Institut Puig Castellar
Santa Coloma de Gramenet

Patitas Conectadas

CFGS Desenvolupament d'Aplicacions WEB
Autores: Fernando Diaz, Mouad Sedjari
Grup: DAW2
Curso académico: 2024/2025
GitHub: Front-Patitas-Conectadas y API-PatitasConectadas

Fernando Diaz y Mouad Sedjari

Resumen del proyecto:

Patitas Conectadas consiste en una red social dedicada a mascotas. En esta red social los dueños de mascotas tendrán un perfil de usuario y otro de sus mascotas, podrán subir contenido de sus mascotas a la red social, enviar mensajes privados a otros usuarios, buscar otros usuarios, valorar a otros usuarios después de dejar a sus mascotas con ellos, crear eventos y grupos.

El objetivo no es tener una red social totalmente acabada y con todas las implementaciones de seguridad que requiere, se pretende crear desde cero para entender la estructura, complejidad y las tecnologías que se utilizan, se espera conseguir una red social funcional y que el usuario no perciba errores. A futuro se puede seguir mejorando y añadiendo funcionalidades. También se puede utilizar como base para otras redes sociales.

Se pretende llevar a cabo un proyecto de desarrollo, pero será inevitable que no se mezcle con una parte de investigación para ver qué tecnologías son las más adecuadas y actuales para desarrollar una red social. Finalizada la investigación de las tecnologías se aplicaran las más óptimas para las funciones deseadas.

El resumen final es que el proyecto pretende abordar las metodologías utilizadas actualmente en desarrollo de aplicaciones web escogiendo las más óptimas para una red social dentro de nuestras capacidades al nivel actual de conocimientos. Con la idea de utilizar el máximo uso de software libre y de crear un proyecto aprovechable para la comunidad de desarrolladores y de usuarios que compartan el amor por sus mascotas.

Palabras clave:

Red social

Animales

Mascotas

Grupos

Eventos

Cuidar

Mascotas

Personas

Abstract:

Connected Pawprints is a social network dedicated to pets. In this platform, pet owners will have both a user profile and profiles for their pets. They can upload content of their pets, send private messages to other users, search for other users, rate them after leaving their pets with them, create events, and groups.

The aim is not to have a fully polished social network with all the necessary security implementations, but to build it from scratch to understand its structure, complexity, and the technologies involved. The goal is to achieve a functional social network where users do not perceive errors. Future iterations will focus on continuous improvement and adding features. This project can also serve as a foundation for other social networks.

The development will involve research to determine the most suitable and current technologies for building a social network. Once the research phase is complete, the optimal technologies will be applied to achieve the desired functionalities.

In summary, the project aims to implement current web application development methodologies, selecting the most optimal ones for a social network within our current capabilities. The goal is to maximize the use of open-source software and create a project beneficial for both the developer and user communities who share a love for their pets.

Keywords:

Social network

Animals

Pets

Groups

Events

Care for

Pets

People

Índex

1. Introducción..... 1
1.1. Contexto..... 1
1.2. Justificación..... 2
1.3. Objetivos..... 2
1.3.1. Objetivo general..... 2
1.3.2. Objetivos específicos..... 3
1.4. Estrategia y planificación del proyecto..... 4
1.4.1. Estrategia elegida..... 4
1.4.2. Justificación y viabilidad de la estrategia elegida..... 5
1.4.2.1. Personalización y enfoque en el usuario..... 5
1.4.2.2. Viabilidad técnica y operativa..... 5
1.4.2.3. Viabilidad económica y de mercado..... 5
1.5. Metodología de trabajo..... 5
1.6. Estudio económico y presupuestario..... 8
1.6.1. Costes de desarrollo..... 8
1.6.2. Costes de lanzamiento y mantenimiento..... 9
1.6.3. Total del proyecto..... 9
2. Descripción del proyecto..... 11
2.1. Análisis de los requisitos..... 11
2.1.1. Requisitos funcionales..... 12
2.1.2. Requisitos no funcionales..... 13
2.2. Tecnologías..... 14
2.2.1. Comparativa de las tecnologías valoradas..... 14
2.2.2. Tecnologías escogidas..... 15
2.2.2.1. Frontend:..... 15
2.2.2.2. Backend:..... 15
2.2.2.3. Base de datos:..... 16
2.3. Estructura del proyecto..... 16
2.3.1. Conexión entre componentes..... 16
2.3.2. Forntend..... 17
2.3.3. Backend..... 18
2.4. Descripción de los componentes..... 19
2.4.1. Componentes del Frontend..... 20
2.4.1.1. Login..... 20
2.4.1.2. Registro..... 21
2.4.1.3. Navegación..... 21
2.4.1.4. Barra lateral izquierda..... 23
2.4.1.5. Barra lateral derecha..... 24
2.4.1.6. Formulario de post..... 25
2.4.1.7. Post..... 26

2.4.1.8. Comentarios..... 27
2.4.1.9. Amigos..... 27
2.4.1.10. Guardados..... 28
2.4.1.11. Grupos..... 29
2.4.1.12. Eventos..... 30
2.4.1.13. Chat..... 31
2.4.1.14. Conversación..... 32
2.4.2. Componentes del Backed..... 33
2.4.2.1. CorsConfig..... 33
2.4.2.2. WebConfig..... 34
2.4.2.3. DTO Request..... 35
2.4.2.4. DTO Response..... 36
2.4.2.5. Controller..... 37
2.4.2.6. Models..... 38
2.4.2.7. Repositories..... 39
2.4.2.8. JwtUtil..... 40
2.4.2.9. SecurityConfig..... 42
2.4.2.10. Services..... 43
2.5. Definición de las funcionalidades..... 44
2.5.1. Perfiles personalizados de usuarios y mascotas..... 44
2.5.1.1. Creación inicial del perfil..... 44
2.5.1.2. Almacenamiento y estructuración de datos..... 44
2.5.1.2. Visualización del perfil..... 44
2.5.1.3. Edición y actualización continua..... 44
2.5.1.4. Relación y navegación entre perfiles..... 44
2.5.2. Sistema de valoraciones y reseñas..... 45
2.5.2.1. Registro de valoraciones y comentarios..... 45
2.5.2.2. Almacenamiento estructurado de reseñas..... 45
2.5.2.3. Integración con perfiles y publicaciones..... 45
2.5.3. Creación y gestión de posts con interacción..... 46
2.5.3.1. Creación de publicaciones..... 46
2.5.3.2. Almacenamiento y gestión del contenido..... 46
2.5.3.3. Interacción mediante comentarios..... 46
2.5.3.4. Notificaciones y seguimiento de actividad..... 47
2.5.3.5. Integración en la red social de la plataforma..... 47
2.5.4. Chat..... 47
2.5.4.1. Inicio de conversación..... 47
2.5.4.2. Interfaz de mensajería..... 48
2.5.4.3. Transmisión en tiempo real..... 48
2.5.4.4. Seguridad y cifrado..... 48
2.5.4.5. Almacenamiento y persistencia..... 48

2.5.4.6. Notificaciones push....49
2.5.5. Sistema de seguidores....49
2.5.5.1. Seguimiento de usuarios y mascotas....49
2.5.5.2. Registro de relaciones de seguimiento....50
2.5.5.3. Personalización del feed....50
2.5.5.4. Descubrimiento de perfiles....50
2.5.5.5. Notificaciones y visibilidad....51
2.5.6. Gestión de eventos y grupos....51
2.5.6.1. Creación de eventos....51
2.5.6.2. Gestión de inscripciones y participación....52
2.5.6.3. Calendario y recordatorios....52
2.5.6.4. Creación y gestión de grupos....53
2.5.6.5. Moderación y permisos....53
2.5.6.6. Promoción y descubrimiento....53
2.5.7. Notificaciones en tiempo real....54
2.5.7.1. Tipos de notificaciones....54
2.5.7.2. Motor de eventos y suscriptores....55
2.5.7.3. Tecnologías utilizadas....55
2.5.7.4. Centro de notificaciones....55
2.5.7.5. Configuración de preferencias....56
2.5.7.6. Escenarios críticos cubiertos....56
2.5.8. Geolocalización....57
2.5.8.1. Obtención de la ubicación del usuario....57
2.5.8.2. Integración con mapas....57
2.5.8.3. Filtros geográficos en búsquedas....57
2.5.8.4. Promoción de eventos y servicios locales....58
2.5.8.5. Conexión entre usuarios cercanos....58
2.5.8.6. Privacidad y control....58
2.5.9. Apartado de recursos y consejos....59
2.5.9.1. Estructura del contenido editorial....59
2.5.9.2. Publicación y administración del contenido....59
2.5.9.3. Interacción con los usuarios....60
2.5.9.4. Acceso y navegación....60
2.5.9.5. Rol educativo y comunitario....60
2.5.9.6. Integración con otras funciones....61
2.5.10. Chatbot de soporte....61
2.5.10.1. Accesibilidad y visibilidad....61
2.5.10.2. Base de conocimiento....61
2.5.10.3. Interacción conversacional....62
2.5.10.4. Escalamiento a soporte humano....62
2.5.10.5. Personalización y contexto....62

2.5.10.6. Mejora continua y analítica..... 63
2.5.11. Pagos seguros integrados..... 63
2.5.11.1. Integración con pasarelas de pago..... 63
2.5.11.2. Flujo de pago en la plataforma..... 64
2.5.11.3. Seguridad y protección de datos..... 64
2.5.11.4. Gestión de transacciones y comprobantes..... 64
2.5.11.5. Interacción con otros módulos..... 65
2.5.11.6. Administración y soporte..... 65
2.5.12. Interfaz web moderna y responsiva..... 65
2.5.12.1. Diseño adaptable y responsivo..... 66
2.5.12.2. Uso de frameworks modernos..... 66
2.5.12.3. Navegación intuitiva..... 66
2.5.12.4. Optimización de rendimiento..... 67
2.5.12.5. Elementos visuales claros y consistentes..... 67
2.5.12.6. Compatibilidad y pruebas..... 67
2.5.13. API REST documentada y robusta..... 68
2.5.13.1. Diseño RESTful y estandarizado..... 68
2.5.13.2. Documentación completa y accesible..... 68
2.5.13.3. Seguridad y control de acceso..... 69
2.5.13.4. Manejo detallado de errores..... 69
2.5.13.5. Escalabilidad y mantenimiento..... 69
3. Licencia..... 70
4. Plan de desarrollo estructurado..... 71
5. Análisis de mercado..... 72
6. Diseño visual y comparación con competencia..... 74
7. Conclusion..... 75
7.1. Conclusión general del proyecto..... 75
7.2. Conclusión de los objetivos..... 75
7.3. Valoración de la metodologia y planificacion..... 76
7.4. Vision de futuro..... 76
8. Glosario..... 77
9. Bibliografia..... 78
9.1. Links..... 78
9.2. Videos..... 79
9.3. Documentación técnica y APIs..... 79
10. Agradecimientos..... 80
11. Anexos..... 81
11.1. Bibliotecas utilizadas..... 81
11.1.1. Frontend:..... 81
12.1.2. Backend:..... 82
12.1.2.1. Spring Boot Starters..... 82

12.1.2.2. Base de Datos..... 82
12.1.2.3. Seguridad..... 82
12.1.2.4. Documentación..... 82
12.1.2.5. Servidor..... 83
13.1. Herramientas de Desarrollo:..... 83
13.2. Diagramas de clases..... 84
13.3.1. Frontend..... 84
13.3.1.1. src..... 84
13.3.1.2. components..... 84
13.3.1.2.1. amigos..... 84
13.3.1.2.2. auth..... 85
13.3.1.2.3. chat..... 85
13.3.1.2.4. common..... 85
13.3.1.2.5. eventos..... 86
13.3.1.2.6. feed..... 86
13.3.1.2.7. groups..... 87
13.3.1.2.8. home..... 87
13.3.1.2.9. layout..... 88
13.3.1.2.10. notificaciones..... 88
13.3.1.2.11. post..... 89
13.3.1.2.12. profiles..... 90
13.3.1.2.13. routes..... 90
13.3.1.2.14. Savedpost..... 91
13.3.1.3. context..... 91
13.3.1.4. routes..... 92
13.3.1.5. services..... 92
13.3.1.6. types..... 92
13.3.1.7. utils..... 93
13.3.1.8. views..... 93
13.3.2. Backend..... 93
13.3.2.1. config..... 93
13.3.2.2. controllers..... 94
13.3.2.3. models..... 95
13.3.2.4. repositories..... 95
13.3.2.5. security..... 96
13.3.2.6. services..... 96
13.4. Plan de prevención de riesgos laborales..... 96

Lista de figura

F1. Trello del proyecto donde se pueden observar las tareas organizadas..... 7
F2. PlantUML - Diagrama de conexión conceptual..... 16
F3. PlantUML - Diagrama Frontend Conceptual.....17
F4. PlantUML - Backend Conceptual.....18
F5. PlantUML - Diagrama de componentes Frontend..... 19
F6. PlantUML - Diagrama de componentes Backend.....19
F7. Login de inicio.....20
F8. Pantalla de registro.....21
F9. Componente de navegación.....21
F10. Componente, barra lateral izquierda..... 23
F11. Componente, barra lateral derecha..... 24
F12. Componente par publicar un post.....25
F13. Componente para visualizar un post..... 26
F14. Componente Comentarios.....27
F15. Componente Amigos..... 27
F16. Componente Guardaos..... 28
F17. Componente Grupos..... 29
F18. Componente Eventos..... 30
F19. Componente Chat.....31
F20. Componente Conversación..... 32
F21. PlantUML - Diagrama conceptual de CorsConfig.....33
F22. PlantUML - Diagrama conceptual de WebConfig..... 34
F23. PlantUML - Diagrama conceptual de DTO Request.....35
F24. PlantUML - Diagrama conceptual de DTO Response..... 36
F25. PlantUML - Diagrama conceptual de Controller..... 37
F26. PlantUML - Diagrama conceptual de Model..... 38
F27. PlantUML - Diagrama conceptual de Repositories.....39
F28. PlantUML - Diagrama conceptual de JwtUtil..... 41
F29. PlantUML - Diagrama conceptual de SecurityConfig..... 42
F30. PlantUML - Diagrama conceptual de Services.....43
F31. Wireframe del proyecto donde se plantea la estructura del Frontend..... 74
F32. PlantUML - src diagrama de clases UML..... 84
F33. PlantUML - components diagrama de clases UML..... 84
F34. PlantUML - components.amigos diagrama de clases UML.....85
F35. PlantUML - components.auth diagrama de clases UML..... 85
F36. PlantUML - components.chat diagrama de clases UML.....85
F37. PlantUML - components.common diagrama de clases UML.....85
F38. PlantUML - components.eventos diagrama de clases UML.....86
F39. PlantUML - components.feed diagrama de clases UML..... 86
F40. PlantUML - components.groups diagrama de clases UML..... 87

F41. PlantUML - components.home diagrama de clases UML..... 87
F42. PlantUML - components.layout diagrama de clases UML.....88
F43. PlantUML - components.notificaciones diagrama de clases UML.....88
F44. PlantUML - components.post diagrama de clases UML.....89
F45. PlantUML - components.profile diagrama de clases UML.....90
F46. PlantUML - components.routes diagrama de clases UML..... 90
F47. PlantUML - components.Savedposts diagrama de clases UML.....91
F48. PlantUML - context diagrama de clases UML..... 92
F49. PlantUML - routes diagrama de clases UML..... 92
F50. PlantUML - services diagrama de clases UML.....92
F51. PlantUML - types diagrama de clases UML..... 93
F52. PlantUML - utils diagrama de clases UML.....93
F53. PlantUML - views diagrama de clases UML..... 93
F54. PlantUML - config diagrama UML de clases..... 93
F55. PlantUML - controller.dto diagrama UML de clases..... 94
F56. PlantUML - controller.rest diagrama UML de clases..... 94
F57. PlantUML - models diagrama UML de clases..... 95
F58. PlantUML - repositories diagrama UML de clases..... 96
F59. PlantUML - security diagrama UML de clases.....96
F60. PlantUML - services diagrama UML de clases.....96

1. Introducción

En la era digital actual, el vínculo entre las personas y sus mascotas ha tomado una nueva dimensión, extendiéndose al mundo virtual a través de plataformas sociales.

Patitas Conectadas nace como una red social especializada para amantes de las mascotas, ofreciendo un espacio donde los dueños puedan compartir experiencias, encontrar recursos útiles y conectarse con otros usuarios que comparten la misma pasión. El objetivo principal de este proyecto es proporcionar una plataforma integral donde los dueños de mascotas puedan interactuar de manera segura y eficiente, promoviendo la creación de una comunidad activa y comprometida.

Se plantea como una aplicación web que combina las funcionalidades clásicas de una red social con características específicas enfocadas en el cuidado y bienestar de las mascotas. A través de esta plataforma, los usuarios tendrán la posibilidad de crear perfiles tanto para ellos mismos como para sus mascotas, permitiendo una representación virtual que facilite la interacción en la comunidad.

1.1. Contexto

En los últimos años, el cuidado y bienestar de las mascotas ha cobrado una gran relevancia, y cada vez más personas buscan plataformas para compartir sus experiencias, encontrar información útil y conectarse con otros dueños de mascotas. Sin embargo, actualmente, las redes sociales generales como Instagram o Facebook no están adaptadas específicamente para satisfacer estas necesidades, lo que deja un vacío para una comunidad enfocada exclusivamente en el mundo de las mascotas. Los usuarios buscan más allá de compartir fotos, desean acceder a recursos, recomendaciones, y establecer relaciones de confianza para servicios como paseos o cuidado temporal de mascotas.

Además, el aumento del uso de dispositivos móviles ha impulsado el desarrollo de aplicaciones especializadas en nichos específicos, como el cuidado de mascotas. A pesar de que existen algunas plataformas para encontrar cuidadores de mascotas o veterinarios, no hay una solución integral que combine funcionalidades de red social.

Patitas Conectadas surge para cubrir esta necesidad, brindando un espacio seguro y dedicado exclusivamente a dueños de mascotas, donde se integran las funcionalidades clave de una red social junto con herramientas adicionales específicas del ámbito de las mascotas.

1.2. Justificación

El desarrollo de Patitas Conectadas responde a una demanda creciente de los dueños de mascotas por disponer de una plataforma digital especializada que cubra sus necesidades específicas. Hoy en día, las mascotas son consideradas miembros de la familia, y sus dueños buscan no solo compartir contenido sobre ellas, sino también acceder a información confiable, encontrar servicios locales y conectar con personas de confianza para el cuidado de sus animales. La carencia de una aplicación que combine estas funcionalidades en un solo lugar representa una oportunidad clara para crear una solución innovadora y de valor añadido.

El proyecto, por tanto, no solo cubre una necesidad del mercado, sino que también fomenta el desarrollo de un entorno más seguro y accesible para los amantes de las mascotas.

1.3. Objetivos

1.3.1. Objetivo general

El objetivo general de Patitas Conectadas es crear una plataforma digital especializada que sirva como un punto de encuentro integral para los dueños de mascotas, facilitando la conexión, el intercambio de información y el acceso a servicios relacionados con el cuidado de sus animales. La aplicación busca ser una herramienta que centraliza todas estas necesidades en un entorno seguro, confiable y fácil de usar, promoviendo así la creación de una comunidad activa y comprometida que mejore la experiencia de los usuarios y el bienestar de sus mascotas.

1.3.2. Objetivos específicos

Crear perfiles personalizados para usuarios y mascotas, donde se pueda compartir información relevante como nombre, raza, género, edad, peso y fotos, fomentando la interacción y el sentido de comunidad.

Incorporar un sistema de valoraciones y reseñas, que fomente la confianza al permitir que los usuarios califiquen sus experiencias en servicios o actividades con otros miembros.

Permitir la creación de publicaciones con texto e imágenes, ofreciendo a los usuarios un espacio para compartir experiencias, consejos o novedades, junto con la posibilidad de comentar y participar activamente en el contenido de otros.

Implementar un sistema de mensajería directa entre usuarios, que facilite la comunicación privada, el intercambio de información y la organización de encuentros o actividades relacionadas con sus mascotas.

Fomentar la interacción social mediante un sistema de seguidores, permitiendo a los usuarios mantenerse conectados, descubrir nuevas mascotas y generar una red activa dentro de la comunidad.

Incorporar la gestión de eventos y grupos, ofreciendo herramientas para organizar actividades, encuentros o campañas, así como la posibilidad de unirse a grupos de interés común.

Diseñar un sistema de notificaciones en tiempo real, para mantener a los usuarios informados sobre comentarios, nuevos seguidores, eventos y otras interacciones relevantes dentro de la plataforma.

Implementar un sistema de valoraciones y reseñas, que permita a los usuarios calificar servicios, publicaciones o interacciones con otros miembros, fomentando la confianza y la transparencia.

Integrar funcionalidades de geolocalización, para ayudar a los usuarios a encontrar personas, eventos y servicios relacionados con mascotas en su área.

Crear un apartado de recursos y consejos, donde los usuarios puedan acceder a información relevante sobre el cuidado y bienestar de sus mascotas.

Desarrollar un chatbot de soporte, para asistir a los usuarios en el uso de la plataforma y resolver sus dudas de manera inmediata.

Implementar pagos seguros dentro de la plataforma, para servicios y actividades, facilitando la contratación de servicios de cuidado de mascotas o

productos relacionados.

Desarrollar una interfaz web moderna y responsiva, que ofrezca una navegación intuitiva y una experiencia de usuario fluida en todo tipo de dispositivos.

Proporcionar una API REST documentada y completa, que facilite la integración con otros sistemas, el desarrollo de futuras aplicaciones móviles o servicios externos, y un manejo de errores robusto.

1.4. Estrategia y planificación del proyecto

Para llevar a cabo el desarrollo de *Patitas Conectadas*, se han considerado distintas estrategias:

Desarrollar un producto nuevo desde cero, con funcionalidades diseñadas específicamente para el público objetivo

Adaptar una aplicación existente, reutilizando una base tecnológica de una red social genérica o una aplicación de gestión.

Utilizar herramientas de desarrollo low-code/no-code, que permiten acelerar el desarrollo, aunque con limitaciones en personalización y escalabilidad.

1.4.1. Estrategia elegida

La estrategia seleccionada ha sido el desarrollo de un producto completamente nuevo, basado en el concepto de una red social, pero adaptado a las necesidades específicas del público objetivo. Se ha optado por crear una plataforma desde cero para garantizar una experiencia centrada en el cuidado de mascotas y la interacción entre usuarios, integrando funcionalidades como perfiles de mascotas, mensajería privada, valoraciones y seguimientos.

Aunque la idea se inspira en redes sociales existentes, se ha considerado que ninguna solución previa ofrece una integración eficaz de estas funcionalidades en un mismo entorno. Por ello, adaptar una plataforma genérica resultaría técnica y funcionalmente limitada frente a los objetivos definidos para este proyecto.

1.4.2. Justificación y viabilidad de la estrategia elegida

1.4.2.1. Personalización y enfoque en el usuario

Desarrollar un producto desde cero, inspirado en el funcionamiento de una red social, ha permitido alinear la experiencia de usuario con las necesidades reales del público objetivo: personas que conviven con mascotas y buscan una comunidad funcional, intuitiva y segura. Esta aproximación facilita la integración específica de funcionalidades clave, imposibles de lograr eficazmente mediante la adaptación de soluciones ya existentes.

1.4.2.2. Viabilidad técnica y operativa

El equipo de desarrollo cuenta con experiencia en tecnologías modernas (React, Spring Boot, PostgreSQL).

Se utilizarán buenas prácticas de desarrollo desde el inicio.

La estrategia contempla una metodología ágil que permite adaptarse a cambios.

1.4.2.3. Viabilidad económica y de mercado

Existe un alto interés en aplicaciones de nicho orientadas a mascotas.

Se prevé un modelo de monetización en la aplicación. La inversión inicial está justificada por el potencial de crecimiento y retorno a largo plazo.

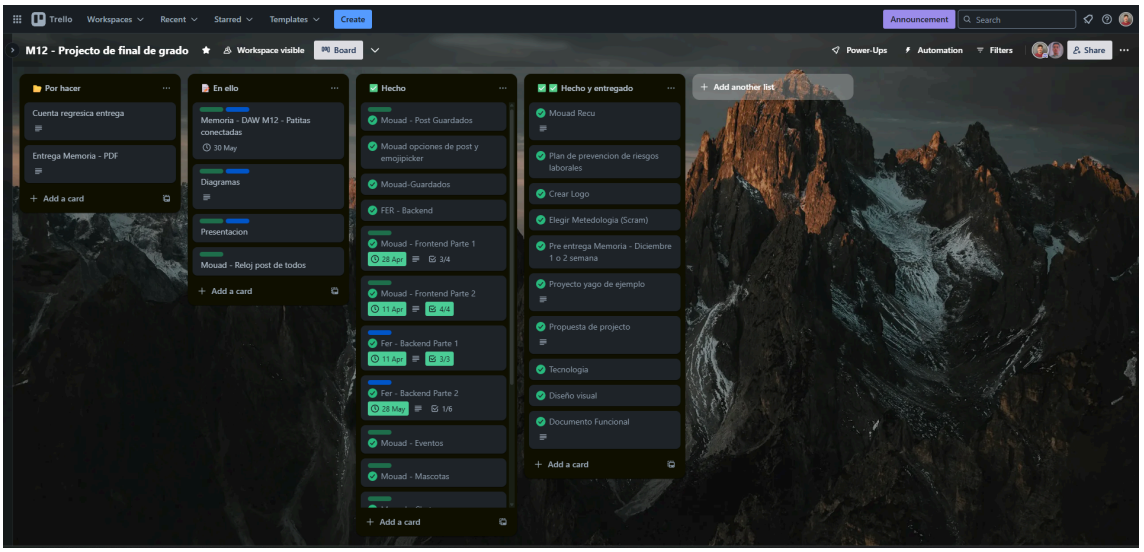
1.5. Metodología de trabajo

Para el desarrollo del proyecto *Patitas Conectadas*, se ha optado por seguir una metodología ágil, concretamente Scrum, que permite una gestión iterativa e incremental del trabajo. Esta metodología ha sido aplicada a través del uso de la herramienta Trello, que ha servido para organizar, asignar y hacer seguimiento de las tareas entre los dos miembros del equipo de manera clara y colaborativa.

Comparativa entre metodologías:

Categoría	Waterfall	Scrum	Kanban
Tipo de metodología	Tradicional	Ágil	Ágil visual
Estructura del trabajo	Fases secuenciales:	Iteraciones cortas llamadas sprints	Flujo continuo con tareas en

	análisis → diseño → desarrollo → pruebas → mantenimiento	(2–4 semanas) o Por hacer, En proceso, Hecho (To Do, Doing Done)	columnas (To Do, In Progress, Done)
Flexibilidad	Baja. Los cambios son costosos una vez iniciado el desarrollo	Alta. Permite adaptarse a cambios en cada sprint	Muy alta. Ideal para entornos donde las prioridades cambian frecuentemente
Definición de requisitos	Completamente definidos desde el inicio	Evolutivos. Pueden ajustarse a medida que el proyecto avanza	Adaptativos. Se trabajan conforme aparecen o cambian las necesidades
Colaboración del equipo	Limitada durante el desarrollo. Participación fuerte solo en etapas iniciales	Colaboración constante del equipo con el cliente y revisiones periódicas	Alta colaboración visual y continua mediante el tablero
Entrega de valor	Al final del proyecto	Parcial y frecuente tras cada sprint	Continua, a medida que las tareas se completan
Adecuado para…	Proyectos bien definidos y estables (por ejemplo, sistemas bancarios o administrativos)	Proyectos cambiantes, dinámicos o centrados en el usuario final	Continua, a medida que las tareas se completan
Ejemplos de herramientas	MS Project, diagramas de Gantt	Trello, Jira, Taiga	Trello, Kanbanize, Notion



F1. Trello del proyecto donde se pueden observar las tareas organizadas.

1.6. Estudio económico y presupuestario

1.6.1. Costes de desarrollo

Concepto	Detalle	Coste estimado
Salario de desarrolladores	2 desarrolladores × 6 meses × 1.500 €/mes	18.000 €
Herramientas y software	Hosting, repositorio, dominio, Trello Pro, etc.	300 €
Diseño y recursos gráficos	Icons, UI kits, diseño gráfico adicional	200 €
Formación	Cursos, guías, documentación	100 €
Gastos operativos	Oficina, conectividad, electricidad	300 €
Otros	Fondo para licencias, backups, incidencias menores	100 €

Subtotal Costes de Desarrollo: 19.000 €

1.6.2. Costes de lanzamiento y mantenimiento

Concepto	Detalle / Justificación	Coste estimado
Dominio y hosting anual	Dominio personalizado y servidor cloud	200 €
Mantenimiento y soporte inicial	Correcciones, pequeños ajustes, mejoras menores	500 €
Publicidad	Campañas en redes, banners, contenido promocional	200 €
Consultoría legal mínima (protección de datos)	Documentación básica RGPD y condiciones legales	100 €

Subtotal Costes de Lanzamiento y Mantenimiento: 1.000 €

1.6.3. Total del proyecto

Total estimado del proyecto
20.000 €

Viabilidad y oportunidad de beneficio:

Coste controlado y ajustado para el desarrollo completo de una red social funcional desde cero.

Se hace uso intensivo de tecnologías open source y software libre, reduciendo licencias.

Escalabilidad futura: el proyecto puede crecer o servir como base para productos similares.

Posibilidades de monetización:

La red social *Patitas Conectadas* presenta un amplio potencial de beneficios tanto a corto como a largo plazo:

Publicidad segmentada: Espacios publicitarios para tiendas de mascotas, clínicas veterinarias, educadores caninos o marcas de productos.

Modelo freemium: Acceso básico gratuito con funciones avanzadas o exclusivas para usuarios premium (por ejemplo, ubicación de servicios, perfil verificado o estadísticas de perfil).

Marketplace especializado: Espacio para publicar productos, servicios o adopciones de mascotas, con opción de comisiones o membresías.
Eventos patrocinados: Promoción de eventos comunitarios, ferias o campañas de adopción con patrocinadores locales o regionales.

Licenciamiento de la plataforma: Uso de la plataforma por asociaciones, ONGs o ayuntamientos interesados en una comunidad digital para mascotas.

Estas oportunidades hacen que el proyecto no solo sea sostenible, sino también rentable y escalable, con potencial de convertirse en un referente en el sector digital de las mascotas.

Decisión del cliente:

Este presupuesto orientativo permite al cliente valorar:

Si desea continuar con el desarrollo, asumiendo un posible margen de desviación del 10–15 %.

O bien, si decide aplazar o cancelar el proyecto con base en la inversión estimada.

2. Descripción del proyecto

2.1. Análisis de los requisitos

- **Autenticación y Usuarios**
  - Sistema de registro e inicio de sesión completamente funcional
  - Perfiles de usuario completos y editables
- **Gestión de Mascotas**
  - Registro completo de mascotas con todos sus datos
  - Subida y gestión de fotos de mascotas
  - Vinculación correcta entre mascotas y propietarios
- **Publicaciones y Contenido**
  - Creación de posts con texto e imágenes
  - Sistema de comentarios completamente funcional
- **Chat**
  - Sistema de chat privado entre usuarios
- **Eventos y Grupos**
  - Creación y gestión de eventos
  - Sistema de registro a eventos
  - Creación y gestión de grupos
- **Interfaz de Usuario**
  - Diseño responsive
  - Navegación intuitiva y accesible
  - Feedback visual para todas las acciones
- **API**
  - Documentación completa de endpoints
  - Manejo de errores consistente
- **Notificaciones**
  - Envío de notificaciones en tiempo real
  - Personalización de preferencias de notificación
  - Notificaciones para mensajes, eventos, valoraciones y seguimientos
  - Almacenamiento y consulta del historial de notificaciones

2.1.1. Requisitos funcionales

Aquí están los requisitos funcionales más importantes para Patitas Conectadas:

- **Gestión de Usuarios**
  - Registro de nuevos usuarios con email y contraseña
  - Perfil de usuario personalizable con información básica
  - Búsqueda de usuarios por nombre y apellido
- **Gestión de Mascotas**
  - Registro de mascotas asociadas a usuarios
  - Información detallada de mascotas (nombre, género, raza)
  - Visualización de mascotas por usuario
- **Sistema de Publicaciones**
  - Creación de posts con contenido multimedia
  - Visualización de publicaciones en feed
  - Interacción con publicaciones (comentarios)
- **Sistema de Comentarios**
  - Comentarios en publicaciones
  - Visualización de comentarios por publicación
- **Sistema de Valoraciones**
  - Crear valoraciones por usuarios
  - Visualización de valoraciones en perfil
  - Gestion de valoraciones (Crear, eliminar)
- **Sistema de Chat**
  - Mensajería privada entre usuarios
  - Historial de conversaciones
- **Gestión de Eventos**
  - Creación de eventos relacionados con mascotas
  - Visualización de eventos disponibles
  - Gestión de asistencia a eventos
- **Sistema de Grupos**
  - Creación de grupos temáticos
  - Unión de usuarios a grupos
- **Sistema de Seguimiento**
  - Seguir a otros usuarios
  - Visualización de seguidos
  - Feed personalizado basado en seguidos
- **Gestión de Archivos**
  - Subida de imágenes y archivos multimedia
  - Almacenamiento de archivos
  - Visualización de contenido multimedia
- **Requisitos de Notificaciones**
  - Envío de notificaciones en tiempo real

2.1.2. Requisitos no funcionales

Aquí están los requisitos no funcionales más importantes para Patitas Conectadas:

- **Seguridad**
  - Encriptación de contraseñas
  - Validación de datos en entrada y salida
- **Usabilidad**
  - Interfaz intuitiva y responsive
  - Feedback visual para acciones del usuario
  - Mensajes de error claros y descriptivos
- **Mantenibilidad**
  - Código documentado y bien estructurado
  - Patrones de diseño consistentes
  - Separación clara de responsabilidades
- **Interoperabilidad**
  - API REST bien documentada
  - Formato de datos estandarizado (JSON)
- **Cumplimiento Legal**
  - Política de privacidad
- **Calidad de Código**
  - Documentación inlin

2.2. Tecnologías

2.2.1. Comparativa de las tecnologías valoradas

Categoría	Tecnología 1	Tecnología 2	Comparativa
Framework Frontend	Vue	React	Vue es más fácil de aprender y tiene una sintaxis más sencilla. React ofrece mayor flexibilidad y una comunidad más amplia.
Herramienta de build	Vite	Webpack	Vite es más moderno, rápido y sencillo de configurar. Webpack es más maduro, personalizable pero complejo.
Lenguaje Frontend	JavaScript	TypeScript	JavaScript es más simple y directo, ideal para proyectos pequeños. TypeScript aporta tipado estático y escalabilidad para proyectos grandes.
Lenguaje Backend	PHP	Java con Spring Boot	PHP es más sencillo de implementar en hosting compartido y adecuado para proyectos pequeños. Java con Spring Boot ofrece robustez, estructura y escalabilidad para proyectos complejos.
Base de datos	PostgreSQL	MySQL	PostgreSQL es más avanzado en características (JSON, índices, integridad de datos), ideal para proyectos exigentes. MySQL es más rápido en lectura y tiene una curva de aprendizaje menor.

2.2.2. Tecnologías escogidas

Se realizó un proyecto de prueba para testear PHP con Bootstrap y se descartó debido a la complejidad del proyecto. https://github.com/Fernandodg97/Postea

2.2.2.1. Frontend:

- **React**
  - **Razón de elección:** React es una de las bibliotecas más populares y maduras para el desarrollo de interfaces de usuario. Su componente virtual DOM y su enfoque en componentes reutilizables lo hacen ideal para construir aplicaciones web modernas y mantenibles.
- **TypeScript**
  - **Razón de elección:** TypeScript proporciona tipado estático sobre JavaScript, lo que ayuda a prevenir errores comunes durante el desarrollo y mejora la mantenibilidad del código.

Herramientas de Desarrollo

- **Vite**
  - **Razón de elección:** Vite es un bundler moderno que ofrece tiempos de desarrollo extremadamente rápidos gracias a su arquitectura basada en ESM.

2.2.2.2. Backend:

- **Spring Boot**
  - **Razón de elección:** Spring Boot es un framework maduro y robusto que simplifica el desarrollo de aplicaciones Java empresariales. Su enfoque en la configuración automática y la convención sobre la configuración lo hace ideal para desarrollar APIs REST de manera eficiente.

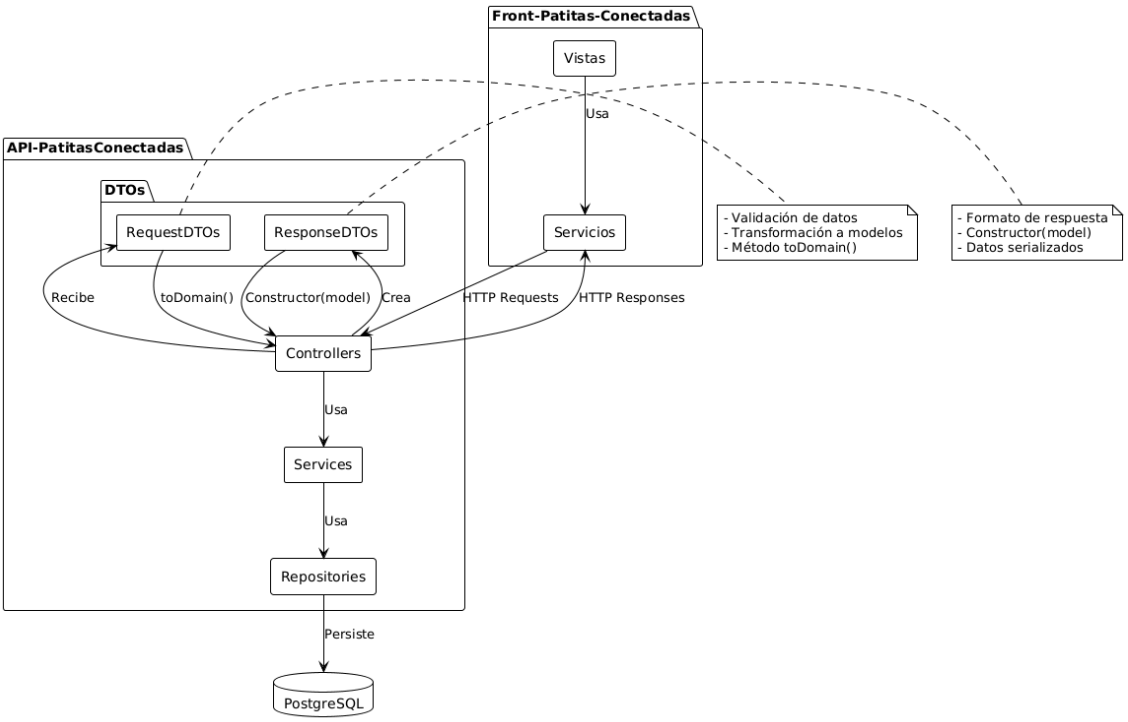
- **Java 21**
  - **Razón de elección:** Java 21 es la versión LTS más reciente que ofrece mejoras significativas en rendimiento y nuevas características como virtual threads y pattern matching.

2.2.2.3. Base de datos:

- **PostgreSQL**
  - Razón de elección: PostgreSQL es una base de datos relacional de código abierto que ofrece características avanzadas como soporte JSON, transacciones ACID, y excelente rendimiento.

2.3. Estructura del proyecto

2.3.1. Conexión entre componentes

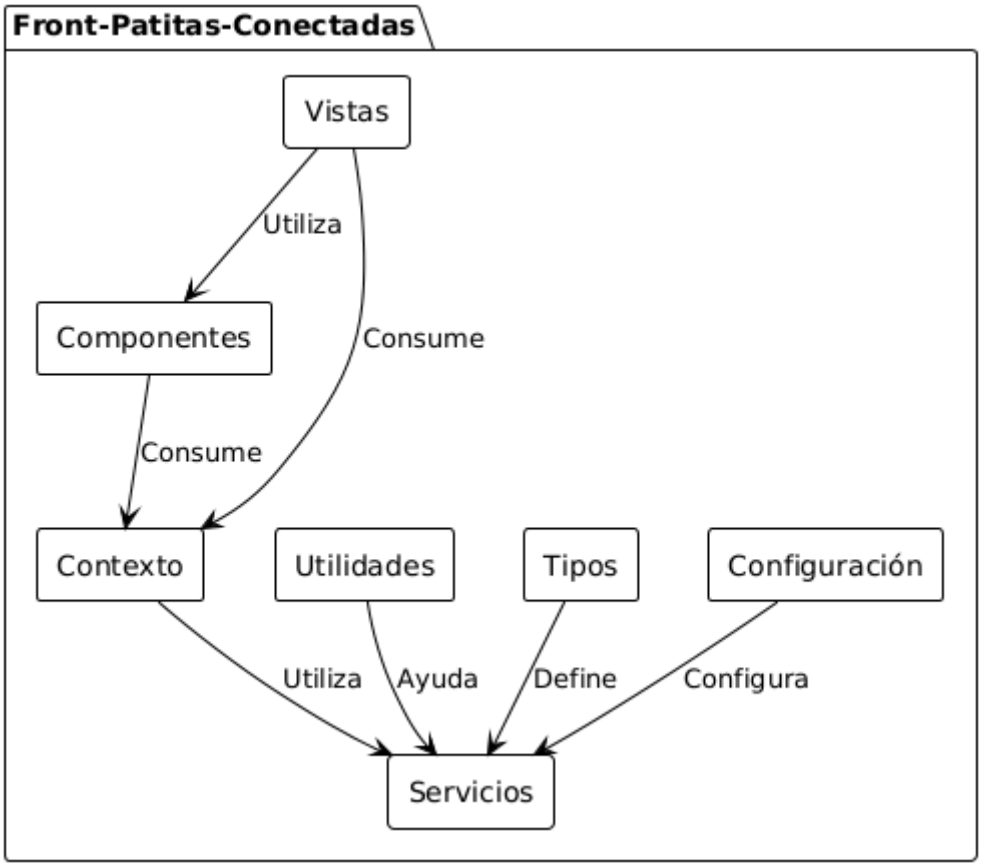


F2. [PlantUML - Diagrama de conexión conceptual]

2.3.2. Forntend

Un frontend es como un edificio bien organizado que:

- Separa las responsabilidades en capas claras
- Utiliza herramientas modernas de desarrollo
- Sigue patrones de diseño establecidos
- Facilita el mantenimiento y escalabilidad
- Optimiza el rendimiento y la experiencia de usuario

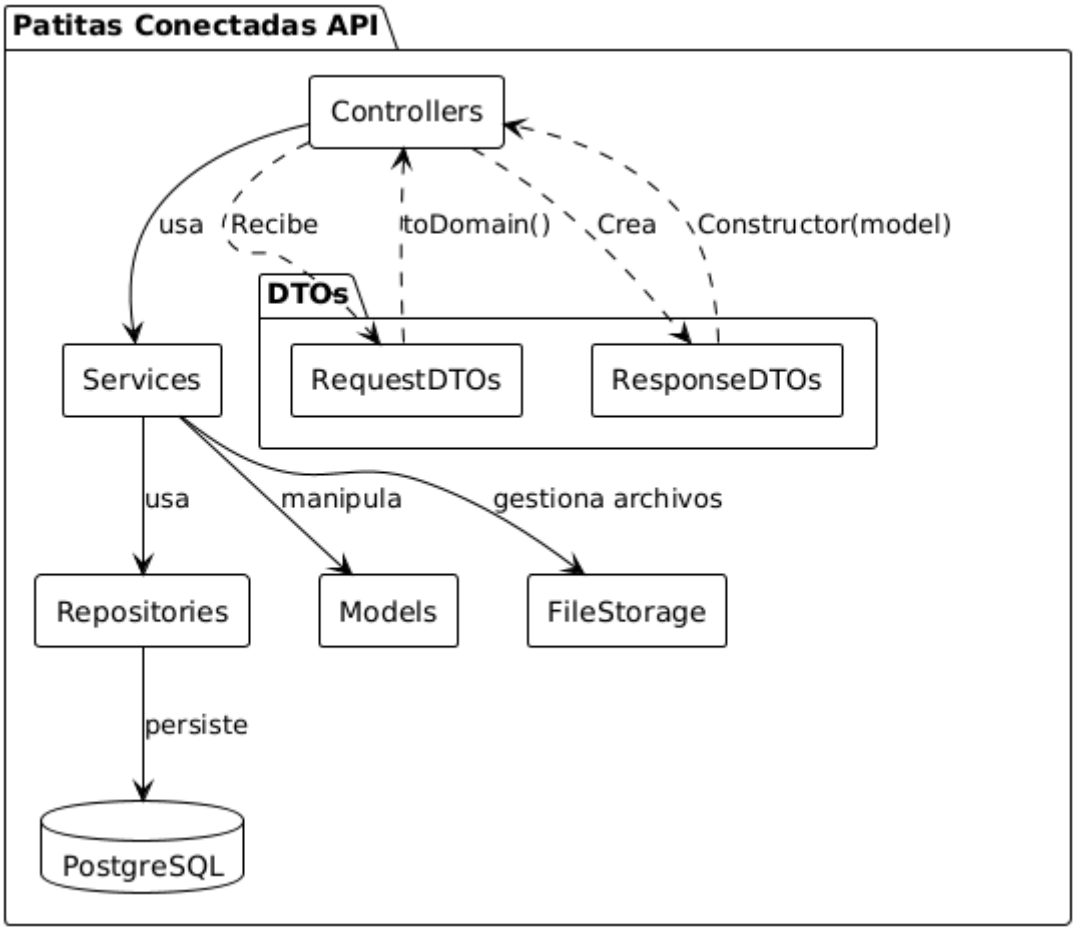


F3. PlantUML - Diagrama Frontend Conceptual

2.3.3. Backend

Un proyecto backend en Spring Boot es como una ciudad bien planificada que:

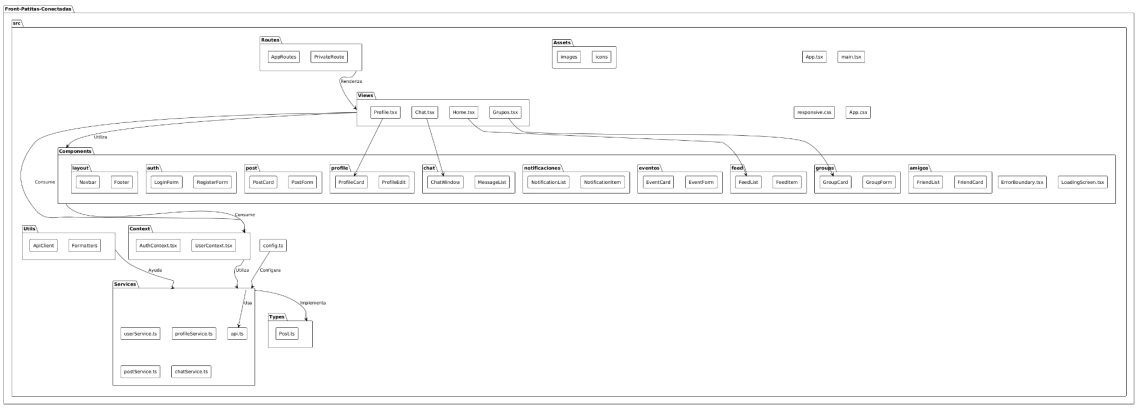
- Organiza los servicios en capas lógicas
- Gestiona los datos de forma segura
- Proporciona APIs bien definidas
- Mantiene la seguridad y el rendimiento
- Facilita el mantenimiento y la escalabilidad



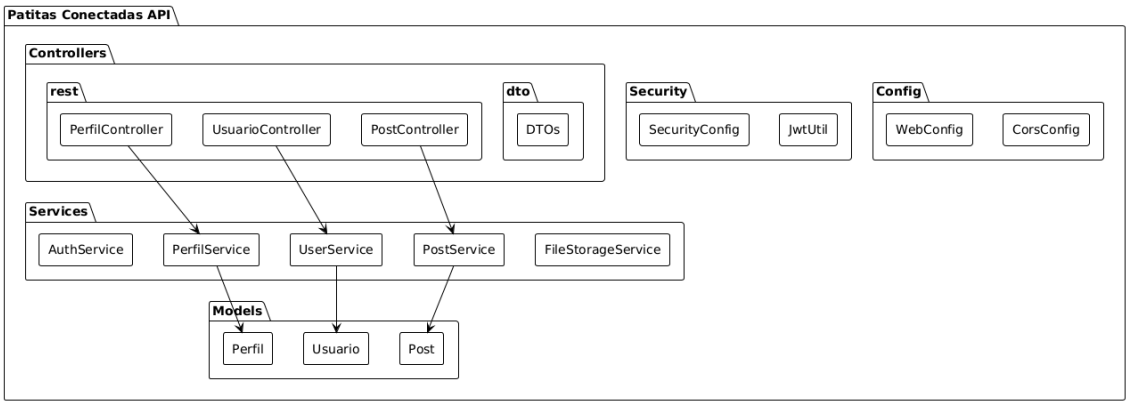
F4. PlantUML - Backend Conceptual

2.4. Descripción de los componentes

En esta sección se encuentra una explicación general de los componentes del proyecto, para ver las clases de cada componente ir a Anexos - Diagrama de clases.



F5. PlantUML - Diagrama de componentes Frontend



F6. PlantUML - Diagrama de componentes Backend

2.4.1. Componentes del Frontend

2.4.1.1. Login

F7. Login de inicio.

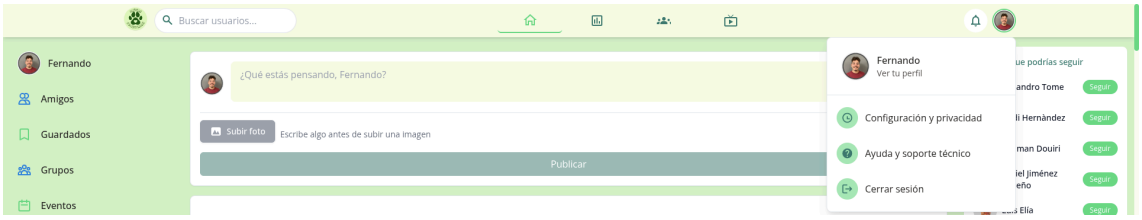
Este componente es un formulario que envía un json a la API Rest para comprobar la existencia del usuario, recibe un token por json que vuelve a enviar a la API Rest en un endpoint para comprobar que es correcto. Una vez validado se avanza al Home.

2.4.1.2. Registro

F8. Pantalla de registro.

Este componente es un formulario que envía un json a la API Rest para crear un nuevo usuario, la API Rest devuelve un json con los datos del usuario. Una vez recibe estos datos replica la funcionalidad del login para avanzar al Home.

2.4.1.3. Navegación

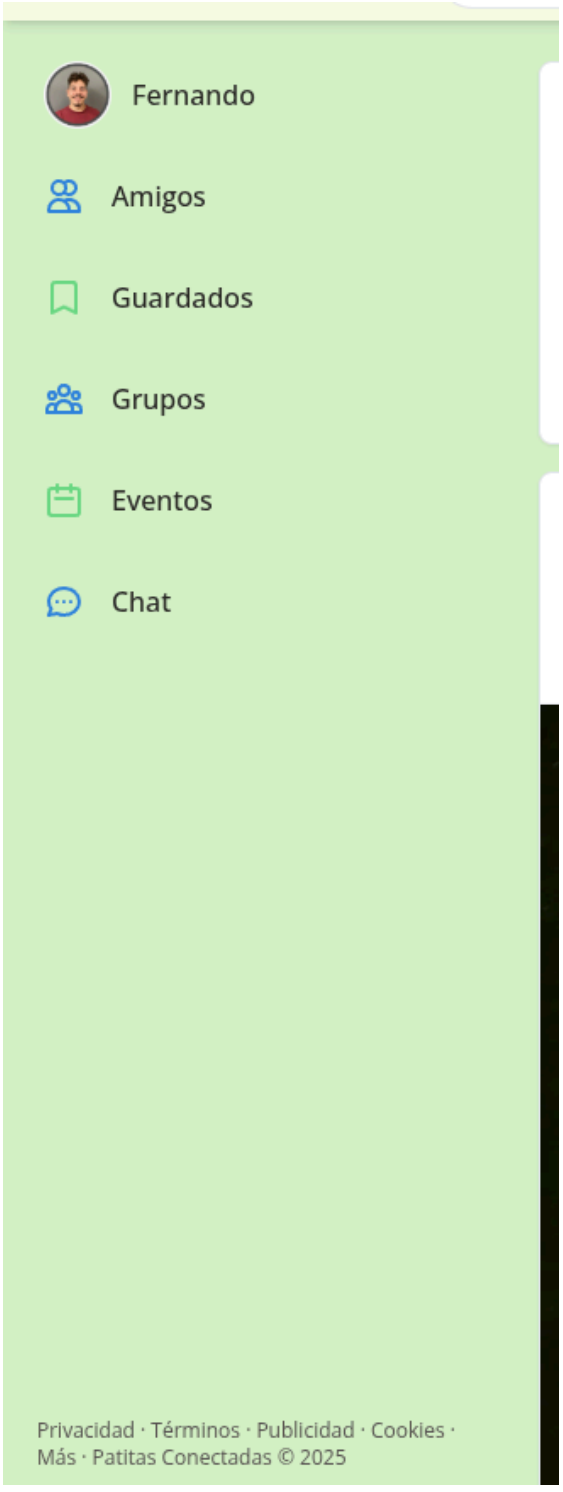


F9. Componente de navegación

Este componente se compone de varias funciones:

- Función que llama a la API Rest y recibe un json con los datos del usuario.
- Función que envía el id del usuario a la API Rest y recibe un json con los datos del perfil. Luego muestra estos datos.
- Función que detecta si en el input de búsqueda se introducen 3 o más caracteres, luego realiza una llamada a la API Rest y recibe un json con los usuarios. Al realizar clic sobre el nombre del usuario te dirige a su perfil.
- Links para navegar a Inicio, Descubrir, Eventos, Grupos.
  Función que realiza una llamada a la API Rest con el id del usuario y recibe un json con todas las notificaciones del usuario.
- Sub menú que muestra los datos del usuario y muestra links a Configuración y privacidad, Ayuda y soporte técnico. También cuenta con una función que destruye la sesión y elimina el token del navegador.

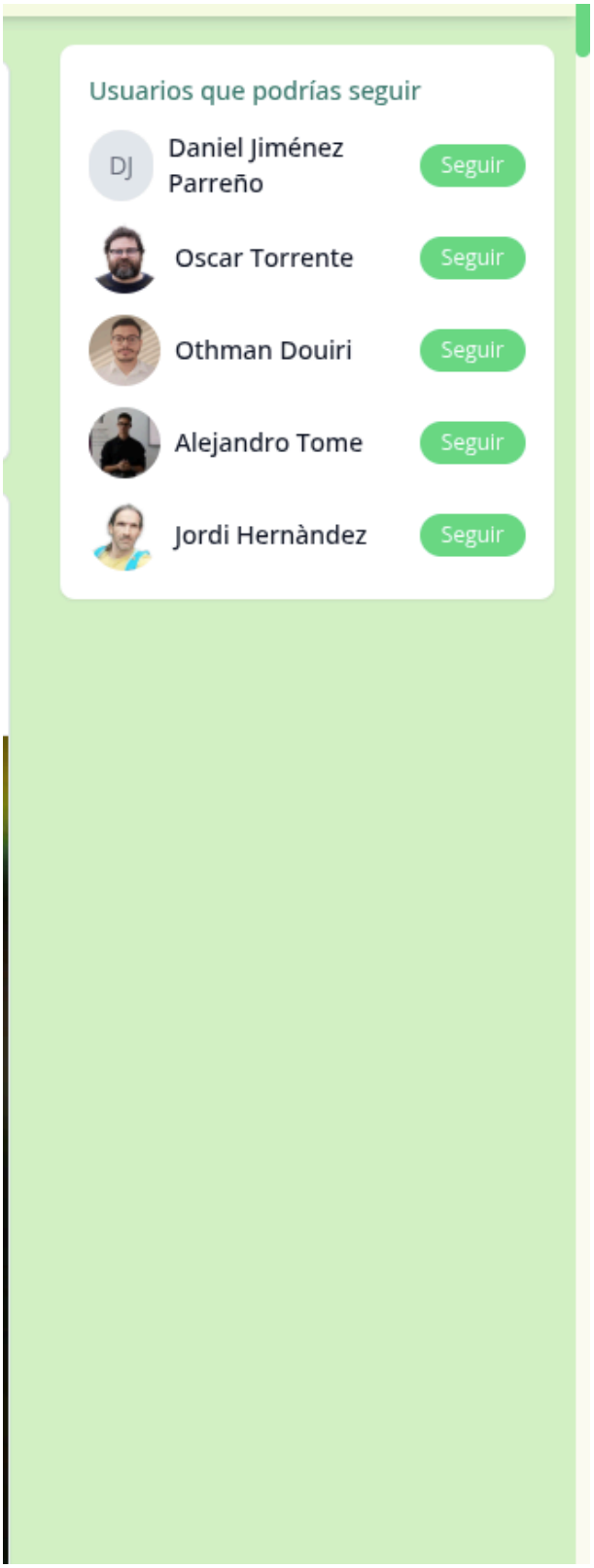
2.4.1.4. Barra lateral izquierda



F10. Componente, barra lateral izquierda.

Este componente se compone funciona como navegador y te dirige a tu perfil, Amigos, Guardados, Grupos, Eventos, Chat. Para mostrar la información de tu perfil realiza una llamada a la API Rest con el id del usuario y recibe un json con los datos, construye una ruta al perfil. Muestra el footer.

2.4.1.5. Barra lateral derecha



F11. Componente, barra lateral derecha.

Este componente se compone de varias funciones:

- Realiza una llamada a la API Rest con el id del usuario y recibe un json con todos los usuarios que sigue.
- Realiza una llamada a la API Rest y recibe un json con todos los usuarios.
- Función que compara todos los usuarios con los seguidos y muestra solamente los usuarios que no son seguidos, con un botón que realiza una llamada a la API Rest enviado el id del usuarios y el id del usuario a seguir.
- Un bucle que recorre los usuarios y realiza una llamada a la API Rest con el id del usuario, recibe un json con los datos del perfil. Muestra la información de cada usuario.

2.4.1.6. Formulario de post

F12. Componente par publicar un post.

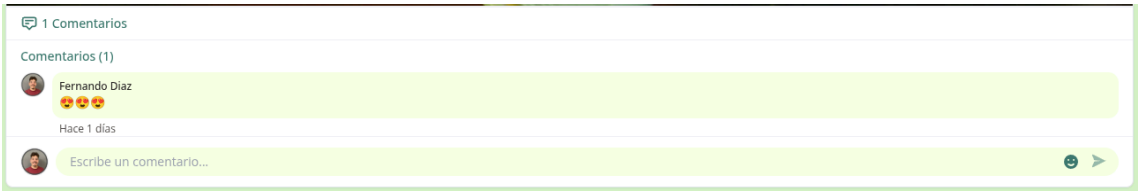
Este componente se compone de un formulario que recoge texto e imágenes, y realiza una llamada a la API Rest mediante un FormData.

2.4.1.7. Post

F13. Componente para visualizar un post.

Este componente realiza una llamada a la API Rest y recibe un json con todos los post, luego muestra la información de cada uno dentro de un card. Construye las URLs, optimiza tamaños. Dispone de un menú que comprueba si el id del usuario en sesión es el mismo que el del autor, si es igual muestra el botón eliminar que realiza una llamada a la API Rest enviando el id del post y elimina el post. Si el id es diferente al del autor muestra un botón guardar que realiza una llamada a la API Rest enviando el id del usuario y el id del post.

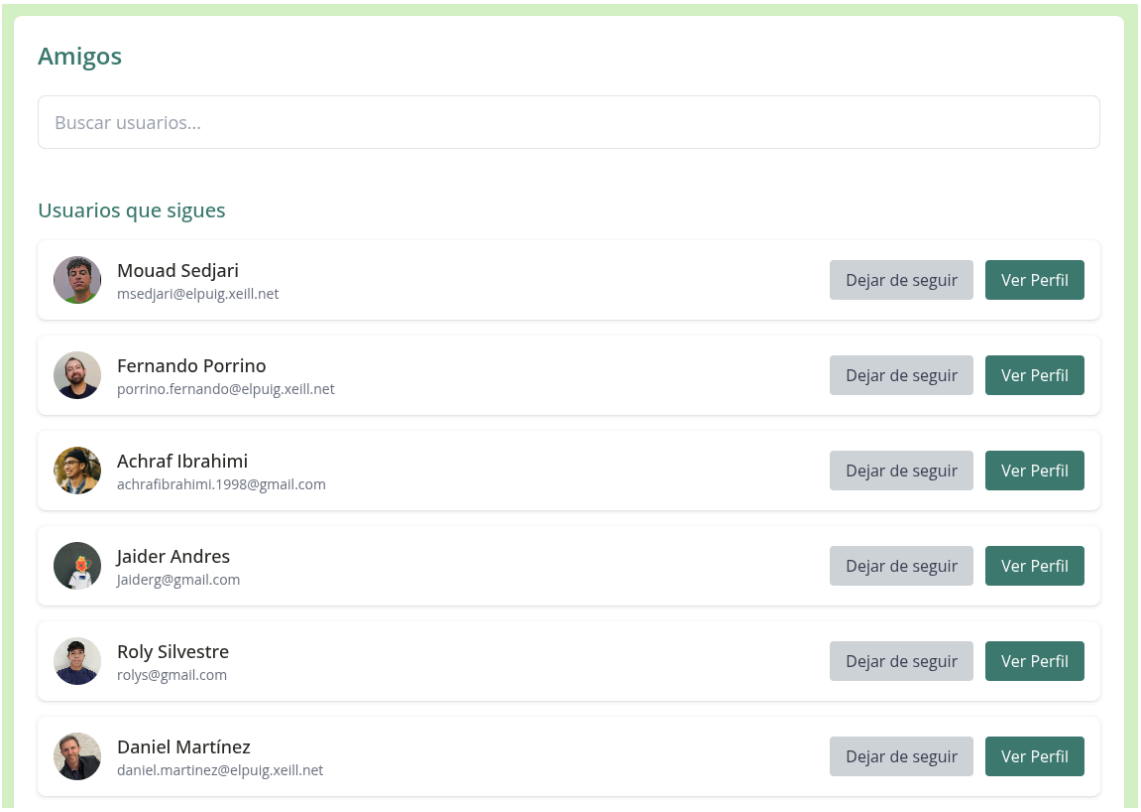
2.4.1.8. Comentarios



F14. Componente Comentarios

Este componente realiza una llamada a la API Rest y recibe un json con todos los comentarios. Cuenta con un formulario que realiza una llamada a la API Rest y envía un json con el comentario.

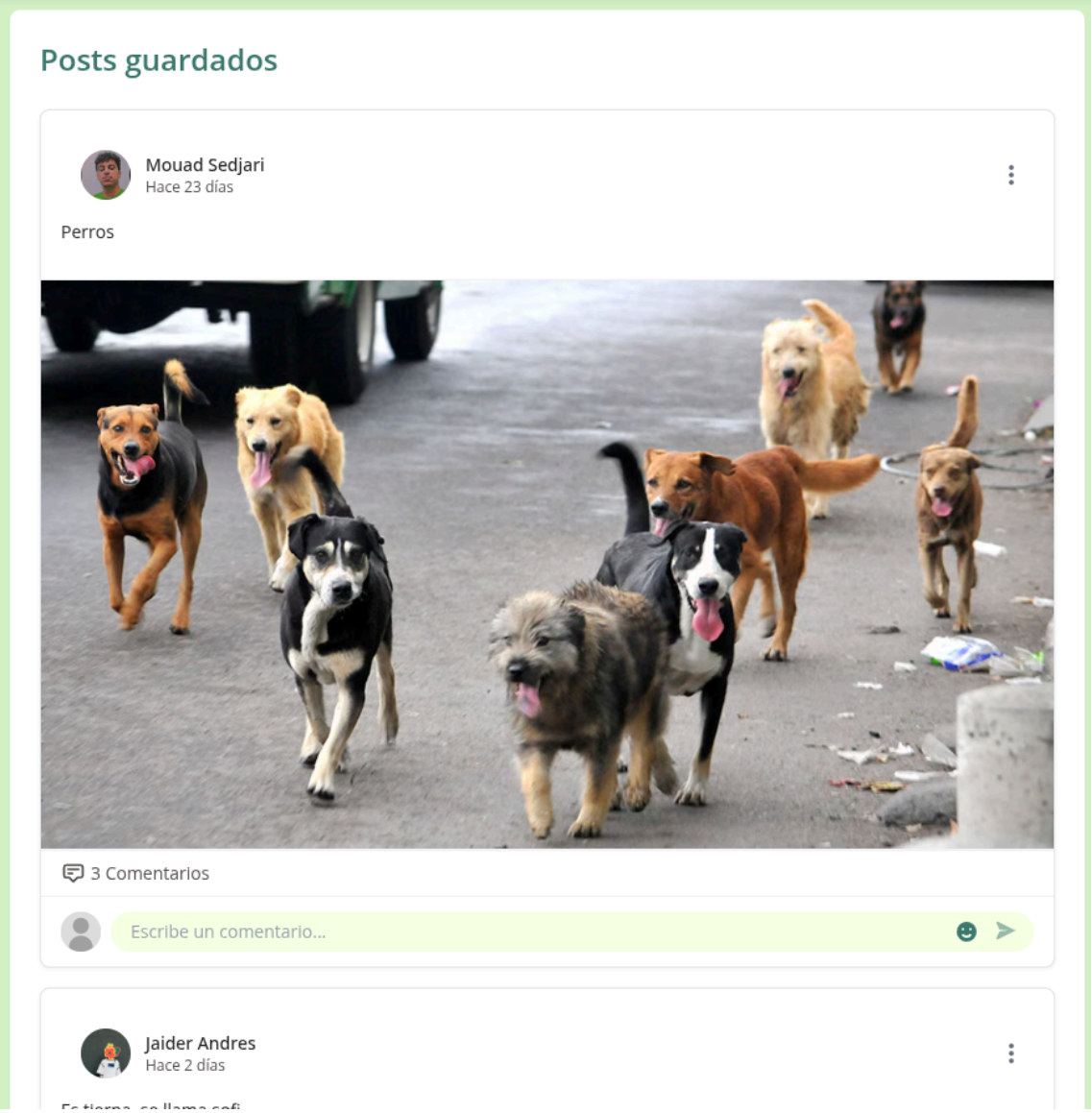
2.4.1.9. Amigos



F15. Componente Amigos.

Este componente realiza una llamada a la API Rest y recibe un json con todos los seguidos. Muestra todos los amigos en una lista. Cuenta con una función que realiza una llamada a la API Rest y envía el id para dejar de seguir.

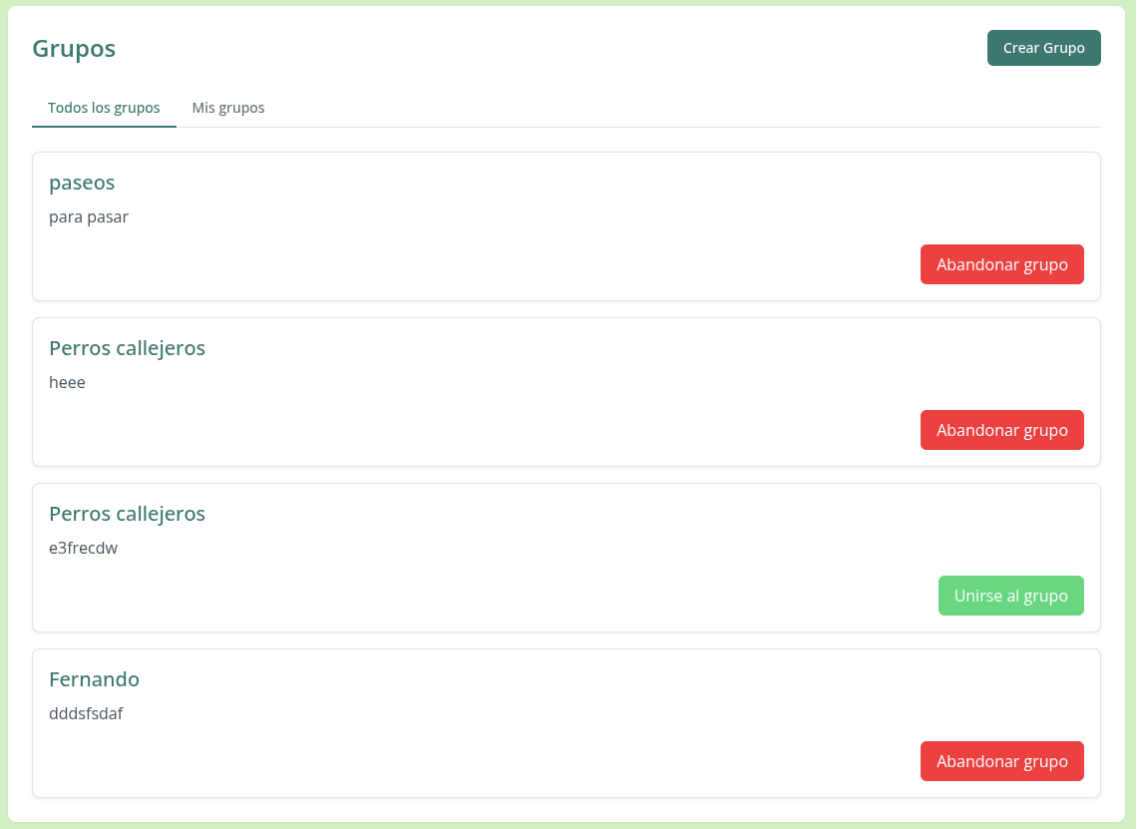
2.4.1.10. Guardados



F16. Componente Guardaos.

Este componente realiza una llamada a la API Rest y recibe un json con todos los post guardados por el usuario.

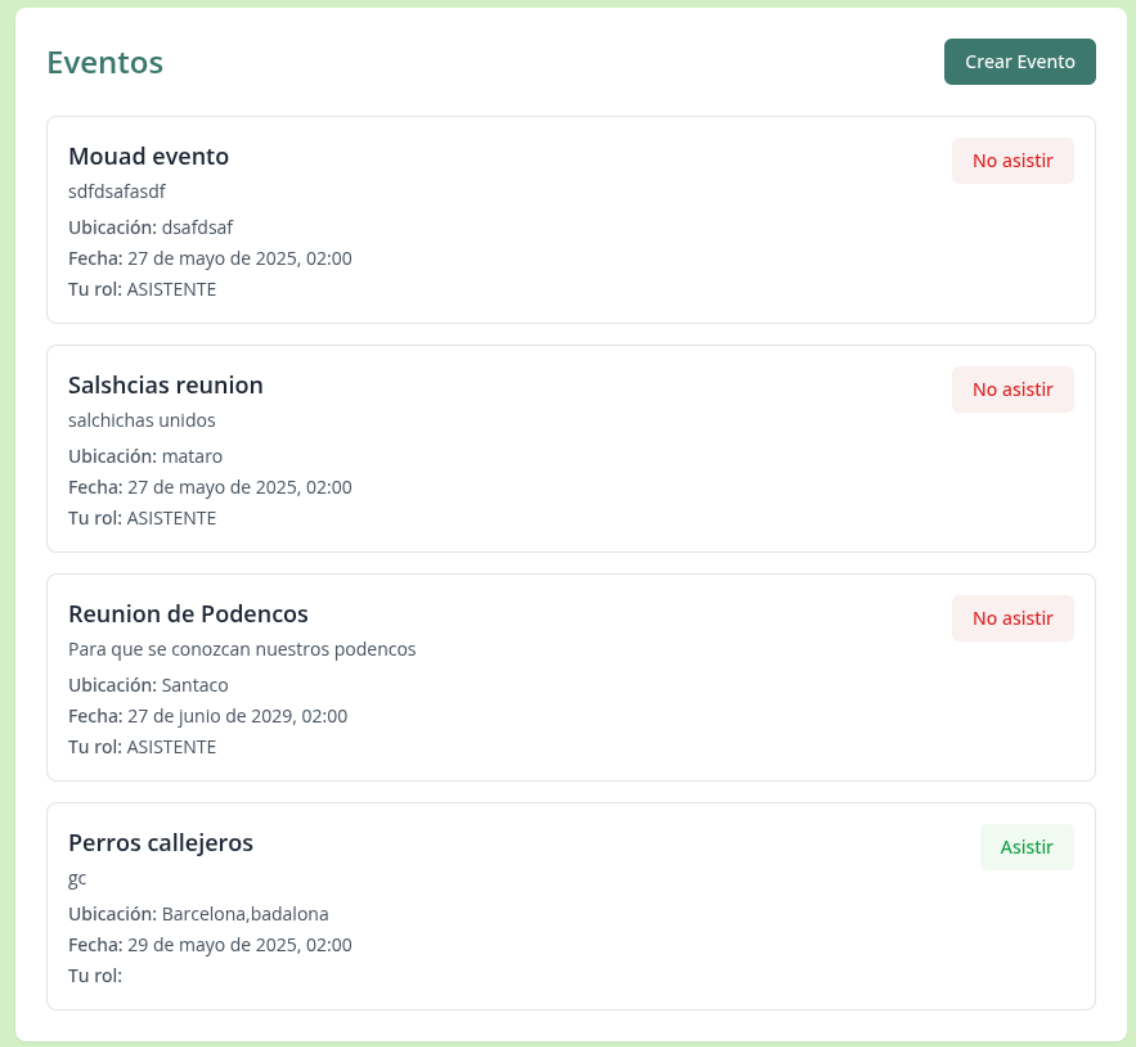
2.4.1.11. Grupos



F17. Componente Grupos.

Este componente realiza una llamada a la API Rest y recibe un json con todos los grupos guardados por el usuario. Cuenta con las funciones para unirse, abandonar, crear, editar y eliminar un grupo.

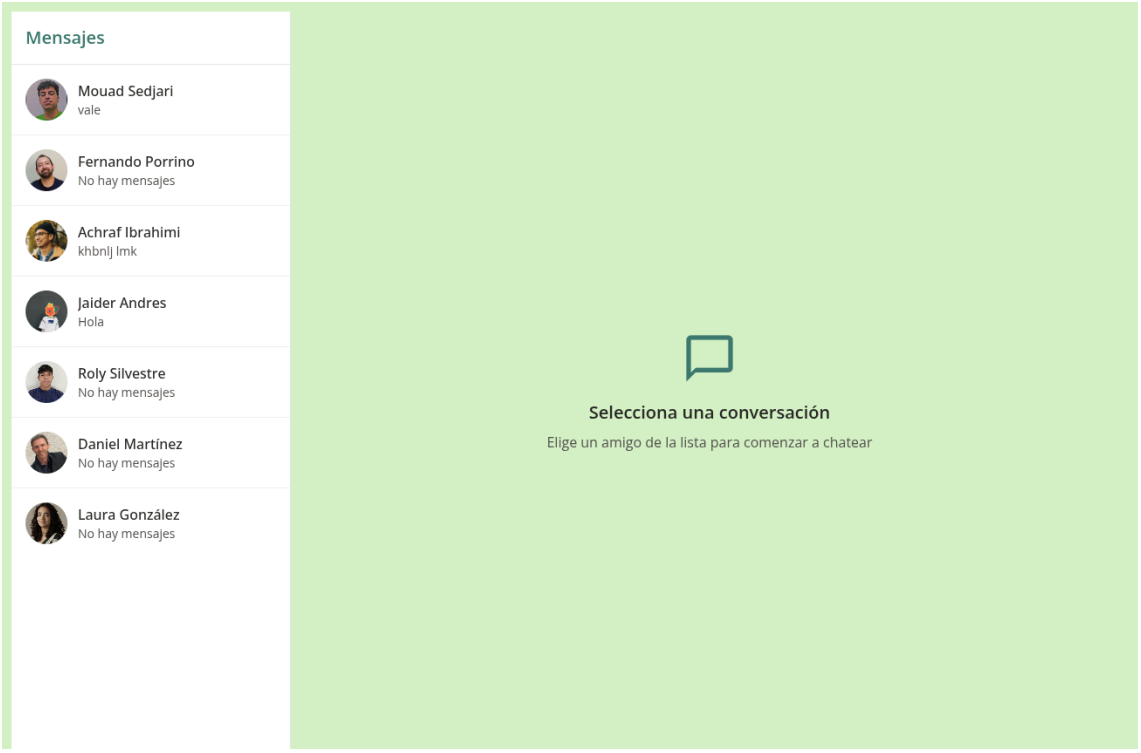
2.4.1.12. Eventos



F18. Componente Eventos.

Este componente realiza una llamada a la API Rest y recibe un json con todos los eventos guardados por el usuario. Cuenta con las funciones para unirse, abandonar, crear, editar y eliminar un evento.

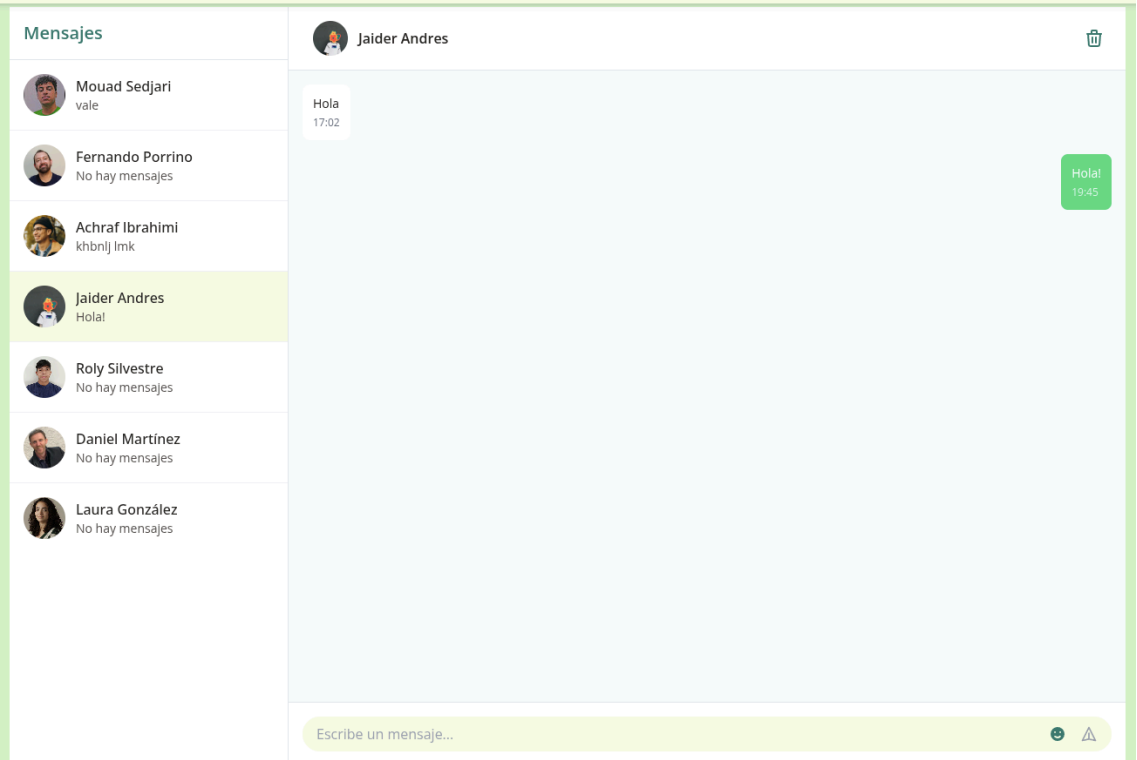
2.4.1.13. Chat



F19. Componente Chat.

Este componente realiza una llamada a la API Rest y recibe un json con todos los amigos del usuario, después realiza otra llamada a la API Rest con el id de cada usuario y recupera un json con la conversación para mostrar el último mensaje. Al realizar clic sobre el usuario mostrará el componente conversación.

2.4.1.14. Conversación



F20. Componente Conversación.

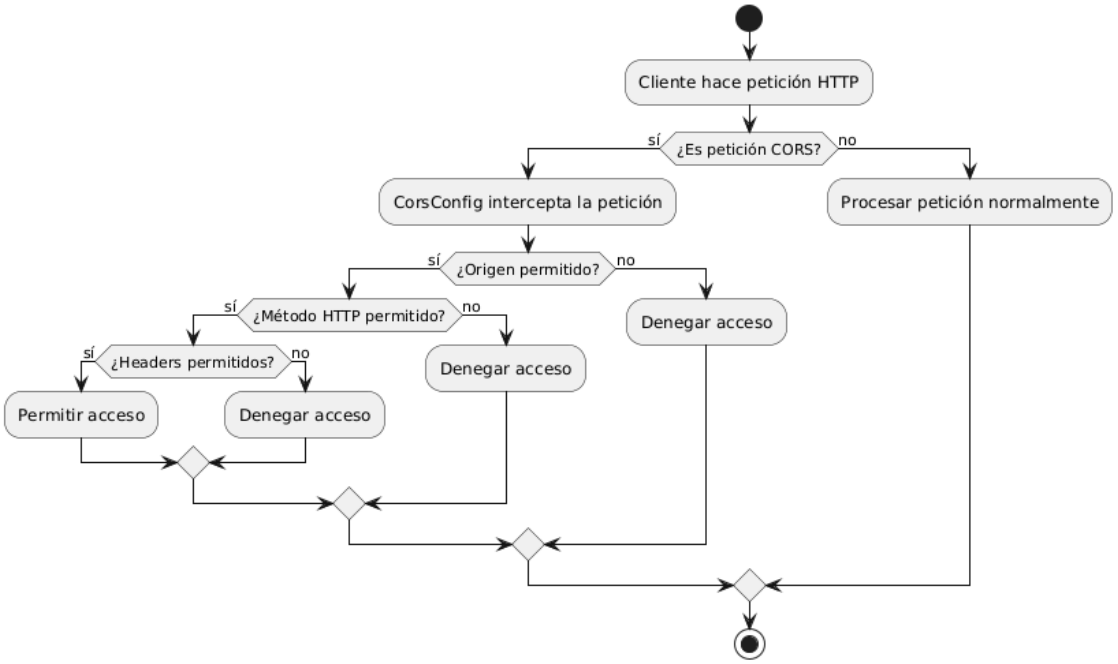
Este componente realiza una llamada a la API Rest y recibe un json con la conversación. Dispone de un formulario que envía un json a la API Rest para escribir mensajes.

2.4.2. Componentes del Backed

2.4.2.1. CorsConfig

El CorsConfig es un componente de seguridad que actúa como un "portero" de nuestra API. Su función principal es controlar quién puede acceder a nuestros recursos y cómo pueden hacerlo. Imaginemos que nuestra API es como un edificio:

- El CorsConfig es el guardia de seguridad
- Las peticiones son los visitantes
- Los orígenes son las diferentes entradas al edificio
- Los métodos HTTP son las diferentes acciones que los visitantes pueden realizar
- Los headers son las credenciales o permisos que los visitantes deben mostrar

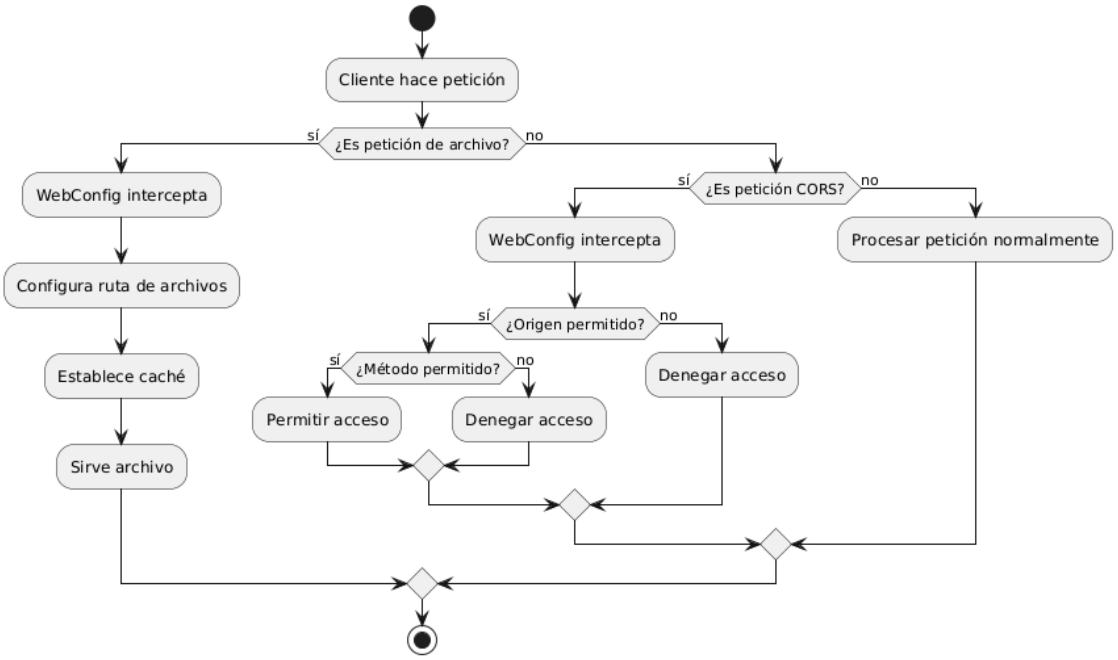


F21. PlantUML - Diagrama conceptual de CorsConfig

2.4.2.2. WebConfig

El WebConfig es un componente de configuración en Spring Boot que actúa como un "gestor de recursos web". Su propósito principal es configurar dos aspectos fundamentales:

- La gestión de archivos subidos (uploads)
- La configuración de CORS para el frontend

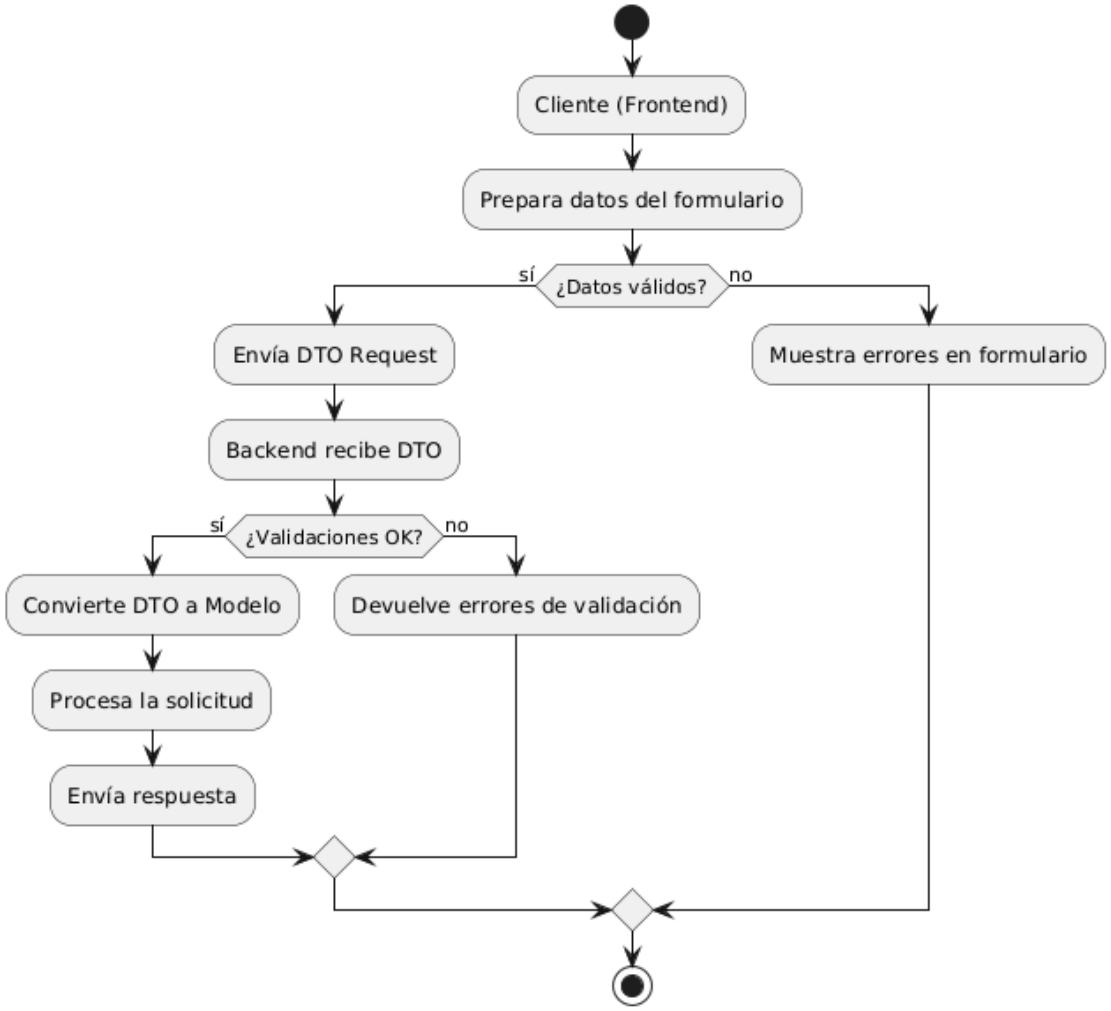


F22. [PlantUML - Diagrama conceptual de WebConfig]

2.4.2.3. DTO Request

Un DTO (Data Transfer Object) Request es como un formulario digital que:

- Recoge datos del cliente (frontend)
- Valida la información antes de procesarla
- Transforma los datos a un formato que la aplicación puede entender
- Actúa como una capa de seguridad entre el cliente y el servidor

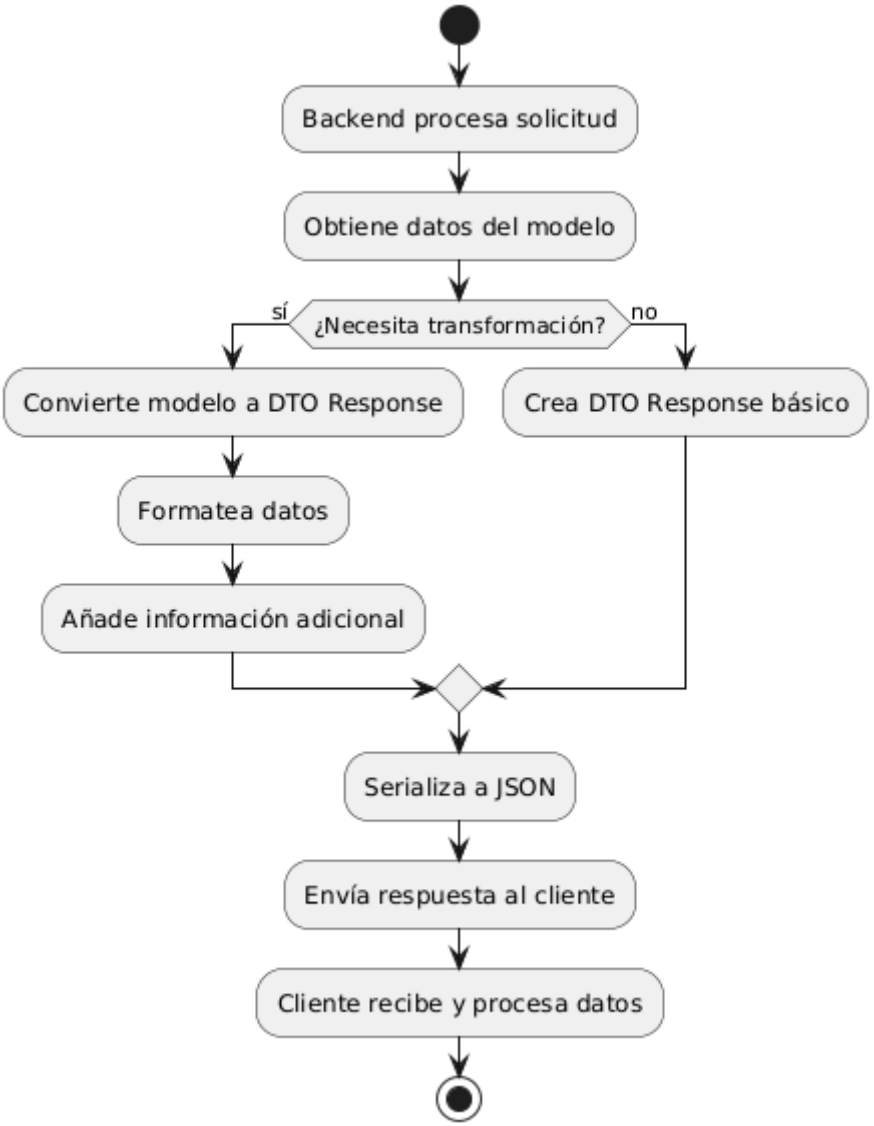


F23. PlantUML - Diagrama conceptual de DTO Request

2.4.2.4. DTO Response

Un DTO Response es como un paquete de datos que:

- Contiene la información que el servidor envía al cliente
- Formatea los datos de manera segura y eficiente
- Incluye solo la información necesaria para el cliente
- Actúa como una capa de presentación de datos

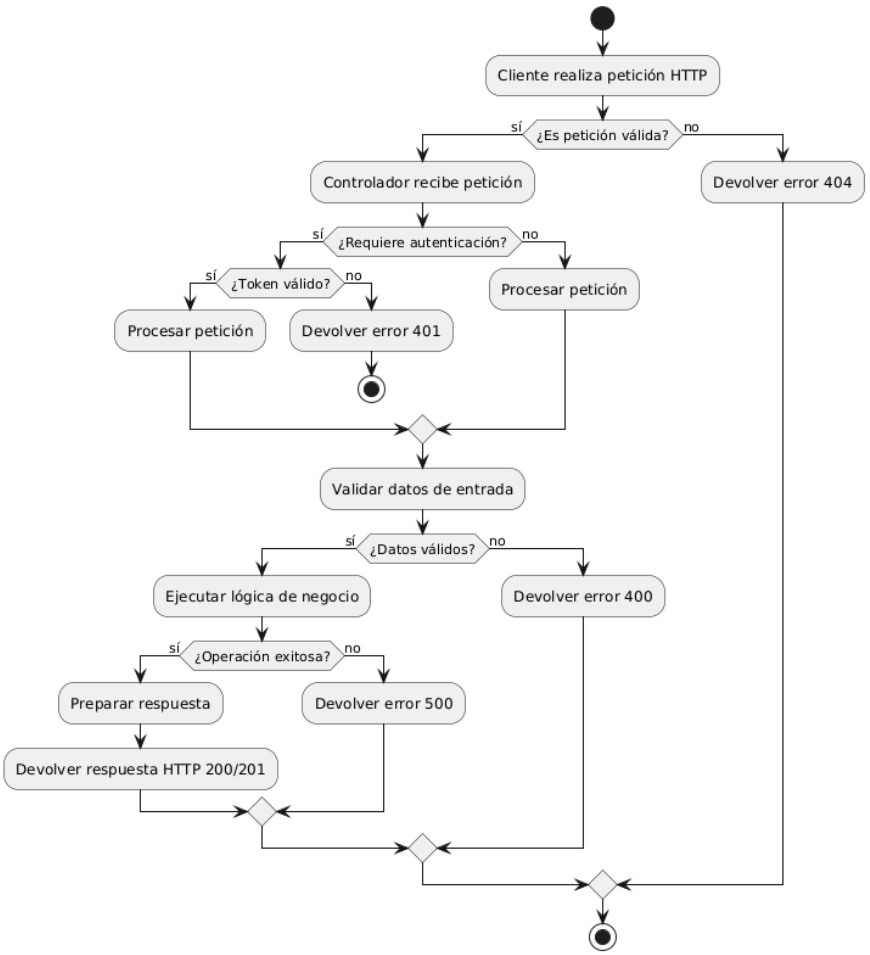


F24. PlantUML - Diagrama conceptual de DTO Response

2.4.2.5. Controller

Un controlador (Controller) en una aplicación Spring Boot es como un "portero inteligente" que:

- **Recibe Peticiones:**
  - Actúa como punto de entrada para las solicitudes HTTP
  - Maneja diferentes tipos de peticiones (GET, POST, PUT, DELETE, etc.)
  - Procesa los datos de entrada (parámetros, cuerpos de petición, etc.)
- **Coordina Operaciones:**
  - Delega la lógica de negocio a los servicios
  - Gestiona las respuestas HTTP
  - Maneja errores y excepciones
- **Estructura Básica:**
  - Anotado con @RestController
  - Define rutas con @RequestMapping
  - Inyecta servicios necesarios con @Autowired

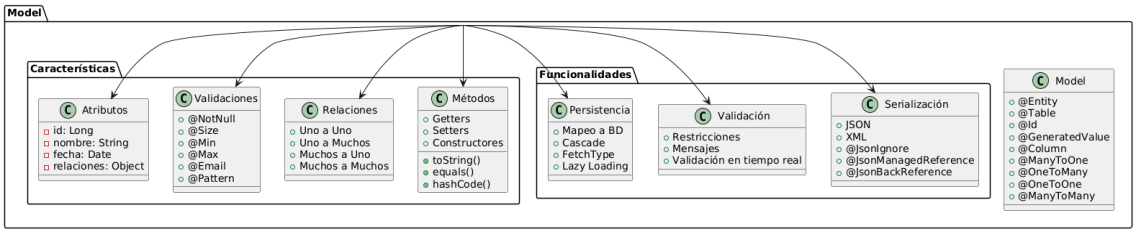


F25. PlantUML - Diagrama conceptual de Controller

2.4.2.6. Models

Un modelo en el contexto de una aplicación Spring Boot es una representación de la estructura de datos que se almacenará en la base de datos. Es una clase Java que:

- Se anota con @Entity para indicar que es una entidad persistente
- Mapea directamente a una tabla en la base de datos
- Define las relaciones entre diferentes entidades
- Contiene los atributos que se convertirán en columnas
- Incluye validaciones y restricciones de datos
- Proporciona getters y setters para acceder a los datos

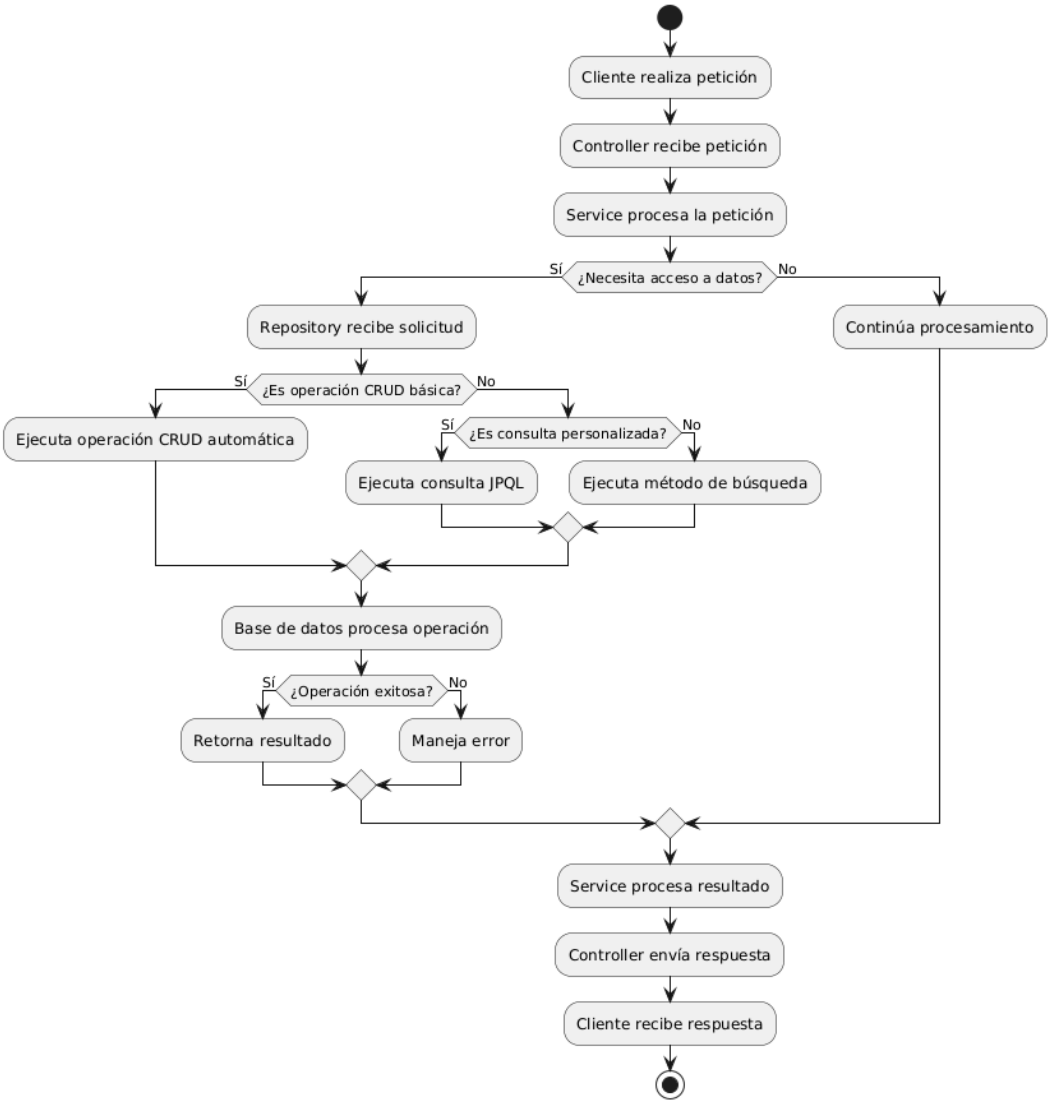


F26. PlantUML - Diagrama conceptual de Model

2.4.2.7. Repositories

Es como el archivador inteligente de una aplicación. Su función principal es gestionar el acceso a los datos: sabe cómo guardar, buscar, actualizar y eliminar información sin necesidad de escribir consultas manuales.

- **Actúa como:**
  - Bibliotecario de datos: conoce dónde está cada dato y cómo manejarlo.
  - Puente entre la base de datos y la lógica de negocio.
- **Características clave:**
  - Se extiende de JpaRepository o CrudRepository.
  - Automatiza operaciones CRUD.
  - Permite consultas personalizadas.
  - Maneja relaciones, paginación y ordenamiento.



F27. [PlantUML - Diagrama conceptual de Repositories]

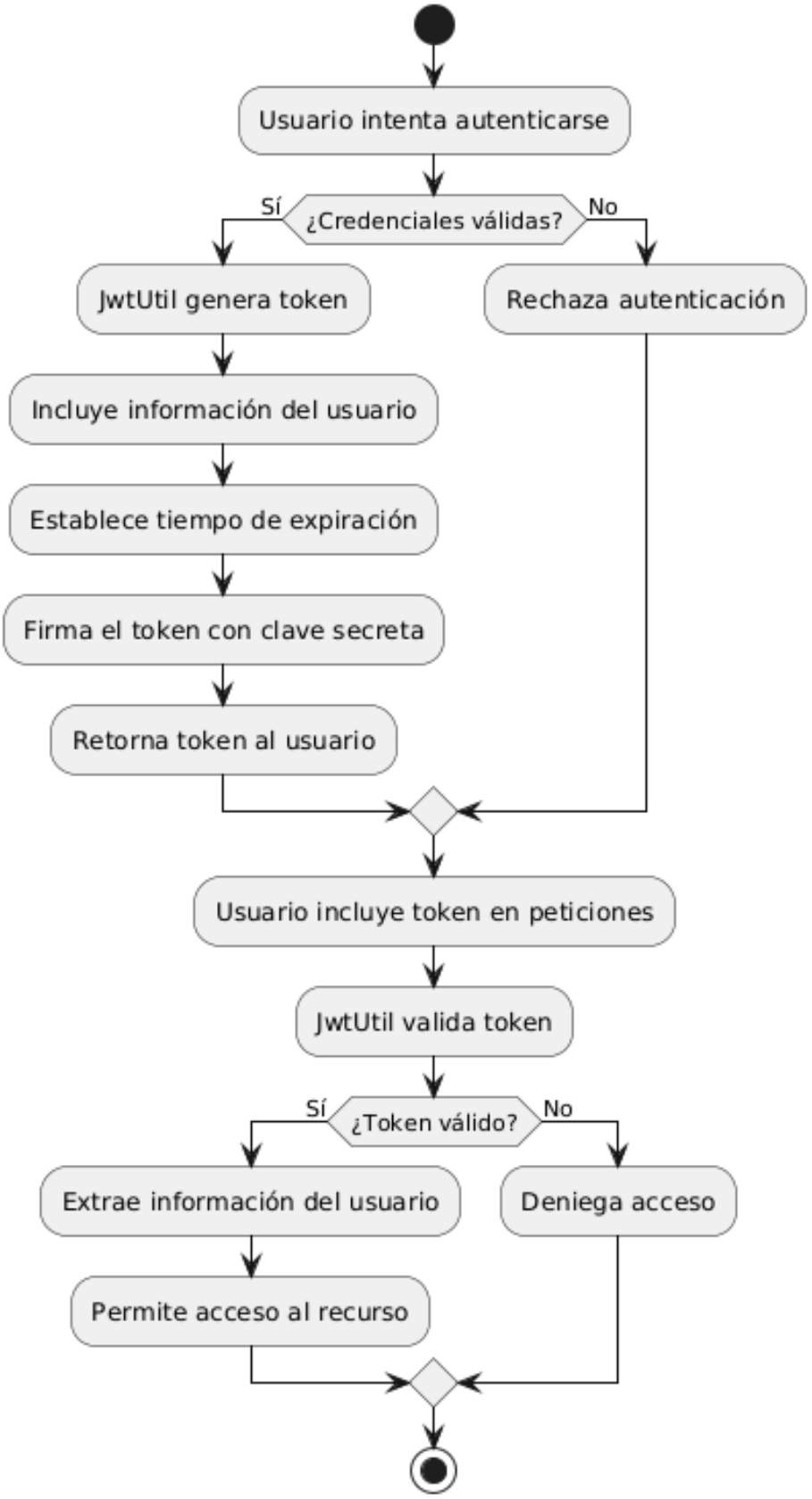
2.4.2.8. JwtUtil

Un JwtUtil es como un guardia de seguridad digital que:

- Genera tokens de autenticación únicos para cada usuario
- Verifica la identidad de los usuarios mediante tokens
- Gestiona la seguridad de las sesiones de usuario
- Proporciona un mecanismo seguro para la autenticación
- Mantiene la integridad de las comunicaciones cliente-servidor

Características principales:

- Utiliza JWT (JSON Web Tokens) para la autenticación
- Mantiene una clave secreta para firmar los tokens
- Establece un tiempo de expiración para los tokens
- Extrae información del usuario del token
- Proporciona métodos para generar y validar tokens



F28. PlantUML - Diagrama conceptual de JwtUtil

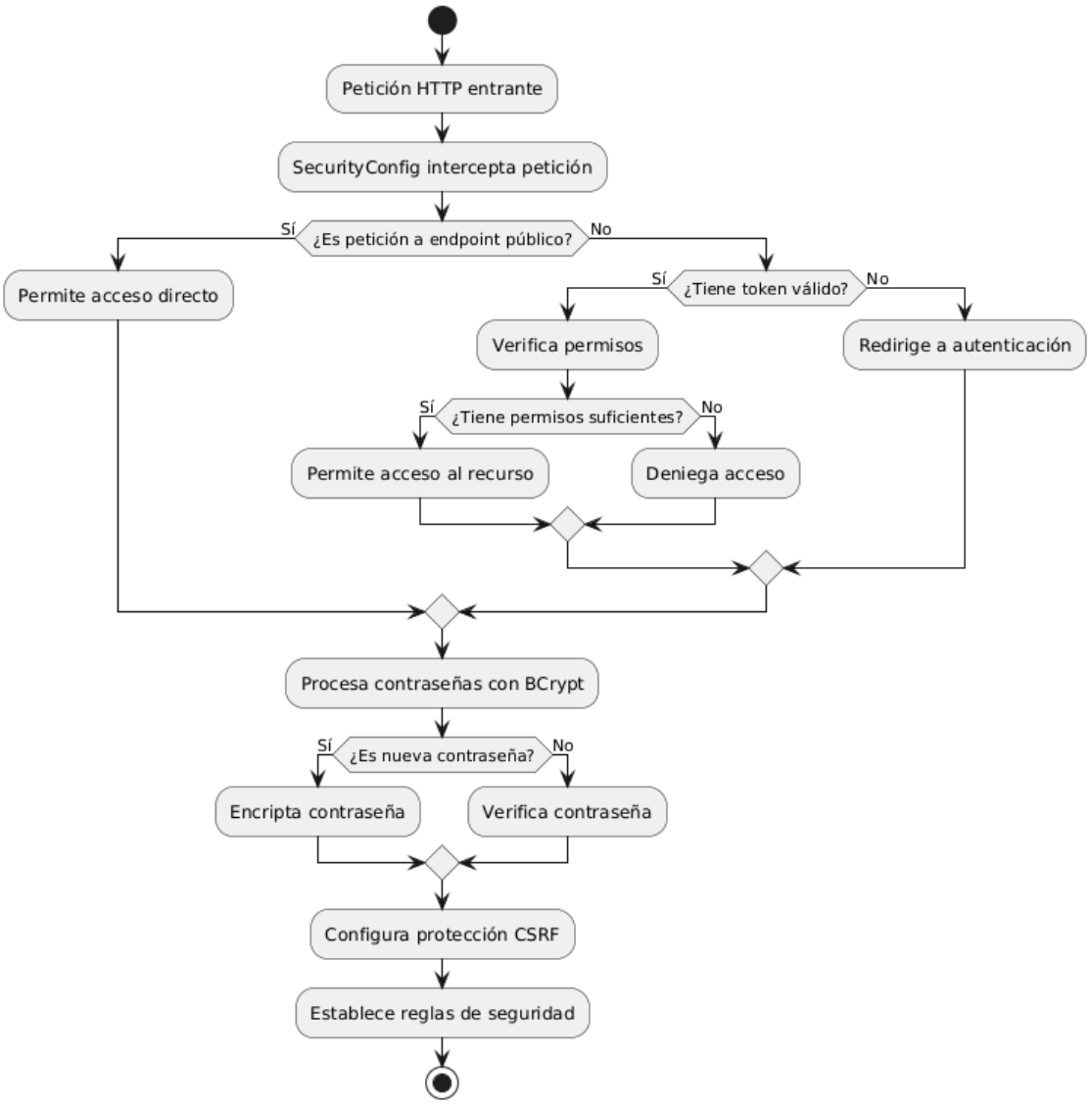
2.4.2.9. SecurityConfig

Un SecurityConfig es como un sistema de seguridad central que:

- Define las reglas de seguridad de la aplicación
- Configura el acceso a los endpoints
- Establece el sistema de encriptación de contraseñas
- Gestiona la autenticación y autorización
- Protege los recursos de la aplicación

Características principales:

- Se anota con @Configuration y @EnableWebSecurity
- Define la cadena de filtros de seguridad
- Configura el codificador de contraseñas
- Establece las reglas de acceso a URLs
- Gestiona la protección CSRF

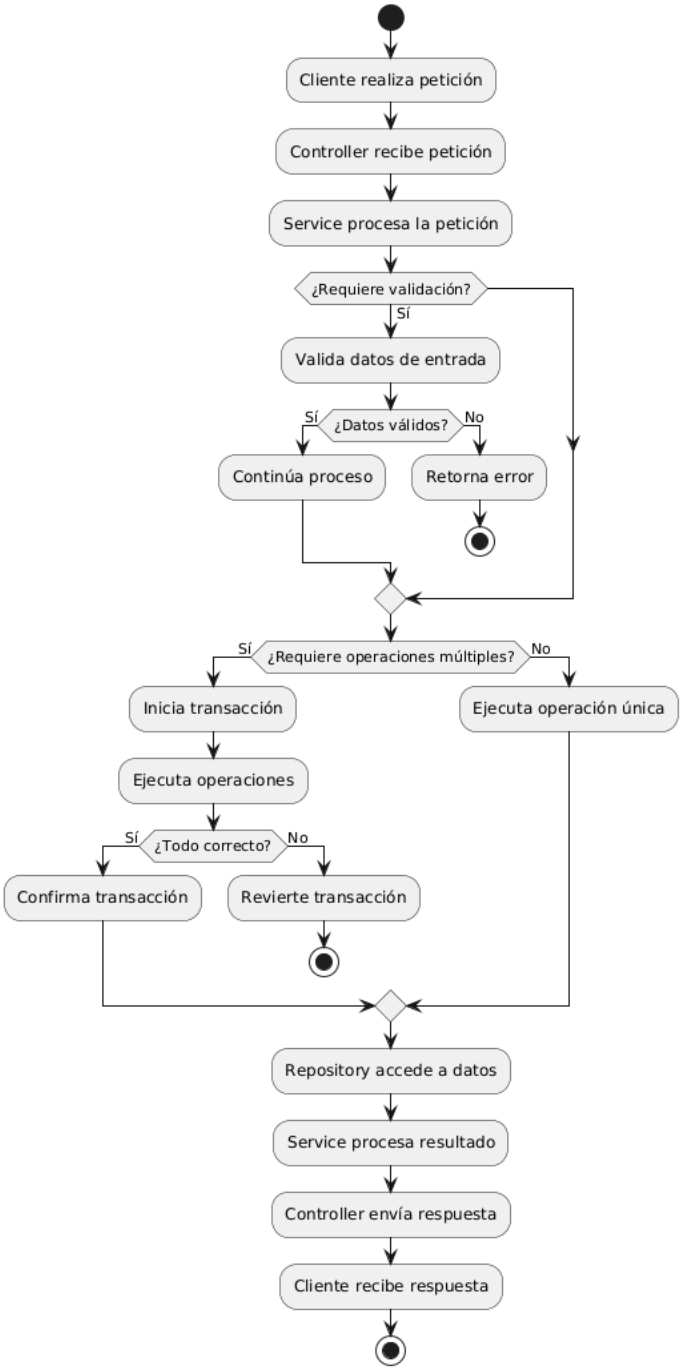


F29. PlantUML - Diagrama conceptual de SecurityConfig

2.4.2.10. Services

Un Service es como un coordinador de operaciones que:

- Implementa la lógica de negocio de la aplicación
- Coordina las operaciones entre controladores y repositorios
- Valida y procesa los datos antes de persistir los datos
- Gestiona las transacciones y reglas de negocio
- Proporciona una capa de abstracción para la lógica de negocio



F30. PlantUML - Diagrama conceptual de Services

2.5. Definición de las funcionalidades

2.5.1. Perfiles personalizados de usuarios y mascotas

La funcionalidad de perfiles personalizados está diseñada para permitir que cada usuario configure y mantenga una representación digital clara y completa tanto de sí mismo como de sus mascotas, facilitando la interacción dentro de la comunidad.

2.5.1.1. Creación inicial del perfil

Al registrarse, el usuario completa un formulario con sus datos personales básicos (como nombre, correo, foto de perfil) que se almacenan en la base de datos. Luego, puede crear uno o varios perfiles de mascotas asociados a su cuenta, ingresando información estructurada para cada mascota: nombre, raza, género, edad, peso y fotos. Esta información se valida en el frontend para asegurar que los datos ingresados tengan el formato correcto (ej. edad numérica, fotos en formato aceptado).

2.5.1.2. Almacenamiento y estructuración de datos

Los datos personales y de las mascotas se almacenan en una base de datos estructurada, en tablas con referencias entre usuario y mascotas. Esto permite una recuperación rápida y eficiente, así como mantener la integridad de la relación entre perfiles y sus dueños.

2.5.1.2. Visualización del perfil

La plataforma ofrece una interfaz donde se muestran estos datos en un formato amigable y visualmente atractivo, similar a una “tarjeta de presentación” o ficha digital. Las fotos se presentan en galerías o miniaturas, y los datos clave (como raza y edad) se destacan para facilitar la identificación rápida. Esta visualización permite que otros usuarios conozcan las características de cada miembro y sus mascotas, promoviendo conexiones y conversaciones relevantes.

2.5.1.3. Edición y actualización continua

Los usuarios pueden acceder a su perfil y modificar la información cuando deseen. Esto se realiza a través de formularios con los datos pre-cargados para facilitar la edición, incluyendo la posibilidad de cambiar fotos o actualizar datos conforme las mascotas crecen o cambian. Las actualizaciones se validan y sincronizan inmediatamente con la base de datos para mantener la información vigente.

2.5.1.4. Relación y navegación entre perfiles

Los perfiles funcionan como nodos dentro de la red social de la plataforma. Otros usuarios pueden buscar, visitar y seguir estos perfiles para estar al tanto de novedades o interacciones. La estructura permite enlazar perfiles de

mascotas con sus dueños, garantizando claridad en la propiedad y facilitando la comunicación dirigida.

Implementación: Completada

2.5.2. Sistema de valoraciones y reseñas

El sistema de valoraciones y reseñas permite a los usuarios expresar su opinión sobre interacciones con otros miembros de la comunidad. Esta funcionalidad cumple un rol esencial para construir un entorno confiable y transparente, donde las experiencias compartidas influyen en las decisiones y fortalecen la credibilidad de los actores involucrados. A continuación, se describe su funcionamiento conceptual.

2.5.2.1. Registro de valoraciones y comentarios

Los usuarios acceden a un formulario donde pueden asignar una calificación numérica (por ejemplo, de 1 a 5 estrellas) y escribir un comentario o reseña textual que complemente su puntuación

2.5.2.2. Almacenamiento estructurado de reseñas

Cada reseña se almacena en una estructura de datos que incluye:

- Identificador del usuario que reseña
- Calificación numérica
- Comentario textual
- Fecha y hora de publicación

2.5.2.3. Integración con perfiles y publicaciones

Las calificaciones se integran directamente en los perfiles de usuarios. Esto permite que otros usuarios consulten las experiencias previas antes de interactuar o contratar servicios, fortaleciendo la toma de decisiones informadas.

Implementación: Completada

2.5.3. Creación y gestión de posts con interacción

Esta funcionalidad permite a los usuarios expresar ideas, compartir contenido e interactuar a través de publicaciones tipo “post”, que pueden incluir texto e imágenes. Está diseñada para fomentar la participación activa dentro de la comunidad, ofreciendo herramientas intuitivas para crear, comentar y mantenerse informado sobre las interacciones relevantes.

2.5.3.1. Creación de publicaciones

Los usuarios autenticados acceden a un editor de publicaciones donde pueden escribir texto libre (experiencias, consejos, noticias, etc.) y subir una imágenes desde su dispositivo.

El sistema realiza validaciones automáticas, longitud máxima del texto y tipos y tamaños permitidos de imágenes

2.5.3.2. Almacenamiento y gestión del contenido

Una vez publicado, el contenido se guarda en la base de datos con los siguientes atributos:

- ID del usuario autor
- Texto de la publicación
- Rutas de las imágenes asociadas (almacenadas en el sistema de archivos)
- Fecha de creación

La información se organiza para permitir una recuperación rápida y eficiente, mostrando las publicaciones en un feed cronológico o filtrado por relevancia, seguidores o temática.

2.5.3.3. Interacción mediante comentarios

Cada post permite que otros usuarios dejen comentarios, que pueden incluir texto (y eventualmente emojis o reacciones).

- Estos comentarios se actualizan en tiempo real utilizando tecnologías como WebSockets o polling frecuente
- Se asocian al post mediante su identificador
- Incluyen nombre del autor, fecha y contenido

Esto promueve una conversación activa y continua alrededor del contenido publicado.

2.5.3.4. Notificaciones y seguimiento de actividad

Para mantener a los usuarios informados de nuevas interacciones:

- El autor de un post recibe una notificación en tiempo real cuando alguien comenta su publicación
- Los usuarios que han comentado previamente también pueden recibir notificaciones si se genera una nueva interacción
- Las notificaciones pueden mostrarse en la interfaz o enviarse mediante alertas push o por correo, según configuración

2.5.3.5. Integración en la red social de la plataforma

Las publicaciones se visualizan en un “feed” o muro de actividad, que se alimenta con:

- Posts propios
- Posts de usuarios seguidos
- Publicaciones destacadas o populares

Implementación: Completada pero con cambios, no se imprenta notificaciones con WebSockets debido a su complejidad y el tiempo requerido para implementarlo.

2.5.4. Chat

El sistema de chat permite a los usuarios comunicarse de forma directa, privada y en tiempo real. Este módulo es fundamental para facilitar el intercambio de información más detallada que no se comparte públicamente, como coordinación de encuentros, consultas sobre mascotas, o detalles de servicios. Está diseñado para ofrecer una experiencia fluida, segura y confidencial.

2.5.4.1. Inicio de conversación

Los usuarios pueden iniciar una conversación directa desde:

- El perfil de otro usuario
- La lista de seguidores o contactos frecuentes

2.5.4.2. Interfaz de mensajería

La interfaz del chat incluye funcionalidades típicas de mensajería instantánea:

- Entrada de texto
- Vista en tiempo real de mensajes enviados y recibidos
- Indicadores de estado (enviado, entregado, leído, escribiendo…)
- Organización por conversaciones recientes o contactos favoritos

Opcionalmente puede incluir soporte para emojis, enlaces o imágenes pequeñas.

2.5.4.3. Transmisión en tiempo real

La comunicación se basa en tecnologías de transmisión en tiempo real como WebSockets o servicios de mensajería en la nube (ej. Firebase, MQTT), lo que permite:

- Entrega instantánea de mensajes
- Actualización en vivo de la conversación sin recargar la página
- Experiencia fluida similar a apps de mensajería populares

2.5.4.4. Seguridad y cifrado

La privacidad es un eje central de esta funcionalidad. Conceptualmente, el sistema implementa:

- Cifrado de extremo a extremo o por canal seguro (TLS/HTTPS) para proteger los datos en tránsito
- Restricción de acceso: solo los dos usuarios involucrados pueden acceder al historial de su conversación
- Protección contra spam: limitación de mensajes por usuario, bloqueo de usuarios ofensivos y mecanismos de reporte

2.5.4.5. Almacenamiento y persistencia

Todos los mensajes se almacenan en una base de datos estructurada por conversaciones, con metadatos como:

- ID de la conversación
- Emisor y receptor
- Contenido del mensaje
- Timestamp (fecha y hora de envío)
- Estado del mensaje (enviado, leído, eliminado)

Esto permite la recuperación eficiente del historial cuando el usuario vuelve a abrir el chat.

2.5.4.6. Notificaciones push

El sistema incluye soporte para notificaciones push en tiempo real, tanto en la plataforma web como en futuras apps móviles, alertando al usuario cuando:

- Recibe un nuevo mensaje
- Alguien inicia una conversación
- Hay actividad reciente en una conversación activa

Las notificaciones se pueden personalizar por el usuario (activar, silenciar, priorizar).

Implementación: Completada pero con cambios, no se imprenta notificaciones con WebSockets, estados, seguridad y cifrado, iniciar el chat desde el perfil de otro usuario, organización, debido a su complejidad y el tiempo requerido para implementarlo.

2.5.5. Sistema de seguidores

El sistema de seguidores permite a los usuarios establecer conexiones unilaterales con otros usuarios y sus mascotas, generando una red social activa e informativa. Esta funcionalidad se inspira en modelos de redes sociales como Instagram o Twitter, donde "seguir" no requiere reciprocidad, pero sí activa un canal de información relevante entre los involucrados. El objetivo es fortalecer el sentido de comunidad y personalizar la experiencia de uso.

2.5.5.1. Seguimiento de usuarios y mascotas

Los usuarios tienen la opción de seguir:

- A otros usuarios (para estar al tanto de sus publicaciones, comentarios o eventos)
- A mascotas (para seguir sus actividades, fotos y novedades relacionadas)

Desde cada perfil público, la interfaz ofrece un botón claro de “Seguir” o “Dejar de seguir”. Este estado se actualiza instantáneamente y se refleja en el perfil de ambos involucrados.

2.5.5.2. Registro de relaciones de seguimiento

Cada acción de seguimiento se registra en una estructura de datos relacional:

- ID del seguidor
- ID del seguido (usuario o mascota)
- Fecha de inicio de seguimiento
- Estado actual (activo/inactivo)

Esto permite generar una red social dirigida, donde las relaciones son rastreables y explotables para otras funcionalidades como notificaciones y feeds personalizados.

2.5.5.3. Personalización del feed

Uno de los principales beneficios de este sistema es la personalización del contenido. El feed del usuario se construye dinámicamente en base a:

- Publicaciones recientes de los perfiles seguidos
- Actividades destacadas (comentarios, nuevos seguidores, eventos)
- Recomendaciones basadas en conexiones mutuas o intereses compartidos

Este enfoque aumenta la relevancia del contenido mostrado y mejora la retención del usuario dentro de la plataforma.

2.5.5.4. Descubrimiento de perfiles

El sistema de seguidores también permite:

- Sugerencias inteligentes (“Podrías conocer a…” o “Mascotas similares a…”), basadas en la actividad y relaciones del usuario
- Rankings de popularidad o actividad, mostrando usuarios o mascotas más seguidos
- Búsquedas filtradas por intereses, raza de mascota, ubicación, etc.

Todo esto facilita el descubrimiento y la ampliación de la red social, promoviendo la interacción orgánica.

2.5.5.5. Notificaciones y visibilidad

Cuando un usuario es seguido por otro, se genera:

- Una notificación (en tiempo real o en el panel de actividad)
- Una actualización visible en su perfil (contador de seguidores)

También se pueden generar notificaciones cuando:

- Un perfil seguido publica contenido nuevo
- Se realizan acciones destacadas (eventos, actualizaciones de perfil, etc.)

Estas notificaciones pueden personalizarse según preferencias del usuario.

Implementación: Completada pero con cambios, no se imprenta notificaciones con WebSockets, seguir a mascotas, personalización del feed, visibilidad y descubrimiento de perfiles se simula, debido a su complejidad y el tiempo requerido para implementarlo.

2.5.6. Gestión de eventos y grupos

El módulo de eventos y grupos está diseñado para dinamizar la comunidad, ofreciendo espacios organizados donde los usuarios pueden conectarse en torno a actividades comunes o intereses compartidos. Este sistema permite la planificación, promoción y participación activa en eventos, así como la creación de grupos temáticos o geográficos, fomentando la interacción en la vida real o en comunidades virtuales más enfocadas.

2.5.6.1. Creación de eventos

Los usuarios autorizados (usuarios estándar, moderadores o perfiles con permisos especiales) pueden crear eventos a través de un formulario que incluye:

- Título y descripción
- Fecha y hora
- Ubicación física o virtual (incluye soporte para geolocalización)
- Imágenes asociadas (como banners o afiches)
- Límite de participantes (opcional)
- Visibilidad (público, privado o solo para miembros de un grupo)

Cada evento se registra en el sistema con un identificador único y se muestra en el calendario general y/o en feeds filtrados por interés o ubicación.

2.5.6.2. Gestión de inscripciones y participación

Los usuarios pueden:

- Unirse a eventos con un solo clic (botón “Asistir”)
- Recibir confirmación e instrucciones
- Cancelar su inscripción si cambia de opinión
- Ver la lista de participantes (según permisos configurados)

El sistema mantiene un registro de inscripciones por evento, incluyendo:

- ID del evento
- ID del usuario
- Estado (inscrito, cancelado, asistió)
- Timestamp de registro

Este registro permite calcular estadísticas, controlar cupos y enviar comunicaciones personalizadas.

2.5.6.3. Calendario y recordatorios

Los eventos se integran en un calendario accesible desde el perfil del usuario y desde el módulo general de eventos. El sistema genera:

- Vistas por día, semana o mes.
- Recordatorios automáticos (push, correo, notificación interna) antes del evento
- Alertas en caso de cambios de fecha, lugar o cancelación

El objetivo es asegurar la asistencia activa y evitar el olvido de eventos relevantes.

2.5.6.4. Creación y gestión de grupos

Además de eventos puntuales, los usuarios pueden crear grupos de interés, organizados por:

- Temática (adiestramiento, adopción, razas específicas)
- Ubicación (ciudad, barrio, parque)
- Actividad (paseos, rescates, campañas solidarias)

Cada grupo tiene su propia página, con:

- Información general y objetivos
- Lista de miembros
- Espacio para publicaciones internas
- Eventos vinculados exclusivamente al grupo

La administración del grupo incluye roles como administrador, moderador y miembro, con diferentes niveles de permiso.

2.5.6.5. Moderación y permisos

El sistema garantiza un entorno seguro mediante:

- Control de quién puede crear eventos o grupos (según nivel o reputación del usuario)
- Mecanismos de reporte de eventos inadecuados o spam
- Moderación de contenidos publicados dentro de los grupos
- Gestión de roles dentro de grupos (asignación, revocación)

Esto asegura que los espacios creados sean relevantes, seguros y bien gestionados.

2.5.6.6. Promoción y descubrimiento

Los eventos y grupos pueden destacarse en el feed general según:

- Popularidad (cantidad de miembros o asistentes)
- Cercanía geográfica
- Intereses del usuario
- Novedades recientes

Implementación: A medias, por falta de tiempo y complejidad esta función está en su fase beta.

2.5.7. Notificaciones en tiempo real

El sistema de notificaciones en tiempo real tiene como objetivo mantener a los usuarios informados de manera inmediata sobre actividades relevantes dentro de la plataforma, como nuevos mensajes, comentarios, seguidores, inscripciones a eventos o reacciones a publicaciones. Este mecanismo mejora significativamente la reactividad, el compromiso y la experiencia de uso, al proporcionar una sensación de inmediatez y conexión continua.

2.5.7.1. Tipos de notificaciones

Las notificaciones se clasifican en dos grandes categorías:

- Push (externas): Enviadas al dispositivo móvil o navegador del usuario, incluso cuando la app/web no está abierta.
- In-app (internas): Mostradas dentro de la aplicación mientras el usuario la está utilizando (burbujas, íconos, banners).

Ambos tipos de notificación informan sobre:

- Nuevos comentarios en publicaciones propias o seguidas
- Nuevos seguidores
- Mensajes directos recibidos
- Confirmación de inscripción o cambios en eventos
- Publicaciones nuevas de perfiles seguidos
- Recordatorios personalizados

2.5.7.2. Motor de eventos y suscriptores

El sistema se basa en un motor de eventos que reacciona ante acciones del usuario. Por ejemplo:

- Cuando un usuario comenta una publicación, se genera un evento “comentario”.
- El sistema identifica al autor de la publicación como receptor.
- Si el autor tiene las notificaciones activadas, se envía una notificación personalizada.

Para lograr esto, cada usuario actúa como un "suscriptor" a ciertos eventos definidos por su configuración personal o por lógica del sistema.

2.5.7.3. Tecnologías utilizadas

El sistema se apoya en tecnologías de comunicación bidireccional en tiempo real, como:

- WebSockets: canal persistente entre el cliente y el servidor para emitir notificaciones instantáneas.
- Firebase Cloud Messaging (FCM) o servicios similares para push en móviles.
- Service Workers: en navegadores para manejar notificaciones incluso en segundo plano.

Estas tecnologías garantizan que la entrega sea inmediata, eficiente y escalable.

2.5.7.4. Centro de notificaciones

Dentro de la plataforma, cada usuario cuenta con un centro de notificaciones, accesible desde el menú principal, que permite:

- Visualizar el historial de notificaciones (con fecha y estado)
- Marcar como leídas o eliminarlas
- Configurar qué tipos de alertas desea recibir
- Filtrar por categoría (social, eventos, mensajes, etc.)

Esto centraliza la información relevante sin saturar al usuario.

2.5.7.5. Configuración de preferencias

Cada usuario puede personalizar su experiencia mediante un panel de preferencias, donde define:

- Qué tipos de notificaciones desea recibir
- En qué formato (push, in-app, correo electrónico)
- Horarios en los que desea silenciar notificaciones (modo descanso)

Estas opciones se almacenan y respetan para garantizar una experiencia no invasiva pero informativa.

2.5.7.6. Escenarios críticos cubiertos

El sistema está optimizado para responder ante:

- Eventos sensibles con tiempo limitado (inicio de eventos, cambios de lugar)
- Mensajes privados no leídos por más de X minutos
- Alta interacción en una publicación (para el autor)
- Cambios de estado en procesos como solicitudes de grupo o confirmaciones de pago

Esto asegura que la notificación no solo sea inmediata, sino también significativa.

Implementación: No implementada debido a su complejidad y el tiempo requerido para implementarlo.

2.5.8. Geolocalización

El sistema de geolocalización permite contextualizar la experiencia del usuario en función de su ubicación física, integrando esta información para ofrecer contenido, conexiones y servicios personalizados según su cercanía. Esta funcionalidad hace uso de APIs de mapas y servicios de localización en tiempo real para enriquecer la interacción social y funcional dentro de la plataforma.

2.5.8.1. Obtención de la ubicación del usuario

La ubicación puede ser determinada de dos maneras principales:

- Automáticamente, mediante permisos de geolocalización en navegadores o dispositivos móviles (HTML5 Geolocation API, GPS).
- Manual, cuando el usuario define su ubicación principal o zona de interés desde su perfil.

La ubicación se representa como coordenadas geográficas (latitud y longitud), que son almacenadas de forma segura y se actualizan periódicamente según la actividad del usuario.

2.5.8.2. Integración con mapas

Se incorporan servicios como Google Maps, Mapbox o Leaflet para:

- Mostrar la ubicación de usuarios, mascotas perdidas o servicios en un mapa interactivo.
- Indicar puntos de encuentro o eventos con direcciones precisas.
- Navegar entre resultados geográficos con facilidad (zoom, filtros, marcadores).

Estos mapas permiten visualizar información espacial y navegar desde una perspectiva geográfica intuitiva.

2.5.8.3. Filtros geográficos en búsquedas

Cada vez que el usuario realiza una búsqueda (de usuarios, eventos, servicios o publicaciones), el sistema puede:

- Ordenar los resultados por proximidad.
- Mostrar solo resultados dentro de un radio definido (por ejemplo, 10 km a la redonda).
- Permitir la selección de una zona en el mapa para obtener resultados geográficos personalizados.

Este filtrado mejora la relevancia del contenido presentado, adaptándolo al entorno local.

2.5.8.4. Promoción de eventos y servicios locales

Los eventos y servicios se etiquetan geográficamente al momento de su creación, lo que permite:

- Mostrar eventos cercanos directamente en el feed del usuario.
- Enviar notificaciones basadas en proximidad (“Evento en tu zona mañana a las 18:00”).
- Recomendar servicios como veterinarias, paseadores o tiendas según cercanía.

Esto refuerza el sentido de comunidad local y facilita la vida cotidiana del usuario con su mascota.

2.5.8.5. Conexión entre usuarios cercanos

Los usuarios pueden explorar y descubrir otros perfiles en su área a través de:

- Un mapa de comunidad con marcadores por usuario o mascota.
- Listas ordenadas por distancia.
- Sugerencias automáticas en función de coincidencias geográficas.

Esto permite establecer conexiones significativas y viables en el mundo real, como organizar paseos o encuentros entre mascotas.

2.5.8.6. Privacidad y control

La ubicación del usuario nunca se comparte directamente sin su consentimiento. El sistema ofrece:

- Opciones para ocultar la ubicación exacta y solo mostrar ciudad o zona aproximada.
- Control sobre quién puede ver o utilizar esta información (solo amigos, grupos, nadie).
- Cifrado y anonimización en el almacenamiento de coordenadas.

Esto asegura una experiencia segura, controlada y respetuosa con la privacidad del usuario.

Implementación: No implementada debido a su complejidad y el tiempo requerido para implementarlo.

2.5.9. Apartado de recursos y consejos

El apartado de recursos y consejos funciona como un espacio editorial dentro de la plataforma, diseñado para informar, educar y apoyar a los usuarios en el cuidado de sus mascotas. Se estructura como un blog categorizado, donde se publican artículos especializados, guías prácticas y recomendaciones de interés general o específico, fomentando una comunidad más informada y responsable.

2.5.9.1. Estructura del contenido editorial

Cada entrada o artículo se compone de:

- Título atractivo y descriptivo
- Texto enriquecido con formato (negritas, listas, enlaces, etc.)
- Imágenes ilustrativas o infografías
- Categorías temáticas (alimentación, salud, comportamiento, adopción, legislación, etc.)
- Autor y fecha de publicación
- Etiquetas (tags) para facilitar la búsqueda

Esto permite una presentación profesional y organizada del contenido.

2.5.9.2. Publicación y administración del contenido

El contenido puede ser generado por:

- Administradores o expertos autorizados (veterinarios, etólogos, entrenadores)
- Usuarios colaboradores, si se habilita un sistema de contribuciones revisadas

Un panel de administración permite:

- Crear, editar y eliminar artículos
- Programar publicaciones
- Moderar comentarios
- Asignar categorías y etiquetas

Este flujo de trabajo asegura calidad y control editorial.

2.5.9.3. Interacción con los usuarios

Los artículos permiten:

- Comentarios abiertos donde los usuarios pueden aportar experiencias, preguntas o sugerencias
- Reacciones (me gusta, útil, etc.) para medir el impacto del contenido
- Compartir en redes sociales o dentro de la misma plataforma

Esto transforma el apartado en un espacio dinámico y participativo, no solo informativo.

2.5.9.4. Acceso y navegación

El contenido es fácilmente accesible desde una sección destacada del menú principal y permite:

- Filtrar por categoría
- Buscar por palabras clave
- Ver los artículos más recientes o más valorados

Una navegación intuitiva y responsiva garantiza que los usuarios puedan acceder al contenido desde cualquier dispositivo, sin barreras.

2.5.9.5. Rol educativo y comunitario

Más allá de informar, esta sección busca:

- Educar al usuario en temas clave del bienestar animal
- Promover prácticas responsables como la adopción, vacunación y esterilización
- Conectar a la comunidad a través del intercambio de conocimientos y vivencias

También se pueden incluir recursos descargables, enlaces a organismos oficiales o calendarios con campañas de concienciación.

2.5.9.6. Integración con otras funciones

El contenido editorial puede integrarse con:

- Notificaciones para alertar sobre nuevos artículos relevantes
- Perfiles de usuarios que marquen artículos como favoritos o recomendados
- Eventos que profundicen sobre temas del blog (charlas, webinars, etc.)

Esta integración amplifica el alcance y la utilidad del contenido.

Implementación: No implementada debido a su complejidad y el tiempo requerido para implementarlo.

2.5.10. Chatbot de soporte

El chatbot de soporte se implementa como un asistente virtual inteligente, accesible desde cualquier sección de la plataforma, diseñado para brindar ayuda inmediata y autónoma a los usuarios. Su propósito es facilitar la navegación, resolver dudas frecuentes y mejorar la experiencia general, reduciendo la necesidad de asistencia humana directa en situaciones comunes.

2.5.10.1. Accesibilidad y visibilidad

El chatbot está siempre disponible mediante:

- Un botón flotante en la esquina inferior de la pantalla (web o móvil).
- Un atajo desde el menú de ayuda o perfil del usuario.

Esto garantiza que el usuario pueda acceder a soporte en cualquier momento y desde cualquier lugar de la plataforma.

2.5.10.2. Base de conocimiento

El chatbot opera sobre una base de datos dinámica de preguntas frecuentes y flujos guiados, que abordan temas como:

- Registro y configuración del perfil.
- Creación de publicaciones, eventos o grupos.
- Gestión de mascotas.
- Uso de mensajería, valoraciones o notificaciones.
- Privacidad, configuración de cuenta y seguridad.

Estos temas están organizados en bloques temáticos, permitiendo respuestas rápidas y contextualizadas.

2.5.10.3. Interacción conversacional

La interfaz del chatbot está diseñada para ofrecer una experiencia natural y fluida, basada en:

- Reconocimiento de palabras clave y lenguaje natural (NLP).
- Opciones interactivas (botones de respuesta rápida, enlaces a secciones, formularios embebidos).
- Mensajes guiados paso a paso para resolver tareas frecuentes.

Esto reduce la frustración del usuario y mejora la eficiencia del soporte.

2.5.10.4. Escalamiento a soporte humano

Cuando el chatbot detecta una consulta compleja, técnica o no contemplada, activa un proceso de escalamiento:

- Solicita detalles adicionales al usuario.
- Crea un ticket automático con la información recopilada.
- Redirige el caso a un agente humano (por chat, correo o sistema interno).

Este mecanismo garantiza que ningún usuario se quede sin respuesta, manteniendo la continuidad en la atención.

2.5.10.5. Personalización y contexto

El chatbot es capaz de:

- Reconocer si el usuario está autenticado.
- Consultar datos básicos de su perfil o actividad reciente.
- Adaptar sus respuestas al contexto (por ejemplo, mostrar ayuda sobre mascotas si el usuario está editando una ficha de mascota).

Esto permite una asistencia proactiva y personalizada.

2.5.10.6. Mejora continua y analítica

El sistema registra cada interacción para:

- Analizar patrones de uso y necesidades frecuentes.
- Actualizar o expandir la base de conocimiento.
- Medir indicadores de rendimiento (satisfacción, resolución de consultas, tasa de escalamiento).

Estos datos permiten mejorar continuamente el servicio, alineándose con las expectativas reales de los usuarios.

Implementación: No implementada debido a su complejidad y el tiempo requerido para implementarlo.

2.5.11. Pagos seguros integrados

El módulo de pagos seguros integrados permite a los usuarios realizar transacciones directamente en la plataforma, facilitando la contratación de servicios (como cuidado de mascotas, paseos, consultas) y la compra de productos (alimentos, accesorios, etc.) en un entorno protegido y confiable. Esta funcionalidad es clave para cerrar el ciclo de interacción, generando valor económico dentro del ecosistema.

2.5.11.1. Integración con pasarelas de pago

El sistema utiliza pasarelas de pago confiables (como Stripe, PayPal, MercadoPago u otras según la región) que:

- Están certificadas bajo los estándares de seguridad PCI DSS.
- Aceptan múltiples métodos de pago (tarjetas de crédito/débito, billeteras digitales, transferencias).
- Garantizan una experiencia fluida y sin redireccionamientos innecesarios.

La integración se realiza mediante APIs seguras y auditadas, lo que asegura robustez y escalabilidad.

2.5.11.2. Flujo de pago en la plataforma

El proceso de pago está diseñado para ser intuitivo y transparente:

1. El usuario selecciona un producto o servicio desde el catálogo o una publicación.
2. Confirma los detalles (descripción, precio, condiciones).
3. Accede a una pantalla de checkout integrada.
4. Introduce los datos de pago o selecciona un método previamente guardado (si aplica).
5. Recibe una confirmación inmediata y comprobante de la transacción.

Todo el proceso se lleva a cabo dentro de la plataforma, sin necesidad de abandonar el entorno seguro del sitio.

2.5.11.3. Seguridad y protección de datos

Para asegurar la protección financiera del usuario, el sistema:

- No almacena directamente datos sensibles como números de tarjetas.
- Utiliza tokens y cifrado extremo a extremo para las comunicaciones con las pasarelas.
- Implementa mecanismos de validación antifraude y detección de transacciones sospechosas.

El cumplimiento de las normativas locales e internacionales es parte del diseño.

2.5.11.4. Gestión de transacciones y comprobantes

Cada pago realizado queda registrado en una base de datos transaccional, permitiendo:

- Consultar el historial de compras y pagos desde el perfil del usuario.
- Descargar facturas o comprobantes.
- Ver el estado de los servicios contratados (pendiente, confirmado, finalizado).
- Notificaciones automáticas ante cambios o actualizaciones en la transacción.

Esto ofrece transparencia y trazabilidad completa.

2.5.11.5. Interacción con otros módulos

El sistema de pagos está estrechamente vinculado a:

- El módulo de publicaciones y servicios, para habilitar ofertas comerciales.
- La mensajería, permitiendo coordinar entregas o condiciones.
- Las valoraciones, para que el usuario pueda calificar el servicio tras la contratación.
- Notificaciones, para informar al usuario sobre pagos realizados o pendientes.

Esta integración facilita una experiencia coherente y centralizada.

2.5.11.6. Administración y soporte

Desde el panel de administración se pueden:

- Gestionar reclamaciones y devoluciones.
- Monitorear estadísticas de uso del sistema de pagos.
- Configurar promociones, tarifas o comisiones.
- Consultar métricas clave como ingresos, volumen de transacciones o servicios más contratados.

También se pueden habilitar mecanismos de arbitraje en caso de conflictos entre usuarios y proveedores.

Implementación: No implementada debido a su complejidad y el tiempo requerido para implementarlo.

2.5.12. Interfaz web moderna y responsiva

La plataforma cuenta con una interfaz de usuario diseñada para ofrecer una experiencia óptima, intuitiva y adaptada a cualquier dispositivo, asegurando que los usuarios puedan acceder y utilizar todas las funcionalidades de forma cómoda, rápida y efectiva, sin importar el tamaño o tipo de pantalla.

2.5.12.1. Diseño adaptable y responsivo

La interfaz utiliza un enfoque de diseño responsive, que:

- Ajusta automáticamente la disposición, tamaño y visibilidad de los elementos según la resolución y orientación del dispositivo (móvil, tablet, desktop).
- Emplea media queries CSS y componentes flexibles para asegurar legibilidad y usabilidad en cualquier contexto.
- Prioriza la accesibilidad, con fuentes legibles, contraste adecuado, y soporte para navegación mediante teclado y lectores de pantalla.

Esto garantiza una experiencia homogénea y accesible para todos los usuarios.

2.5.12.2. Uso de frameworks modernos

La construcción de la interfaz se apoya en frameworks y librerías actuales (como React, Vue, Angular o Ionic), que facilitan:

- Componentización modular para mantener el código organizado y reutilizable.
- Renderizado eficiente para minimizar tiempos de carga y mejorar la respuesta.
- Integración sencilla con APIs y servicios backend.
- Actualizaciones reactivas que reflejan cambios en tiempo real (por ejemplo, en notificaciones o chats).

Esto asegura rendimiento, escalabilidad y facilidad de mantenimiento.

2.5.12.3. Navegación intuitiva

La plataforma ofrece una estructura de navegación clara y sencilla, con:

- Menús accesibles y visibles en todo momento.
- Rutas bien definidas y consistentes.
- Elementos interactivos evidentes (botones, enlaces, formularios).
- Feedback visual inmediato ante acciones del usuario (animaciones, mensajes, estados de carga).

Estas características minimizan la curva de aprendizaje y aumentan la confianza al usar la plataforma.

2.5.12.4. Optimización de rendimiento

Se aplican técnicas para:

- Reducir el peso de recursos estáticos (imágenes, scripts, estilos).
- Carga progresiva y diferida de componentes para mejorar el tiempo hasta la interacción.
- Uso de caché inteligente para acelerar visitas repetidas.
- Monitoreo constante de métricas de rendimiento (First Contentful Paint, Time to Interactive).

Así se garantiza una experiencia rápida y fluida, incluso en conexiones lentas o dispositivos con recursos limitados.

2.5.12.5. Elementos visuales claros y consistentes

El diseño visual sigue:

- Guías de estilo coherentes en colores, tipografías, iconografía y espaciamiento.
- Uso de patrones de diseño que facilitan el reconocimiento y uso de funciones.
- Diseño accesible para personas con discapacidad visual o motora.

Esto contribuye a una experiencia agradable y profesional que invita a la interacción continua.

2.5.12.6. Compatibilidad y pruebas

Se realizan pruebas continuas en distintos navegadores y dispositivos para asegurar:

- Compatibilidad multiplataforma (Chrome, Firefox, Safari, Edge, iOS, Android).
- Corrección de errores visuales y funcionales.
- Adaptación a futuras tecnologías y tendencias en UX/UI.

El mantenimiento y mejora continua garantizan la evolución constante de la experiencia de usuario.

Implementación: Se ha implementado de la mejor manera posible debido a los conocimientos y capacidades de las que disponemos.

2.5.13. API REST documentada y robusta

La plataforma expone toda su funcionalidad mediante una API RESTful diseñada para ser completa, segura, y fácil de integrar, posibilitando la interoperabilidad con aplicaciones móviles, sistemas externos y futuros desarrollos.

2.5.13.1. Diseño RESTful y estandarizado

La API sigue principios REST para garantizar:

- Uso claro y consistente de métodos HTTP (GET, POST, PUT, DELETE, PATCH).
- Rutas intuitivas que reflejan los recursos gestionados (usuarios, mascotas, publicaciones, eventos, pagos, etc.).
- Respuestas en formato JSON estructurado y legible, facilitando el consumo por cualquier cliente.

Esto hace que la API sea predecible y fácil de usar.

2.5.13.2. Documentación completa y accesible

Se provee documentación actualizada y detallada, que incluye:

- Descripción de cada endpoint con sus parámetros, métodos soportados y ejemplos de solicitud/respuesta.
- Guías para autenticación y autorización (tokens JWT, OAuth u otros).
- Explicación de códigos de estado HTTP y mensajes de error.
- Herramientas interactivas tipo Swagger/OpenAPI para probar la API directamente desde el navegador.

La documentación clara mejora la adopción y reduce errores de integración.

2.5.13.3. Seguridad y control de acceso

La API incorpora mecanismos robustos para:

- Autenticación segura de usuarios y aplicaciones cliente.
- Control de permisos para proteger recursos sensibles y operaciones.
- Prevención de ataques comunes (inyección, CSRF, fuerza bruta).
- Gestión de tokens con expiración y renovación segura.

Esto garantiza que solo usuarios y sistemas autorizados puedan acceder a la información y funcionalidades.

2.5.13.4. Manejo detallado de errores

La API ofrece respuestas claras y detalladas ante errores, incluyendo:

- Códigos HTTP específicos (400, 401, 403, 404, 500, etc.).
- Mensajes explicativos que indican la causa del fallo y posibles soluciones.
- Estructura consistente para facilitar el manejo automático en clientes.

Esto facilita la depuración y mejora la experiencia de desarrollo.

2.5.13.5. Escalabilidad y mantenimiento

El diseño modular y desacoplado permite:

- Añadir nuevas funcionalidades sin afectar las existentes.
- Soportar múltiples versiones para compatibilidad retroactiva.
- Monitorizar el rendimiento y uso para optimizar recursos.
- Facilitar la integración con microservicios o arquitecturas distribuidas.

Esto asegura la longevidad y evolución del sistema conforme crecen las necesidades.

Implementación: Se ha implementado, sin versiones y de la mejor manera posible debido a los conocimientos y capacidades de las que disponemos.

3. Licencia

Se ha decidido licenciar este proyecto bajo la licencia Reconocimiento-NoComercial-CompartirIgual 3.0 España de Creative Commons.

Esta licencia permite compartir el trabajo de forma abierta, fomentando la colaboración y el intercambio de conocimientos entre otros estudiantes, docentes y profesionales. Al elegir la opción de NoComercial, se asegura que nadie pueda utilizar el proyecto con fines lucrativos sin el consentimiento del autor, lo cual es especialmente importante en un contexto académico. Se desea que la obra sea accesible para quienes deseen aprender o contribuir, pero sin que pueda ser explotada comercialmente.

Asimismo, esta licencia garantiza que se reconozca a los autores originales de la obra, algo esencial en un entorno donde el reconocimiento del esfuerzo y la originalidad son fundamentales. Este aspecto asegura que el trabajo será utilizado y compartido, pero siempre manteniendo la atribución correcta hacia los autores.

Por último, el componente de Compartirigual obliga a que cualquier modificación o adaptación del trabajo se distribuya bajo la misma licencia. Esto asegura que futuras versiones del proyecto mantendrán el mismo espíritu colaborativo, promoviendo la creación de recursos accesibles para otros estudiantes y profesionales.

En resumen, esta licencia equilibra la protección de los derechos de autor con la promoción del aprendizaje y la colaboración en un entorno académico, facilitando el acceso al trabajo sin permitir su uso con fines comerciales.

4. Plan de desarrollo estructurado

Para garantizar un avance ordenado y eficiente del proyecto, se ha seguido un plan de desarrollo dividido en fases. Cada etapa ha sido diseñada para abordar aspectos clave del proceso, desde el análisis inicial hasta el lanzamiento final, permitiendo una implementación progresiva y controlada de las funcionalidades.

El proyecto se desarrolló en varias fases:

Fase 1: Análisis de mercado.

Fase 2: Definición de funciones.

Fase 3: Diseño visual.

Fase 4: Comprobar compatibilidad de funciones con diseño.

Fase 5: Planteamiento de implantación de Frontend y Backend.

Fase 6: Definición de las rutas para la API.

Fase 7: Implementación de funcionalidades clave (perfiles, mensajería, valoraciones, publicaciones,seguidos, etc) en la API.

Fase 8: Desarrollo del Frontend

Fase 9: Implementación de funcionalidades no planeadas.

Fase 10: Pruebas manuales de los desarrolladores y con usuarios.

Fase 11: Retoques finales.

Fase 12: Lanzamiento.

5. Análisis de mercado

Se ha realizado un estudio de las principales plataformas digitales que conectan a dueños de mascotas con cuidadores. Aunque ninguna funciona como red social, estas aplicaciones actúan como mercados clasificados para encontrar servicios de cuidado de mascotas. A continuación, se describen las características principales y la disponibilidad en español de cada una.

1. **Rover:**
    - **Descripción:** Rover es una plataforma que conecta a dueños de mascotas con cuidadores y paseadores. Permite a los dueños encontrar cuidadores en su área, ver perfiles, leer reseñas y reservar servicios de cuidado de mascotas.
    - **Características:** Perfiles de cuidadores, mensajería, valoraciones y opción de pago a través de la app.
    - **Idioma:** Disponible en castellano.

2. **Wag!:**
    - **Descripción:** Wag! Se centra en el cuidado de perros y ofrece servicios de paseos, cuidado en casa y cuidado nocturno. Los dueños pueden buscar cuidadores en su área y ver calificaciones y reseñas.
    - **Características:** Función de geolocalización, seguimiento de paseos en tiempo real y sistema de notificaciones.
    - **Idioma:** No dispone de castellano.

3. **PetBacker:**
    - **Descripción:** PetBacker conecta a dueños de mascotas con cuidadores, paseadores y cuidadores de mascotas en su área. También incluye funciones para la adopción de mascotas y la búsqueda de veterinarios.
    - **Características:** Perfiles de cuidadores, opciones de reservas y pagos, y sistema de valoraciones.
    - **Idioma:** Disponible en castellano.

4. **Barkly Pets:**
    - **Descripción:** Esta aplicación permitía a los dueños de perros encontrar paseadores y cuidadores locales, enfocándose en cuidado personalizado y comunicación constante.
    - **Características:** Mensajería, historial de paseos y posibilidad de programar servicios regulares.
    - **Descontinuada**

5. **Pawshake:**
    - **Descripción:** Similar a Rover, Pawshake conecta a dueños de mascotas con cuidadores que ofrecen servicios de hospedaje, cuidado y paseos. Es popular en varios países.
    - **Características:** Perfiles de cuidadores, sistema de pago y herramientas para comunicación y coordinación.
    - **Idioma:** No dispone de castellano.

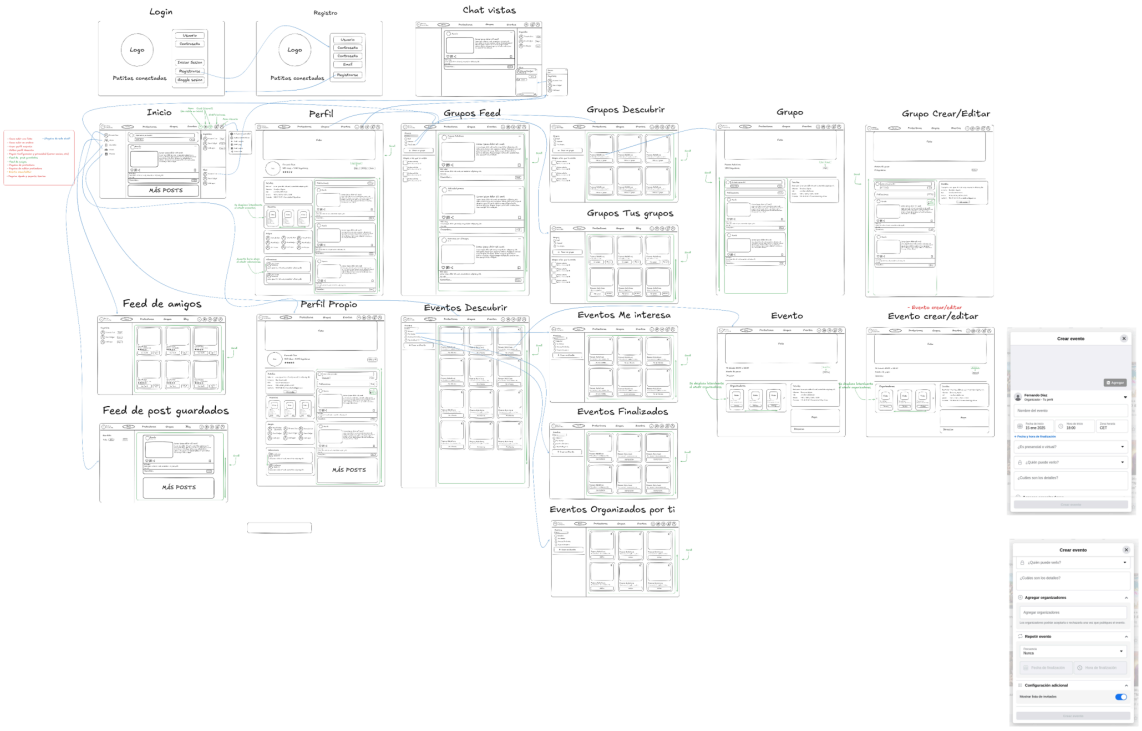
6. **Fetch! Pet Care:**
    - **Descripción:** Fetch! Ofrece servicios de cuidado de mascotas, incluidos paseos, cuidado en casa y alojamiento. Permite a los dueños encontrar cuidadores en su área.
    - **Características:** Sistema de reservas, comunicación directa y enfoque en la seguridad de las mascotas.
    - **Idioma:** No dispone de castellano.

7. **PetSitter:**
    - **Descripción:** Esta plataforma permite a los dueños encontrar cuidadores de mascotas en su área y ofrecer sus servicios. También incluye la opción de crear perfiles detallados.
    - **Características:** Búsqueda por ubicación, valoración de cuidadores y sistema de pago en línea.
    - **Idioma:** Disponible en castellano.

6. Diseño visual y comparación con competencia

Con el fin de visualizar la estructura y funcionalidad de la aplicación, se ha diseñado un wireframe que representa la interfaz. Esta representación esquemática permite anticipar la distribución de los elementos en pantalla y facilita la validación temprana del diseño antes de su implementación.

Para conseguir una navegación intuitiva, moderna y responsiva se ha inspirado en grandes redes sociales como instagram y facebook.



F31. Wireframe del proyecto donde se plantea la estructura del Frontend.

7. Conclusion

7.1. Conclusión general del proyecto

El desarrollo de Patitas Conectadas nos ha aportado un gran valor tanto a nivel académico como profesional. Durante este proceso, hemos tenido la oportunidad de aplicar de forma práctica los conocimientos adquiridos a lo largo del ciclo, utilizando tecnologías actuales como React, Spring Boot y PostgreSQL.

Además, hemos reforzado nuestras habilidades en la gestión de proyectos, lo cual ha sido clave para organizarnos de forma eficiente.

Profesionalmente, este proyecto nos ha permitido trabajar en equipo, enfrentarnos a retos reales del desarrollo web y comprender el ciclo completo de creación de una aplicación funcional desde cero.

7.2. Conclusión de los objetivos

En cuanto a los objetivos definidos al inicio del proyecto, consideramos que en su mayoría han sido alcanzados con éxito:

- Logramos desarrollar perfiles personalizados para usuarios y mascotas, lo que fomenta la interacción en la plataforma.
- Implementamos un sistema de valoraciones y reseñas que permite generar confianza entre los usuarios.
- La funcionalidad de publicaciones con texto e imágenes funciona correctamente, y permite compartir contenido de manera dinámica.
- Conseguimos implementar un sistema de mensajería directa completamente funcional.
- El sistema de seguidores también se encuentra operativo, facilitando la creación de una red activa.
- No se completó del todo:
    - Desarrollamos las bases de herramientas para la gestión de eventos y grupos.
- No se llegó a desarrollar por cuestiones de tiempo y complejidad, pero sienta las bases para futuras mejoras:
    - Las notificaciones.
    - La geolocalización.
    - Un apartado de recursos y consejos.

7.3. Valoración de la metodologia y planificacion

Desde el inicio, decidimos seguir una metodología ágil, concretamente Scrum, utilizando Trello como herramienta de organización. Esta elección nos permitió gestionar el trabajo de manera flexible y adaptarnos a cambios sobre la marcha.

Aunque en algunos momentos fue necesario modificar la planificación inicial debido a imprevistos o nuevas ideas, creemos que la metodología fue muy acertada.

Nos permitió mantener un ritmo constante de trabajo, priorizar tareas según su importancia y mantenernos enfocados en los objetivos principales. En general, la planificación fue respetada y nos ayudó a llegar a una versión funcional del proyecto dentro del plazo previsto.

7.4. Vision de futuro

Somos conscientes de que Patitas Conectadas aún tiene margen de mejora y crecimiento. Nos gustaría continuar con su desarrollo con una mejora más profunda de eventos y grupos. Incorporando funcionalidades que no pudimos finalizar, como el chatbot de soporte completo o la geolocalización.

Además, iniciar el desarrollo de una aplicación móvil que complemente la versión web y haga la plataforma más accesible. También planeamos seguir optimizando la interfaz para ofrecer una mejor experiencia de usuario.

Por último, vemos interesante explorar modelos de monetización sostenibles que puedan hacer viable el mantenimiento y evolución del proyecto a largo plazo.

8. Glosario

Trello, herramienta para organizar las tareas, división de trabajo y llevar un control de lo que se va realizando.

Backend: Parte del desarrollo que se encarga de la lógica del negocio, la gestión de datos y la conexión con la base de datos. No es visible para el usuario final.

Frontend: Parte del desarrollo que se encarga de la interfaz visual con la que interactúa el usuario.

API (Interfaz de Programación de Aplicaciones): Conjunto de definiciones y protocolos que permite la comunicación entre distintas partes del software, como el Frontend y el Backend.

API REST: Tipo de API que utiliza el protocolo HTTP y sigue los principios del estilo arquitectónico REST (Representational State Transfer), permitiendo operaciones como GET, POST, PUT y DELETE. Utiliza comúnmente el formato **JSON** (JavaScript Object Notation) para el intercambio de datos entre cliente y servidor.

API RESTful: API que sigue estrictamente los principios REST, asegurando una estructura coherente y predecible basada en recursos y rutas claras. También utiliza **JSON** como formato estándar para enviar y recibir datos. Representa una implementación más rigurosa y estandarizada del concepto de API REST.

FormData: Objeto nativo de JavaScript que permite construir fácilmente un conjunto de pares clave/valor que representan los campos de un formulario HTML. Se utiliza principalmente para enviar datos mediante peticiones HTTP, especialmente cuando se requiere enviar archivos o datos de tipo multipart/form-data. Facilita el envío de formularios desde el frontend al backend sin necesidad de codificar manualmente los datos.

Spring Boot: Framework de Java para el desarrollo rápido de aplicaciones web. Simplifica la configuración y despliegue de proyectos backend.

React: Librería de JavaScript para construir interfaces de usuario interactivas y dinámicas, especialmente en aplicaciones de una sola página (SPA).

PostgreSQL: Sistema de gestión de bases de datos relacional, potente y de código abierto, utilizado en este proyecto para almacenar y consultar datos.

WebSocket: Protocolo de comunicación que permite establecer una conexión bidireccional entre cliente y servidor en tiempo real.

JWT (JSON Web Token): Método de autenticación que permite transmitir información segura como tokens entre dos partes, usado para gestionar sesiones sin estado (stateless).

JSON (JavaScript Object Notation): Formato ligero de intercambio de datos, fácil de leer y escribir para humanos, y fácil de interpretar para las máquinas.

Scrum: Metodología ágil de gestión de proyectos que promueve la entrega incremental de valor mediante sprints y reuniones regulares de seguimiento.

Trello Board: Panel visual dentro de Trello donde se organizan las tareas en columnas como "Pendiente", "En proceso" y "Hecho".

Responsive Design: Diseño web que se adapta automáticamente al tamaño de la pantalla del dispositivo (móvil, tableta, ordenador), mejorando la experiencia de usuario.

Chatbot: Sistema automatizado de conversación que simula respuestas humanas para asistir a los usuarios, integrado parcialmente en este proyecto.

Geolocalización: Tecnología que permite obtener la ubicación geográfica del usuario para mejorar la personalización del contenido o funcionalidad.

Stripe / PayPal / MercadoPago: Pasarelas de pago seguras utilizadas para procesar pagos electrónicos en línea dentro de plataformas como esta.

9. Bibliografia

9.1. Links

- Cómo instalar PostgreSQL en Ubuntu 24.04 LTS ✔ [2024]
  https://comoinstalar.me/como-instalar-postgresql-en-ubuntu/
- Componentes de Figma
  https://www.figma.com
- Desarrolla tu aplicación Spring Boot con Visual Studio Code
  https://careers.edicomgroup.com/blogtech/desarrolla-tu-aplicacion-spring-boot/

- Documentación de la API web de ASP.NET Core con Swagger/OpenAPI https://learn.microsoft.com/es-es/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-8.0
- Getting Started | Using WebSocket to build an interactive web application https://spring.io/guides/gs/messaging-stomp-websocket
- Guía para definir un proyecto y elaborar una memoria de actividades https://aytosagunto.es/media/yhjpedzd/guia-proyecto-elaborar-memoria-actividades.pdf
- Guía para la elaboración de la Memoria de Prácticas Modalidad A https://www.uam.es/uam/media/doc/1606857975733/guia-modalidad-a
- Microsoft Learn https://learn.microsoft.com/
- REST API Documentation Tool | Swagger UI https://swagger.io/tools/swagger-ui/
- PlantUML – Generador de diagramas UML a partir de texto http://www.plantuml.com
- OpenAI – Documentación oficial de modelos de lenguaje y APIs https://platform.openai.com/docs
- Cómo instalar PostgreSQL en Ubuntu 24.04 LTS ✔️ [2024]

9.2. Videos

- APRENDE ARQUITECTURA HEXAGONAL | Cap. 1 Domain
- Aprende Arquitectura Hexagonal en 10 minutos
- Creación de un API REST usando arquitectura hexagonal con Springboot…
- Cómo Configurar Grillas y Columnas en Figma | Curso Figma 2025 #5
- Desarrolla tu primera API REST desde cero con Spring Boot aplicando bu…
- Login Registration Form Restful Api Using Spring boot React
- Relacion MuchosAMuchos (ManyToMany) con Spring Boot - Spring Data …

9.3. Documentación técnica y APIs

- Spring Boot Guides: https://spring.io/guides
- ReactJS Docs: https://reactjs.org/docs
- PostgreSQL Documentation: https://www.postgresql.org/docs/
- WebSockets con STOMP: https://stomp.github.io/
- Axios (cliente HTTP): https://axios-http.com
- React Toastify: https://fkhadra.github.io/react-toastify/
- Date-fns: https://date-fns.org/

10. Agradecimientos

El desarrollo de esta aplicación y de sus funcionalidades ha sido posible gracias al apoyo de diversas personas que han contribuido en distintas disciplinas, tales como el diseño, el arte y la programación.

Se reconoce a continuación a quienes han colaborado en el avance del proyecto desde diferentes ámbitos.

Yago Morales, brindó orientación sobre cómo iniciarse en Spring Boot, indicó el camino a seguir y estuvo disponible para resolver dudas y problemas.

David Tomas, proporcionó apoyo y orientación en el uso de React, guiando en buenas prácticas y resolviendo dudas para optimizar el desarrollo de la aplicación. Además, colaboró explicando las buenas prácticas para la redacción de esta memoria y aportando su opinión.

Laura González, una gran amiga, colaboró en el diseño, la elección de colores y otros aspectos visuales, aportando sus conocimientos en gráfica impresa y digital.

Achraf Ibrahimi, un compañero de clase, colaboró en el frontend y la conexión con el backend, aportando sus conocimientos.

Sin olvidar a los **Beta Testers:**

- Achraf Ibrahimi
- Alejandro Tome
- Irene Urbano - *Mención honorífica, descubrió un bug en la edición de perfiles.*
- Jaider Andres
- Othman Douiri - *Mención honorífica, expuso muchas vulnerabilidades.*
- Roly Silvestre

11. Anexos

11.1. Bibliotecas utilizadas

11.1.1. Frontend:

- **React Router DOM (v6.22.2):**
  - **Razón de elección:** Solución robusta y flexible para el manejo de rutas en aplicaciones React.
  - **Ventajas:** Soporte para rutas anidadas, lazy loading y navegación programática.
- **React Icons (v5.5.0):**
  - **Razón de elección:** Biblioteca completa de iconos que incluye múltiples conjuntos de iconos populares.
  - **Ventajas:** Fácil integración, optimización de rendimiento y amplia variedad de opciones.
- **React Toastify (v11.0.5):**
  - **Razón de elección:** Solución elegante para mostrar notificaciones al usuario.
  - **Ventajas:** Personalizable, accesible y fácil de implementar.
- **Axios (v1.6.7):**
  - **Razón de elección:** Cliente HTTP robusto para realizar peticiones a APIs.
  - **Ventajas:** Interceptores, manejo de errores mejorado y cancelación de peticiones.
- **Date-fns (v3.3.1):**
  - **Razón de elección:** Biblioteca moderna para el manejo de fechas en JavaScript.
  - **Ventajas:** Modular, inmutable y con excelente soporte para internacionalización.

12.1.2. Backend:

12.1.2.1. Spring Boot Starters

- **Spring Boot Starter Web:**
  - Razón de elección: Proporciona todas las dependencias necesarias para crear una aplicación web con Spring MVC.
- **Spring Boot Starter Data JPA:**
  - Razón de elección: Facilita la implementación de la capa de persistencia de datos utilizando JPA.
- **Spring Boot Starter Security:**
  - Razón de elección: Proporciona seguridad basada en Spring para la aplicación.
- **Spring Boot Starter Validation:**
  - Razón de elección: Permite la validación de datos de entrada utilizando anotaciones.
- **Spring Boot Starter Actuator:**
  - Razón de elección: Proporciona endpoints para monitoreo y gestión de la aplicación.

12.1.2.2. Base de Datos

- **PostgreSQL Driver:**
  - Razón de elección: Conector oficial para la base de datos PostgreSQL.

12.1.2.3. Seguridad

- **JWT (JSON Web Token):**
  - Razón de elección: Implementación de tokens JWT para autenticación stateless.
- **JAXB API:**
  - Razón de elección: Proporciona soporte para el procesamiento de XML en la aplicación.

12.1.2.4. Documentación

- **SpringDoc OpenAPI:**
  - Razón de elección: Genera documentación automática de la API REST.

12.1.2.5. Servidor

- **Spring Boot Starter Tomcat:**
  - Razón de elección: Servidor web embebido para la aplicación.

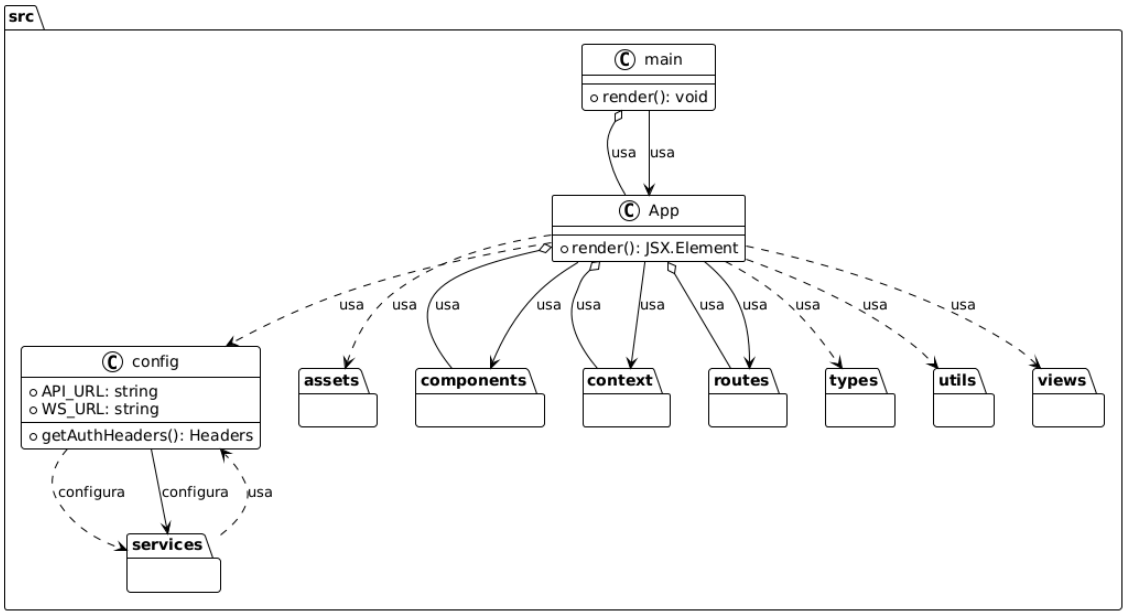
13.1. Herramientas de Desarrollo:

- **Maven:**
  - Razón de elección: Maven es una herramienta de gestión de dependencias y construcción que simplifica el ciclo de vida del proyecto.
  - Ventajas: Gestión de dependencias centralizada, ciclo de vida del proyecto estandarizado, y fácil integración con IDEs.
- **Git:**
  - Razón de elección: Git es el sistema de control de versiones más utilizado, permitiendo un trabajo colaborativo eficiente.
  - Ventajas: Control de versiones robusto, soporte para ramas, y excelente integración con plataformas de desarrollo colaborativo.
- **Herramientas de Documentación:**
  - PlantUML
  - Razón de elección: PlantUML permite crear diagramas UML a partir de texto plano, facilitando la documentación del sistema.
  - Ventajas: Fácil mantenimiento, integración con Markdown, y soporte para múltiples tipos de diagramas.

13.2. Diagramas de clases

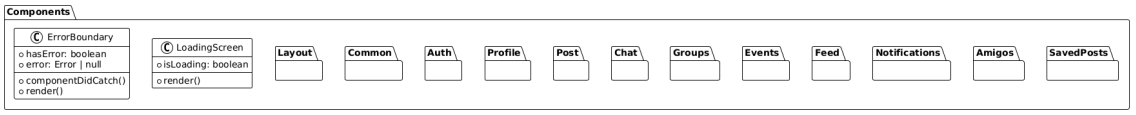
13.3.1. Frontend

13.3.1.1. src



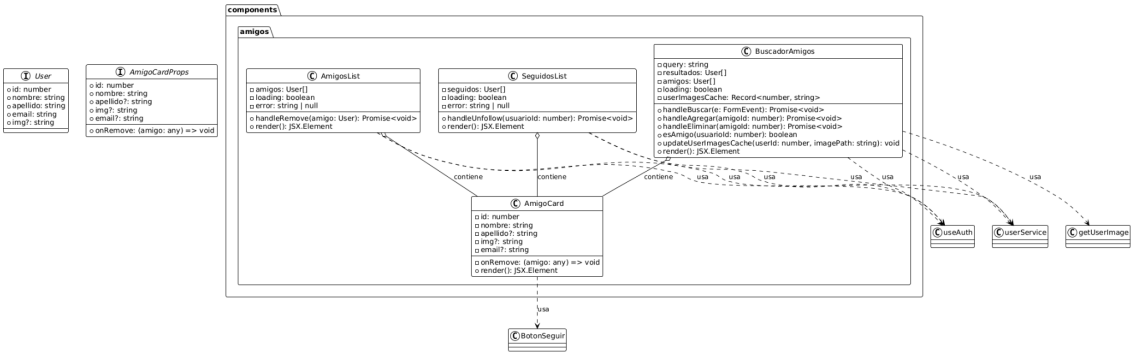
F32. PlantUML - src diagrama de clases UML

13.3.1.2. components



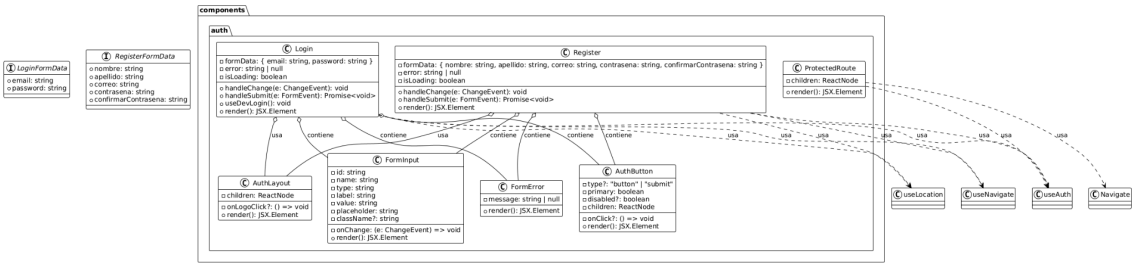
F33. PlantUML - components diagrama de clases UML

13.3.1.2.1. amigos



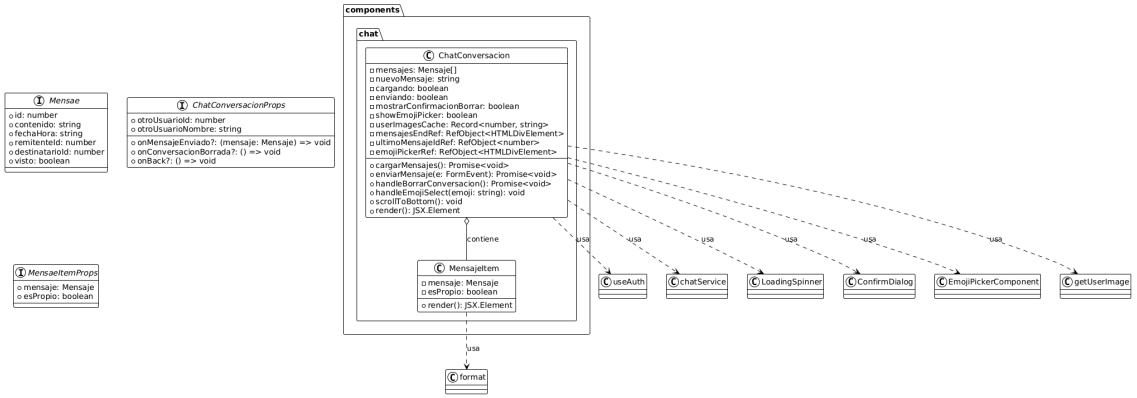
F34. *PlantUML - components.amigos diagrama de clases UML*

13.3.1.2.2. auth



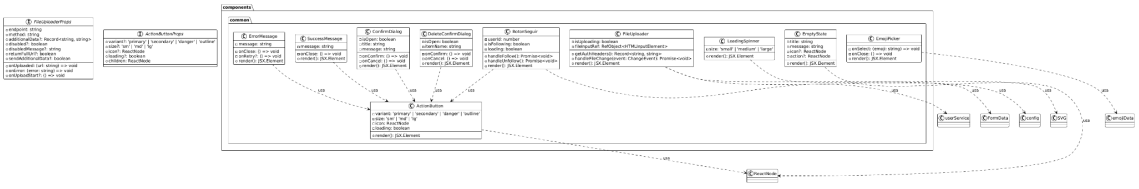
F35. *PlantUML - components.auth diagrama de clases UML*

13.3.1.2.3. chat



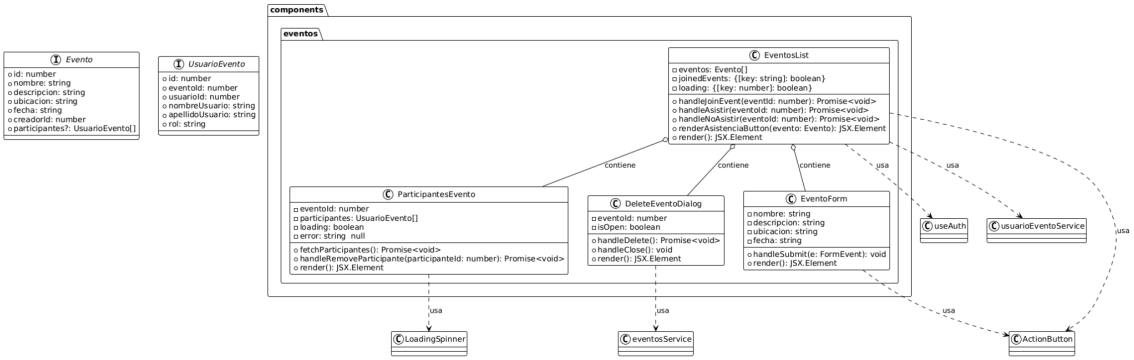
F36. *PlantUML - components.chat diagrama de clases UML*

13.3.1.2.4. common



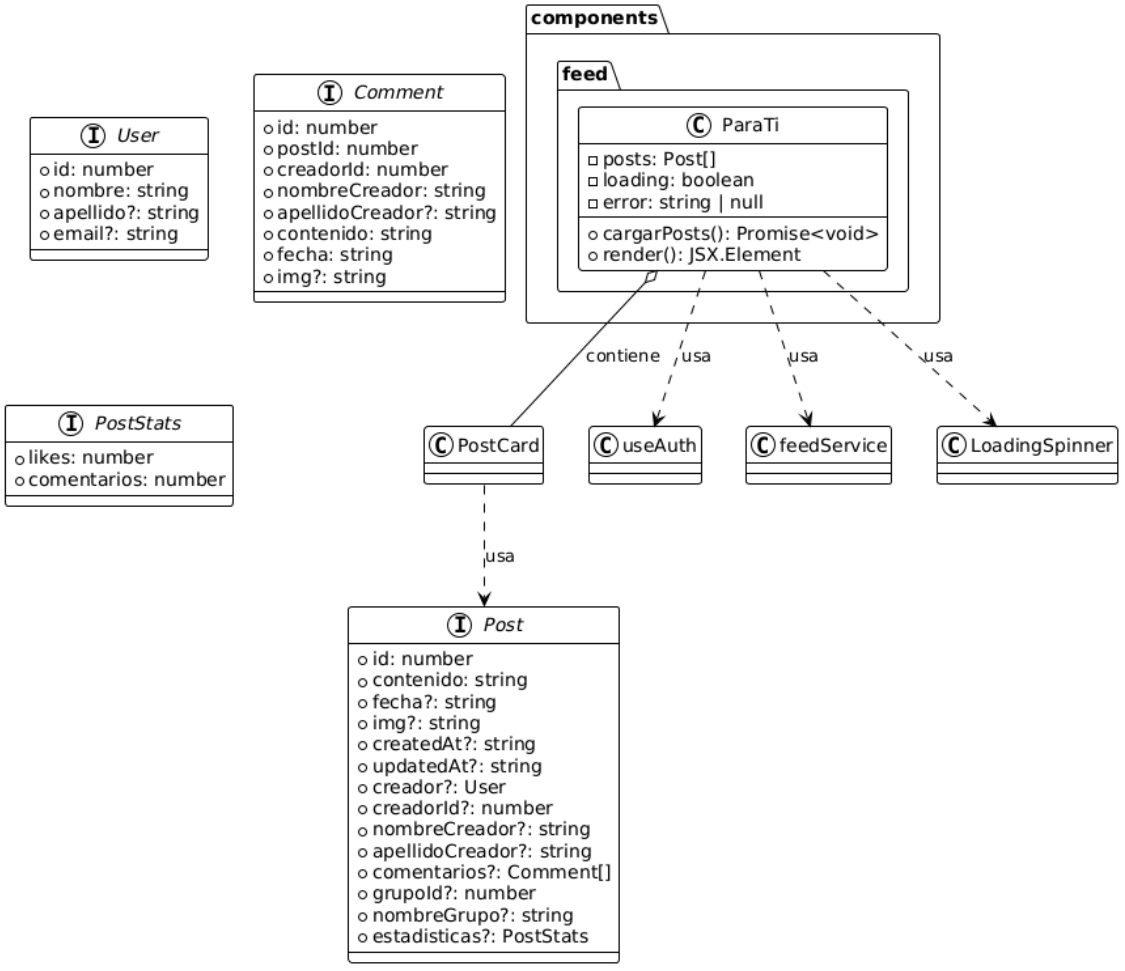
F37. *PlantUML - components.common diagrama de clases UML*

13.3.1.2.5. eventos



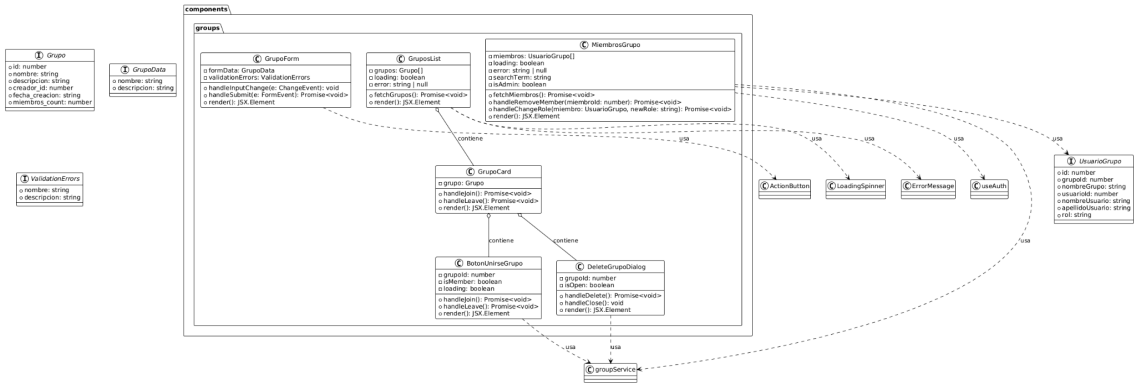
F38. [PlantUML - components.eventos diagrama de clases UML]

13.3.1.2.6. feed



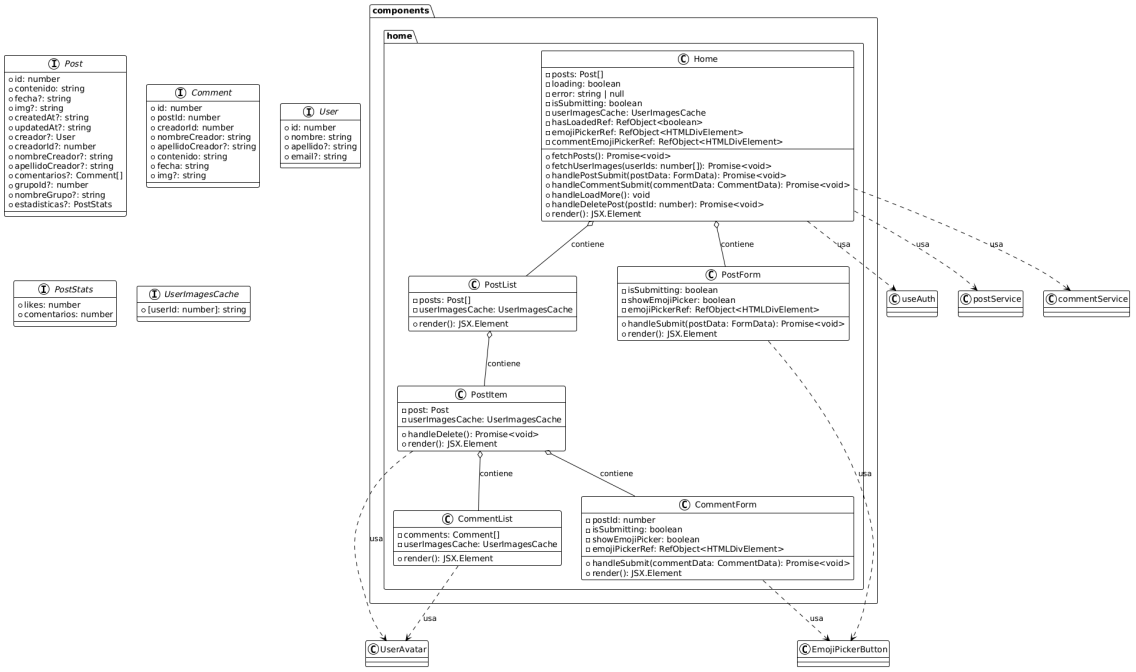
F39. [PlantUML - components.feed diagrama de clases UML]

13.3.1.2.7. groups



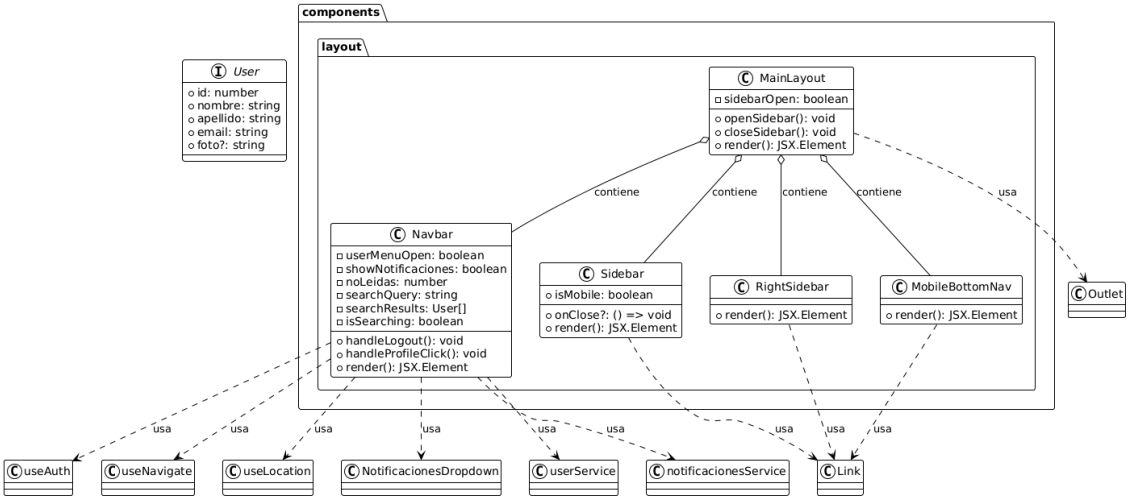
F40. PlantUML - components.groups diagrama de clases UML

13.3.1.2.8. home



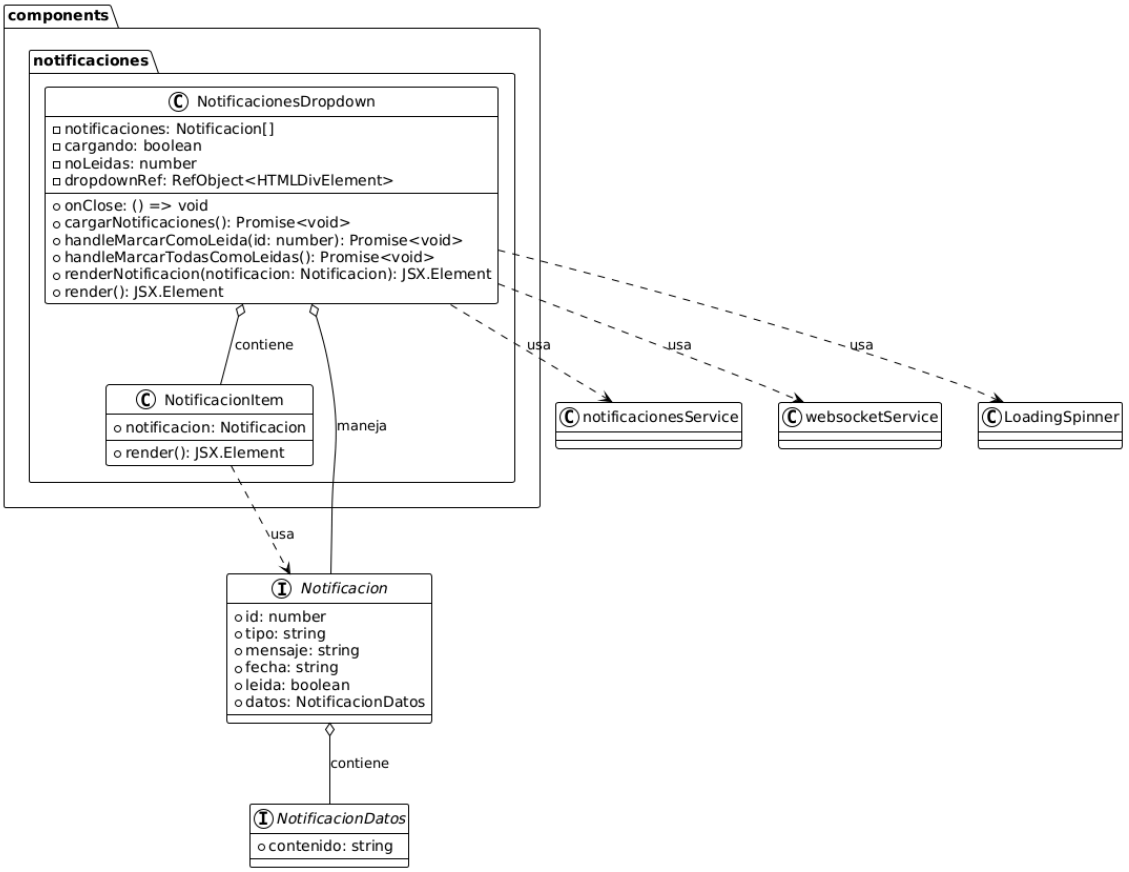
F41. PlantUML - components.home diagrama de clases UML

13.3.1.2.9. layout



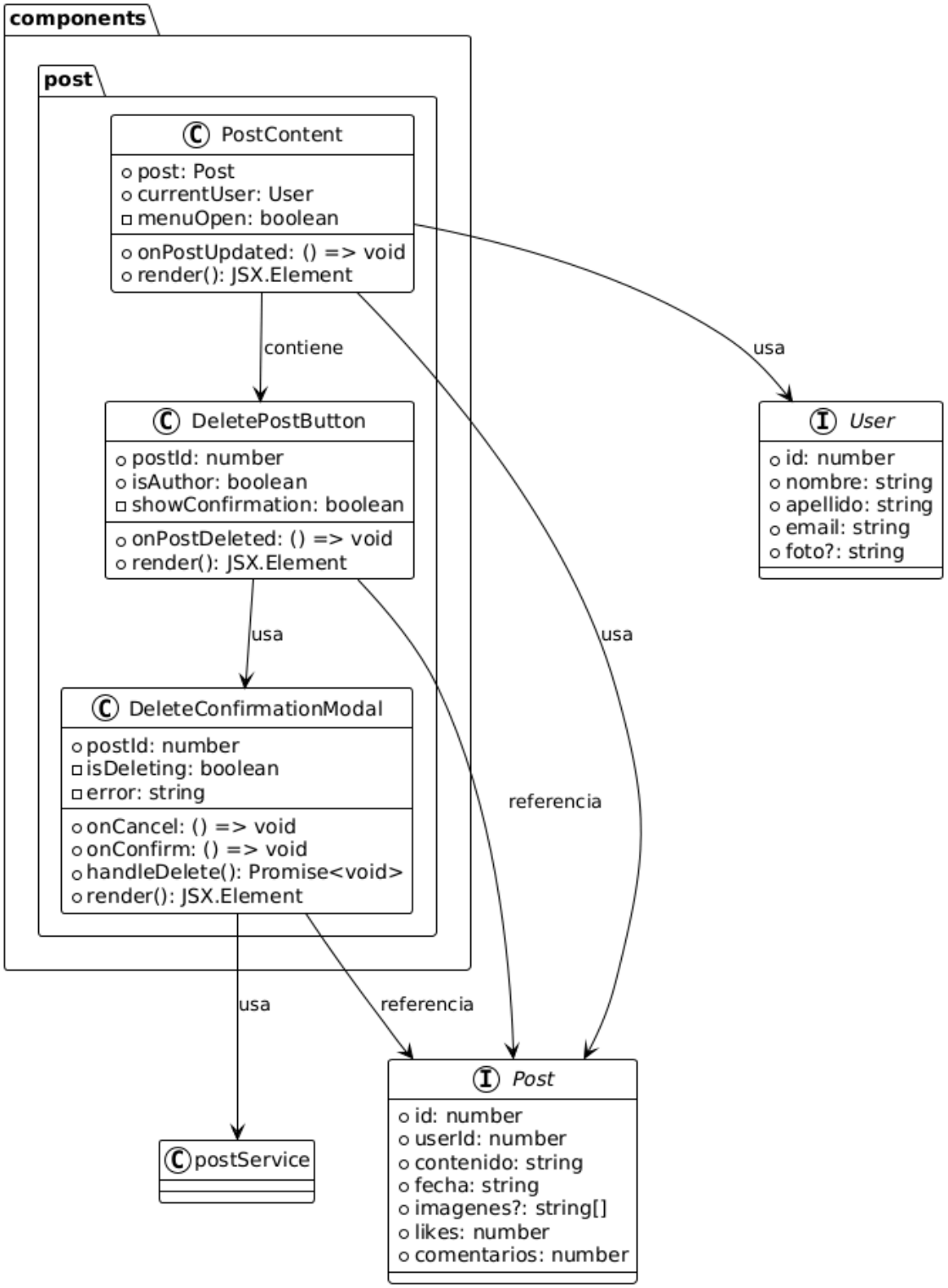
F42. PlantUML - components.layout diagrama de clases UML

13.3.1.2.10. notificaciones



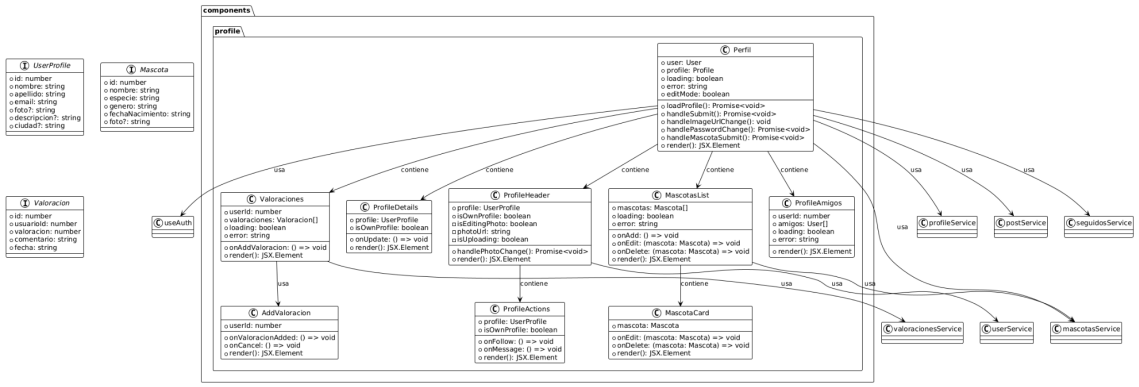
F43. PlantUML - components.notificaciones diagrama de clases UML

13.3.1.2.11. post



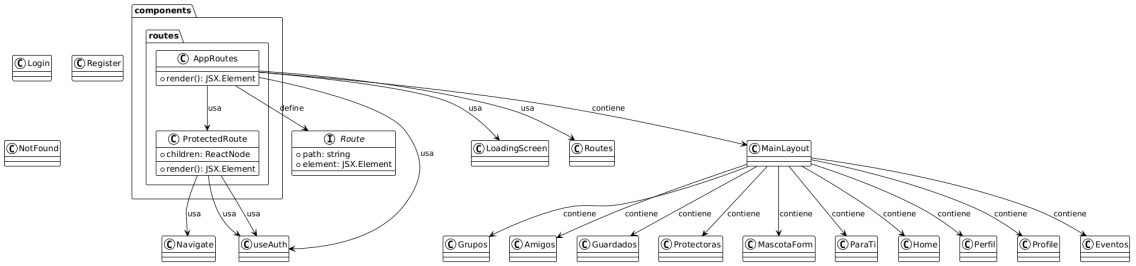
F44. PlantUML - components.post diagrama de clases UML

13.3.1.2.12. profiles



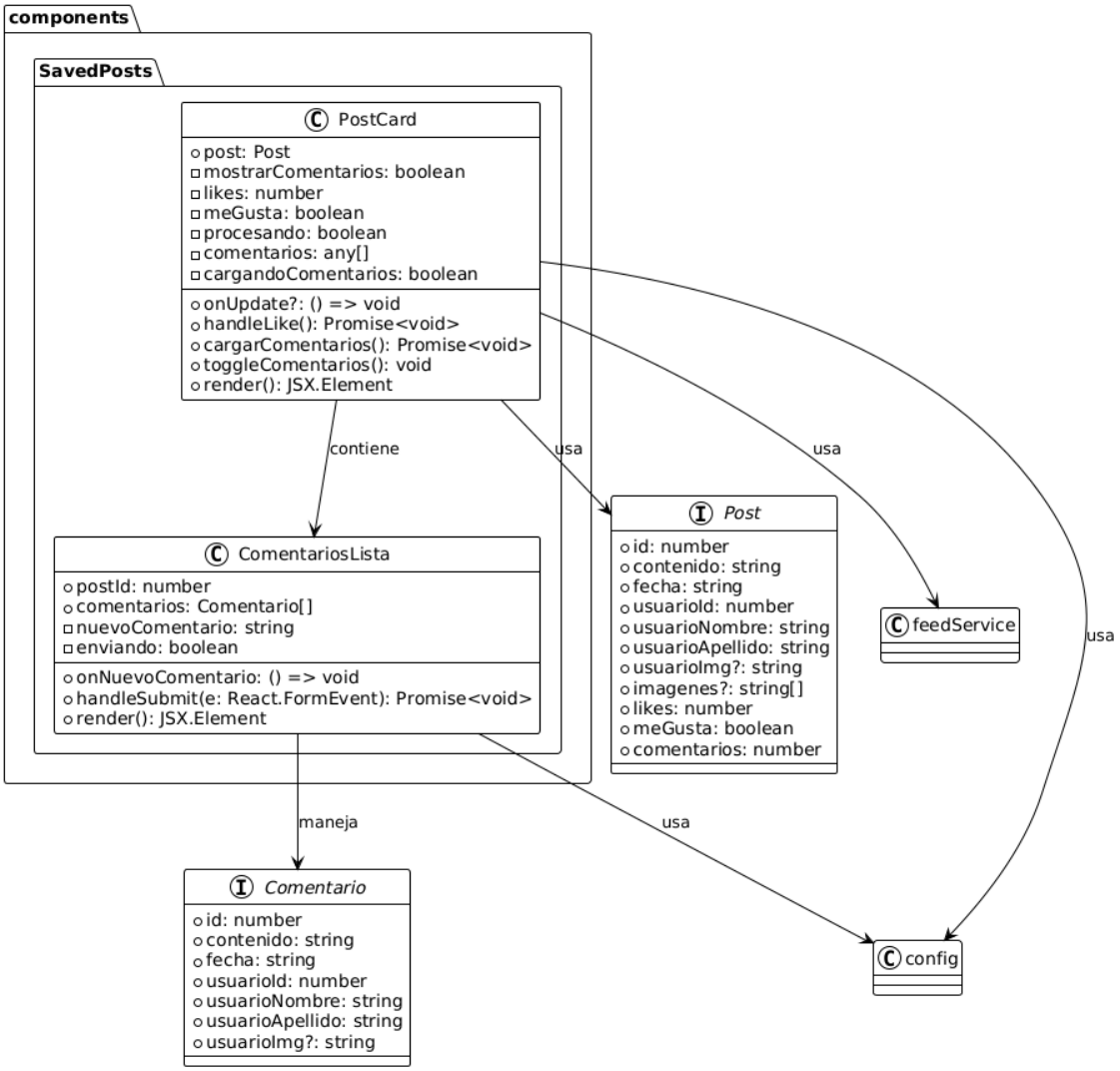
F45. [PlantUML - components.profile diagrama de clases UML]

13.3.1.2.13. routes



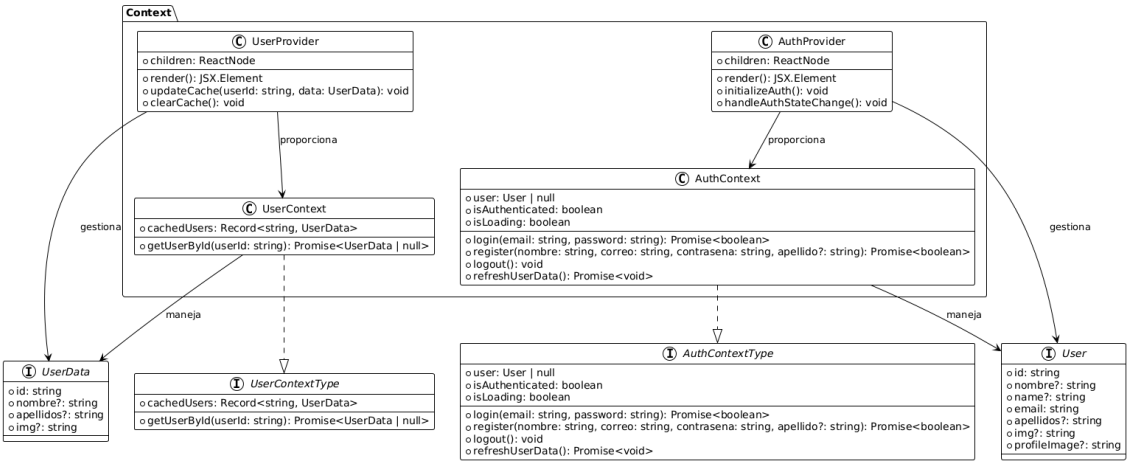
F46. [PlantUML - components.routes diagrama de clases UML]

13.3.1.2.14. Savedpost



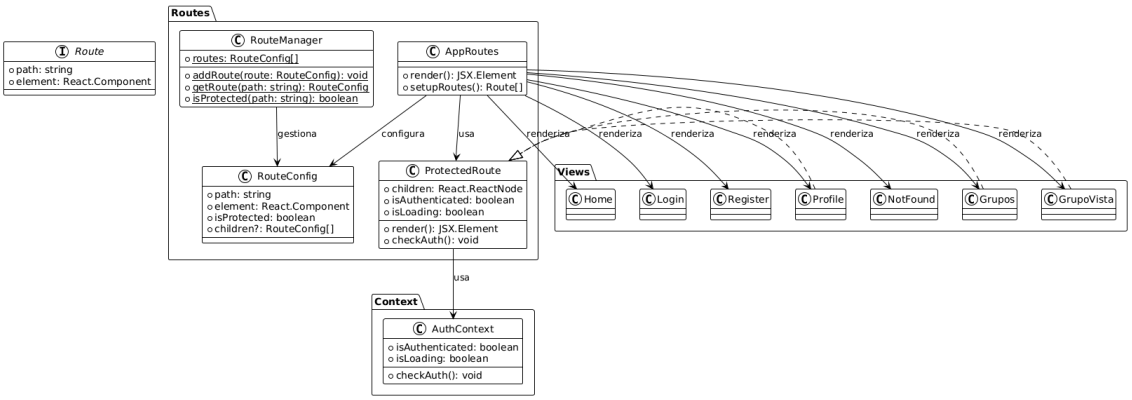
F47. [PlantUML - components.Savedposts diagrama de clases UML]

13.3.1.3. context



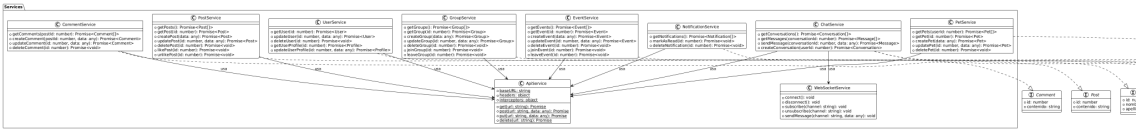
F48. PlantUML - context diagrama de clases UML

13.3.1.4. routes



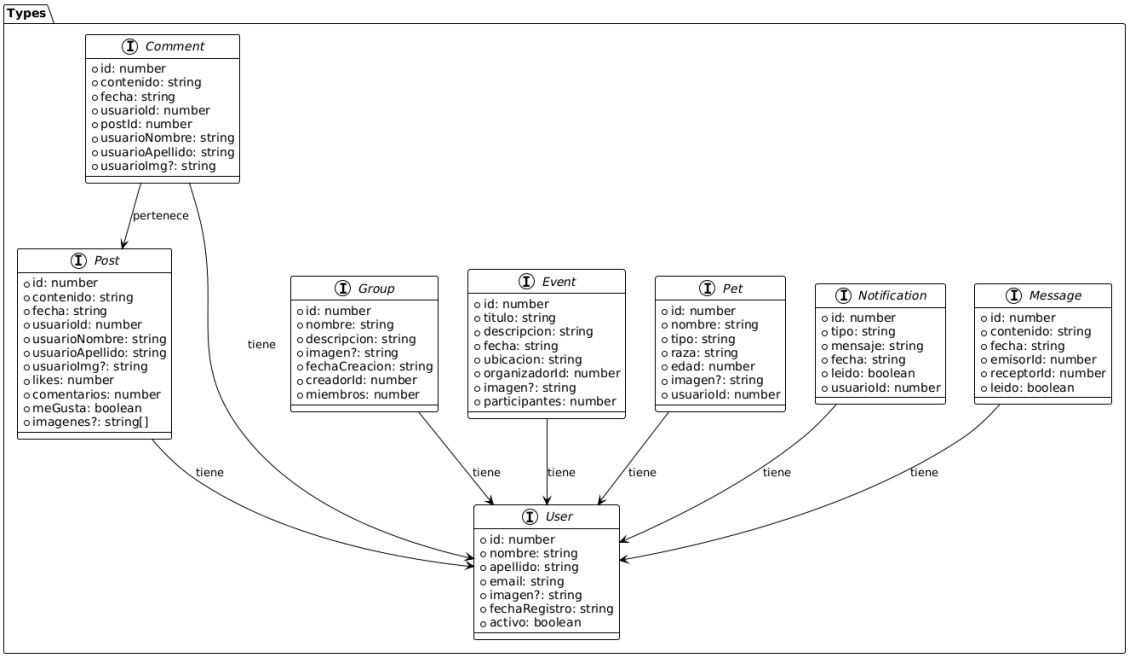
F49. PlantUML - routes diagrama de clases UML

13.3.1.5. services



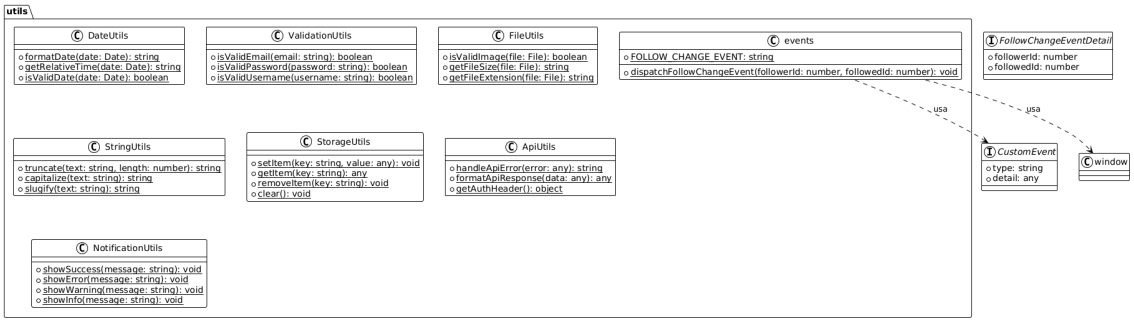
F50. PlantUML - services diagrama de clases UML

13.3.1.6. types



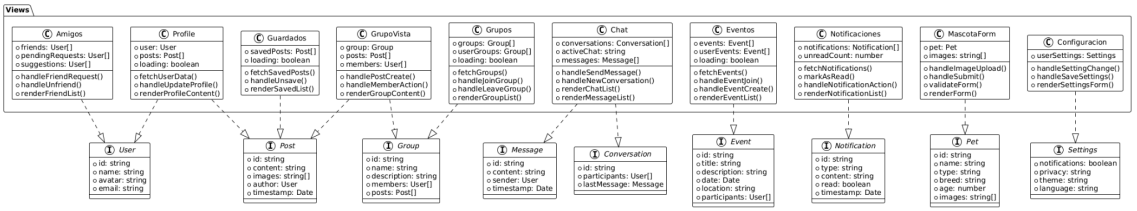
F51. PlantUML - types diagrama de clases UML

13.3.1.7. utils



F52. PlantUML - utils diagrama de clases UML

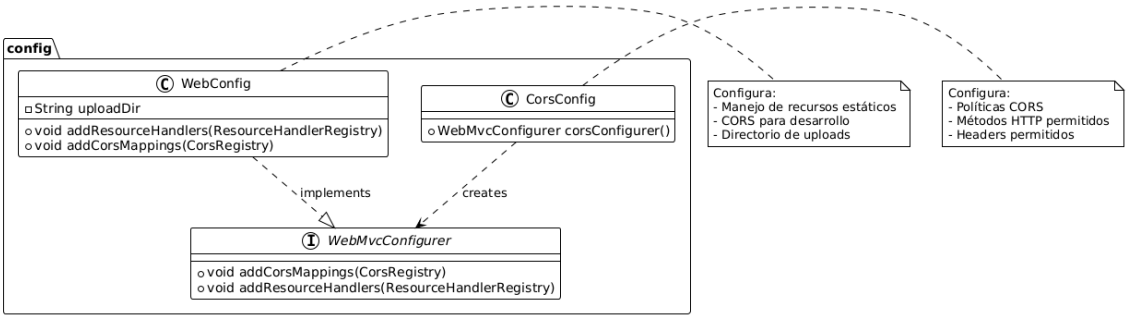
13.3.1.8. views



F53. PlantUML - views diagrama de clases UML

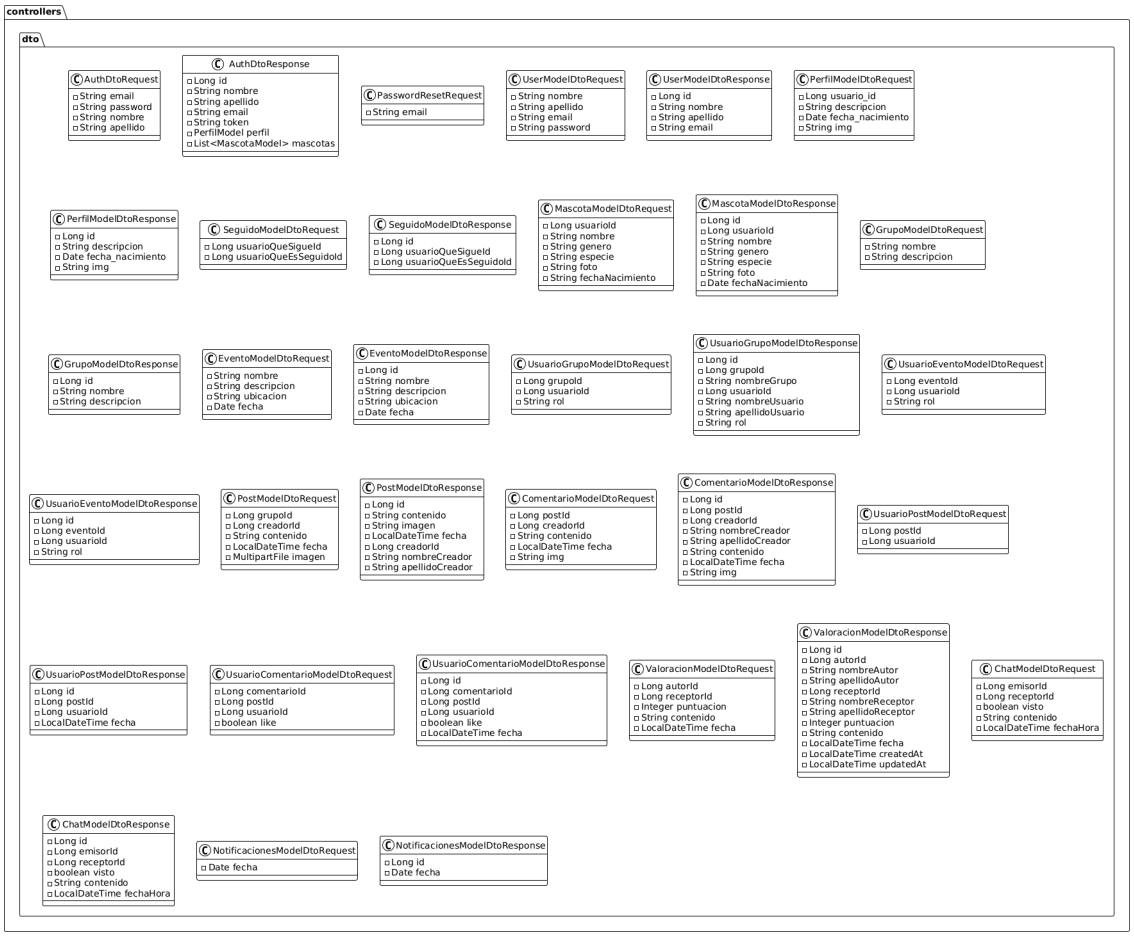
13.3.2. Backend

13.3.2.1. config



F54. PlantUML - config diagrama UML de clases

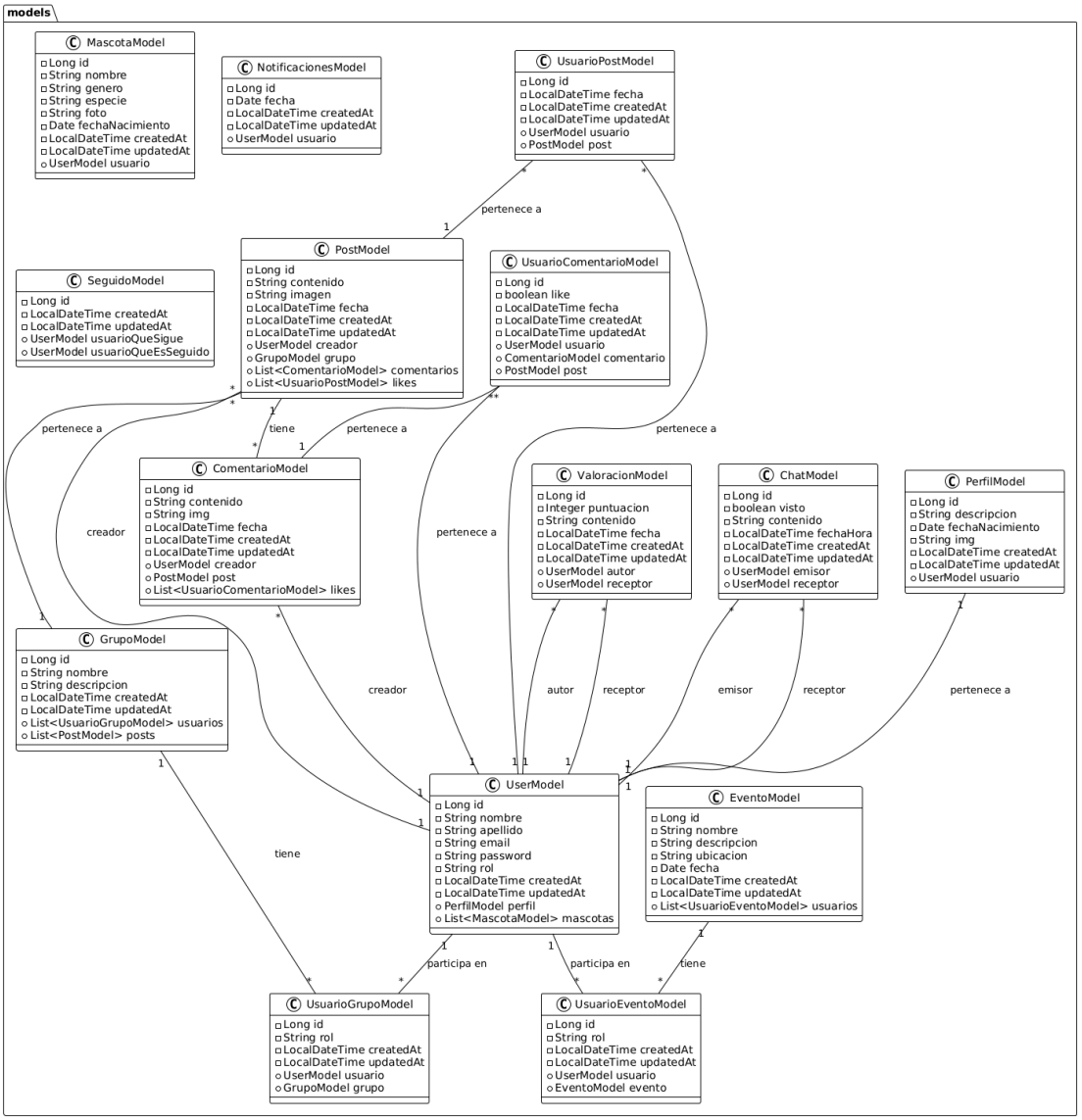
13.3.2.2. controllers



F55. PlantUML - controller.dto diagrama UML de clases

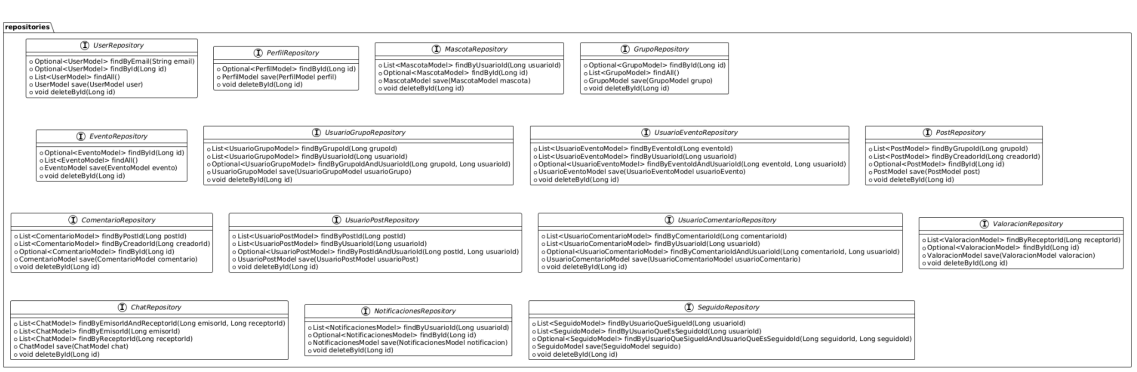
F56. PlantUML - controller.rest diagrama UML de clases

13.3.2.3. models



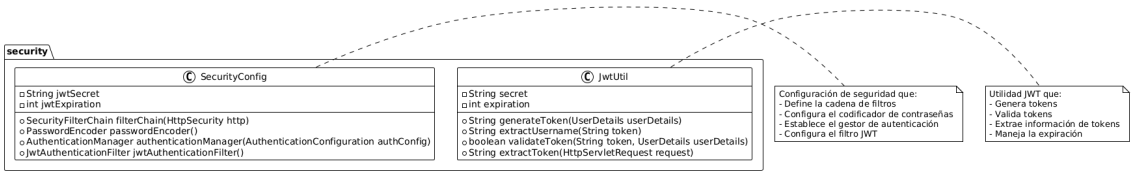
F57. [*PlantUML - models diagrama UML de clases*]

13.3.2.4. repositories



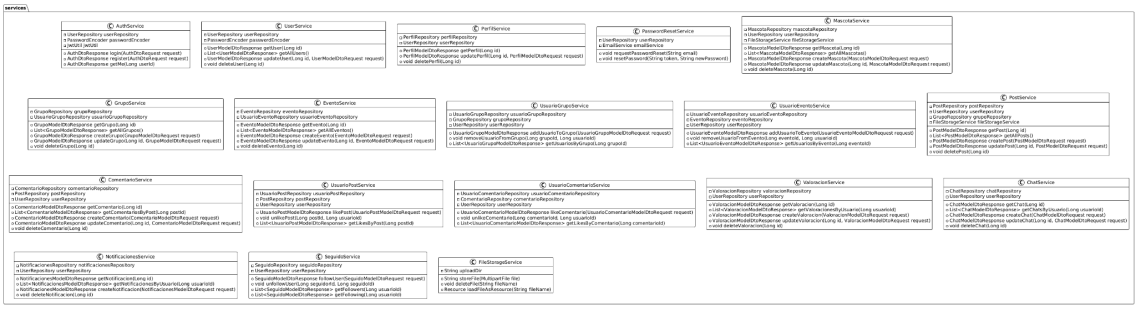
F58. PlantUML - repositories diagrama UML de clases

13.3.2.5. security



F59. PlantUML - security diagrama UML de clases

13.3.2.6. services



F60. PlantUML - services diagrama UML de clases

13.4. Plan de prevención de riesgos laborales

Se crea un plan de prevención de riesgos laborales de la empresa.
Plan de prevención de riesgos laborales Fernando Diaz Mouad Sedjari acabado.pdf