UNIVERSIDAD SANTO TOMAS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
T U N J A

TÍTULO PASANTÍA

Pajareo Colombia: plataforma web con Keycloak, módulo de usuarios y multimedia

PROPONENTE(S)
Julian Santiago Rangel Manrique
2298128

DIRECTOR
Diego Alejandro Vela Beltrán

Tunja
11/02/2026

CONTENIDO

1. FICHA TÉCNICA DE LA PASANTÍA 4

2. PLANTEAMIENTO DEL PROBLEMA 5

3. OBJETIVOS 9

3.1. OBJETIVO GENERAL 9
3.2. OBJETIVOS ESPECÍFICOS 9

4. DESCRIPCIÓN DE LAS ACTIVIDADES REALIZADAS 10

4.1. ACTIVIDADES REALIZADAS EN EL PROYECTO 10
4.2. HERRAMIENTAS UTILIZADAS EN EL DESARROLLO DEL PROYECTO 12

5. CONCLUSIONES 14

6. ANEXOS 15

6.1. RESUMEN DE LOS PROYECTOS DESARROLLADOS DURANTE LA PASANTÍA 15
6.2. MÉTRICAS Y ESTADÍSTICAS TÉCNICAS DEL PROYECTO 17
6.3. HERRAMIENTAS Y TECNOLOGÍAS UTILIZADAS 24
6.4. ARQUITECTURA GENERAL DEL SISTEMA 27
6.5. DIAGRAMA DE FLUJO DEL SISTEMA 29
6.6. DIAGRAMAS UML 30
6.6.1. DIAGRAMA DE CASOS DE USO 30
6.6.2. DIAGRAMA DE COMPONENTES 31
6.6.3. DIAGRAMA ENTIDAD–RELACIÓN 32
6.6.4. DIAGRAMAS DE SECUENCIA 33
6.6.5. EVIDENCIAS DE DOCUMENTACIÓN TÉCNICA 35
6.6.6. CÓDIGOS DE ERROR 37
6.6.7. GLOSARIO 38
6.6.8. ESPECIFICACIÓN DEL MÓDULO DE GESTIÓN DE USUARIOS COMO COMPONENTE REUTILIZABLE 39

6.7. EVIDENCIAS SELECCIONADAS DEL CRONOGRAMA DE ACTIVIDADES 40
6.7.1. EVIDENCIA DE RED SEGURA (TAILSCALE) 40
6.7.2. EVIDENCIA DE VIRTUALIZACIÓN CON PROXMOX VE 41
6.7.3. EVIDENCIA DE INTEGRACIÓN CON KEYCLOAK 42
6.7.4. EVIDENCIA DE LA ARQUITECTURA HEXAGONAL DEL BACKEND 44
6.7.5. EVIDENCIA DEL FRONTEND 45
6.7.6. EVIDENCIA DEL MICROSERVICIO DE MEDIOS 48
6.7.7. EVIDENCIA DE LA GUÍA DE USO DE OPENPROJECT 51

7. REFERENCIAS BIBLIOGRÁFICAS 54

1. FICHA TÉCNICA DE LA PASANTÍA

Título	Pajareo Colombia: plataforma web con Keycloak, módulo de usuarios y multimedia
Nombre Estudiante	Julian Santiago Rangel Manrique
Correo electrónico	julian.rangel@usantoto.edu.co
Director	Diego Alejandro Vela Beltrán
Entidad o sector de la empresa	EnLinea.App SAS
Lugar de ejecución de la pasantía	Tunja
Duración	Seis meses

Los abajo firmantes confirman que todos los datos incluidos en la presente propuesta son correctos y verídicos, que no incumplen ninguna ley o norma vigente. (Incluir nombres y firmas de estudiantes, director y tutor).

Firma del autor
Nombre autor

Firma del director
Nombre director de la pasantía

2. PLANTEAMIENTO DEL PROBLEMA

Pajareo Colombia surge en un contexto donde la generación y uso de información sobre avistamientos de aves depende de flujos digitales cada vez más exigentes. En la práctica, los registros y sus evidencias multimedia (imágenes, audio y video) se gestionan hoy en canales dispersos, con metadatos incompletos o inconsistentes y procesos manuales de moderación que dificultan publicar información confiable y oportuna. Para EnLinea.App —tanto su equipo interno como sus clientes institucionales— esta fragmentación se traduce en fricción operativa, baja trazabilidad y costos crecientes, justo cuando las expectativas de desempeño, seguridad y calidad de datos no dejan de aumentar.

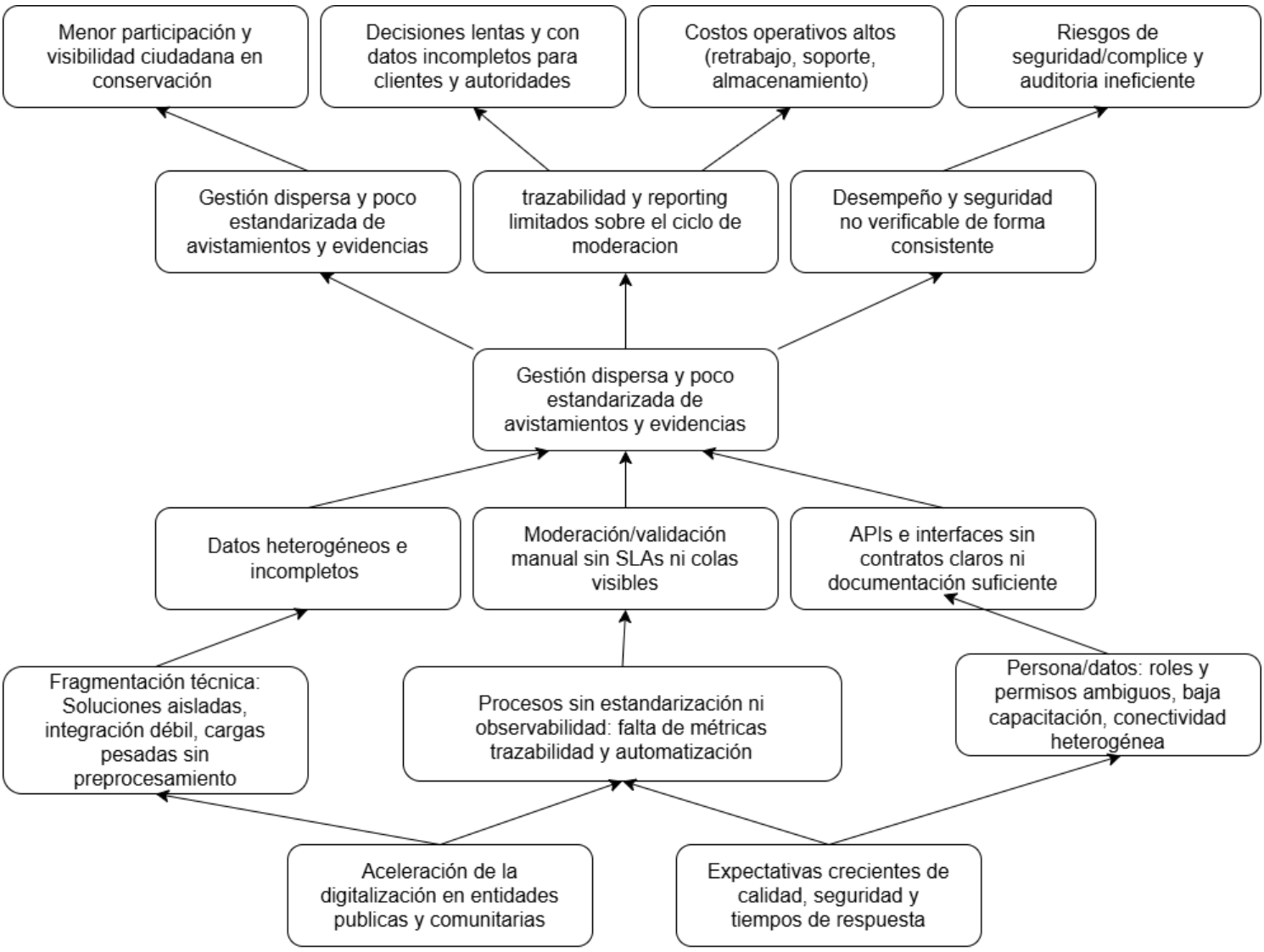
Para **ENLINEA.APP S.A.S**, empresa orientada a la creación de soluciones tecnológicas a la medida, esta problemática se traduce en un obstáculo para el desarrollo de plataformas sostenibles y escalables. La falta de una infraestructura que garantice una correcta autenticación, control de accesos y gestión eficiente de recursos multimedia limita la capacidad de respuesta ante la creciente demanda de servicios digitales confiables. Además, los procesos manuales y la carencia de herramientas automatizadas generan sobrecarga operativa, posibles inconsistencias y mayores tiempos de entrega.

El problema central, por tanto, radica en **la ausencia de una plataforma unificada que integre la captura, moderación y publicación de avistamientos con mecanismos seguros de identidad, autorización y validación de datos multimedia**. Esta carencia impide garantizar una trazabilidad completa del flujo de información y dificulta la generación de valor a partir de los datos recolectados.

De esta manera, surge la necesidad de diseñar e implementar una solución tecnológica que permita **unificar y estandarizar los procesos relacionados con la gestión de avistamientos**, integrando buenas prácticas de arquitectura backend, control de identidad, procesamiento de archivos y observabilidad, en coherencia con los estándares modernos de calidad y seguridad que caracterizan a **ENLINEA.APP S.A.S**.

La **Figura 1** sintetiza esta lógica: en la base, las causas tecnológicas, de proceso y de roles/datos (falta de estandarización, moderación manual sin SLAs, UX irregular y metadatos deficientes); en el centro, la gestión dispersa y poco estandarizada de avistamientos; y en la parte superior, los efectos visibles (menor participación, decisiones lentas y costos altos). El proyecto orienta sus objetivos a cerrar estos vacíos con una solución medible, segura y reutilizable, mejorando la efectividad de EnLinea.App y el impacto de la información en conservación y gestión ambiental.

Figura 1. Árbol de Problemas



Fuente: Autor

En la **Tabla 2** se muestra la matriz de Stakeholders del proyecto, donde se identifican los grupos clave, como el director/a de la pasantía, EnLinea.App (Supervisores/Equipo), junto con sus intereses, niveles de influencia e interés. Para cada Stakeholders se proponen estrategias de gestión que incluyen comunicación frecuente, informes periódicos y la consideración de sus expectativas según su rol en el proyecto. También se mencionan otros actores como la Universidad, el equipo de desarrollo, y los usuarios finales.

Tabla 1. Matriz de Stakeholders

Stakeholder	Intereses	Nivel de influencia	Nivel de interés	Estrategia de gestión
Director/a de la pasantía	Cumplimiento académico, alcance claro, riesgos controlados, entregables medibles.	Alto	Alto	Colaborar / Empoderar (co-definir criterios y metodología)
EnLinea.App — Supervisores / equipo	Calidad técnica, mantenibilidad, seguridad (Keycloak), costos operativos.	Alto	Alto	Colaborar (revisiones de código/PR, definición de estándares)
EnLinea.App — Clientes (municipios/entidades)	Usabilidad, tiempos de atención, reportes claros y oportunos.	Media-Alta	Alto	Consultar / Colaborar (demos, validación de requerimientos)
Universidad	Asegurarse de que el proyecto cumpla con los requisitos académicos y se concluya satisfactoriamente.	Medio	Medio	Entregar informes y resultados según los plazos y directrices establecidos por la universidad.

Stakeholder	Intereses	Nivel de influencia	Nivel de interés	Estrategia de gestión
Equipo de Desarrollo de EnLinea.App	Colaborar en la implementación de las soluciones propuestas, asegurando que sean compatibles con las tecnologías y sistemas existentes.	Alto	Medio	Mantener una comunicación fluida y una colaboración efectiva durante el proceso de desarrollo e integración de las soluciones propuestas.
Usuarios observadores (pajareros)	Carga sencilla de fotos/audio/video, reconocimiento, privacidad.	Media	Medio	Consultar (beta cerrada, feedback guiado)

Fuente: Autor

3. OBJETIVOS

3.1. Objetivo General

Desarrollar la plataforma web Pajareo Colombia en EnLinea.App SAS con arquitectura hexagonal, integración de Keycloak para autenticación y autorización, un microservicio de medios para imágenes, audio y video, y APIs documentadas y protegidas, asegurando que los endpoints críticos tengan p50 ≤ 200 ms y sin hallazgos críticos de seguridad, e incluyendo la gestión de usuarios como módulo independiente reutilizable en otros proyectos.

3.2. Objetivos específicos

- Implementar el backend del sistema en Java 17 y Spring Boot bajo arquitectura hexagonal, con modelado y normalización de PostgreSQL, APIs REST documentadas en OpenAPI, CI/CD con ejecución ≤ 8 min y éxito ≥ 90 %, cobertura unitaria ≥ 80 % en módulos críticos, incorporando observabilidad básica mediante métricas y logs.
- Configurar la autenticación y autorización mediante Keycloak, Spring Security y OAuth2/OIDC, definiendo roles y políticas, protegiendo los endpoints críticos y verificando accesos mediante pruebas de integración y seguridad con cobertura ≥ 95 % de casos positivos y negativos.
- Crear un microservicio de medios en Java/Spring Boot, validando tipo y tamaño de archivos, escalando imágenes a 1280×720, extrayendo metadatos EXIF/GPS, persistiendo referencias en PostgreSQL e integrando con Google Cloud Storage, para garantizar tiempo de procesamiento ≤ 2 s y tasa de éxito ≥ 95 % en cargas válidas.
- Documentar el sistema mediante OpenAPI/Swagger y UML (casos de uso, clases y secuencia) con ejemplos estandarizados, códigos de error, glosario, trazabilidad y especificación del módulo de gestión de usuarios como componente reutilizable, asegurando cobertura completa de los módulos críticos.

4. DESCRIPCIÓN DE LAS ACTIVIDADES REALIZADAS

4.1. Actividades realizadas en el proyecto

En la **Tabla 2** se describen las principales actividades desarrolladas durante la pasantía, organizadas cronológicamente y asociadas a los productos entregados en cada etapa del proyecto. Estas actividades abarcan el diseño de la arquitectura, la implementación del backend, la configuración de la seguridad, el despliegue de infraestructura y la documentación técnica de la solución.

Tabla 2. Cronograma de actividades

Descripción de la actividad	Fecha inicio	Fecha entrega	Producto entregado
Diagnóstico, limpieza y mantenimiento preventivo/correctivo de equipos de cómputo (inventario básico, limpieza interna, pruebas SMART, actualización de BIOS/firmware cuando aplicó).	01/06/2025	15/06/2025	Informe de mantenimiento con hallazgos y acciones
Implementación de servidor de virtualización con Proxmox VE (instalación, almacenamiento, bridges de red, backups) y definición de VMs: DB PostgreSQL, Docker/servicios, web y futura VM para OpenProject.	10/06/2025	25/06/2025	Host Proxmox operativo, VMs creadas (Ver Anexo 6.7.2).
Configuración de red segura de administración con Tailscale y endurecimiento básico (usuarios, claves, fail2ban/logrotate según aplique).	20/06/2025	05/07/2025	Acceso administrativo seguro y registro de accesos (Ver Anexo 6.7.1)
Diseño e implementación del backend con arquitectura hexagonal.	01/07/2025	31/08/2025	Repositorio del servicio con estructura hexagonal (Ver Anexo 6.7.4)
Modelado de base de datos PostgreSQL normalización de tablas núcleo	15/07/2025	15/08/2025	Esquema SQL versionado y diagrama (Ver Anexo 6.6)
Desarrollo del front-end en Angular con temas de PrimeNG y componentes	01/08/2025	15/09/2025	App Angular funcional integrada con el backend (Ver Anexo 6.7.5)

Descripción de la actividad	Fecha inicio	Fecha entrega	Producto entregado
Integración Front–Back con Keycloak (OIDC Authorization Code + PKCE): flujo de inicio de sesión, almacenamiento seguro de tokens, renovación, extracción de roles para guards, consumo de API documentada (Swagger), y manejo unificado de errores.	15/08/2025	30/09/2025	Front autenticado/autorizado y pruebas de consumo de API (Ver Anexos 6.5 y 6.7.5)
Integración de Keycloak: realm de la solución, clientes, roles y políticas; protección de endpoints	15/08/2025	30/09/2025	Realm configurado y endpoints protegidos(Ver Anexos 6.5 y 6.7.3)
Servicio de multimedia (validación de tipo/tamaño, procesamiento de imágenes 1280×720, extracción EXIF/GPS, persistencia de referencias).	01/09/2025	15/10/2025	Endpoint de carga y procesamiento con pruebas (Ver Anexo 6.2)
Capacitación interna: “Introducción a Keycloak (OIDC, roles/policies)” y “Arquitectura Hexagonal aplicada al proyecto”.	01/10/2025	31/10/2025	Proyecto base implementando seguridad con keycloak
Despliegue en AWS–EC2 (Ubuntu): build de Angular, Nginx como servidor estático	15/10/2025	15/11/2025	Sitio productivo en EC2 con Nginx y backend operativo
Bucket S3 para multimedia: creación y versionado, CORS del bucket, políticas IAM de mínimo privilegio, lifecycle rules (transición/expiración), y uso de URLs prefirmadas desde el backend para cargas/descargas seguras.	01/11/2025	30/11/2025	Bucket S3 configurado
Montaje de OpenProject en VM dedicada (repositorio, instalación, servicio systemd, respaldo, usuarios y proyectos iniciales).	01/11/2025	20/11/2025	OpenProject operativo (Ver Anexo 6.3).
Documentación final (manual de administración, de despliegue y de usuario funcional de OpenProject).	15/11/2025	30/11/2025	Conjunto de manuales y guías operativas (Ver Anexo 6.7.7)

4.2. Herramientas utilizadas en el desarrollo del proyecto

Tabla 3. Herramientas

Herramienta utilizada	Para qué la utilizó
Proxmox VE	Virtualización del entorno de servidores y despliegue de VMs.
Ubuntu Server	SO de las VMs para servicios (DB, OpenProject, etc.).
PostgreSQL	Motor de base de datos principal para los proyectos
Docker	Empaquetado y ejecución de servicios auxiliares.
OpenProject	Gestión de proyectos, tareas y seguimiento de avances.
Keycloak (OIDC/OAuth2)	Gestión de identidad, roles y autorización; protección de APIs.
Java 17 / Spring Boot	Backend bajo arquitectura hexagonal, controladores y servicios.
Spring Security / Resource Server	Validación y autorización de JWT emitidos por Keycloak.
OpenAPI/Swagger	Documentación de contratos de API y pruebas interactuables.
Git/GitHub (o GitLab)	Control de versiones, PRs y revisión de código.
Tailscale	Acceso seguro de administración al entorno.
Nginx	Servir estáticos de Angular y reverse proxy al backend.
JUnit/Mockito/Testcontainers	Pruebas unitarias/integración con entornos reproducibles.
PlantUML/Lucidchart	Diagramación (casos de uso, clases, secuencia).
Angular	Front-end SPA modular, rutas con lazy loading, interceptores y guards.
PrimeNG	Componentes UI, temas, tablas, diálogos, notificaciones y carga de archivos.
AWS EC2	Servidor de aplicación (front y proxy, backend).

Herramienta utilizada	Para qué la utilizó
AWS S3	Almacenamiento de multimedia con versionado y CORS.

Fuente: Autor

5. CONCLUSIONES

- De manera general, la pasantía permitió consolidar una solución web modular para Pajareo Colombia, integrando backend bajo arquitectura hexagonal, autenticación centralizada con Keycloak, gestión de archivos multimedia y documentación técnica del sistema. La solución quedó soportada por una infraestructura híbrida de desarrollo y pruebas, con criterios de mantenibilidad, seguridad y trazabilidad acordes con el contexto empresarial en el que fue desarrollada.
- Para dar cumplimiento al primer objetivo específico, se implementó el backend de la plataforma en Java 17 y Spring Boot bajo arquitectura hexagonal, con modelo relacional normalizado en PostgreSQL y APIs REST documentadas, evidenciando una estructura modular, mantenible y alineada con buenas prácticas de ingeniería de software (Ver Anexos 6.2, 6.4, 6.6 y 6.7).
- Para dar cumplimiento al segundo objetivo específico, se implementó un sistema de autenticación y autorización basado en Keycloak, aplicando el estándar OAuth2/OpenID Connect mediante la configuración de realms, clientes, roles y políticas de acceso, lo cual permitió proteger los endpoints del backend y controlar el acceso a los recursos según el perfil del usuario (Ver Anexos 6.5 y 6.7).
- Para dar cumplimiento al tercer objetivo específico, se desarrolló un microservicio independiente para la gestión de archivos multimedia, encargado de validar tipo y tamaño de archivo, procesar imágenes, extraer metadatos y persistir referencias asociadas, garantizando una operación consistente y coherente con los requerimientos funcionales definidos para la plataforma (Ver Anexo 6.2).
- Para dar cumplimiento al cuarto objetivo específico, se documentó el sistema mediante OpenAPI/Swagger y diagramas UML, facilitando la comprensión funcional y técnica de la solución, así como su mantenimiento, futura evolución y posible reutilización en otros proyectos de la empresa (Ver Anexos 6.3 y 6.6).
- Adicionalmente, el desarrollo de este proyecto fortaleció de manera significativa mis competencias profesionales en ingeniería de software, al permitirme aplicar de forma práctica arquitecturas modernas, seguridad basada en estándares, diseño de servicios desacoplados y documentación técnica orientada a mantenibilidad. Asimismo, la interacción con un entorno empresarial real consolidó habilidades

transversales como la planificación de actividades, la comunicación técnica, el trabajo colaborativo y la entrega de resultados medibles.

6. ANEXOS

6.1. Resumen de los proyectos desarrollados durante la pasantía

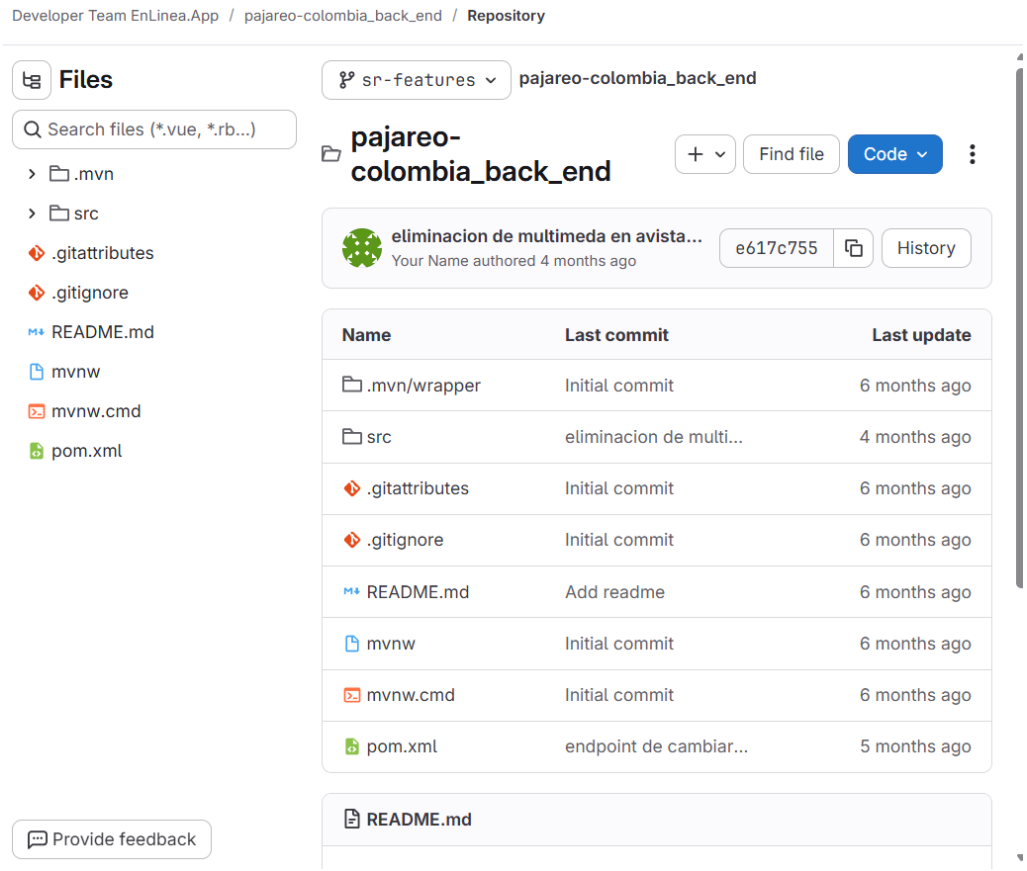
Durante la pasantía se participó activamente en el diseño, implementación y despliegue de la plataforma Pajareo Colombia, así como en actividades de infraestructura y soporte tecnológico asociadas. Los principales proyectos y frentes de trabajo fueron los siguientes:

- **Plataforma Pajareo Colombia (Backend y Seguridad)**
  Se diseñó e implementó el backend de la plataforma bajo arquitectura hexagonal, empleando Java 17 y Spring Boot. Se desarrollaron APIs REST documentadas, con modelado relacional normalizado en PostgreSQL, orientadas a garantizar mantenibilidad, trazabilidad y reutilización de componentes.
- **Gestión de identidad y control de accesos**
  Se integró Keycloak como servidor de identidad bajo OAuth2/OIDC, definiendo realms, clientes, roles y políticas de autorización. Los endpoints críticos fueron protegidos mediante Spring Security como Resource Server, garantizando autenticación robusta y control de accesos basado en roles.
- **Microservicio de gestión multimedia**
  Se desarrolló un microservicio independiente para la carga y procesamiento de archivos multimedia (imágenes, audio y video), incorporando validación de tipo y tamaño, escalado de imágenes a 1280×720, extracción de metadatos EXIF/GPS y persistencia de referencias en base de datos. El almacenamiento se integró con servicios de objetos en la nube.
- **Infraestructura y despliegue**
  Se implementó un entorno de virtualización con Proxmox VE para desarrollo y pruebas, y un entorno productivo en AWS EC2. Se configuró Nginx como servidor web y proxy inverso, así como almacenamiento de multimedia en AWS S3 con políticas de mínimo privilegio y URLs prefirmadas.

- **Gestión de proyectos y documentación**

  Se realizó la instalación y puesta en marcha de OpenProject para la gestión de tareas y seguimiento del proyecto. Adicionalmente, se elaboraron manuales técnicos, documentación de despliegue, guías de administración y documentación funcional de los sistemas implementados.

Figura 2. Vista general del proyecto Pajareo Colombia en repositorio



Fuente: Autor

6.2. Métricas y estadísticas técnicas del proyecto

Con el fin de evaluar objetivamente el cumplimiento de los objetivos definidos, se establecieron métricas técnicas y operativas, cuyos resultados se resumen a continuación:

Tabla 4. Indicadores técnicos alcanzados

Indicador	Resultado alcanzado
Arquitectura backend	Hexagonal (dominio, aplicación, infraestructura)
Lenguaje / Framework	Java 17 / Spring Boot
Base de datos	PostgreSQL normalizada
Autenticación y autorización	OAuth2 / OIDC con Keycloak
Latencia en endpoints críticos	p50 ≤ 200 ms
Tiempo de procesamiento multimedia	≤ 2 segundos por carga válida
Tasa de éxito en cargas válidas	≥ 95 %
Cobertura de pruebas en módulos críticos	≥ 80 %
Documentación de APIs	OpenAPI / Swagger completa
Infraestructura productiva	AWS EC2 + Nginx
Almacenamiento multimedia	AWS S3 con versionado y CORS

Fuente: Autor

Con el propósito de complementar los indicadores técnicos definidos, se presenta a continuación un resumen de las principales pruebas ejecutadas durante el desarrollo del proyecto. Este check list permite evidenciar las validaciones realizadas sobre el backend principal, el microservicio de medios, los mecanismos de autenticación y la integración general entre componentes.

Tabla 4.1. Check list de pruebas ejecutadas

Tipo de prueba	Endpoints o componente	Descripción	Resultado esperado
Registro público de usuario	POST /api/public/create/user	Alta de usuario con datos y foto opcional	Usuario creado y respuesta exitosa
Consulta pública de avistamientos	GET /api/public/views	Listado de avistamientos visibles públicamente	Respuesta 200 con listado
Consulta pública de artículos	GET /api/public/articles	Listado de artículos públicos	Respuesta 200 con listado
Creación de avistamiento	POST /api/views	Registro de avistamiento con archivos	Avistamiento creado correctamente
Actualización de avistamiento	PATCH /api/views/{id}	Modificación de datos y archivos asociados	Respuesta de actualización exitosa
Carga de media	POST /api/v1/media/with-media	Subida de imágenes, audios, videos o documentos	Batch procesado correctamente
Actualización de media	PATCH /api/v1/media/update	Reenvío de lote para actualización	Respuesta exitosa

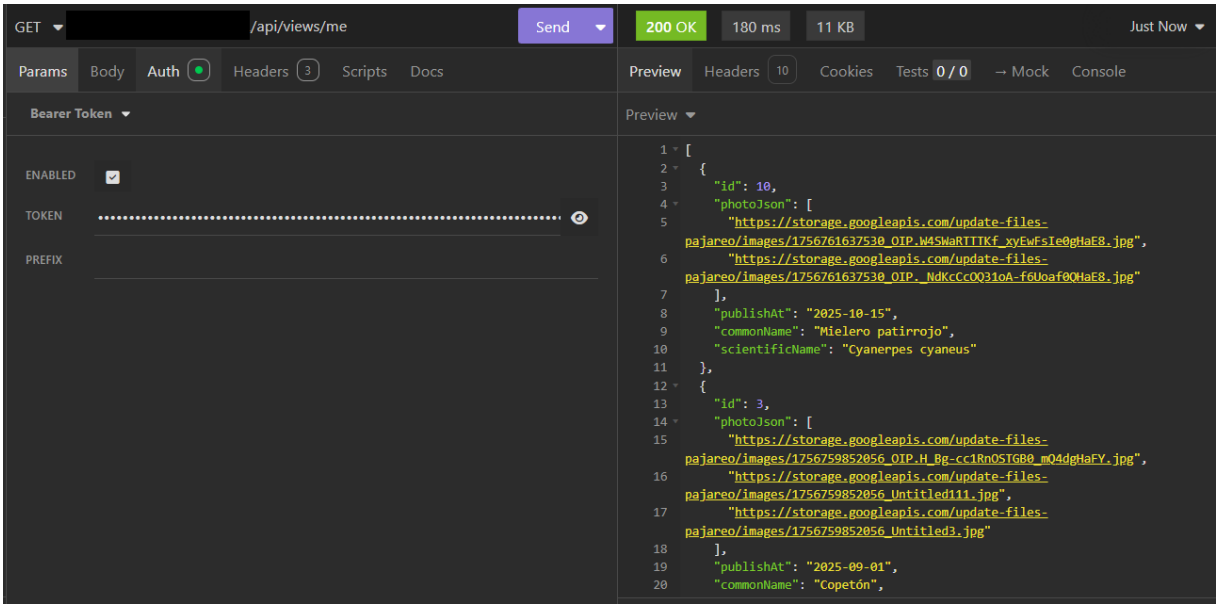
Tipo de prueba	Endpoints o componente	Descripción	Resultado esperado
Borrado de media por URL	DELETE /api/v1/media?url=...	Eliminación de recurso multimedia	Respuesta de borrado correcta
Consulta de usuario por ID	GET /api/keycloak/user/{id}	Obtención de información del usuario autenticado	Respuesta 200
Cambio de contraseña	PUT /api/keycloak/user/password	Cambio de contraseña por correo	Confirmación de operación
Protección de endpoints	Spring Security + Keycloak	Validación de acceso por rol	Restricción correcta según permisos
Swagger / OpenAPI	/swagger-ui.html	Verificación de documentación de endpoints	Documentación visible y navegable

Fuente: Autor

Las siguientes evidencias corresponden a pruebas funcionales realizadas sobre los servicios REST del sistema mediante la herramienta Insomnia. Estas pruebas permitieron verificar el correcto funcionamiento de los endpoints desarrollados, así como la obtención de respuestas exitosas dentro de los parámetros definidos en los indicadores técnicos del proyecto, tales como tiempos de respuesta adecuados, integridad de la información y correcta ejecución de algunas operaciones CRUD, en concordancia con los resultados presentados en la **Tabla 4**.

En la **Figura 3** se valida el endpoint encargado de la consulta y listado de artículos del sistema, ejecutado mediante Insomnia. La respuesta obtenida confirma la correcta comunicación con el backend, la recuperación adecuada de los datos desde la base de datos PostgreSQL y el cumplimiento de los tiempos de respuesta definidos para los endpoints críticos, contribuyendo al logro de los indicadores técnicos establecidos.

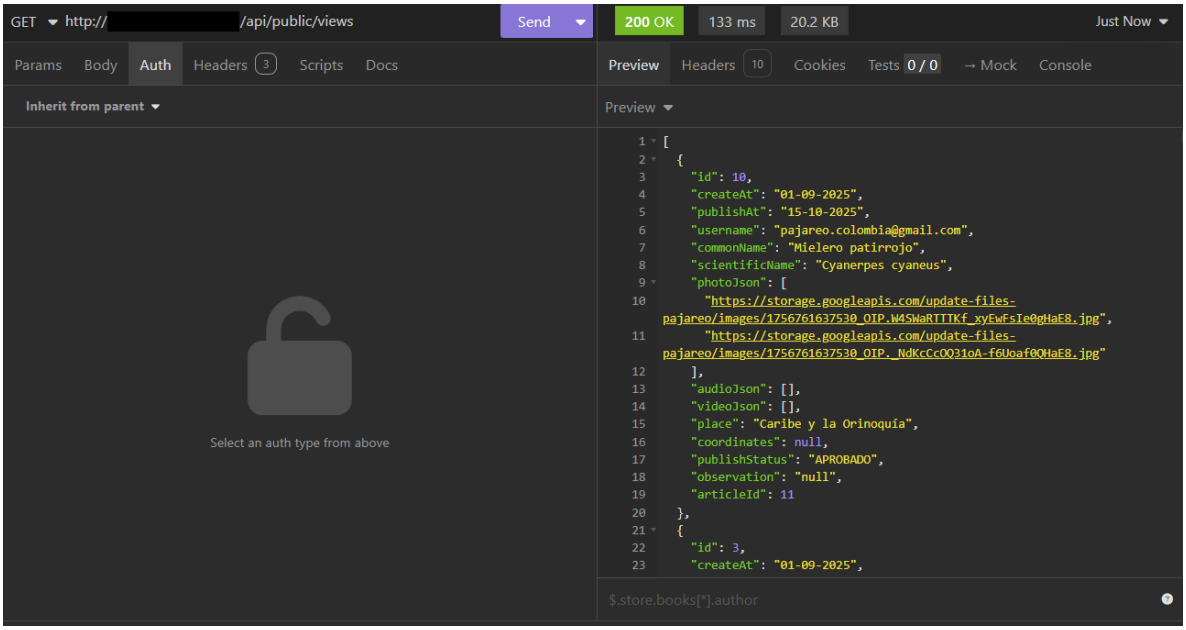
Figura 3. Prueba de listado de artículos



Fuente: Autor

En la **Figura 4** se evidencia corresponde a una prueba funcional del endpoint de listado de avistamientos, realizada desde Insomnia. Los resultados obtenidos demuestran la correcta ejecución de la operación, la consistencia de los datos retornados y la estabilidad del servicio, evidenciando el cumplimiento de los objetivos asociados a la gestión de información y al desempeño del backend.

Figura 4. Prueba de listado de avistamientos

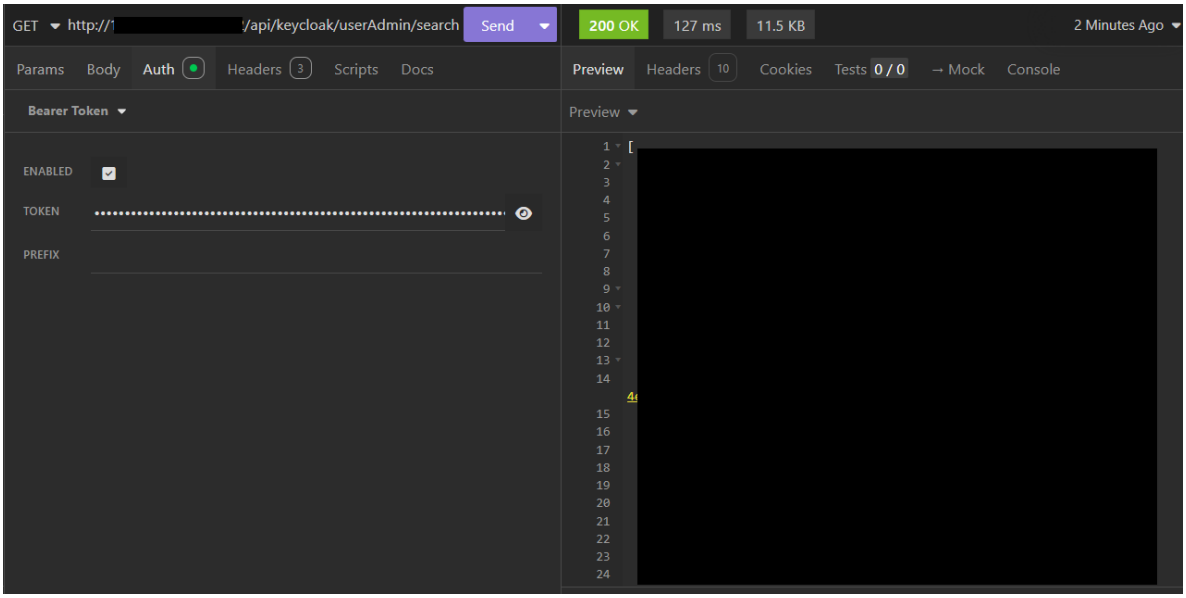



```
[
  {
    "id": 10,
    "createAt": "01-09-2025",
    "publishAt": "15-10-2025",
    "username": "pajareo.colombia@gmail.com",
    "commonName": "Mielero patirrojo",
    "scientificName": "Cyanerpes cyaneus",
    "photoJson": [
      "https://storage.googleapis.com/update-files-pajareo/images/1756761637530_OIP.W4SWaRTTTKf_xyEwfsIe0gHaE8.jpg",
      "https://storage.googleapis.com/update-files-pajareo/images/1756761637530_OIP._NdKcCcOQ31oA-f6Uoaf0QHaE8.jpg"
    ],
    "audioJson": [],
    "videoJson": [],
    "place": "Caribe y la Orinoquía",
    "coordinates": null,
    "publishStatus": "APROBADO",
    "observation": "null",
    "articleId": 11
  },
  {
    "id": 3,
    "createAt": "01-09-2025",
```


Fuente: Autor

En la **Figura 5** se valida el endpoint de consulta de usuarios del sistema, ejecutado mediante Insomnia con credenciales y permisos válidos. La respuesta exitosa confirma la correcta integración del backend con el módulo de gestión de usuarios, así como el adecuado manejo de los datos y el cumplimiento de los criterios de autorización definidos en la arquitectura del sistema.

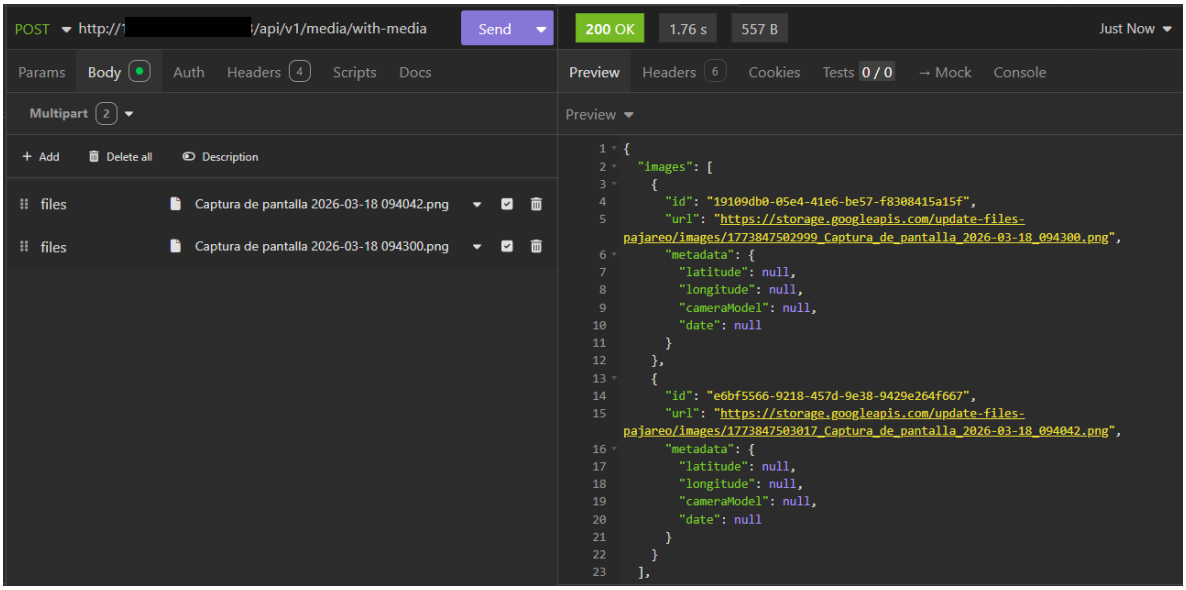
Figura 5. Prueba de listado usuarios



Fuente: Autor

En la **Figura 6** se evidencia la prueba del endpoint de carga de archivos multimedia, realizada mediante Insomnia. El resultado obtenido demuestra la correcta validación del archivo, su procesamiento conforme a los parámetros definidos (tipo y tamaño), y la respuesta exitosa del servicio, respaldando los indicadores técnicos relacionados con el tiempo de procesamiento y la tasa de éxito en cargas válidas establecidos en el proyecto.

Figura 6. Prueba de subida de imagen



Fuente: Autor

6.3. Herramientas y tecnologías utilizadas

El presente anexo describe las principales herramientas y tecnologías empleadas durante el desarrollo de la pasantía, las cuales fueron seleccionadas con el fin de garantizar escalabilidad, seguridad, mantenibilidad y eficiencia en la construcción de la plataforma. Estas tecnologías respaldan las decisiones técnicas adoptadas y permitieron el cumplimiento de los objetivos específicos del proyecto.

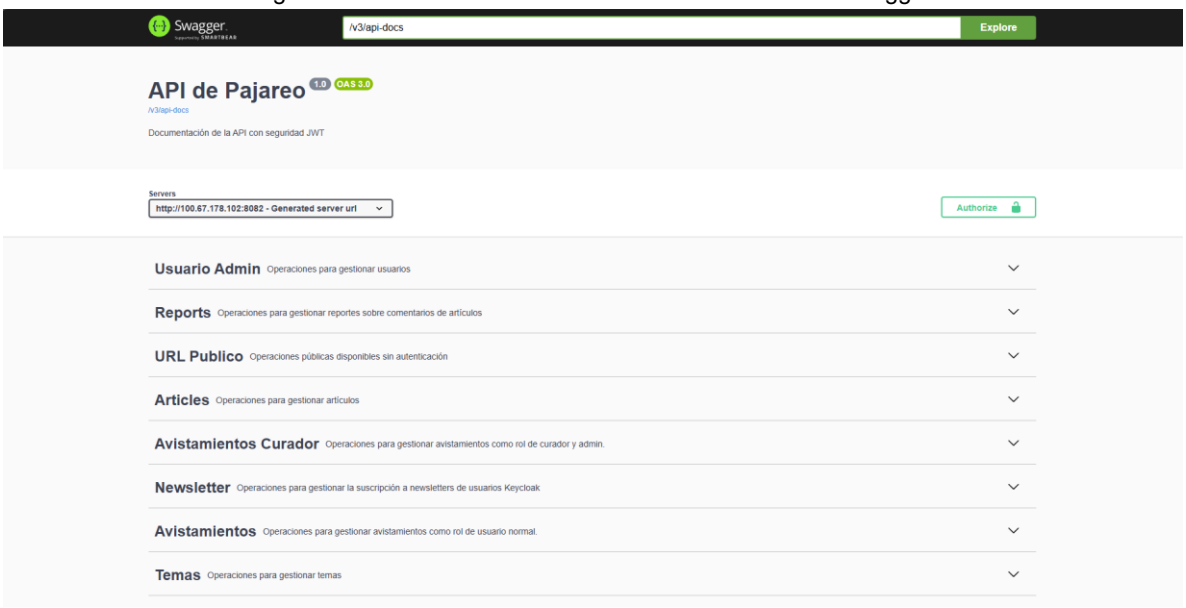
Tabla 5. Herramientas y tecnologías utilizadas

Herramienta	Propósito
Proxmox VE	Virtualización de servidores y VMs
Ubuntu Server	Sistema operativo para servicios
PostgreSQL	Gestión de datos relacionales
Java 17 / Spring Boot	Desarrollo backend
Spring Security	Protección de APIs
Keycloak	Gestión de identidad y accesos
Angular + PrimeNG	Desarrollo frontend
OpenAPI / Swagger	Documentación de APIs
Docker	Ejecución de servicios auxiliares
AWS EC2	Despliegue productivo
AWS S3	Almacenamiento de multimedia
Nginx	Servidor web y proxy inverso
Git	Control de versiones
JUnit / Mockito / Testcontainers	Pruebas automatizadas
OpenProject	Gestión del proyecto
PlantUML / Lucidchart	Diagramación UML
Insomnia	Rest Cliente para pruebas de API

Fuente: Autor

La **Figura 7** corresponde a la documentación de los servicios REST del sistema, generada automáticamente mediante Swagger/OpenAPI. Esta herramienta permitió visualizar de forma estructurada los endpoints disponibles, sus métodos, parámetros y respuestas, facilitando tanto el desarrollo como la validación de los servicios implementados.

Figura 7. Documentación de servicios REST mediante Swagger



Fuente: Autor

La **Figura 7.1** corresponde a la documentación de las URLs públicas del sistema, generada automáticamente mediante Swagger/OpenAPI. Esta herramienta permitió visualizar de forma estructurada los endpoints disponibles, sus métodos, parámetros y respuestas, facilitando tanto el desarrollo como la validación de los servicios implementados.

Figura 7.1. Documentación de URLs públicas del sistema mediante Swagger

Fuente: Autor

La **Figura 8** muestra el uso de la herramienta OpenProject para la gestión y seguimiento de las actividades realizadas durante la pasantía. Esta plataforma permitió organizar tareas, establecer tiempos de ejecución y dar seguimiento al progreso del proyecto de manera estructurada.

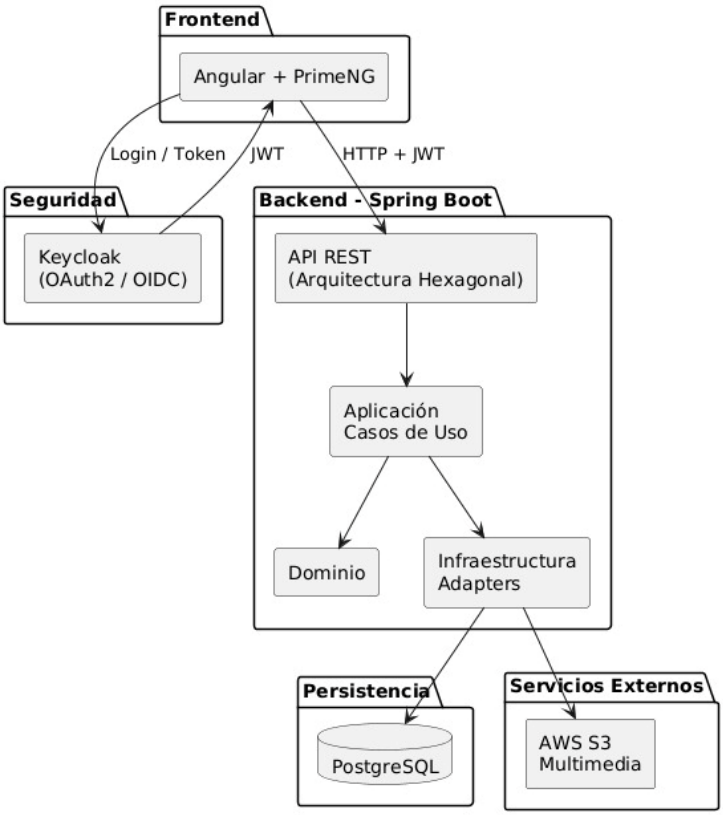
Figura 8. Gestión de actividades del proyecto en OpenProject

Fuente: Autor

6.4. Arquitectura general del sistema

La **Figura 9** presenta la arquitectura general de la plataforma Pajareo Colombia, diseñada bajo el enfoque de arquitectura hexagonal. En ella se evidencia la separación de responsabilidades entre el frontend desarrollado en Angular, la capa de seguridad basada en Keycloak mediante OAuth2/OIDC, el backend implementado en Java y Spring Boot, y los servicios de persistencia y almacenamiento externo. Esta arquitectura facilita la mantenibilidad, escalabilidad y reutilización de componentes, al tiempo que garantiza una correcta gestión de identidad, control de accesos y trazabilidad de la información.

Figura 9. Arquitectura general de la plataforma Pajareo Colombia

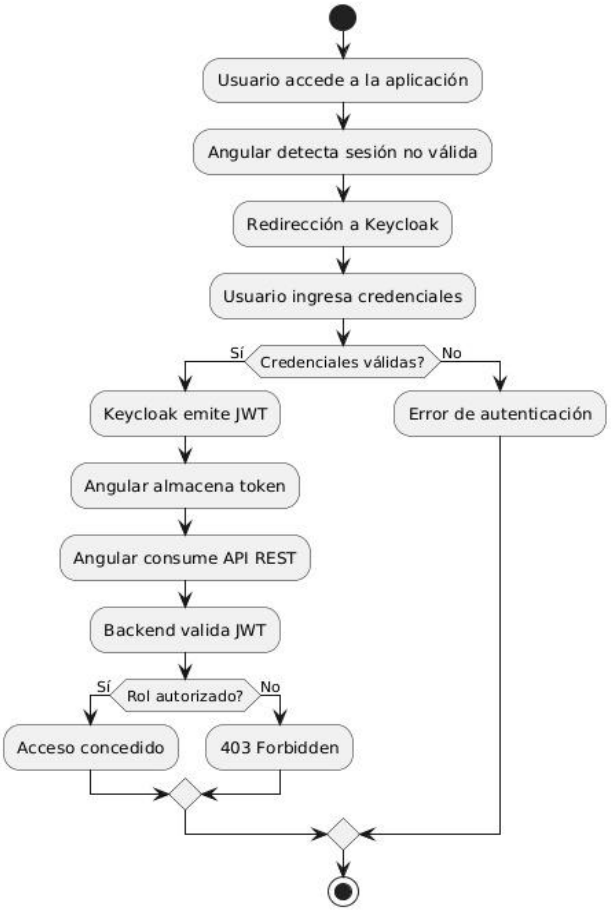


Fuente:Autor

6.5. Diagrama de flujo del sistema

La **Figura 10** ilustra el flujo de autenticación y autorización implementado en la plataforma, basado en el protocolo OAuth2/OpenID Connect mediante Keycloak. El diagrama describe el proceso desde el acceso inicial del usuario, la validación de credenciales, la emisión y gestión de tokens JWT, hasta la autorización de solicitudes hacia el backend. Este flujo asegura el acceso controlado a los recursos del sistema, reforzando los mecanismos de seguridad y cumplimiento de los objetivos definidos para la pasantía.

Figura 10. Flujo de autenticación y autorización con Keycloak



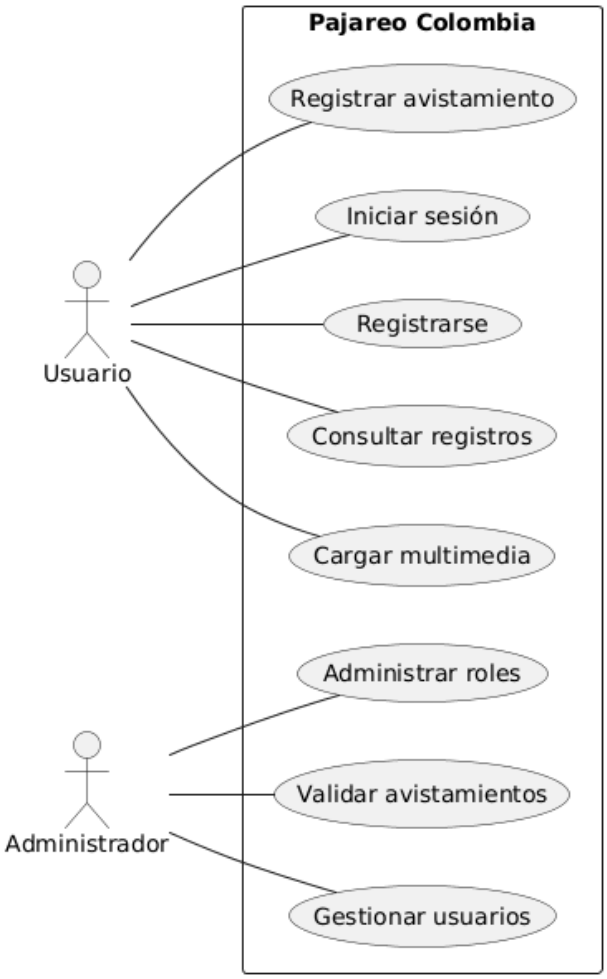
Fuente:Autor

6.6. Diagramas UML

6.6.1. Diagrama de casos de uso

La **Figura 11** muestra el diagrama de casos de uso de la plataforma Pajareo Colombia, en el cual se representan los principales actores del sistema y sus interacciones con las funcionalidades disponibles. Este diagrama permite identificar los procesos clave como la autenticación de usuarios, el registro de avistamientos, la carga de contenido multimedia y la gestión administrativa, proporcionando una visión funcional del sistema desde la perspectiva del usuario.

Figura 11. Caso de uso

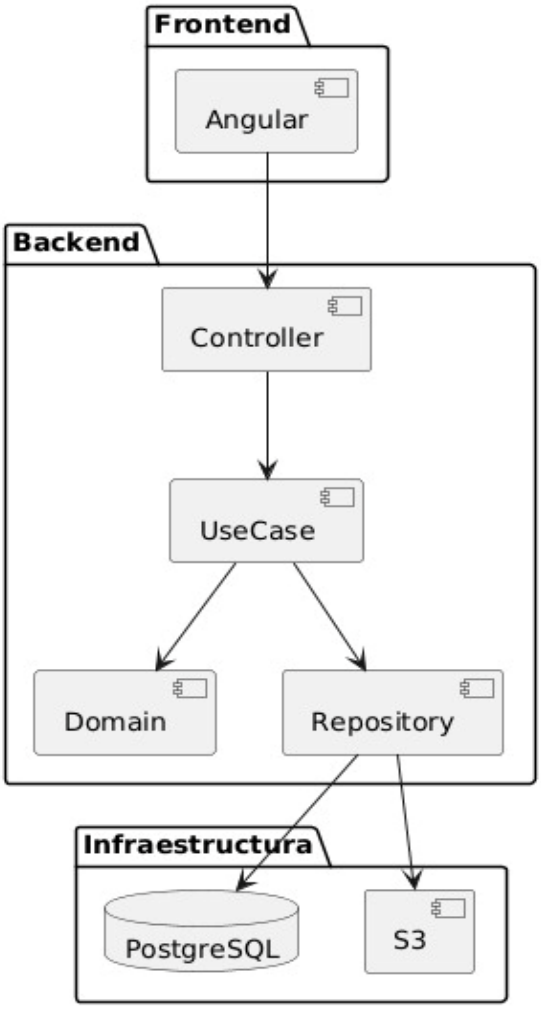


Fuente: Autor

6.6.2. Diagrama de componentes

La **Figura 12** presenta el diagrama de componentes del sistema, donde se detalla la organización interna de los módulos que conforman la plataforma. Se evidencia la interacción entre el frontend, los controladores, los casos de uso, el dominio y los adaptadores de infraestructura, así como su comunicación con los servicios de persistencia y almacenamiento. Este diagrama refuerza la aplicación de la arquitectura hexagonal y la modularidad del backend desarrollado durante la pasantía.

Figura 12. Diagrama de componentes del sistema

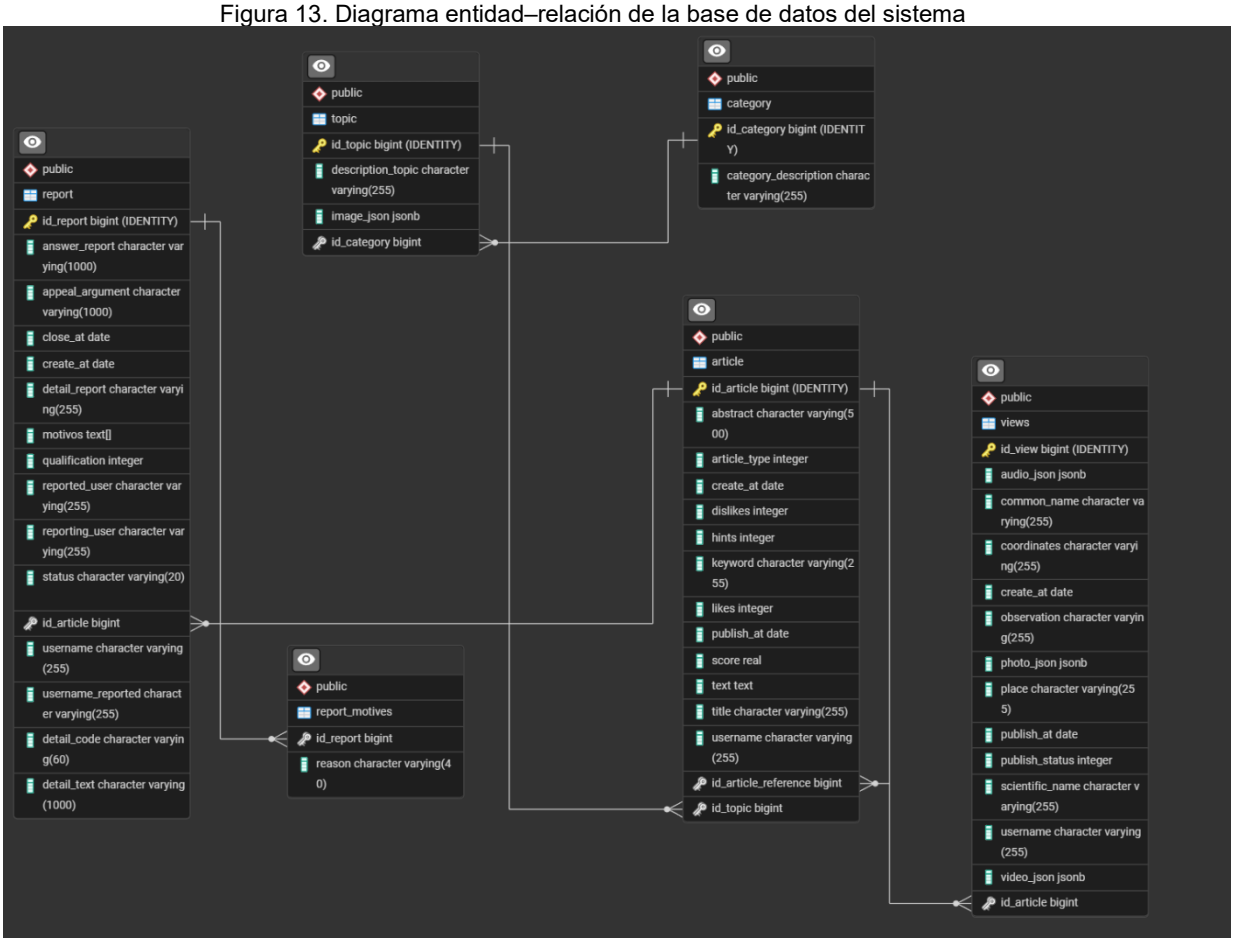


Fuente: Autor

6.6.3. Diagrama entidad–relación

La **Figura 13** presenta el diagrama entidad–relación de la base de datos del sistema Pajareo Colombia, el cual refleja el modelado y la normalización de las entidades principales que soportan la operación del backend. Este diagrama evidencia la correcta estructuración de la información, las relaciones entre entidades y la integridad referencial del sistema, en coherencia con los objetivos definidos para la pasantía.

Figura 13. Diagrama entidad–relación de la base de datos del sistema

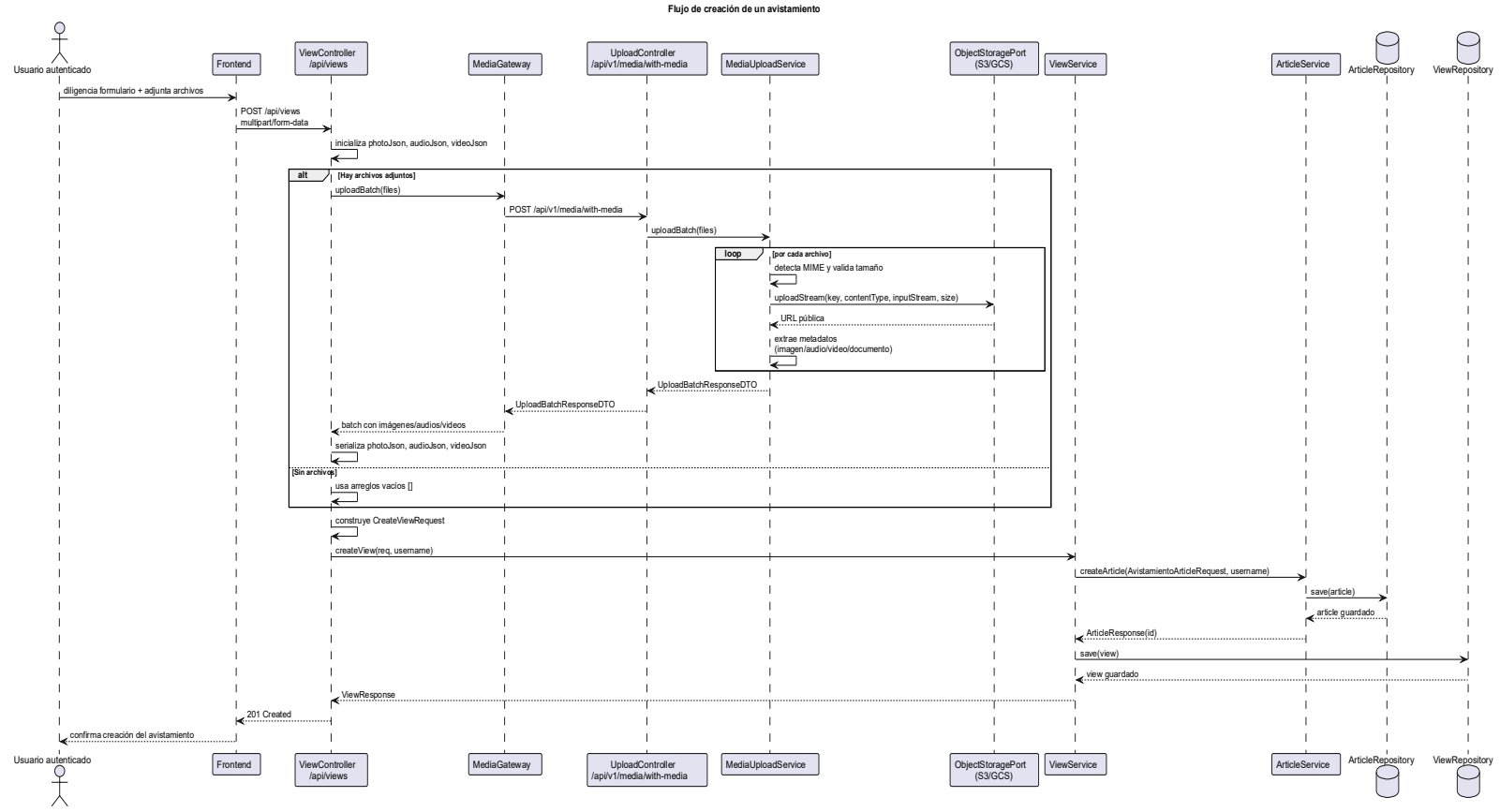


Fuente: Autor

6.6.4. Diagramas de secuencia

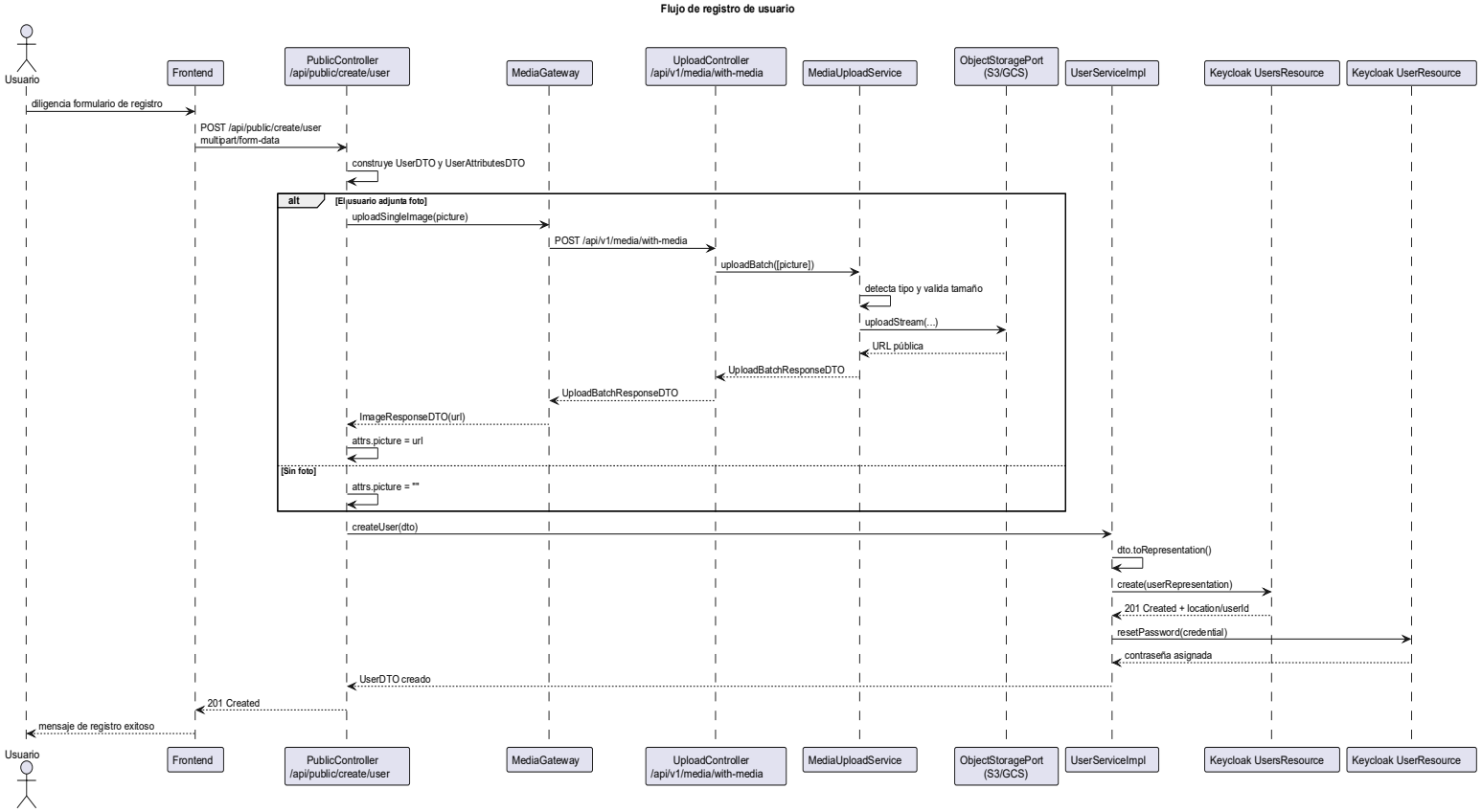
Con el fin de complementar la documentación del sistema y representar de forma dinámica la interacción entre actores, controladores, servicios y repositorios, se incluyen a continuación diagramas de secuencia asociados a dos procesos clave de la plataforma: el registro de usuarios y la creación de avistamientos con carga de archivos multimedia.

Figura 14. Diagrama de secuencia del flujo de creación de un avistamiento



Fuente: Autor

Figura 15. Diagrama de secuencia del flujo de registro de usuarios



Fuente: Autor

6.6.5. Evidencias de documentación técnica

Además de los diagramas UML, se incorporan evidencias adicionales de documentación técnica relacionadas con los contratos de la API, el manejo de errores, la terminología del sistema, la trazabilidad entre objetivos y evidencias, y la especificación del módulo de gestión de usuarios como componente reutilizable. Esta información responde a los lineamientos definidos para el último objetivo específico del proyecto.

La **Tabla 6** presenta ejemplos estandarizados de algunos de los endpoints más representativos del sistema, incluyendo su ruta, método HTTP, tipo de entrada y propósito funcional.

Tabla 6. Ejemplos estandarizados de endpoints implementados

Módulo	Endpoint	Método	Tipo de entrada	Descripción
Registro público de usuario	/api/public/create/user	POST	multipart/form-data	Registra un nuevo usuario con datos básicos y foto opcional
Avistamientos	/api/views	POST	multipart/form-data	Crea un avistamiento con nombre común, nombre científico, lugar y archivos
Consulta pública de avistamientos	/api/public/views	GET	Parámetros simples	Lista avistamientos públicos

Módulo	Endpoint	Método	Tipo de entrada	Descripción
Gestión de usuarios admin	/api/keycloak/userAdmin/create	POST	multipart/form-data	Crea usuarios desde administración
Actualización de usuario	/api/keycloak/user/update/{userId}	PATCH	multipart/form-data	Actualiza perfil y foto
Carga de media	/api/v1/media/with-media	POST	multipart/form-data	Sube lote heterogéneo de imágenes, audios, videos y documentos

Fuente: Autor

6.6.6. Códigos de error

El sistema implementa códigos y respuestas de error orientados a facilitar la interpretación de fallos por parte de clientes y desarrolladores. A continuación, se resumen algunos de los códigos más representativos identificados en el backend principal y en el microservicio de medios.

Tabla 7. Códigos de error del backend principal

Código	Nombre	Descripción funcional
400	BAD_REQUEST	Datos inválidos o solicitud mal formada
401	UNAUTHORIZED	Usuario no autenticado o credenciales inválidas
403	FORBIDDEN	Usuario autenticado sin permisos suficientes
404	NOT_FOUND	Recurso solicitado no existe
409	CONFLICT	Conflicto de datos
1401	VIEW_NOT_FOUND	Avistamiento no encontrado
1404	VIEW_CREATE_FAILED	Error al registrar avistamiento
1405	VIEW_UPDATE_FAILED	Error al actualizar avistamiento

Fuente: Autor

Tabla 8. Códigos de error del backend principal

Código	Nombre	Descripción funcional
4100	FILE_EMPTY	El archivo está vacío
4101	FILE_TOO_LARGE	El archivo supera el tamaño permitido
4102	FILE_TYPE_NOT_ALLOWED	Tipo de archivo no permitido
4103	FILE_UPLOAD_ERROR	Error durante la subida
4104	FILE_STORAGE_ERROR	Error al almacenar el archivo
4105	FILE_NOT_FOUND	Archivo no encontrado
4108	MEDIA_METADATA_ERROR	Error extrayendo metadatos
4200	AUDIO_TOO_LARGE	Audio supera tamaño permitido
4201	AUDIO_TYPE_NOT_ALLOWED	Tipo de audio no permitido

Fuente: Autor

6.6.7. Glosario

Con el fin de unificar la interpretación de los conceptos utilizados en el desarrollo del proyecto, se presenta el siguiente glosario técnico con algunos de los términos más relevantes.

Tabla 9. Glosario de términos técnicos del sistema

Término	Definición
Avistamiento	Registro de observación de un ave con información descriptiva y multimedia
Keycloak	Proveedor de identidad utilizado para autenticación y autorización
JWT	Token firmado utilizado para transportar credenciales y roles
Realm	Espacio de configuración de usuarios, clientes y roles dentro de Keycloak
Endpoint	Ruta expuesta por la API para ejecutar una operación
Arquitectura hexagonal	Estilo arquitectónico que separa dominio, aplicación e infraestructura
Media Gateway	Adaptador del backend principal que integra el microservicio de media
ObjectStoragePort	Puerto que abstrae la persistencia de archivos en servicios de objetos
Multipart	Tipo de solicitud HTTP usada para enviar archivos y campos en una sola petición
DTO	Objeto de transferencia de datos usado para entrada y salida entre capas

Fuente: Autor

6.6.8. Especificación del módulo de gestión de usuarios como componente reutilizable

El módulo de gestión de usuarios fue concebido como un componente reutilizable, apoyado en Keycloak como proveedor de identidad. Su diseño permite registrar, consultar, actualizar y administrar usuarios desde distintos niveles del sistema, manteniendo una separación clara entre lógica de negocio, integración y control de acceso.

Tabla 10. Especificación del módulo de gestión de usuarios como componente reutilizable

Aspecto	Descripción
Nombre del componente	Módulo de gestión de usuarios
Responsabilidad principal	Registro, consulta, actualización, administración de usuarios y gestión de credenciales
Integración base	Keycloak como proveedor de identidad
Endpoints públicos	Registro de usuario, recuperación y cambio de contraseña
Endpoints autenticados	Consulta y actualización de perfil
Endpoints administrativos	Alta, baja, habilitación, deshabilitación y asignación de roles
Entrada principal	Datos básicos de usuario, atributos personalizados y foto opcional
Salidas principales	Usuario creado, usuario actualizado, listado de roles, mensajes de error estandarizados
Dependencias	IUserService, Keycloak Admin API, MediaGateway
Posibilidad de reutilización	Puede integrarse en otros proyectos que utilicen Keycloak y requieran gestión centralizada de usuarios

Fuente: Autor

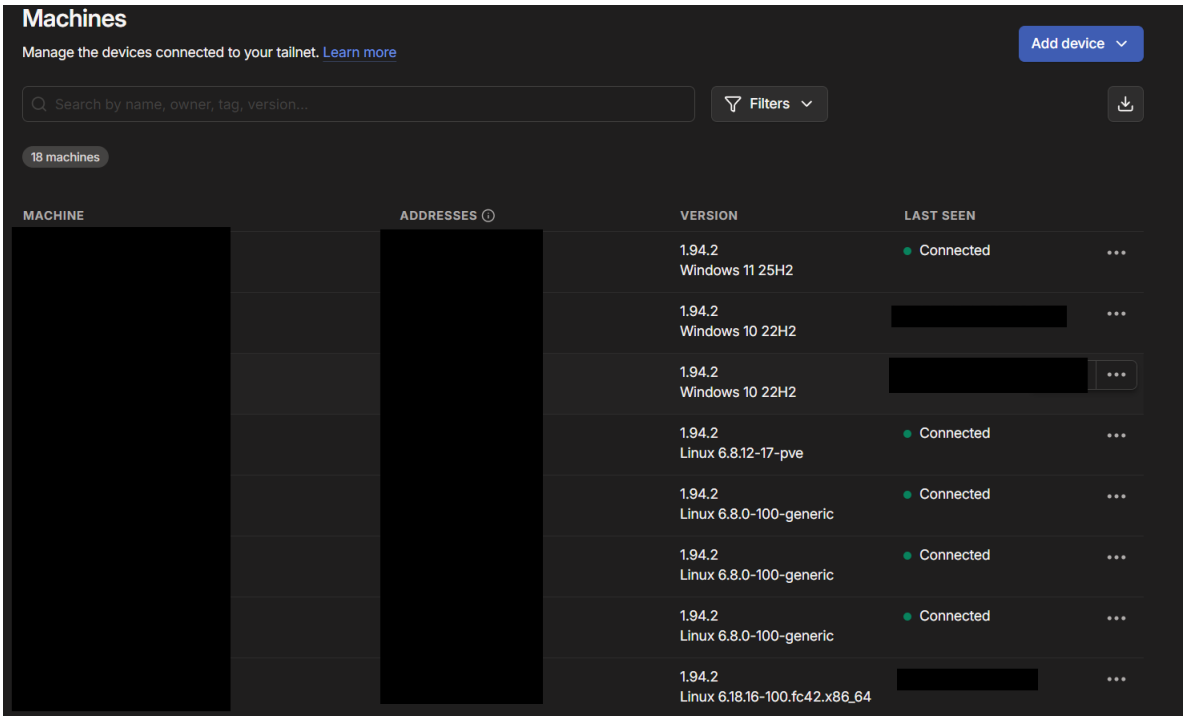
6.7. Evidencias seleccionadas del cronograma de actividades

El presente anexo contiene evidencias seleccionadas de las actividades desarrolladas durante la pasantía, las cuales corresponden a tareas clave del cronograma de trabajo. Estas evidencias permiten verificar la implementación real de la infraestructura, la seguridad del sistema y la arquitectura del backend, respaldando las actividades descritas en la **Tabla 2**.

6.7.1. Evidencia de red segura (Tailscale)

La **Figura 16** muestra la implementación de una red segura para la administración de los servidores del proyecto, realizada mediante la herramienta Tailscale. Esta configuración permitió establecer conexiones privadas cifradas entre los equipos autorizados, garantizando el acceso remoto seguro a la infraestructura utilizada durante la pasantía.

Figura 16. Red segura implementada mediante Tailscale

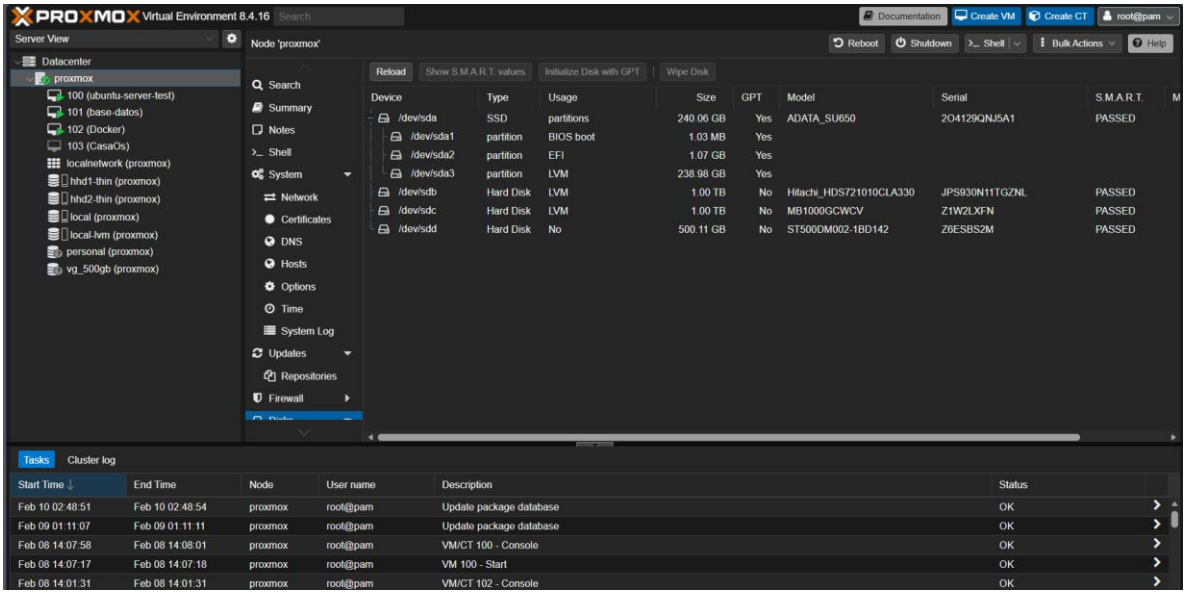


Fuente: Autor

6.7.2. Evidencia de virtualización con Proxmox VE

En la **Figura 17** se observa el servidor de virtualización implementado con Proxmox VE, el cual fue utilizado para la creación y administración de máquinas virtuales destinadas a los entornos de desarrollo y pruebas del proyecto. Esta infraestructura permitió una gestión eficiente de los recursos y facilitó el despliegue de los servicios del sistema.

Figura 17. Servidor de virtualización Proxmox VE con máquinas virtuales activas

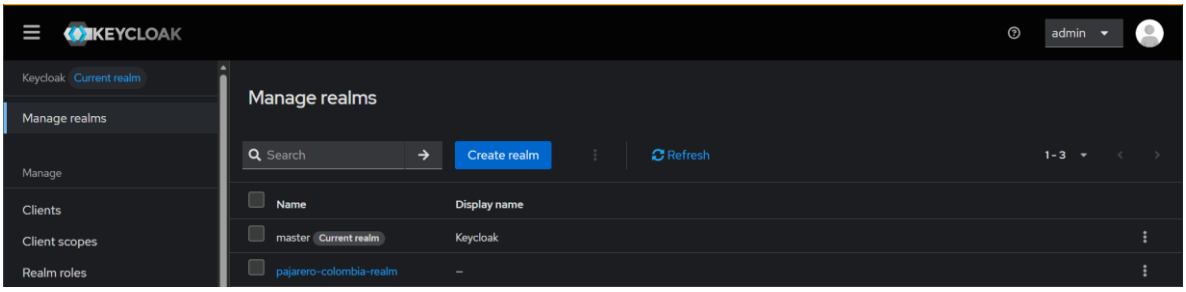


Fuente: Autor

6.7.3. Evidencia de integración con Keycloak

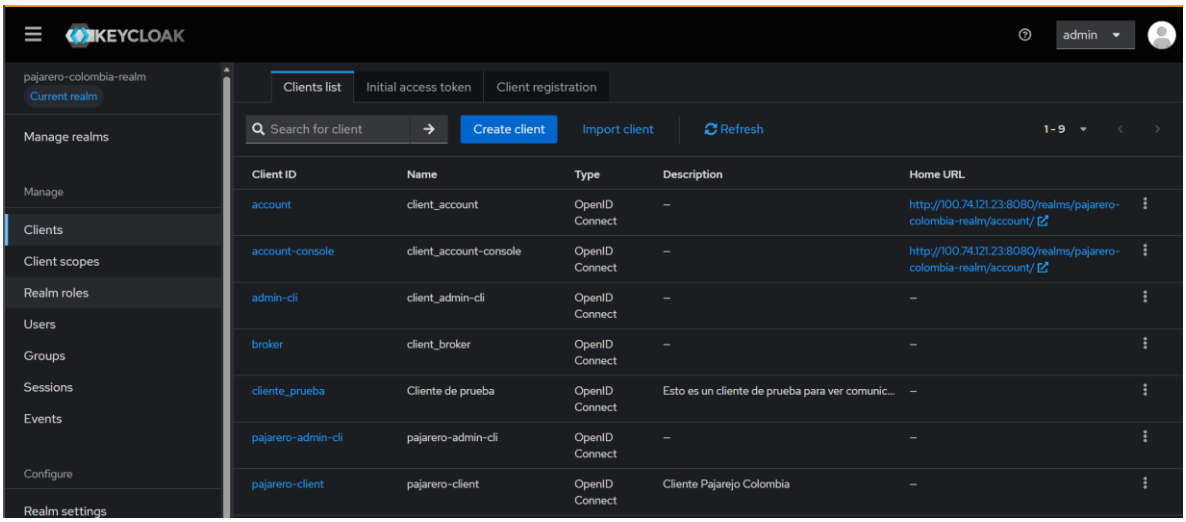
Las siguientes evidencias corresponden a la integración del sistema con Keycloak como proveedor de identidad, donde se configuraron los elementos necesarios para la autenticación y autorización de los usuarios. Esta integración permitió aplicar controles de acceso basados en roles y proteger los servicios del backend conforme a los estándares OAuth2/OpenID Connect.

Figura 18. Realms del Keycloak



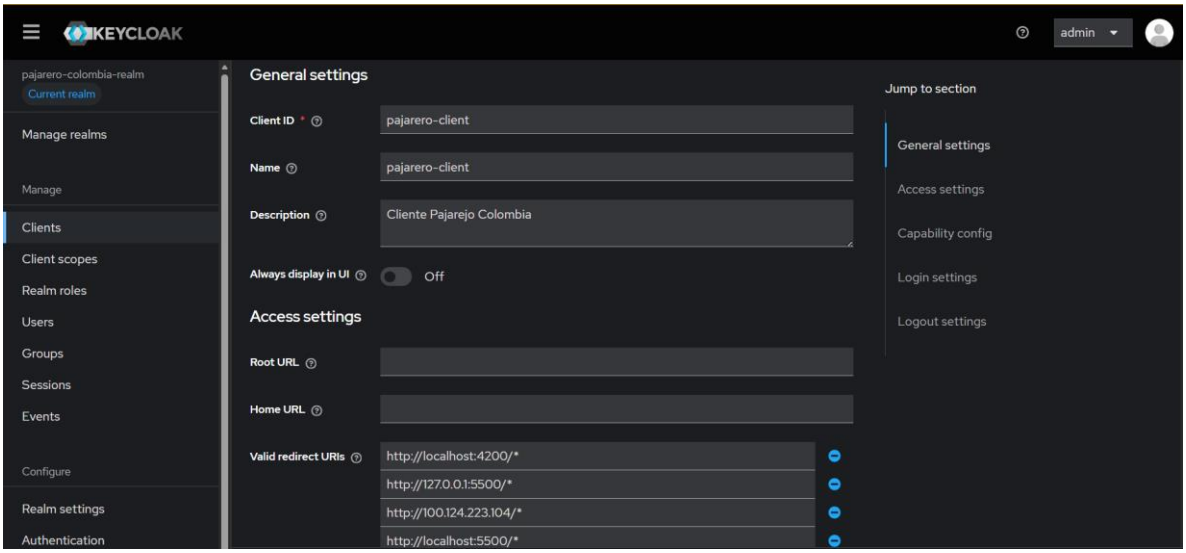
Fuente: Autor

Figura 18.1. Clientes configurados en el proveedor Keycloak



Fuente: Autor

Figura 18.2. Parte de la configuración del cliente de pajareo

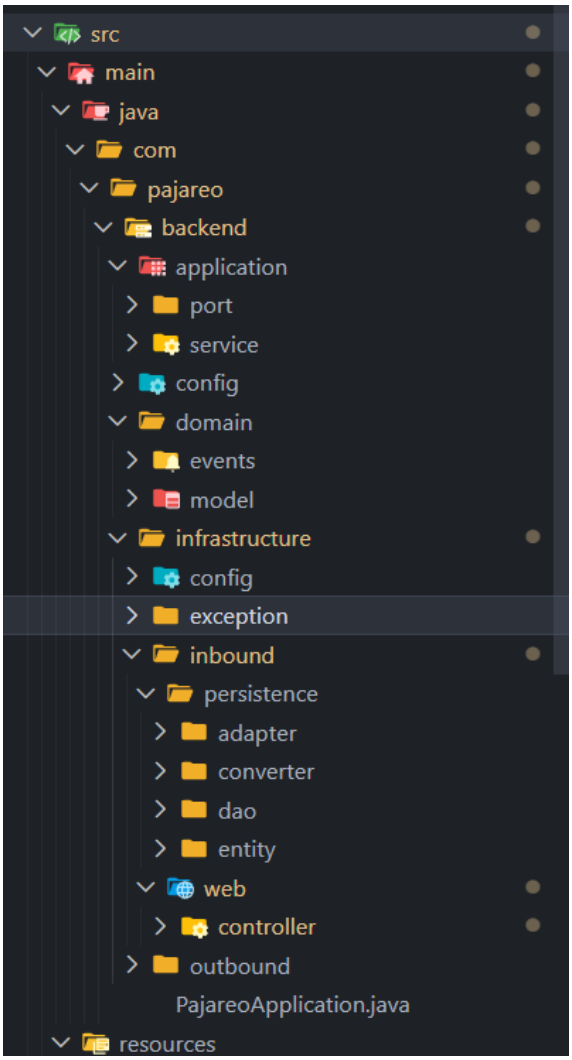


Fuente: Autor

6.7.4. Evidencia de la arquitectura hexagonal del backend

La **Figura 19** muestra la estructura del backend del sistema organizada bajo el enfoque de arquitectura hexagonal. Esta estructura evidencia la separación de responsabilidades entre las capas de dominio, aplicación e infraestructura, facilitando la mantenibilidad, escalabilidad del sistema desarrollado durante la pasantía.

Figura 19. Estructura del backend bajo arquitectura hexagonal

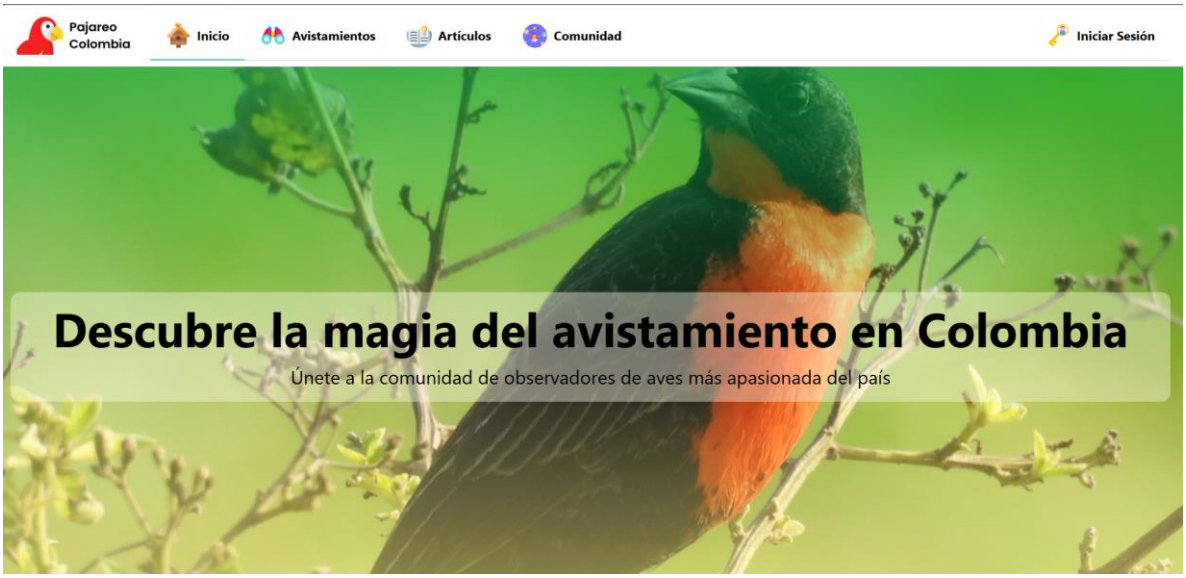


Fuente: Autor

6.7.5. Evidencia del frontend

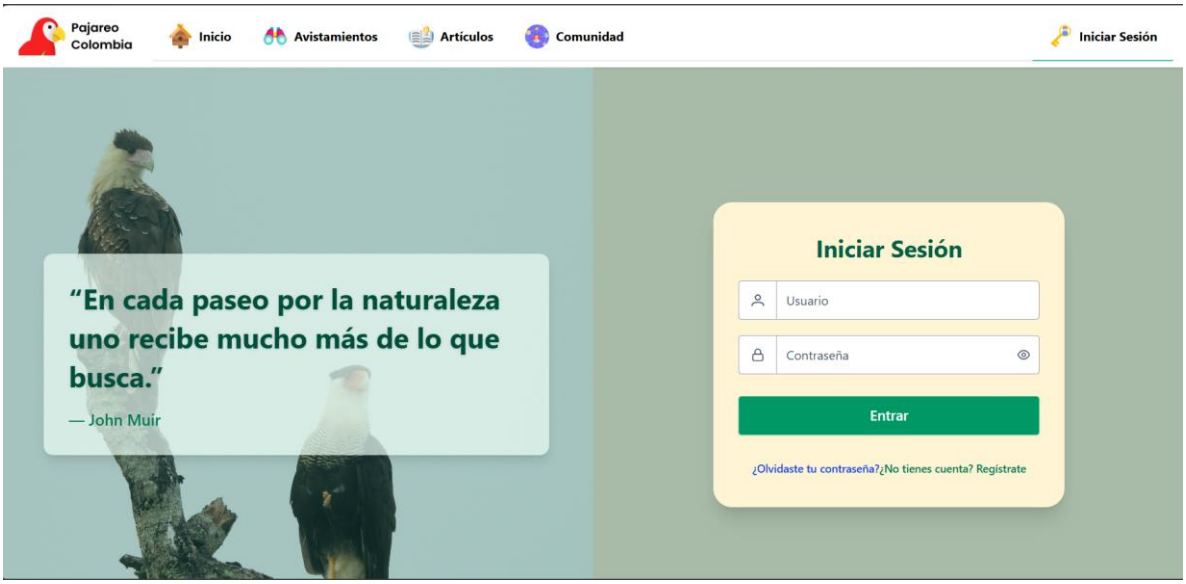
Las **Figuras 20, 21, 22 y** 23 muestra el desarrollo del frontend de la plataforma Pajareo Colombia, implementado en Angular y apoyado en componentes de PrimeNG. Esta capa permitió construir la interfaz de usuario del sistema, integrando formularios, rutas, validaciones y componentes visuales conectados con los servicios del backend.

Figura 20. Interfaz funcional del frontend de la plataforma Pajareo Colombia



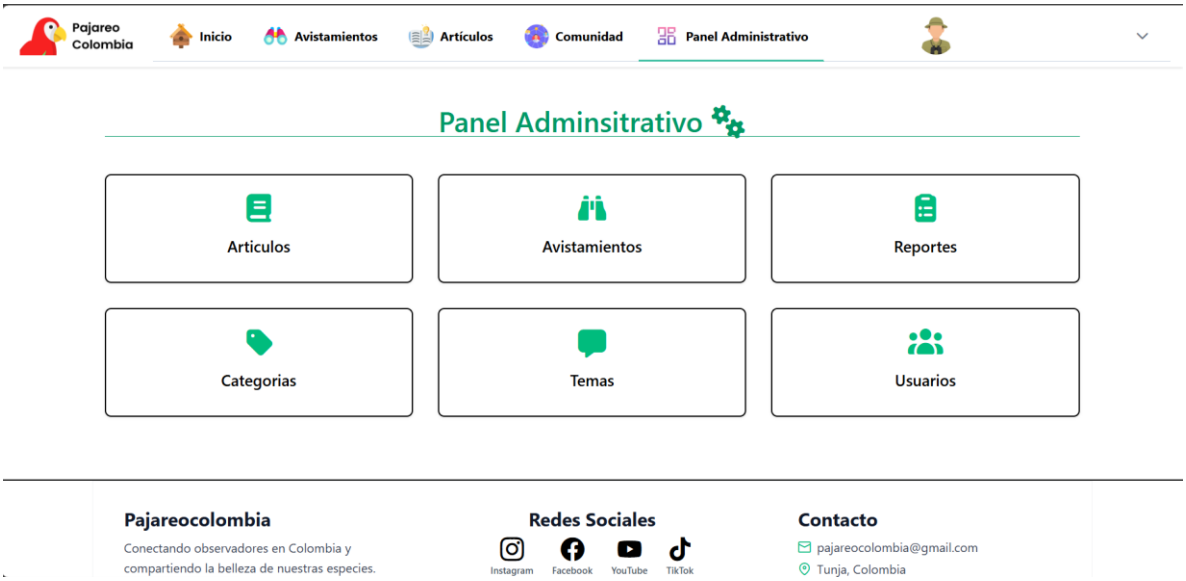
Fuente: Autor

Figura 21. Interfaz inicio de sesión de la plataforma Pajareo Colombia



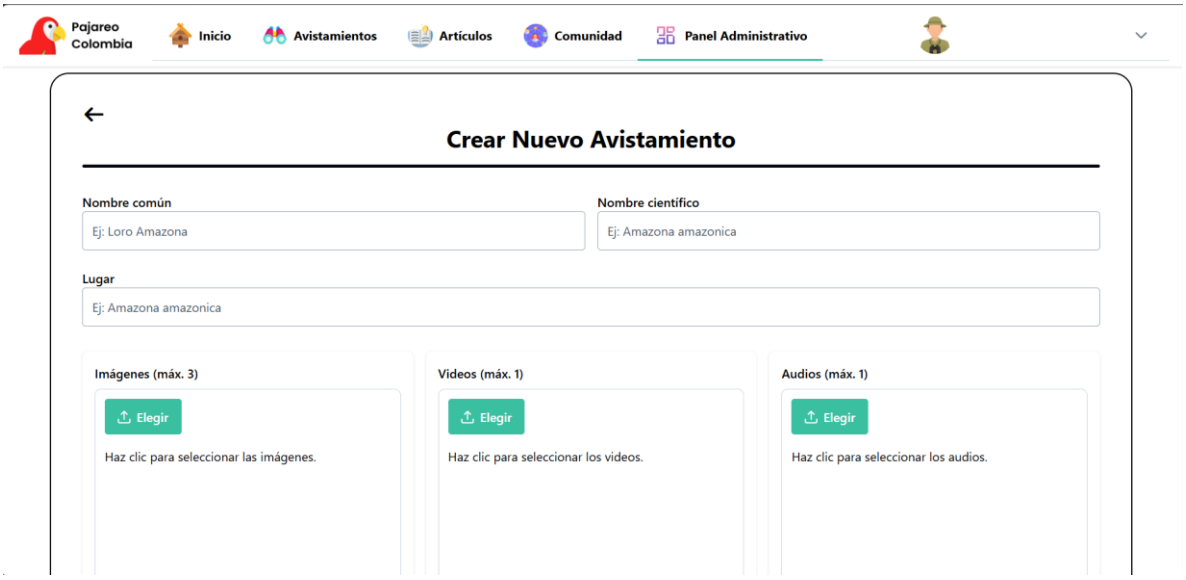
Fuente: Autor

Figura 22. Interfaz panel administrativo de la plataforma Pajareo Colombia



Fuente: Autor

Figura 23. Interfaz creación avistamiento de la plataforma Pajareo Colombia

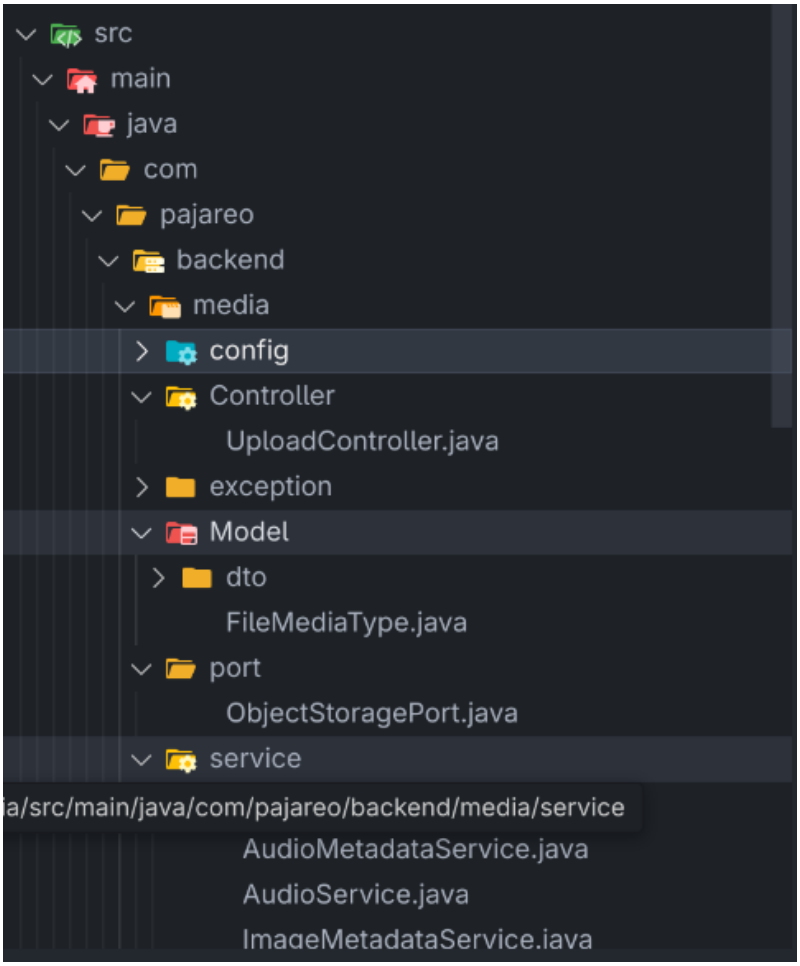


Fuente: Autor

6.7.6. Evidencia del microservicio de medios

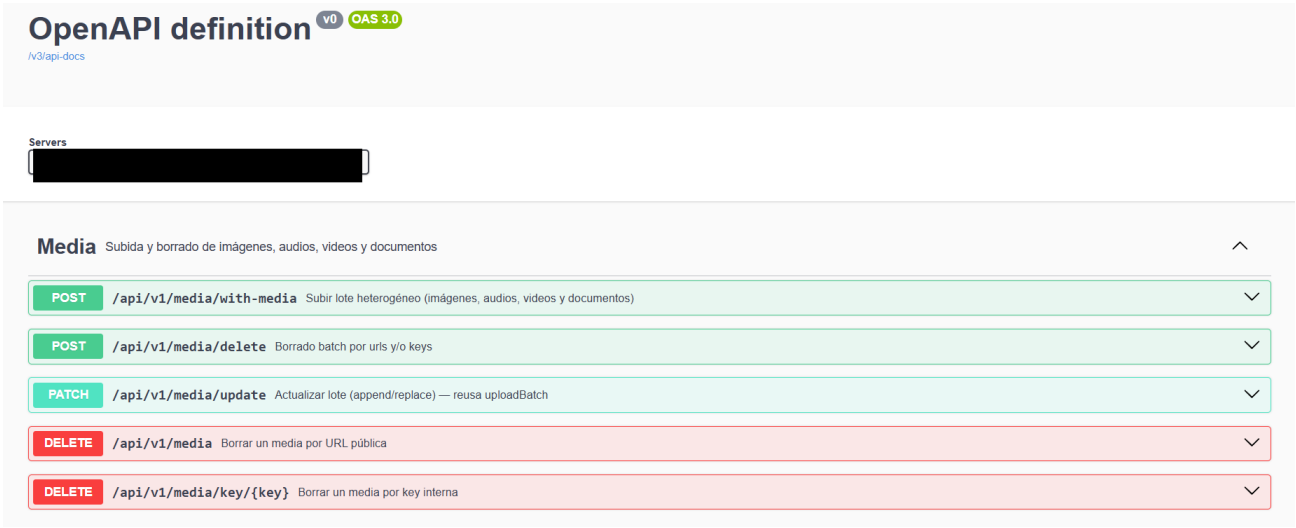
Las **Figuras 24 y 25** corresponde al microservicio de medios desarrollado como componente independiente del sistema, implementado en Spring Boot y expuesto mediante endpoints REST bajo la ruta /api/v1/media. Este microservicio fue diseñado para recibir archivos multimedia, validar su tipo y tamaño, procesarlos según su naturaleza y delegar su almacenamiento en un servicio de objetos desacoplado.

Figura 24. Estructura del microservicio de medios



Fuente: Autor

Figura 25. Documentación REST mediante Swagger servicio multimedia



Fuente: Autor

La **Tabla 11** presenta un resumen de los principales componentes implementados dentro del microservicio de medios y su función dentro de la solución.

Tabla 11. Componentes funcionales del microservicio de medios

Componente	Evidencia en código	Función real implementada
UploadController	Controlador REST	Expone endpoints para carga y borrado de media
MediaUploadService	Servicio principal	Orquesta la validación, clasificación y subida de archivos
ObjectStoragePort	Puerto de salida	Desacopla la lógica del proveedor de almacenamiento
S3StorageService	Implementación de almacenamiento	Gestiona subida y borrado en Amazon S3
TikaMetadataService	Servicio de metadatos	Detecta MIME y extrae metadatos
ImageMetadataService, AudioMetadataService, VideoMetadataService	Servicios especializados	Procesan metadatos por tipo de archivo

Fuente: Autor

6.7.7. Evidencia de la guía de uso de OpenProject

La siguiente evidencia corresponde al manual de uso y capacitación en OpenProject, elaborado como parte de la documentación final del proyecto. Este documento fue diseñado con el propósito de facilitar la adopción de la herramienta dentro de la empresa, dejando orientaciones prácticas sobre módulos funcionales, flujos de trabajo, reuniones, tableros, documentación y buenas prácticas de gestión. El manual completo se encuentra en este enlace.
https://drive.google.com/file/d/14UDSLBFPCUt3mDHPu5j42aVcKNnZ8OsH/view?usp=sharing

Figura 26. Portada del manual

MANUAL DE USO Y CAPACITACIÓN EN OPENPROJECT

Introducción

OpenProject es una plataforma web de **gestión de proyectos colaborativa** que permite planificar, organizar, monitorear y documentar todas las fases de un proyecto desde un entorno centralizado.
Su enfoque combina metodologías **ágiles (Scrum/Kanban)** y **tradicionales (Gantt/PMI)**, lo que la convierte en una herramienta muy flexible para equipos de desarrollo, diseño y gestión.

En OpenProject cada proyecto se estructura en **tareas (Works Packages)**, que pueden organizarse en cronogramas, tableros, versiones, sprints o dependencias. Además, permite almacenar documentación, registrar tiempo, coordinar reuniones, compartir archivos y generar reportes.

Figura 27. Contenido del manual

Arquitectura conceptual de OpenProject

En la siguiente imagen se evidencia la barra lateral de OpenProject, la cual nos permitirá acceder a diferentes funcionalidades, siendo la opción Proyectos la que nos permite ver el listado de proyectos creados en OpenProject

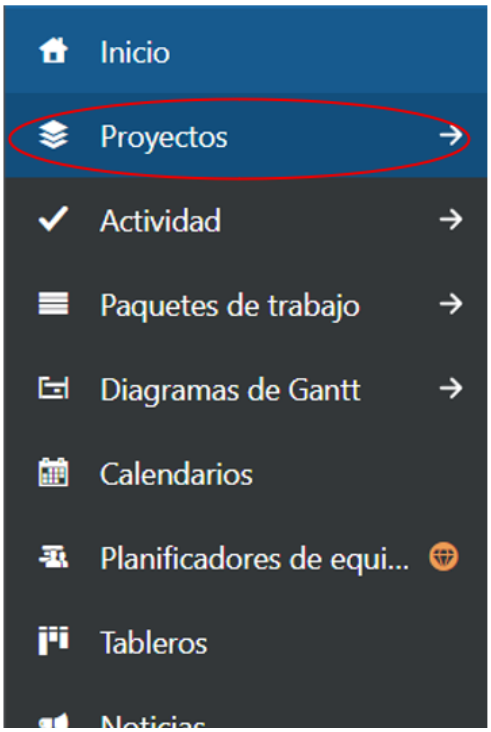
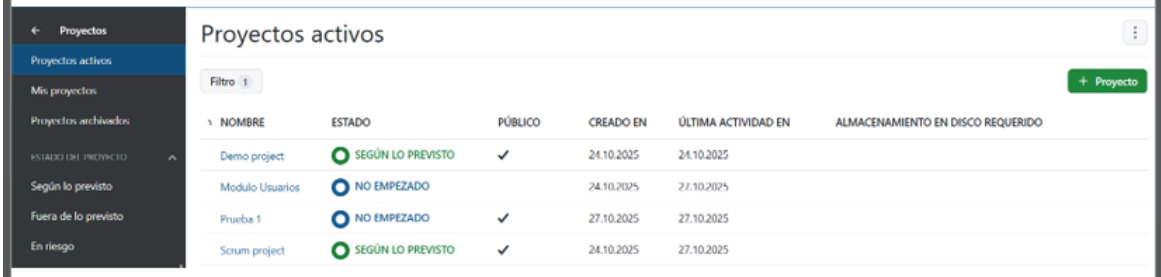
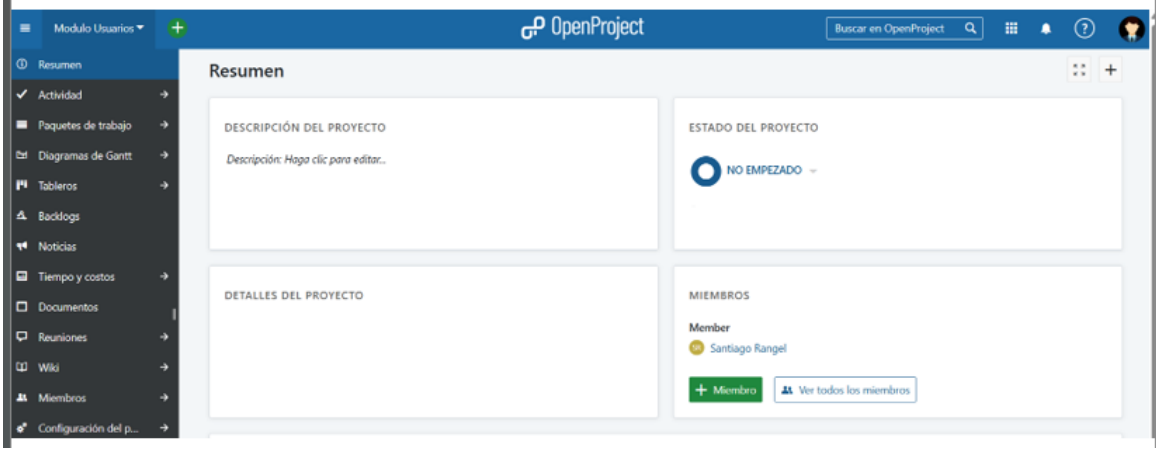


Figura 27.1. Contenido del manual

Una vez en el apartado de proyectos veremos el listado de proyectos creados junto con una barra lateral que nos permitirá ver proyectos activos, ver los proyectos asignados y un filtro según el estado del proyecto.



Una vez demos clic en el nombre de cualquier proyecto, veremos un dashboard del proyecto, junto a una barra lateral donde se despliegan las diferentes funcionalidades que podremos hacer en el proyecto.



7. REFERENCIAS BIBLIOGRÁFICAS

- Spring. (n.d.). *Spring Boot documentation*. Recuperado el 18 de marzo de 2026. URL: https://docs.spring.io/spring-boot/index.html
- Keycloak. (n.d.). *Keycloak documentation*. Recuperado el 18 de marzo de 2026. URL: https://www.keycloak.org/documentation
- PostgreSQL Global Development Group. (n.d.). *PostgreSQL documentation*. Recuperado el 18 de marzo de 2026. URL: https://www.postgresql.org/docs/
- Angular Team. (n.d.). *Angular documentation*. Recuperado el 18 de marzo de 2026. URL: https://angular.dev/
- SmartBear Software. (n.d.). *Swagger documentation*. Recuperado el 18 de marzo de 2026. URL: https://swagger.io/docs/
- PrimeTek. (n.d.). *PrimeNG documentation*. Recuperado el 18 de marzo de 2026. URL: https://primeng.org/
- Docker, Inc. (n.d.). *Docker Engine documentation*. Recuperado el 18 de marzo de 2026. URL: https://docs.docker.com/engine/
- NGINX. (n.d.). *NGINX documentation*. Recuperado el 18 de marzo de 2026. URL: https://nginx.org/en/docs/
- Amazon Web Services. (n.d.). *What is Amazon S3?* Recuperado el 18 de marzo de 2026. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html
- OpenProject GmbH. (n.d.). *OpenProject documentation*. Recuperado el 18 de marzo de 2026. URL: https://www.openproject.org/docs/
- Oracle. (n.d.). *Java documentation*. Recuperado el 18 de marzo de 2026. URL: https://docs.oracle.com/en/java/
- Hardt, D. (2012). *The OAuth 2.0 Authorization Framework (RFC 6749)*. RFC Editor. URL: https://www.rfc-editor.org/rfc/rfc6749
- Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., & Mortimore, C. (2014). *OpenID Connect Core 1.0*. OpenID Foundation. URL: https://openid.net/specs/openid-connect-core-1_0-18.html