Proyecto final

Proyecto final del ciclo de desarrollo de aplicaciones multiplataforma

Jose Javier Martínez Samperio

Proyecto de DAM

Índice

[Resumen 2](#_Toc195432459)

[Palabras clave 2](#_Toc195432460)

[Introducción 2](#_Toc195432461)

[Objetivos 3](#_Toc195432462)

[Análisis del contexto 4](#_Toc195432463)

[Entorno General (Macroentorno) 4](#_Toc195432464)

[Entorno Específico (Microentorno) 5](#_Toc195432465)

[Análisis DAFO 5](#_Toc195432466)

[Diseño 6](#_Toc195432467)

[Diseño de la interfaz 6](#_Toc195432468)

[Modelo de datos 6](#_Toc195432469)

[Diagrama de clases 8](#_Toc195432470)

[Planificación 8](#_Toc195432471)

# Resumen

El proyecto consiste en una aplicación de gestión para una clínica veterinaria. Con ella los clientes podrían crear cuentas y loggearse en el sistema para gestionar sus propias citas y ver las fichas de sus animales.

Por otro lado, los empleados de la clínica tendrían acceso a las fichas de los pacientes que tengan asignados, permitiéndoles gestionar sus citas y acceder a los historiales de sus pacientes, así como a información de los dueños.

También habrá una cuenta de administrador para el gerente de la clínica, que le permite un acceso más amplio a todas las características de la aplicación, como por ejemplo el de poder crear nuevas cuentas para empleados o ver toda la información de empleados, clientes y pacientes.

Para el funcionamiento de esta aplicación se desarrolla un backend con java spring boot, permitiendo así crear un API Rest que será consumido por la aplicación creada en Android con Kotlin.

# Palabras clave

Desarrollo de aplicaciones, aplicación móvil, multiplataforma, veterinaria, gestión de clínicas, framework, Spring Boot, Java, API REST, backend, Android, Kotlin, base de datos, gestión de citas, usuarios, autenticación, roles y permisos, dueños de mascotas, veterinarios, administrador, fichas médicas, historial clínico, seguridad, interfaz gráfica, CRUD, microservicios, optimización, escalabilidad…

# Introducción

La gestión e interacción con los clientes en empresas de tamaño pequeño y mediano puede resultar una tarea laboriosa, tanto para los clientes como para los empleados. En muchos casos, estas empresas utilizan sistemas tradicionales basados en papel o software desactualizado, lo que genera ineficiencias, dificultades en la organización de la información y posibles errores en la gestión. Con el avance de la tecnología y la digitalización de los servicios, es fundamental contar con herramientas que optimicen estos procesos y mejoren la experiencia de los usuarios.

Este proyecto propone el desarrollo de una aplicación móvil multiplataforma para la gestión de una clínica veterinaria, facilitando la administración de citas y fichas médicas de los pacientes. La aplicación está diseñada para cubrir las necesidades tanto de los clientes como del personal de la clínica, proporcionando un entorno intuitivo y accesible.

Desde la perspectiva del cliente, la aplicación permite crear cuentas personales, gestionar citas y acceder a la información de sus mascotas de forma rápida y sencilla. Por otro lado, los empleados pueden consultar las fichas de los pacientes asignados, gestionar su agenda y acceder a los historiales clínicos, optimizando el tiempo de trabajo y mejorando la atención veterinaria. Además, se incluirá un rol de administrador para el gerente de la clínica, quien podrá supervisar la gestión integral del sistema, crear nuevas cuentas de empleados y acceder a la información general de la clínica.

Técnicamente, el proyecto se desarrollará utilizando un backend basado en Java con el framework Spring Boot, lo que permitirá la creación de una API REST para gestionar la comunicación con la base de datos. La aplicación móvil será desarrollada en Kotlin para Android, garantizando una interfaz moderna, fluida y adaptada a las necesidades del usuario.

Uno de los aspectos más destacados de esta aplicación es su escalabilidad y capacidad de adaptación a otros sectores. Con pequeños ajustes en la interfaz y en la base de datos, el sistema podría aplicarse a otras empresas que requieran gestión de citas y administración de clientes, como consultorios médicos, peluquerías o talleres mecánicos. Esta versatilidad aumenta su potencial de aplicación y expande las posibilidades de evolución del proyecto.

En un contexto donde la digitalización de los servicios es cada vez más relevante, este proyecto busca ofrecer una solución eficiente, segura y accesible para mejorar la gestión de clínicas veterinarias, optimizando el tiempo y la organización tanto de clientes como de empleados. A largo plazo, podría incorporar funcionalidades adicionales, como recordatorios automáticos de citas o integraciones con otros sistemas, lo que permitiría seguir mejorando la experiencia del usuario y la eficiencia del negocio.

# Objetivos

El objetivo principal del prooyecto es diseñar y desarrollar una aplicación móvil multiplataforma que permita a los clientes gestionar sus citas, visualizar el historial médico de sus mascotas y recibir notificaciones sobre eventos importantes, como recordatorios de citas o vacunaciones.

Para lograr esto debemos pasar por objetivos más específicos como:

1. Implementar un sistema de autenticación y gestión de usuarios con distintos roles (clientes, empleados y administradores), asegurando que cada usuario tenga acceso a la información y funcionalidades adecuadas según su perfil.
2. Desarrollar un backend en Java con Spring Boot para gestionar la comunicación entre la aplicación y la base de datos mediante una API REST, garantizando eficiencia y seguridad en el intercambio de datos.
3. Crear una interfaz gráfica intuitiva y accesible en Kotlin para Android, asegurando una navegación sencilla y fluida que facilite la experiencia del usuario, tanto para clientes como para empleados.
4. Optimizar la gestión de citas e historiales clínicos, permitiendo a los empleados registrar y actualizar la información médica de los pacientes de manera estructurada y accesible en tiempo real.
5. Garantizar la seguridad de los datos de clientes, empleados y mascotas mediante el uso de cifrado y protocolos adecuados de almacenamiento, cumpliendo con las normativas de protección de datos.

# Análisis del contexto

## Entorno General (Macroentorno)

1. Factores Político-Legales

* La gestión de datos médicos de mascotas debe cumplir con regulaciones de privacidad, como el RGPD en Europa.
* Políticas gubernamentales de apoyo a la digitalización de pequeñas empresas podrían beneficiar la adopción de la aplicación.

1. Factores Económicos

* Nuevas leyes, como la española sobre el bienestar animal, puede influir en un aumento de la obtención de mascotas por parte de la gente, lo que influye directamente el flujo de clientes y datos de una clínica veterinaria.
* Crisis económicas pueden afectar la capacidad de pago de los clientes y la inversión en nuevas tecnologías por parte de las clínicas.
* Costes de mantenimiento y desarrollo de la aplicación deben ser equilibrados con su modelo de negocio.

1. Factores Socioculturales

* Aumento del número de hogares con mascotas y mayor preocupación por su bienestar.
* Creciente demanda de servicios digitales y facilidad de gestión a través de aplicaciones móviles.
* Cambio en el comportamiento del consumidor, prefiriendo sistemas de autogestión de citas en lugar de llamadas telefónicas.

1. Factores Tecnológicos

* Avances en la conectividad móvil y la adopción de smartphones impulsan el uso de aplicaciones.
* Desarrollo de API seguras y eficientes con tecnologías como Spring Boot y Kotlin.

## Entorno Específico (Microentorno)

1. Clientes

* Clínicas veterinarias que buscan mejorar la gestión de citas y la organización de historiales médicos.
* Dueños de mascotas que requieren un acceso rápido y fácil a la información sobre sus animales.
* Posibilidad de segmentación del mercado en función del tamaño de la clínica y la cantidad de pacientes.

1. Competidores

* Existen plataformas genéricas de gestión de citas, pero pocas especializadas en el sector veterinario.
* Diferenciación basada en una interfaz intuitiva y funciones adaptadas al sector.

1. Proveedores

* Servicios de hosting y bases de datos en la nube.
* APIs externas para autenticación y gestión de pagos si se implementa un modelo premium.
* Empresas de seguridad informática para garantizar el cumplimiento de normativas de protección de datos.

1. Productos Sustitutivos

* Software de escritorio que algunas clínicas ya utilizan para gestión interna.
* Servicios de reservas telefónicas tradicionales.
* Aplicaciones genéricas de gestión empresarial.

## Análisis DAFO

1. Debilidades

* Requiere inversión inicial en desarrollo y marketing.
* Dependencia de la adopción tecnológica por parte de las clínicas.

1. Amenazas

* Competencia de software veterinario ya existente.
* Resistencia de algunas clínicas a cambiar sus métodos tradicionales.

1. Fortalezas

* Aplicación intuitiva y especializada en el sector veterinario.
* Uso de tecnologías modernas y escalables.

1. Oportunidades

* Expansión a otros sectores que requieran gestión de citas.
* Crecimiento del mercado de mascotas y digitalización del sector.

# Diseño

## 

## Diseño de la interfaz

### Pantallas

#### Login

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.En esta pantalla los usuarios de la aplicación podrán iniciar sesión, si ya tienen una cuenta, para acceder a la aplicación. En caso de no tener una cuenta registrada en el sistema podrán acceder a la pantalla de registro

La pantalla se compone del logo de la aplicación.

Dos cuadros de introducción de texto:

* Email: campo para que el usuario introduzca un correo que este registrado en la aplicación
* Contraseña: campo para que el usuario introduzca su contraseña

Dos botones:

* Iniciar sesión: validará la información introducida del usuario y llevará a la siguiente pantalla
* Llevará al usuario a la pantalla de registro

#### Registro

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

La pantalla de registro dispone de 4 inputText dos de ellos con formato de contraseña un chekBox para aceptar términos y condiciones. Un botón para enviar el registro y un enlace para volver al login.

Cada inputText resalta sus validaciones en caso de que se introduzca de manera incorrecta:

Imagen que contiene Aplicación

El contenido generado por IA puede ser incorrecto.

Forma, Rectángulo

El contenido generado por IA puede ser incorrecto.

Forma, Rectángulo

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Como se puede ver en la imagen el usuario puede usar el botón situado a la derecha del inputText para mostrar la contraseña escrita

Forma, Rectángulo

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

#### Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico El contenido generado por IA puede ser incorrecto.Navegador

La actividad principal de la aplicación es un navegador para que el usuario pueda moverse entre los distintos fragmentos que componen la aplicación.

En la parte superior del navegador (header) aparecerá la imagen del usuario.

A continuación, tendrá los items para la navegación, que le llevará por las diferentes pantallas.

Pulsando fuera del menú de navegación podrá ver en pantalla completa el fragmento que en el que se encuentre situado en ese momento. Es decir, cierra el desplegable del menú.

### Fragmentos

#### Home

Texto, Carta

El contenido generado por IA puede ser incorrecto.

En la pantalla de “Home” el usuario puede tanto ver como cambiar su información personal.

también tendrá un botón para cerrar la aplicación.

#### Animales

Texto

El contenido generado por IA puede ser incorrecto.

En esta pantalla el usuario puede ver la información de sus animales.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.Además, en la parte superior izquierda dispone de un buscador para filtrar la lista de animales por nombre, especie o raza

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

#### Citas

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

En este fragmento se pueden ver las citas que tenga el usuario que no estén cerradas, es decir, que todavía no se hayan realizado.

Se le indicará al usuario el estado de la cita, si esta confirmada o pendiente de confirmación.

En cada cita aparece el motivo de la consulta, la fecha y la mascota para la cual es la cita.

Además, aparecerá también el veterinario encargado de realizar la consulta médica.

## Caso de uso de los usuarios

Diagrama, Dibujo de ingeniería

El contenido generado por IA puede ser incorrecto.

La primera pantalla que aparecerá a la vista de los usuarios es la del Login. En esta pantalla pueden darse dos casos:

1. El usuario está registrado

Si el usuario está registrado deberá cumplimentar la información requerida y con el botón de “Iniciar Sesión” podrá acceder a la aplicación.

1. El usuario no está registrado

Si el usuario no está registrado puede pulsar en el texto de “¿No tienes cuenta, Regístrate?” y acceder a la pantalla de registro.

En la pantalla de registro el usuario puede volver a la pantalla de Login sin registrarse o rellenar los datos requeridos para el registro y pulsando el botón de registrarse, lo que también le llevará de vuelta a la pantalla de registro.

Una vez el usuario hay iniciado sesión en la aplicación verá la pantalla principal.

Esta dispone de un menú lateral desplegable que le permitirá navegar entre las distintas opciones que ofrece la aplicación.

1. Pantalla de “home”:

Aquí los usuarios pueden ver su información personal.

Desde esta pantalla pueden acceder a la pantalla para cambiar su información personal.

* 1. Cambiar información. Deberá rellenar los datos que desee cambiar. Una vez validada la información puede darle al botón para cambiar la información, lo que le llevará de vuelta a la pantalla de Login.
  2. Volver. Si decide no cambiar nada de la información registrada puede volver atrás en la aplicación.

1. Pantalla de “Citas”:

El usuario podrá ver la información de las citas que tenga disponible.

En caso de ser un usuario externo, podrá acceder mediante el botón de “Solicitar cita” a una nueva pantalla, donde deberá cumplimentar la información necesaria para la solicitud.

Si es un veterinario podrá ver las solicitudes de cita pendientes y aceptarlas.

1. Pantalla de “Animales”:

En esta pantalla le aparecerá la lista con todos los animales que están registrados a su nombre en caso de ser un usuario externo. En el caso de que el usuario sea un veterinario le saldrán sus pacientes y si es el administrador, le aparecerán todos los animales registrados en la aplicación.

1. Log Out:

Pulsar en este menú le devolverá a la pantalla de Login

## Modelo de datos

El motor de base de datos que se utilizará para el proyecto será Mariadb. MariaDB es un sistema de gestión de bases de datos relacionales que es compatible con MySQL, ofreciendo un alto rendimiento para aplicaciones como esta que requieren gestión de transacciones y consultas complejas. A continuación, se presenta el diagrama relacional de la base de datos, realizado con la herramienta Draw.io

Diagrama

El contenido generado por IA puede ser incorrecto.

Comenzaremos la explicación con la tabla **Usuarios**. En esta tabla se recogerán los datos de las personas que van a utilizar la aplicación. A parte de los campos necesarios para recoger estos datos, como nombre, apellidos, user (que será el nombre del usuario dentro de la aplicación) …, tiene el campo ID\_Rol, que se trata de la relación con la tabla **Roles**. Esta relación será 1 a 1. Esto lo he decidido así para poder manejar después los permisos de la aplicación sin tener que hacer tablas diferentes para distintos tipos de usuario, ya que la aplicación tendrá usuarios de tipo cliente, veterinarios y administrador. Al hacerlo así permite también añadir nuevos roles en un futuro sin tener que modificar la estructura de la base de datos.

Siguiendo con las relaciones desde la tabla usuarios tenemos la tabla **Pacientes**. Esta se trata de la tabla con los principales datos de los animales, además, esta relacionada con la tabla de usuarios ya que marcará quien es el dueño del animal.

Continuamos las relaciones hacia la derecha de la imagen, pasando a la tabla **Citas**. Esta tabla está dedicada a las consultas de los pacientes. Se relaciona con el paciente que debe acudir a la cita y con el veterinario que le atiende, mediante una clave foránea con la tabla Usuarios. También se relaciona con la tabla **Pagos**, que contendrá la información del método de pago usado en la cita (si es que se ha pagado).

Volviendo a la tabla pacientes, esta vez hacia la izquierda de la imagen, tiene una relación de 1 a 1 con la tabla **Fichas**. Esto es así para facilitar después el poder identificar el historial de una paciente en concreto. Para ello, desde la tabla fichas (que contiene más información relativa al animal como la raza, especie, etc.) vamos hacia la tabla **Historiales**. Esta contiene todo el historial clínico de todos los animales de la clínica. Cada animal tendrá un historial. Esto facilitará la búsqueda de enfermedades padecidas por cada paciente y los tratamientos que se utilizaron.

Ya que cada animal puede haber tenido o tener varias enfermedades y cada enfermedad puede ser tenida por cualquier paciente se requiere una relación entre las tablas de historiales y **Enfermedades** de n a m. Por tanto, necesitamos una tabla de paso entre ellas dos. La tabla **historiales\_enfermedades** contiene la información relativa al tratamiento de cada animal y, como se ha comentado anteriormente, relaciona la tabla de historiales con enfermedades, la cual contiene el nombre y una breve descripción de las enfermedades que pueden padecer los pacientes.

Este modelo proporciona una base sólida para desarrollar un sistema completo de gestión veterinaria con seguimiento médico, agenda de citas, gestión de usuarios y sistema de facturación.

## Diagrama de clases

UML de la API

Para no complicar visualmente el diagrama de clases de la API, se han omitido en este tanto los modelos como los servicios, ya que no aportan demasiada información dado que las relaciones se pueden ver bien con los controladores y los modelos.

Diagrama, Esquemático

El contenido generado por IA puede ser incorrecto.

# Planificación

Diagrama de Gantt

# Implementación

## Implementación de la API REST

La API se desarrolla en el lenguaje de programación, Java utilizando el framework Spring Boot. Esto simplificará de forma considerable la configuración y el desarrollo, permitiendo crear la API sin tener que preocuparse mucho de configuraciones iniciales.

Además, Spring Boot cuenta con una gran cantidad de herramientas integradas que facilitan tareas comunes como el acceso a base de datos, lo cual es esencial para la programación de una API REST.

Para iniciar el proyecto con Java y Spring Boot se utilizará la página de [Spring Initializr](https://start.spring.io/), lo que permitirá descargarse un proyecto, como se menciona anteriormente, con la configuración inicial necesaria para el desarrollo de la aplicación.

El sistema de construcción que se ha elegido para la implementación es Maven. Principalmente porque facilita la gestión de dependencias mediante un archivo de configuración (pom.xml).

Por otra parte, facilita la escalabilidad del proyecto a futuro, ya que se trata de un sistema estandarizado, con una comunidad activa y estable, lo que permite poder buscar nuevas herramientas, las cuales disponen de mucha información y documentación detallada.

Además, como el entorno de desarrollo elegido para la creación es NetBeans, Maven dispone de compatibilidad inmediata con este IDE, sin tener que buscar configuraciones adicionales para el funcionamiento tanto del lenguaje como del framework.

Las principales dependencias, ajenas a la configuración inicial de java, que se utilizarán son:

* **Spring Data JPA**

Es una dependencia pensada para facilitar el acceso y la persistencia en base de datos relacionales, MariaDB en este caso. Utiliza Hibernate como proveedor por defecto.

Usar este conjunto de bibliotecas permitirá la eliminación de código repetitivo que normalmente se escribe en las capas de acceso. Nos da acceso a unas interfaces, como JpaRepository, que permitirán a Spring generar las implementaciones en tiempo de ejecución y que traen métodos predefinidos como save, findById, findAll, delete, etc., que facilitan el trabajo a la hora de implementar las funcionalidades de la API.

* **Conector MariaDB**

Driver necesario para conectar la aplicación con la base de datos. Traduce las instrucciones que el programa envía para que el motor de base de datos las pueda entender y ejecutar.

* **Spring Web**

Es una de las dependencias fundamentales cuando se trabaja con Spring Boot para desarrollar un API REST. Integra automáticamente varios componentes esenciales como Tomcat y bibliotecas para el manejo de JSON, como Jakson.

La principal utilidad es facilitar la creación de endpoints HTTP.

Incluye funcionalidades como el manejo automático de errores, como devolver un 404 si no se encuentra el recurso.

Para hacer el código mas limpio, el programa se ha dividido en paquetes según la utilidad de las clases que contienen:

* Modelos

Este paquete contiene todas las clases que conforman las entidades, es decir, la equivalencia de objetos en Java con las tablas de la base de datos.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

* Repositorios
* Servicios
* Controladores

## Implementación gráfica de la aplicación