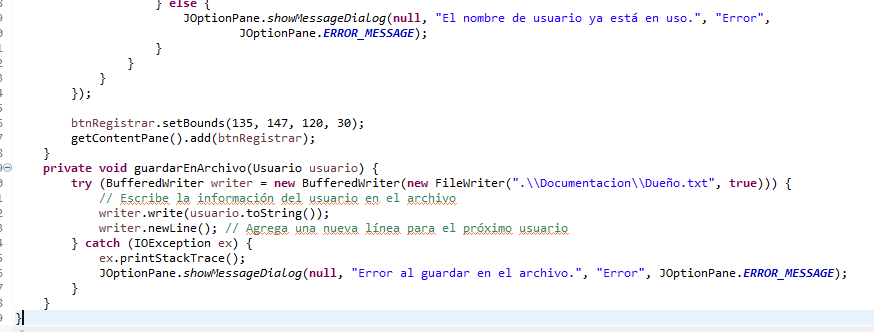


pág. 18

* **Clase RegistroDueño:** En esta el dueño de la mascota puede ver todo su proceso dentro de la veterinaria.

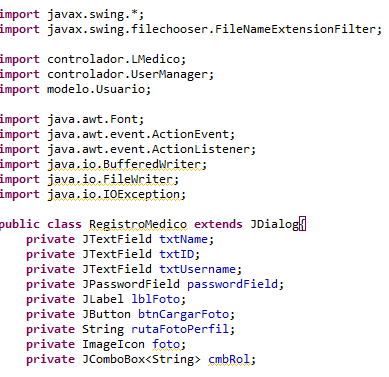
****

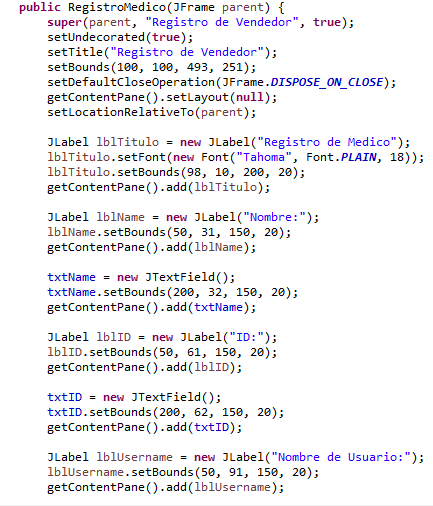
****

****

pág. 19

* Clase RegistroMedico: En esta clase se ve registrada la interfaz del registro en el rol de medico.

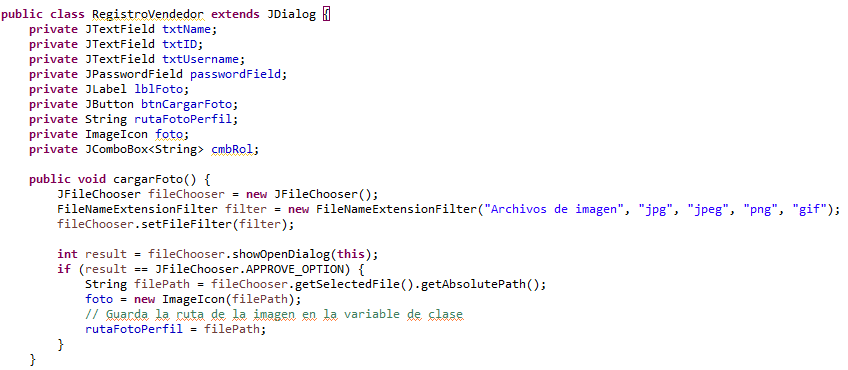


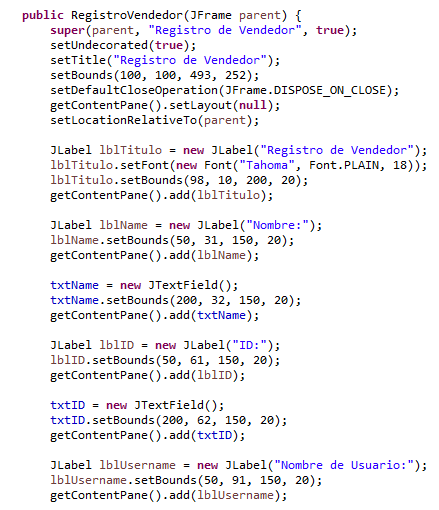


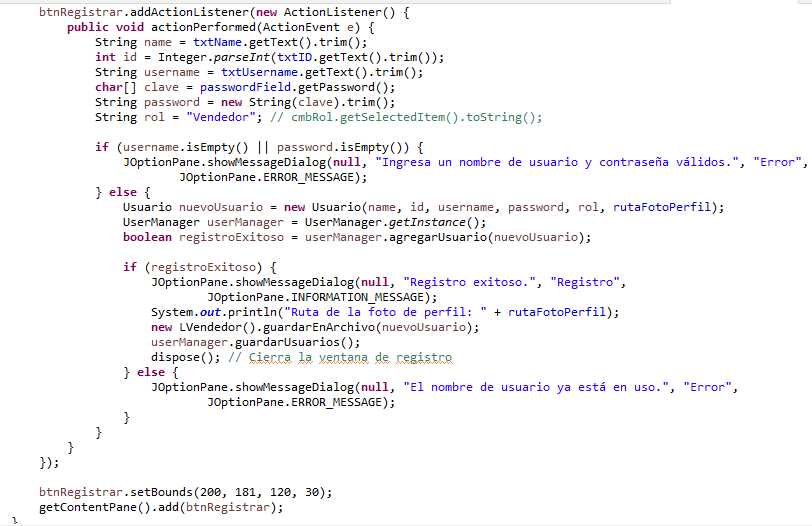


pág. 20

* **Clase RegistroVendedor:** En esta clase se muestra la interfaz gráfica del rol de vendedor.

****

****

****

pág. 21

# 

# Conclusiones

Para concluir con este trabajo, mostramos todas las funcionalidades y como estas han sido desarrolladas e implementadas en cada parte de la aplicación, para ver como cada parte se une a la otra y evidenciar el manejo de paneles gráficos y programación orientada a objetos en una misma aplicación, además de hacer uso del manejo de archivos planos para el almacenamiento de datos y funcionalidad dentro de la aplicación haciendo practica de todo lo hecho en clase.

Esta aplicación parte de la base de ser práctica y útil en todos sus sentidos, dando así una mejor experiencia para el usuario a la hora de cambiar sus ámbitos laborales.

Todas las herramientas, utilidades y funciones que han sido aplicadas son administradas para darle un mejor manejo al programa y a su vez darle simpleza y eficacia a todos los usuarios.

pág. 22