

Implementación de ajustes y nuevas funcionalidades en la aplicación web “VariaMos”

Luis Alfonso Giraldo Murillo

Trabajo de grado para optar al título de Ingeniero de Sistemas

Modalidad de Práctica
Trabajo de Grado

Diana Margot Lopez Herrera, Magíster en Ingeniería Industrial

Universidad de Antioquia
Facultad de Ingeniería
Ingeniería de Sistemas
Medellín
2025

Cita	Giraldo-Murillo [1]
Referencia Estilo IEEE (2020)	[1] Giraldo-Murillo, “Implementación de ajustes y nuevas funcionalidades en la aplicación web “VariaMos”” Proyecto de grados, Ingeniería de Sistemas, Universidad de Antioquia Medellín, 2025.



Grupo de Investigación Ingeniería y Tecnologías de las Organizaciones y de la Sociedad (ITOS).

Ingeniería.



Centro de Documentación Ingeniería (CENDOI)

Repositorio Institucional: http://bibliotecadigital.udea.edu.co

Universidad de Antioquia - www.udea.edu.co

El contenido de esta obra corresponde al derecho de expresión de los autores y no compromete el pensamiento institucional de la Universidad de Antioquia ni desata su responsabilidad frente a terceros. Los autores asumen la responsabilidad por los derechos de autor y conexos.

Dedicatoria

A mis padres, hermanos y esposa por todo el apoyo recibido para culminar este proceso.

Agradecimientos

A los profesores Diana y Raúl por el apoyo, acompañamiento y por creer en mis capacidades.

TABLA DE CONTENIDO

RESUMEN..... 9
ABSTRACT..... 10
I. INTRODUCCIÓN..... 11
II. OBJETIVOS..... 12
A. Objetivo general..... 12
B. Objetivos específicos..... 12
III. MARCO TEÓRICO..... 13
Líneas de producto de software..... 13
VariaMos..... 13
Arquitectura Limpia..... 13
Microservicios y API REST..... 13
Autenticación y Autorización..... 14
Bases de Datos PostgreSQL..... 14
Desarrollo Web: React y Bootstrap..... 14
Integración Continua y Despliegue Continuo..... 15
Enfoque Metodológico: Scrum Adaptado..... 16
Reuniones Semanales..... 16
Refinamiento de Requisitos..... 16
Uso de GitHub para Gestión del Proyecto..... 17
Fases del Proyecto..... 17
V. ANÁLISIS DE RESULTADOS..... 18
Desarrollos realizados..... 18
Gestión de usuarios..... 19
Gestión de Roles y Permisos..... 26
Estadísticas de visita y uso..... 30
Monitoreo de Micro-servicios..... 37
VI. CONCLUSIONES Y RECOMENDACIONES..... 39
REFERENCIAS..... 41

LISTA DE TABLAS

TABLA I
ESTRUCTURA DE LA TABLA DE USUARIOS “USER”..... 19
TABLA II
ESTRUCTURA DE LA TABLA DE ROLES “ROLE”..... 27
TABLA III
ESTRUCTURA DE LA TABLA DE PERMISOS “PERMISSION”..... 28
TABLA IV
ESTRUCTURA DE LA TABLA PIVOTE ENTRE ROLES Y USUARIOS “USER_ROLE”.... 29
TABLA V
ESTRUCTURA DE LA TABLA PIVOTE ENTRE ROLES Y PERMISOS
“ROLE_PERMISSION”..... 30
TABLA VI
ESTRUCTURA DE LA TABLA DE PAÍSES “COUNTRY”..... 32
TABLA VII
ESTRUCTURA DE LA TABLA DE VISITAS POR PÁGINA “PAGE_VISIT”..... 32
TABLA VIII
ESTRUCTURA DE LA TABLA DE RESUMEN DE VISITAS “VISITS_SUMMARY”..... 33
TABLA IX
ESTRUCTURA DE LA TABLA DE VISITAS DE USUARIOS “USER_VISIT”.....34
TABLA X
ESTRUCTURA DE LA TABLA DE RESUMEN ANUAL DE VISITAS DE USUARIOS
“YEARLY_VISITS_SUMMARY”.....34

LISTA DE FIGURAS

Fig. 1. Comparación del login de VariaMos.....20
Fig. 2. Nuevo formulario de registro.....21
Fig. 3.Diagrama de secuencia de inicio de sesión mediante Google.....22
Fig. 4. Listado de usuarios.....23
Fig. 5. Página “Mi cuenta”.....24
Fig. 6. Página de detalle de usuario.....25
Fig. 7. Flujo de autorización.....26
Fig. 8. Listado y creación de roles.....27
Fig. 9. Detalle de los roles.....28
Fig. 10. Página de listado de permisos.....29
Fig. 11. Página de métricas.....31
Fig. 12. Página de métricas - Visitas al año por país.....31
Fig. 13. Flujo de recolección de métricas de visitas.....35
Fig. 14. Página de métricas.....37
Fig. 15. Página de monitoreo - Listado de microservicios.....38
Fig. 16. Visualización de logs en la página de monitoreo.....39

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

IEEE	Institute of Electrical and Electronics Engineers
PL	Product Lines (Líneas de producto)
SPL	Software Product Lines (Líneas de producto de software)
JSON	JavaScript Object Notation
JWT	JSON Web Token
API	Interface de programación de aplicaciones (Application Programming Interface)
UdeA	Universidad de Antioquia

RESUMEN

El presente trabajo tuvo como objetivo implementar ajustes y nuevas funcionalidades en la aplicación web "VariaMos", una herramienta que permite hacer ingeniería a líneas de productos de software. Este proyecto abordó la mejora de la arquitectura basada en microservicios y React, asegurando la correcta integración de los cambios realizados con los módulos existentes, como la gestión de lenguajes de modelado y proyectos. La metodología empleada es ágil, basada en Scrum, con entregas incrementales cada semana. A través de este proceso, se mejoró la escalabilidad, seguridad y usabilidad de la plataforma, garantizando que cumpla con los estándares actuales de desarrollo web y los requisitos de los usuarios.

Palabras clave **— VariaMos, Líneas de Producto de Software, Desarrollo web, Ingeniería de Software, Sistemas autoadaptativos.**

ABSTRACT

The objective of this project was to implement adjustments and new functionalities in the web application "VariaMos," a tool that enables software product line engineering. This project focused on improving the microservices-based architecture and React framework, ensuring the proper integration of the implemented changes with existing modules, such as modeling language and project management. The methodology used was agile, based on Scrum, with incremental weekly deliveries. Through this process, the platform's scalability, security, and usability were enhanced, ensuring compliance with current web development standards and user requirements..

Keywords **— VariaMos, Software Product Lines, Web Development, Software Engineering, Auto-adaptive Systems.**

I. INTRODUCCIÓN

VariaMos es una plataforma web que facilita la definición, el modelado y análisis de Líneas de Producto (PL). La herramienta permite a los usuarios diseñar lenguajes de modelado personalizados y ejecutar tareas de razonamiento como verificar la satisfacibilidad de modelos y configurar productos. Desarrollada con tecnologías como TypeScript y React, VariaMos utiliza una arquitectura modular que soporta la creación de modelos complejos a través de interfaces gráficas y archivos JSON [1]. Gracias a su capacidad de razonamiento automatizado y enfoque en la extensibilidad, se ha convertido en una solución valiosa para la ingeniería de PL[2].

Este trabajo de grado tuvo como objetivo implementar nuevas funcionalidades y realizar ajustes a las ya existentes en el aplicativo web VariaMos (Variability Models), una herramienta que ha evolucionado significativamente a lo largo de cuatro versiones. VariaMos comenzó como un plugin para Eclipse, pasó por una versión Java independiente con arquitectura cliente-servidor, luego una versión web basada en Vue.js, y finalmente llegó a su versión más avanzada con una arquitectura de microservicios y un front-end en React [1]. Con esta evolución, surge la necesidad de continuar mejorando el aplicativo, adaptándolo a los nuevos requisitos y asegurando su robustez y escalabilidad. El trabajo se centrará en (i) estudiar la arquitectura y el estado actual del desarrollo de VariaMos, (ii) elucidar y analizar los nuevos requisitos y las nuevas funcionalidades de cada módulo, y (iii) diseñar e implementar soluciones que sean compatibles con la versión actual del aplicativo. Esto incluye evaluar la factibilidad de los cambios propuestos, asegurar que se integren de manera coherente con la arquitectura existente y documentarlos correctamente.

II. OBJETIVOS

A. Objetivo general

Implementar nuevas funcionalidades y realizar ajustes a funcionalidades existentes en el aplicativo web VariaMos.

B. Objetivos específicos

- Estudiar la arquitectura y estado actual del desarrollo del aplicativo VariaMos versión 4.0.
- Comprender los requisitos de ajuste en cada uno de los módulos de la aplicación web VariaMos, su impacto y factibilidad de implementación.
- Comprender los requerimientos de nuevas funcionalidades en cada uno de los módulos de la aplicación web VariaMos y su factibilidad de implementación.
- Diseñar una solución que satisfaga las funcionalidades nuevas y los ajustes, de manera que sean funcionalmente compatible y arquitectónicamente orquestada con la versión actual de la aplicación web Variamos.
- Desarrollar la solución diseñada para las funcionalidades nuevas y los ajustes, incorporando altos estándares de calidad.
- Desarrollar la solución diseñada para las funcionalidades nuevas y los ajustes, incorporando altos estándares de calidad.
- Completar la documentación del aplicativo web VariaMos, con la información técnica y de uso de las funcionalidades implementadas.

III. MARCO TEÓRICO

Líneas de producto de software

Las Líneas de Productos de Software (SPL) representan un enfoque estratégico para la creación de familias de sistemas de software, permitiendo la reutilización efectiva de activos a través de un conjunto común de características. Este enfoque se basa en la identificación y gestión de variaciones dentro de un dominio específico, lo que facilita el desarrollo ágil y eficiente de múltiples productos derivados de una misma base. Según Clements y Northrop, las SPL permiten a las organizaciones responder rápidamente a las cambiantes demandas del mercado, optimizando recursos y reduciendo costos de desarrollo al aprovechar las sinergias entre productos relacionados [3]. Esta metodología no solo mejora la calidad del software, sino que también promueve la sostenibilidad a largo plazo de las soluciones de software, asegurando que los productos sean escalables y adaptables a futuras necesidades [3].

VariaMos

VariaMos es una herramienta avanzada para el modelado de líneas de productos que ha evolucionado significativamente desde su versión inicial como plug-in de Eclipse hasta convertirse en una aplicación web moderna basada en microservicios [1]. La última versión, 4.0, se ha rediseñado utilizando TypeScript, React y una arquitectura de microservicios, lo que ha permitido mejorar la extensibilidad, mantenibilidad y capacidad de razonamiento [1].

Arquitectura Limpia

La Arquitectura Limpia, propuesta por Robert C. Martin, es fundamental en el diseño tanto del frontend como del backend de VariaMos. Esta arquitectura promueve la separación de responsabilidades y la independencia de las capas del sistema, facilitando la mantenibilidad y escalabilidad [4]. La arquitectura limpia se aplica en el backend para organizar los microservicios y en el frontend para estructurar los componentes de React de manera modular y reutilizable.

Microservicios y API REST

En VariaMos 4.0, el backend está basado en una arquitectura de microservicios, lo que permite que los distintos componentes del sistema sean independientes y escalables. Los

servicios REST son expuestos a través de APIs, facilitando la comunicación entre el frontend y el backend [5]. Estas APIs permiten al frontend interactuar con los servicios de manera eficiente y estandarizada.

Autenticación y Autorización

La seguridad del sistema se gestiona mediante JWT (JSON Web Tokens), que se utiliza para la autenticación y autorización segura. OAuth 2.0 proporciona un marco para la autorización, mientras que JWT asegura la transmisión de información de usuario de manera confiable [6][7]. La autorización se gestiona mediante Role-Based Authorization y Permission-Based Authorization, permitiendo un control detallado sobre el acceso a los recursos y funcionalidades del sistema [6].

El algoritmo bcrypt se utiliza para asegurar las contraseñas almacenadas en la base de datos. bcrypt es un algoritmo de hash diseñado para proteger contraseñas contra ataques de fuerza bruta mediante la incorporación de un "salt" y la capacidad de ajustar la complejidad del hash [8].

Bases de Datos PostgreSQL

PostgreSQL se utiliza como el sistema de gestión de bases de datos para VariaMos, ofreciendo soporte avanzado para datos JSON y JSONB. Esta capacidad es crucial para manejar la complejidad de los modelos de datos utilizados en los lenguajes de modelado [9]. PostgreSQL permite realizar consultas eficientes y manipular datos semiestructurados, lo que es esencial para la flexibilidad y el análisis de los modelos en VariaMos.

Desarrollo Web: React y Bootstrap

React se empleó para construir la interfaz de usuario de VariaMos, aprovechando su capacidad para crear componentes reutilizables y mejorar la experiencia del usuario [10]. El frontend se comunica con el backend a través de las APIs REST, facilitando una integración fluida y dinámica.

Bootstrap fue utilizado para diseñar la interfaz de usuario, proporcionando un marco de CSS que facilita la creación de interfaces accesibles y responsivas. Su uso acelera el desarrollo del

frontend al ofrecer componentes y estilos predefinidos que mejoran la experiencia de usuario (UX) [11].

Integración Continua y Despliegue Continuo

En el desarrollo de VariaMos, se implementa un proceso de integración continua (CI) y despliegue continuo (CD) para asegurar la calidad del código y facilitar la entrega de nuevas versiones. La integración continua permite integrar cambios en el código frecuentemente y desencadenar pruebas automáticas, lo que asegura que los nuevos cambios no introduzcan errores en el sistema [12]. GitHub Actions se utiliza para automatizar este proceso, permitiendo la ejecución de pruebas, construcción del código y otras tareas al detectar eventos como commits o pull requests.

GitHub Actions es clave en el flujo de CI/CD de VariaMos, ya que permite gestionar la automatización de todo el proceso, desde la integración hasta el despliegue. Gracias a su capacidad de personalización, GitHub Actions permite definir flujos de trabajo que realizan tareas como pruebas automáticas y despliegue en entornos controlados, lo que facilita la entrega continua de nuevas funcionalidades con mayor confiabilidad y rapidez [13].

IV. METODOLOGÍA

Enfoque Metodológico: Scrum Adaptado

La metodología adoptada para este proyecto es una versión adaptada de Scrum, un marco ágil ampliamente utilizado en el desarrollo de software. Scrum permite trabajar de manera colaborativa e incremental, dividiendo el proyecto en ciclos llamados sprints, con entregas parciales que permiten una retroalimentación continua [14]. Sin embargo, en este proyecto no se realizaron las reuniones diarias típicas de Scrum, conocidas como daily stand-ups. En su lugar, se llevaron a cabo reuniones semanales cada viernes, en las que se discutieron los avances, se revisaron los resultados y se abordaron problemas o bloqueos que puedan haber surgido [15].

Esta adaptación permite mantener la estructura iterativa y colaborativa de Scrum, ajustada a las necesidades y disponibilidad del equipo, evitando las reuniones diarias y enfocándose en las reuniones semanales como el único punto de contacto fijo.

Reuniones Semanales

Las reuniones semanales, que se realizaron cada viernes, servían como puntos clave para evaluar el progreso del proyecto y planificar el trabajo para la siguiente semana. En estas reuniones se mostró el trabajo completado y se discutieron ajustes o clarificaciones en los requisitos cuando fue necesario. Este enfoque aseguró que, aunque no hubiera revisiones diarias, se mantuviese el control y la visibilidad del proyecto [16].

Refinamiento de Requisitos

En caso que durante las reuniones semanales se detectaran ambigüedades o se requiriera mayor precisión en los requisitos, se buscaron espacios adicionales para realizar reuniones virtuales con los responsables del proyecto. El refinamiento de requisitos permite clarificar los objetivos y ajustar el backlog de tareas de manera dinámica, asegurando que el producto final cumpla con las expectativas [17].

Uso de GitHub para Gestión del Proyecto

GitHub fue la herramienta principal para el seguimiento de actividades y centralización del código. Se utilizó para gestionar las tareas y avances del proyecto a través de repositorios organizados por microservicios y frontend, similar al concepto de microfrontends. Esto facilitó una gestión modular y escalable, permitiendo que cada componente se desarrolle de manera independiente pero con una integración clara en el proyecto global [18].

Fases del Proyecto

1. **Fase de Planificación:** Para definir los objetivos iniciales y el backlog de tareas, desglosando el proyecto en sprints semanales.
2. **Frase de diseño:** Se llevó a cabo una fase de diseño. El diseño se compartió con el equipo cuando éste tenía un gran impacto, con el fin de obtener retroalimentación y realizar las mejoras correspondientes. Algunos ejemplos de esto incluyen los módulos de usuario, permisos y roles.
3. **Desarrollo Iterativo:** Se realizó un ciclo semanal de desarrollo, con entregas incrementales de productos funcionales en cada reunión.
4. **Integración Final y Pruebas:** Al finalizar el desarrollo, se integraron todos los componentes y se realizaron pruebas de funcionamiento.

V. ANÁLISIS DE RESULTADOS

Desarrollos realizados

Los desarrollos realizados y sus respectivos requerimientos se listan a continuación:

1. Módulo de gestión de usuarios. Requerimientos:
    a. Crear Usuarios: Permitir la creación de nuevos usuarios con roles específicos (admin, usuario, invitado).
    b. Editar Usuarios: Permitir la edición de la información de los usuarios, incluyendo roles, permisos y datos personales.
    c. Listar Usuarios: Permitir el listado de la información de los usuarios, incluyendo roles, permisos y datos personales.
    d. Eliminar Usuarios: Permitir la eliminación de usuarios.
    e. Activar/Desactivar Usuarios: Permitir la activación o desactivación de cuentas de usuarios.
2. Módulo de gestión de permisos y roles. Requerimientos:
    a. Crear Roles: Permitir la creación de nuevos roles de usuario con permisos personalizados.
    b. Editar Roles: Permitir la edición de roles existentes.
    c. Asignar Roles a Usuarios: Permitir la asignación de roles a usuarios específicos.
    d. Gestionar Permisos: Permitir la gestión de permisos específicos para cada rol.
3. Módulo de estadísticas de visita y uso. Requerimientos:
    a. Monitorear Visitas: Mostrar estadísticas de visitas a la herramienta (número de visitas, visitantes únicos, páginas más visitadas, etc.) por mes y por año.
    b. Monitorear Uso: Mostrar estadísticas de uso de la herramienta (usuarios activos, proyectos activos, lenguajes más utilizados, etc.) por mes y por año.
4. Módulo monitoreo de Micro-servicios. Requerimientos:

a. Listar Micro-servicios: Mostrar una lista de los micro-servicios de VariaMos.
b. Estado de Microservicios: Mostrar el estado actual de cada micro-servicio (activo, inactivo, en error).
c. Iniciar/Detener Micro-servicios: Permitir iniciar y detener microservicios específicos.
d. Reiniciar Micro-servicios: Permitir reiniciar micro-servicios específicos.

Gestión de usuarios

La gestión de usuarios requería funciones para registrar, editar, listar, eliminar, desactivar y activar usuarios. Esto se logró mediante la interacción entre las entidades de usuario, roles y permisos, cuyas estructuras y relaciones se detallan en las Tablas I, II, III, IV y V.

TABLA I
ESTRUCTURA DE LA TABLA DE USUARIOS “USER”

Columna	**Descripción**	**Tipo de dato**
id	Identificador del usuario (PK)	text (uuid)
user	Nombre de usuario	text
name	Nombre completo del usuario	text
email	Correo electrónico del usuario	text
password	Contraseña del usuario	text
is_enabled	Indica si el usuario está habilitado	boolean
is_deleted	Indica si el usuario está eliminado	boolean
created_at	Fecha de creación del usuario	timestamp with time zone
last_login	Última fecha de inicio de sesión	timestamp with time zone
country_code	Código del país del usuario	character varying

La aplicación originalmente solo permitía el inicio de sesión a través del sistema OAuth de Google, pero se ha mejorado para incluir la opción de registrarse e iniciar sesión con usuario y contraseña. Se han implementado validaciones en los formularios de registro e inicio de sesión

para garantizar la calidad de los datos ingresados y para informar a los usuarios sobre cualquier error.

Es especialmente importante que la contraseña cumpla con los requisitos de seguridad, como tener una longitud mínima de 8 caracteres e incluir letras mayúsculas y minúsculas, números y caracteres especiales.

Estas validaciones son fundamentales para prevenir ataques desde el frontend y para proteger los servicios web del backend que gestionan el inicio de sesión y el registro, evitando el envío de formularios con datos faltantes o incorrectos.

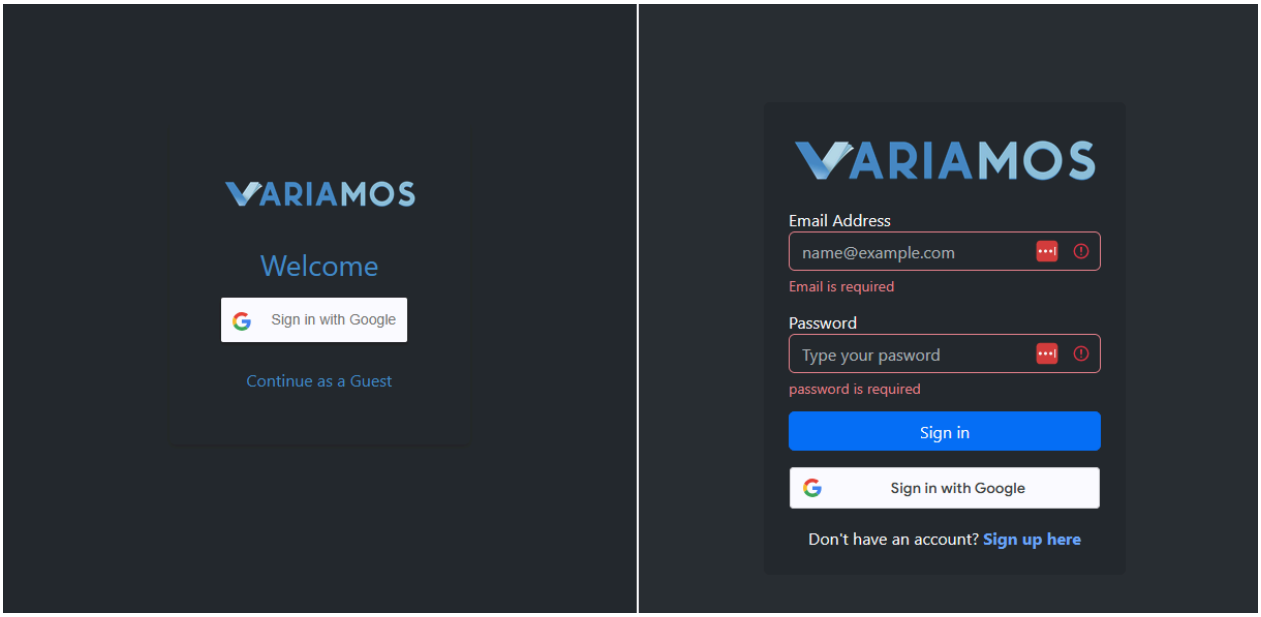

Fig. 1. Comparación del login de VariaMos.

En la figura 1 se ilustran los cambios realizados a la página de inicio de sesión (login) a la izquierda la versión anterior y a la derecha la nueva versión. En la nueva versión se pueden observar las validaciones mencionadas anteriormente, se agregó el link de registro “Sign up here” y se conserva el inicio de sesión mediante Google.

La Figura 2 muestra el nuevo formulario de registro, que incluye validaciones y la opción de volver a la página de inicio de sesión. El botón de Google se ha mantenido, pero el texto se ha

cambiado a "Registrarse con Google" para mayor coherencia. Para mejorar la seguridad y prevenir la enumeración de usuarios, las respuestas de los servicios web nunca revelan si un usuario existe o no, y se mantienen uniformes tanto en el inicio de sesión como en el registro para evitar proporcionar información útil a posibles atacantes.

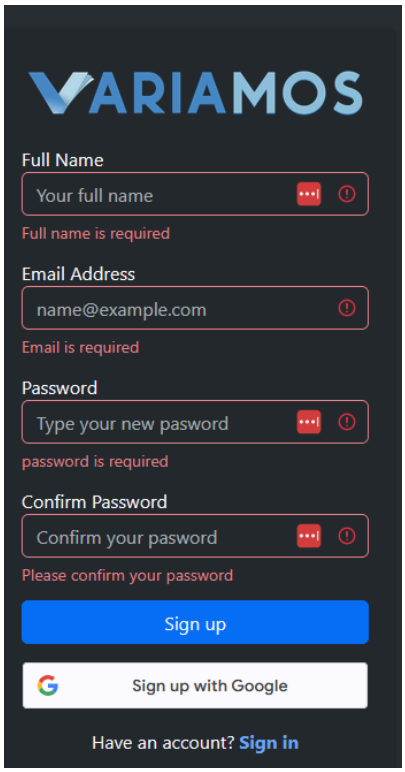

Fig. 2. Nuevo formulario de registro.

Dado que la aplicación permite registrarse con usuario y contraseña, pero también debe mantener la funcionalidad de inicio de sesión con Google, es necesario controlar la sesión desde la aplicación. Se elige JWT por ser uno de los estándares de la industria a la hora de implementar OAuth (Google, por ejemplo, devuelve un JWT al iniciar sesión).

En la Figura 3 se ilustra la secuencia para iniciar sesión mediante Google: el usuario indica el método de inicio de sesión al sistema, la aplicación redirige al inicio de sesión de Google, el usuario ingresa su información de inicio de sesión en este flujo externo y, si todo es correcto, la aplicación recibe una llamada a su servicio web con un código que se puede intercambiar por un token que permite obtener información del usuario, como su nombre y correo electrónico.

Tras obtener una respuesta exitosa y un token de Google, se valida la existencia del usuario; si no existe, se registra y, finalmente, se genera un JWT utilizando una clave privada. Este token podrá ser validado por cualquier otro microservicio utilizando la clave pública. El token JWT incluye datos como nombres de usuario, correo electrónico, ID de usuario, roles y permisos, lo que permite que cualquier aplicación o microservicio verifique los accesos del usuario sin generar dependencia del microservicio de administración, que es el encargado de realizar el flujo de inicio de sesión.

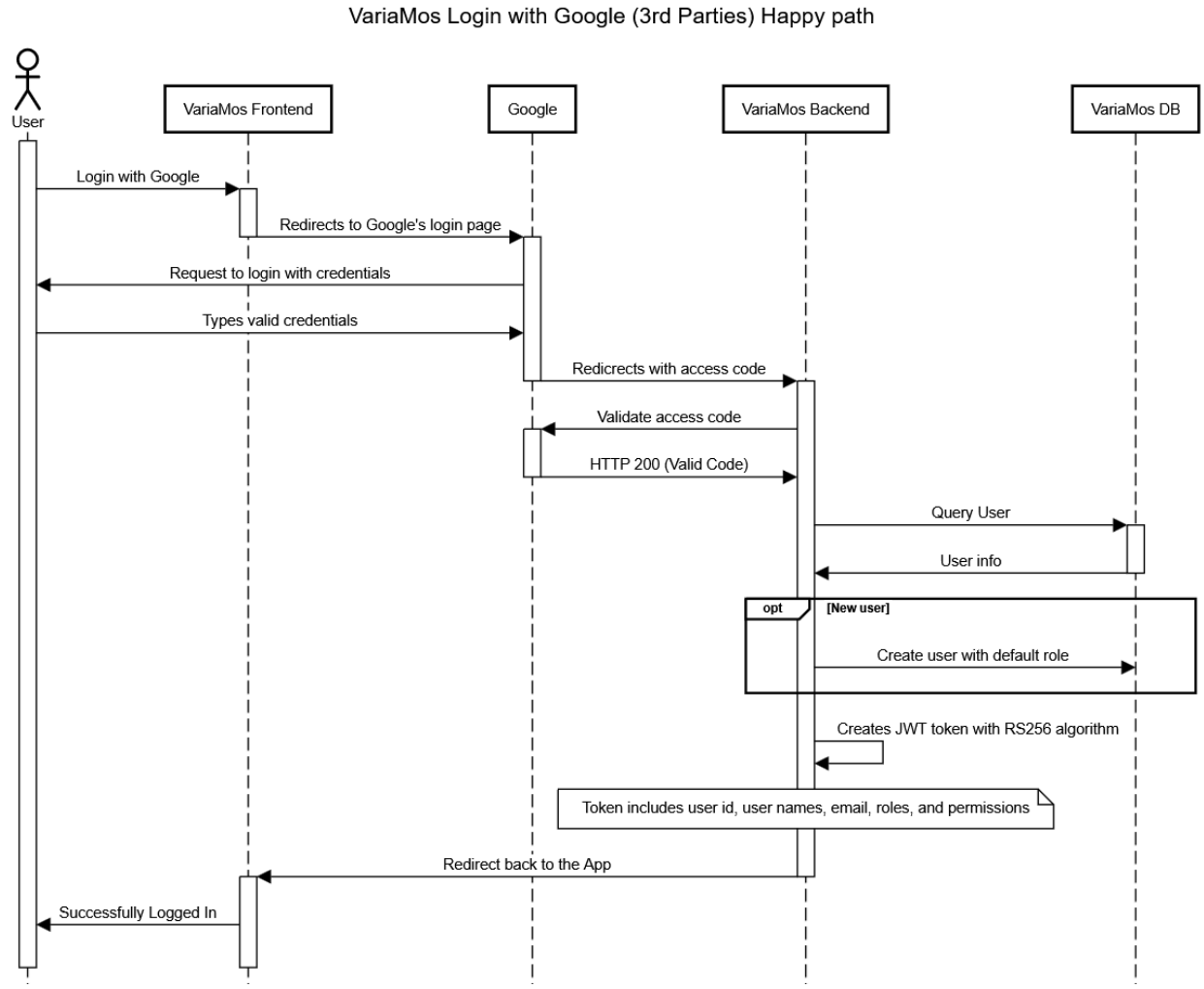

Fig. 3.Diagrama de secuencia de inicio de sesión mediante Google.

Cabe destacar que la figura 3 muestra el escenario ideal. Sin embargo, la Tabla I, que detalla la estructura de la entidad de usuario, incluye los campos "is_deleted" e "is_enabled". Si un usuario

intenta iniciar sesión (ya sea con usuario y contraseña o a través de Google) y su cuenta está marcada como eliminada o desactivada, recibirá un mensaje que le informará sobre el estado de su cuenta. En el caso de inicio de sesión con usuario y contraseña, el sistema primero valida que la contraseña sea correcta para prevenir la enumeración de usuarios.

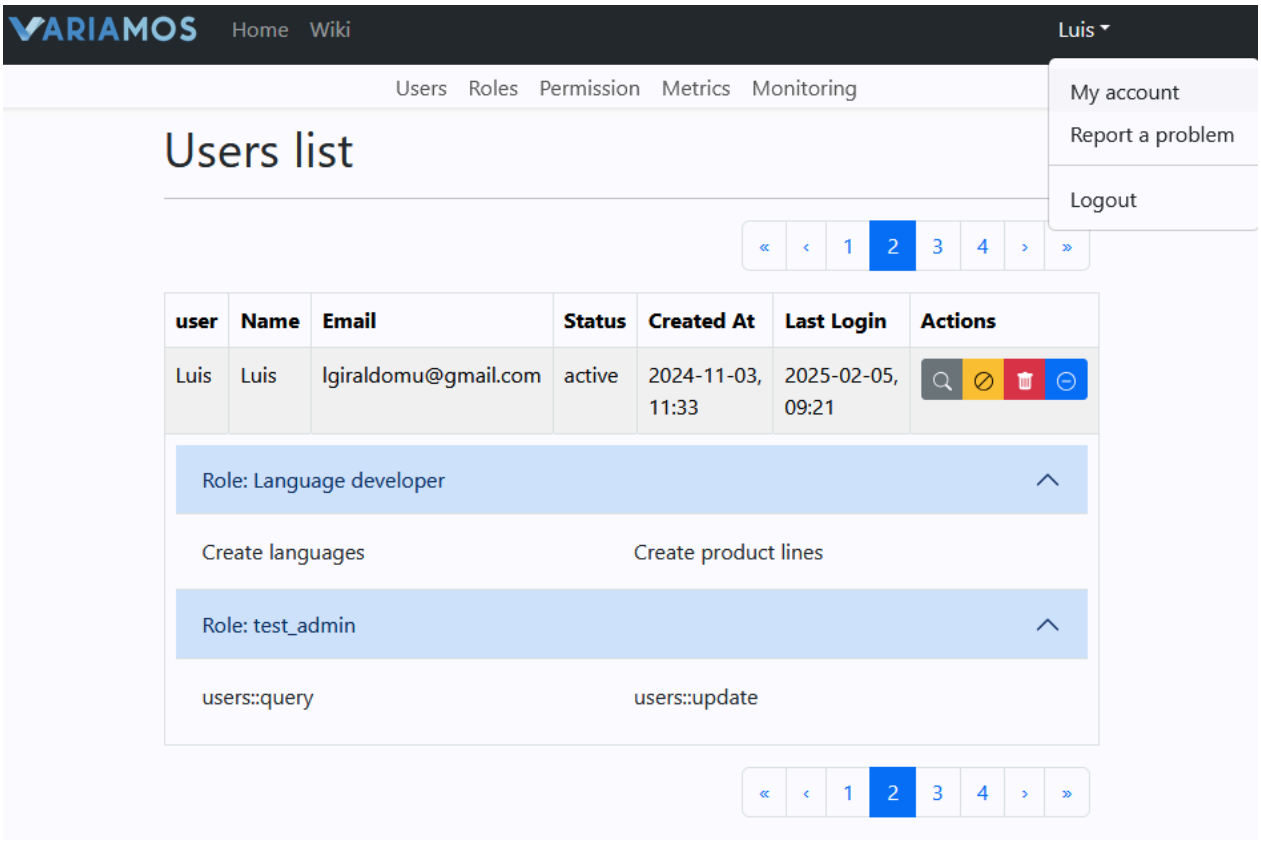

Fig. 4. Listado de usuarios.

La Figura 4 presenta la página de listado de usuarios, una interfaz diseñada exclusivamente para el acceso y gestión por parte de los administradores del sistema. Con el objetivo de mantener un rendimiento óptimo incluso con grandes volúmenes de datos, se ha implementado un sistema de paginación en todos los servicios web que generan listados.

Esta página proporciona una vista resumida de los usuarios del sistema, mostrando información esencial como la fecha de creación de la cuenta y el registro del último inicio de sesión. Además de estos datos básicos, la interfaz ofrece a los administradores un conjunto de herramientas para la gestión de usuarios, incluyendo la capacidad de ver detalles ampliados de un usuario

específico, deshabilitar o habilitar una cuenta, eliminar de manera lógica un usuario del sistema, y visualizar los permisos y roles asignados a cada usuario. Para facilitar la consulta de los permisos y roles, que pueden ser extensos, se ha implementado un sistema de acordeón que permite expandir y contraer la información de forma interactiva, mejorando significativamente la legibilidad y la experiencia de usuario.

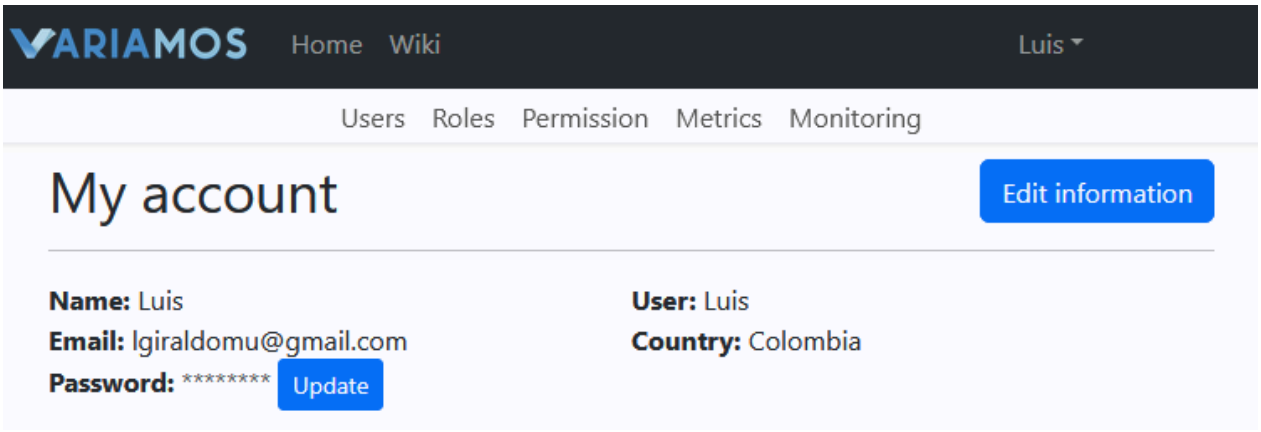

Fig. 5. Página “Mi cuenta”.

En la parte superior de la página, se encuentra un menú que incluye un enlace a "Mi cuenta", donde los usuarios pueden actualizar su contraseña y gestionar sus datos de ubicación como se puede ver en la figura 5.

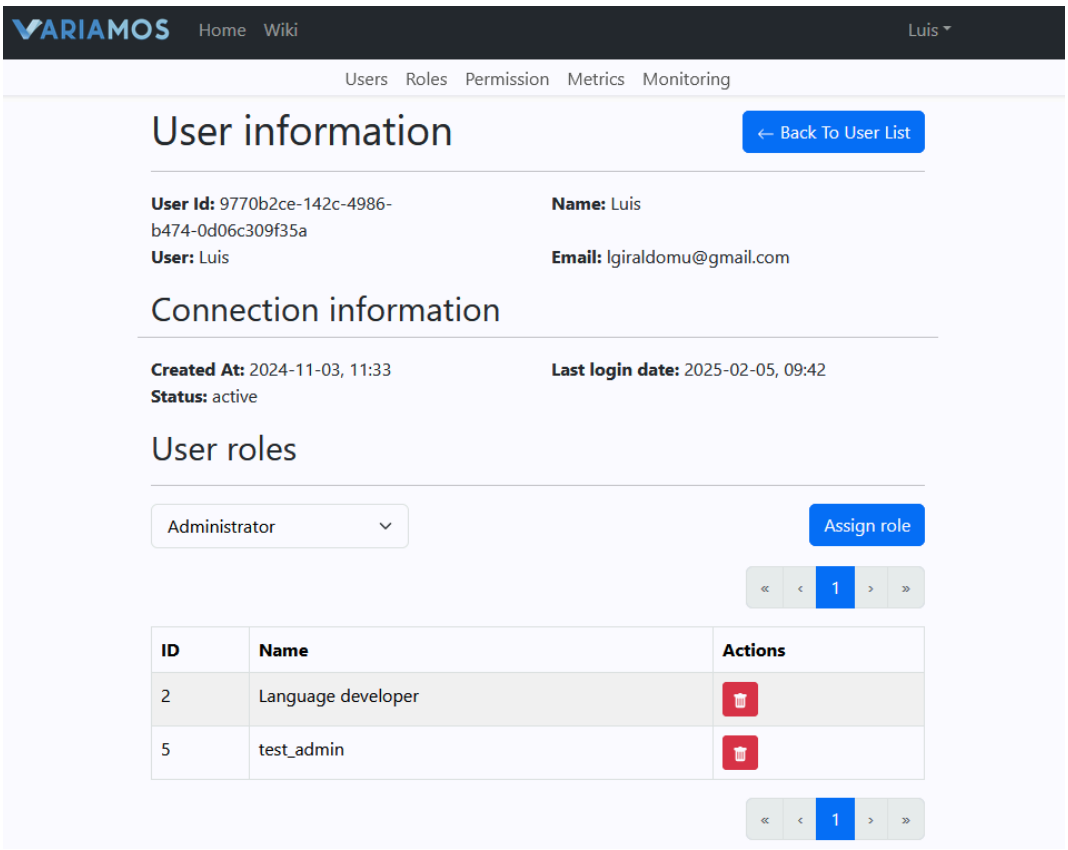

Fig. 6. Página de detalle de usuario.

La figura 6 muestra la página de detalles del usuario, que permite a los administradores ver información detallada de la cuenta y asignar o eliminar roles de usuario.

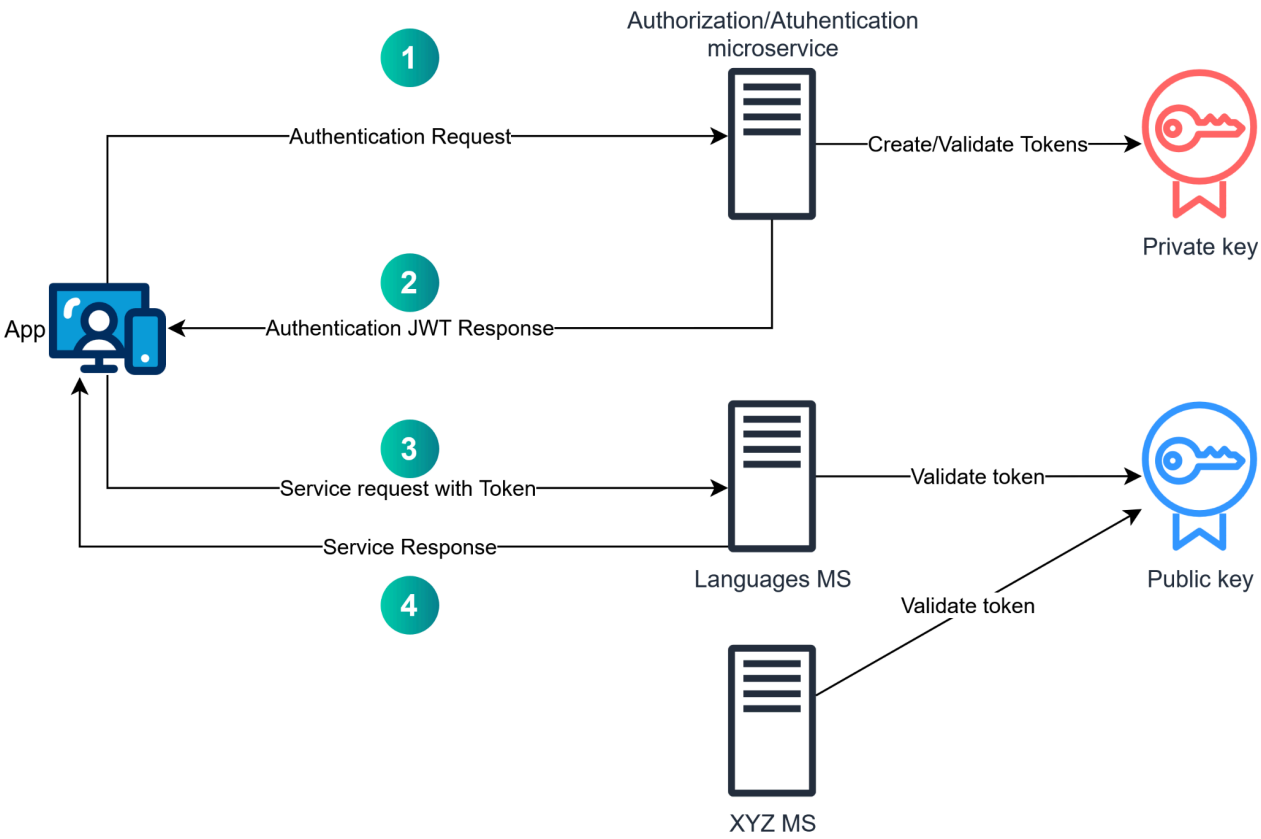

Fig. 7. Flujo de autorización.

La arquitectura de microservicios, utilizada en VariaMos, requiere un mecanismo de autorización eficiente para acceder a los servicios web. Para ello, se emplea el algoritmo RS256 durante la generación del JWT (JSON Web Token). Este algoritmo utiliza un par de claves pública y privada, permitiendo compartir la clave pública con aplicaciones frontend o clientes. Así, estos pueden validar la autenticidad de la sesión directamente al verificar el token JWT. Aunque un sistema con API Gateway podría centralizar esta validación, en su ausencia, este método evita el acoplamiento entre los microservicios y el encargado de autenticar a los usuarios.

Gestión de Roles y Permisos

Las siguientes tablas ilustran la estructura de las entidades de rol y permiso, y su relación entre sí y con el usuario.

TABLA II
ESTRUCTURA DE LA TABLA DE ROLES “ROLE”

Columna	Descripción	Tipo de dato
id	Identificador del rol (PK)	bigint
name	Nombre del rol	text

La Tabla II contiene la estructura de los roles de la aplicación. Cada rol puede estar asociado con múltiples permisos, como se muestra en la Tabla V, y con múltiples usuarios, como se muestra en la Tabla IV. La asignación de roles permite una gestión de permisos versátil y eficiente, ya que se pueden asociar múltiples permisos a un usuario simplemente asignándole un rol.

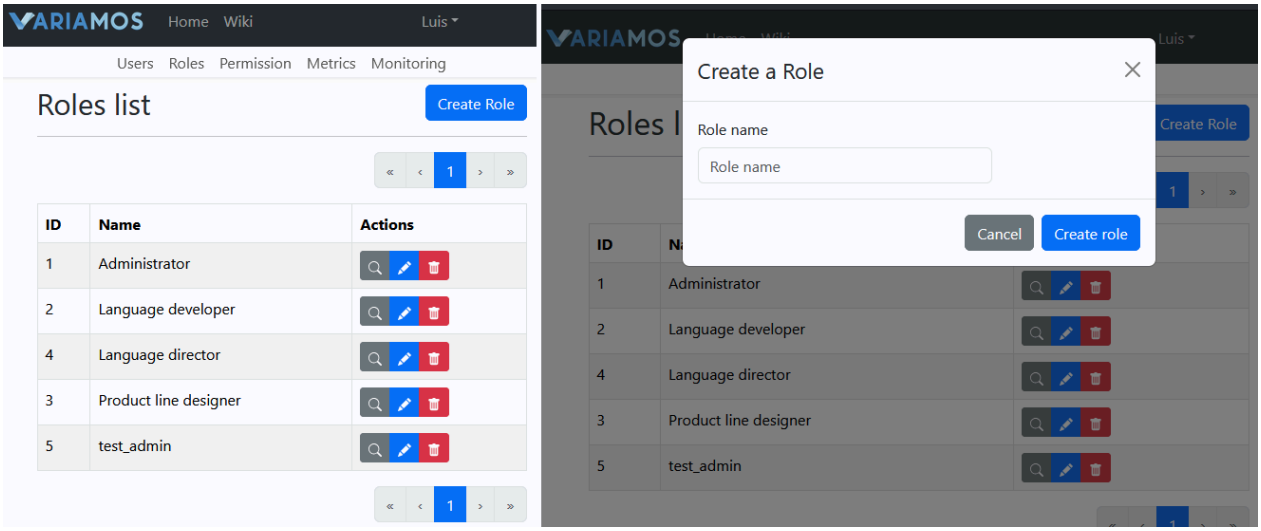

Fig. 8. Listado y creación de roles.

La Figura 8 presenta la página de listado de roles, una interfaz exclusiva para los administradores del sistema. Esta página proporciona una vista detallada de cada rol existente, y otorga a los administradores la capacidad de editar, eliminar o crear nuevos roles según las necesidades del sistema.

Por otro lado, la Figura 9 muestra la página de detalle de un rol específico. En esta interfaz, los administradores pueden profundizar en la configuración de un rol y gestionar sus permisos

asociados. Específicamente, tienen la capacidad de asignar nuevos permisos a un rol para ampliar sus capacidades, o eliminar permisos existentes para restringir sus acciones.

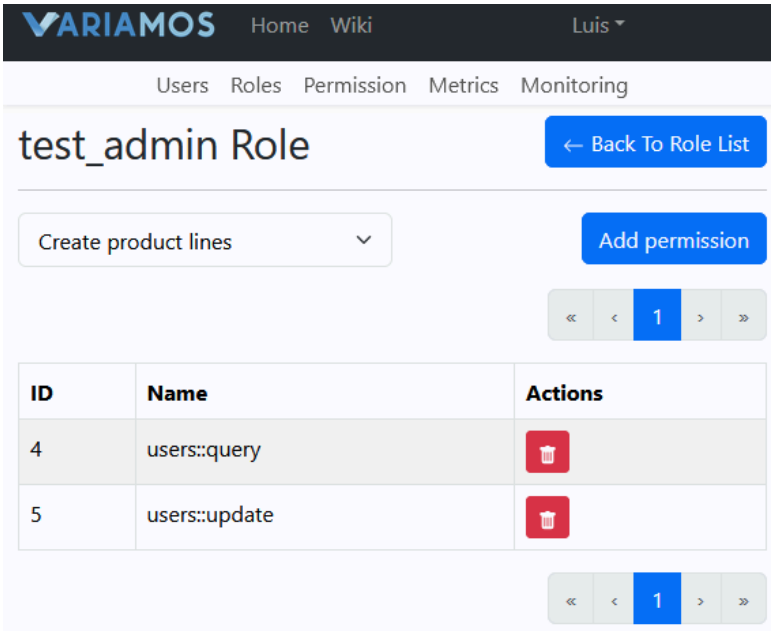

Fig. 9. Detalle de los roles.

TABLA III
ESTRUCTURA DE LA TABLA DE PERMISOS "PERMISSION"

Columna	Descripción	Tipo de dato
id	Identificador del permiso (PK)	bigint
name	Nombre del permiso	text

La Tabla III contiene la estructura de los permisos de la aplicación. Cada permiso puede estar asociado con múltiples roles, como se muestra en la Tabla V. Los roles son la estructura más granular que permite el acceso al sistema. Algunas funcionalidades solo necesitan validar el rol, mientras que otras requieren un permiso específico. Esto permite comportamientos como permitir a un usuario visualizar información sin editarla, dar permisos de escritura pero no de borrado, o todas las anteriores.

En la Figura 10, se observa que los permisos, al igual que los roles, cuentan con su propia página dedicada a su gestión. Esta página permite realizar acciones como listar, editar y eliminar

permisos. Toda la información pertinente a cada permiso se encuentra centralizada en esta página, eliminando la necesidad de una vista de detalles debido a la simplicidad inherente de esta entidad. Un permiso únicamente requiere un nombre para su definición, y el sistema se encarga de asignar automáticamente y de forma secuencial un identificador único (ID) a cada permiso.

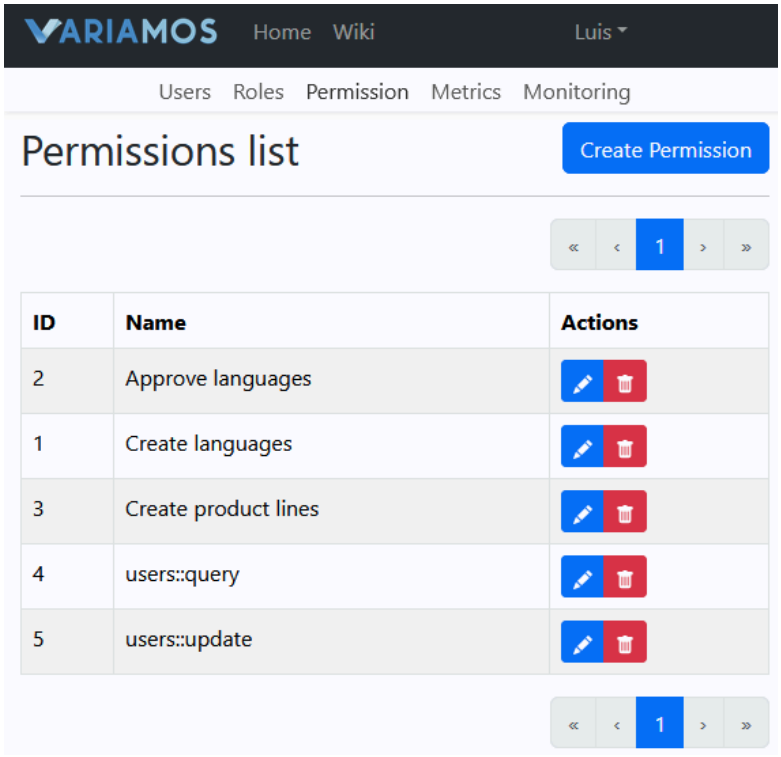

Fig. 10. Página de listado de permisos.

TABLA IV
ESTRUCTURA DE LA TABLA PIVOTE ENTRE ROLES Y USUARIOS "USER_ROLE"

Columna	**Descripción**	**Tipo de dato**
user_id	Identificador del usuario (PK)	text
role_id	Identificador del rol (PK)	bigint

La Tabla IV, diseñada como una tabla pivote, establece una relación de muchos a muchos entre los usuarios y los roles dentro del sistema. Esta estructura flexible permite que un mismo usuario pueda desempeñar múltiples roles según las necesidades o requerimientos del sistema. De esta forma, un usuario podría tener privilegios de administrador en un momento dado, y posteriormente cambiar a un rol de usuario con acceso restringido. Esta asignación dinámica de

roles a usuarios proporciona una capa adicional de seguridad y control, al tiempo que facilita la administración de permisos y accesos dentro del sistema.

TABLA V
ESTRUCTURA DE LA TABLA PIVOTE ENTRE ROLES Y PERMISOS “ROLE_PERMISSION”

Columna	**Descripción**	**Tipo de dato**
role_id	Identificador del rol (PK)	bigint
permission_id	Identificador del permiso (PK)	bigint

La Tabla V es una tabla pivote (muchos a muchos) entre los roles y los permisos, esto permite que se le asigne al usuario la lista de permisos asociados con un rol específico al asignarle dicho rol.

Estadísticas de visita y uso

- **Visualizar el número de visitas diarias por página:** Esta métrica muestra cuántas veces se accede a cada página de la aplicación por día, lo que permite identificar las páginas más populares y las fluctuaciones en el tráfico diario.
- **Visualizar el número de visitas únicas por página:** Esta métrica muestra cuántos usuarios individuales visitan cada página de la aplicación, lo que proporciona información sobre qué herramientas de la aplicación son más utilizadas.
- **Visualizar el número mensual de visitas por página:** Esta métrica realiza un seguimiento del número total de visitas mensuales a cada página de la aplicación durante un período de hasta 24 meses. Esto permite la comparación directa del uso de la página entre el año actual y el anterior, lo que facilita la identificación de tendencias de uso a lo largo del tiempo y los cambios en la forma en que se utilizan las páginas.
- **Visualizar el top 10 de páginas más visitadas en los últimos 3 meses:** Esta métrica muestra las páginas más populares de la aplicación durante los últimos tres meses, lo que proporciona información valiosa sobre las preferencias de los usuarios y las áreas de la aplicación que generan más interés.
- **Visualizar un mapa mundial con las visitas únicas anuales por país:** Esta métrica muestra la distribución geográfica de los usuarios de la aplicación, lo que permite identificar las regiones con mayor adopción y adaptar la aplicación a las necesidades y preferencias de los usuarios en diferentes países.

En general, la página de estadísticas proporciona una herramienta poderosa para comprender cómo se utiliza la aplicación, lo que permite a los administradores tomar decisiones informadas sobre el desarrollo futuro, la optimización del rendimiento y la mejora de la experiencia del

usuario.

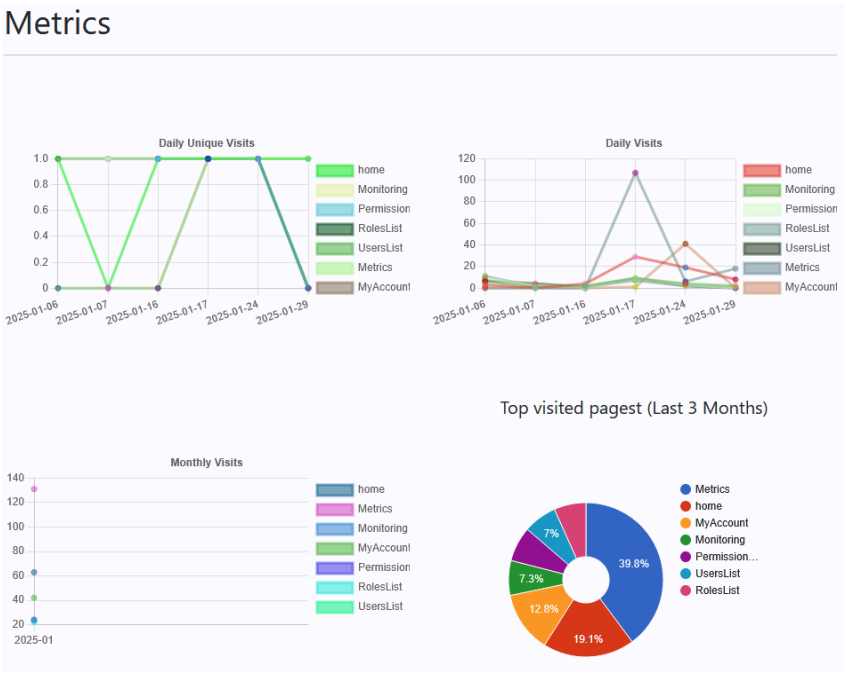

Fig. 11. Página de métricas.

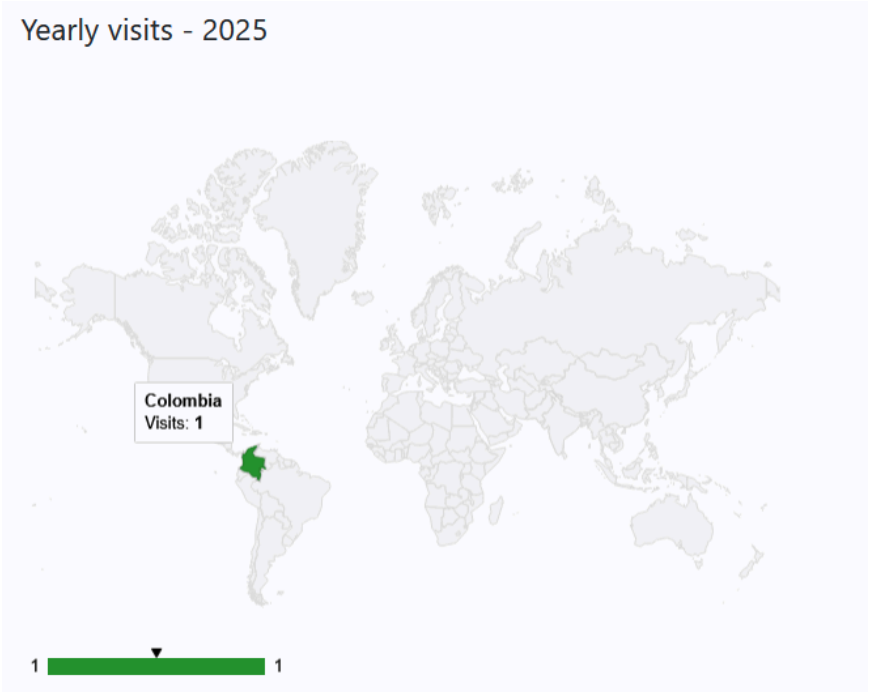

Fig. 12. Página de métricas - Visitas al año por país.

La tabla VI detalla la estructura de la entidad "países", esencial para la implementación del gráfico de visitas por país. Esta tabla almacena las coordenadas de cada país (latitud y longitud), así como sus códigos ISO 3166-1 alpha-2 y alpha-3. Al incluir estos datos, se facilita la integración con otros repositorios de información o bibliotecas de geolocalización, lo que a su vez posibilita la representación gráfica de la información.

TABLA VI
ESTRUCTURA DE LA TABLA DE PAÍSES “COUNTRY”

Columna	**Descripción**	**Tipo de dato**
code	Código del país (PK) ISO 3166-1 alpha-2	character varying
name	Nombre del país	character varying
latitude	Latitud del país	numeric
longitude	Longitud del país	numeric
code3	Código de tres letras del país ISO 3166-1 alpha-3	character varying

La tabla VII detalla la estructura de la entidad "visita por página", esta es la tabla que se encarga de registrar todas y cada una de las visitas a las páginas de la aplicación cada día.La Tabla VII, detallada a continuación, describe la estructura de la entidad "visita por página". Esta tabla es esencial para el seguimiento y registro de la actividad en la aplicación, ya que almacena un registro completo de cada visita realizada a las distintas páginas de la aplicación a lo largo del día. Cada fila de la tabla representa una visita individual, lo que permite un análisis granular del comportamiento del usuario y de las métricas de uso.

TABLA VII
ESTRUCTURA DE LA TABLA DE VISITAS POR PÁGINA “PAGE_VISIT”

Columna	**Descripción**	**Tipo de dato**
page_id	Identificador de la página	character varying
organization	Organización asociada	character varying
user_id	Identificador del usuario	text

visit_date	Fecha de la visita	date
country_code	Código del país	character varying

La Tabla VIII proporciona un resumen completo de la actividad diaria del sitio web, presentando un recuento detallado tanto de las visitas únicas como del total de visitas (incluyendo las repetidas). Este registro se actualiza automáticamente cada día a medianoche UTC, gracias a un trabajo programado implementado en la base de datos PostgreSQL. Esta tarea programada garantiza que los datos de la tabla estén siempre al día, proporcionando una visión precisa y actualizada del tráfico del sitio web.

TABLA VIII
ESTRUCTURA DE LA TABLA DE RESUMEN DE VISITAS “VISITS_SUMMARY”

Columna	**Descripción**	**Tipo de dato**
page_id	Identificador de la página (PK)	text
unique_visits	Número de visitas únicas	bigint
visits	Número total de visitas	bigint
visits_date	Fecha de las visitas (PK)	date

La Tabla IX, diseñada para almacenar un registro completo de las visitas de los usuarios por cada año, se actualiza diariamente. Esta actualización se basa en la información proporcionada por la tabla de base de datos "PAGE_VISIT". El objetivo principal de esta tabla es recopilar un conjunto de datos integral que abarque todo el año, identificando usuarios únicos junto con su ubicación por país.

Es importante tener en cuenta que la Tabla IX está diseñada para adaptarse a los cambios en la ubicación del usuario. En el caso de que un usuario actualice su ubicación, la tabla registrará automáticamente la última ubicación registrada durante cualquiera de las visitas del usuario. Esto garantiza que la tabla mantenga un registro preciso y actualizado de la ubicación del usuario, incluso si esta cambia con el tiempo.

TABLA IX
ESTRUCTURA DE LA TABLA DE VISITAS DE USUARIOS “USER_VISIT”

Columna	**Descripción**	**Tipo de dato**
user_id	Identificador del usuario (PK)	text
country_code	Código del país	character varying
visit_date	Fecha de la visita (PK)	date

La Tabla X, diseñada para almacenar un resumen de visitas por país y año, se actualiza periódicamente con los datos de la tabla "USER_VISIT". Esta actualización constante permite mantener un registro histórico consolidado de varios años, evitando así la necesidad de realizar operaciones complejas y costosas en grandes volúmenes de datos de forma regular. Es importante destacar que esta tabla se enfoca principalmente en consolidar los datos de años anteriores. Los datos del año en curso se mantienen en la tabla USER_VISIT para asegurar que sólo se cuente una vez a cada usuario.

Al comienzo de cada año, se lleva a cabo un proceso de mantenimiento crucial: los registros del año anterior se eliminan de la tabla “USER_VISIT” y se consolidan en la tabla "YEARLY_VISITS_SUMMARY". Este procedimiento garantiza que la Tabla “USER_VISIT” mantenga su enfoque en los datos del año actual, optimizando su rendimiento y evitando la acumulación innecesaria de datos. La tabla "YEARLY_VISITS_SUMMARY", por otro lado, se convierte en el repositorio histórico de resúmenes anuales de visitas, permitiendo análisis y comparaciones a largo plazo.

TABLA X
ESTRUCTURA DE LA TABLA DE RESUMEN ANUAL DE VISITAS DE USUARIOS “YEARLY_VISITS_SUMMARY”

Columna	**Descripción**	**Tipo de dato**
count	Conteo de visitas	bigint
visit_year	Año de las visitas	date
country_code	Código del país	character varying

El proceso de recopilación de métricas de uso en VariaMos, como se detalla en la figura 13, se inicia en el momento en que un usuario accede a una página. En ese preciso instante, el frontend

envía una solicitud al servicio de métricas con el objetivo de registrar dicha visita. Acto seguido, el servicio de métricas procede a consultar la base de datos para verificar si el usuario que ha visitado la página tiene registrada información sobre su ubicación.

En caso de que la información de ubicación esté disponible en la base de datos, el sistema la utiliza directamente. De lo contrario, el sistema toma la dirección IP del usuario y realiza una llamada a un servicio web externo especializado en proporcionar una ubicación aproximada basada en la dirección IP. Si bien la precisión de este servicio web suele ser alta en cuanto al país de origen de la IP, puede ser menos precisa en cuanto a la ciudad. No obstante, esta información resulta suficiente para ubicar espacialmente al usuario de manera general.

Finalmente, con la información recopilada tanto de la base de datos como del servicio web, el sistema procede a almacenar todos los datos relevantes en la tabla "PAGE_VISITS", consolidando así el registro de la visita del usuario y su ubicación asociada.

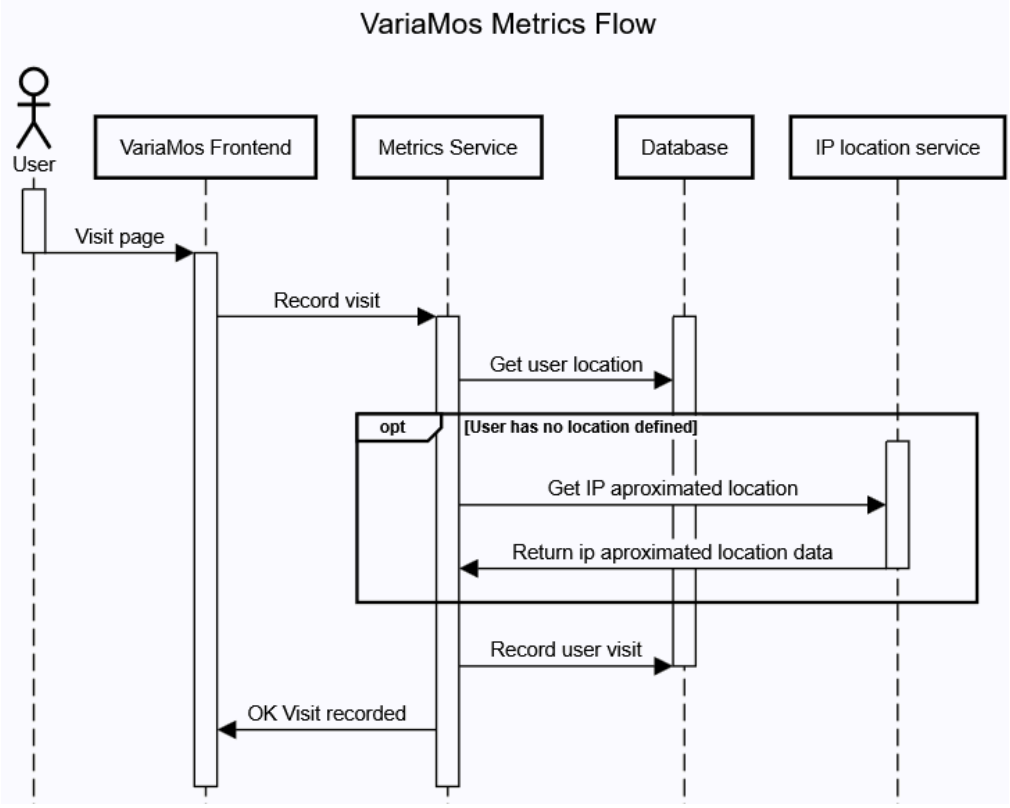

Fig. 13. Flujo de recolección de métricas de visitas.

El diagrama de flujo de métricas (Figura 14) muestra el proceso de la aplicación para el manejo diario de datos. El proceso se inicia a medianoche (UTC) por el Agente de PostgreSQL, PgAgent, que activa una tarea programada. Los primeros tres pasos de esta tarea cargan datos en la tabla

VISITS_SUMMARY. Esta tabla es esencial para mostrar métricas como las páginas más visitadas, visitas mensuales, diarias y únicas por página.

El segundo paso carga la tabla USER_VISITS usando datos de PAGE_VISITS. Esto permite el seguimiento de usuarios únicos por año. A diferencia de USER_VISITS (que acumula datos de todo el año), VISITS_SUMMARY no distingue entre usuarios nuevos y recurrentes en días consecutivos por eso la tabla USER_VISITS es necesaria.

El tercer paso carga datos en YEARLY_VISITS_SUMMARY usando la información del año anterior de USER_VISITS. USER_VISITS solo contiene datos del año anterior al inicio del año, ya que estos se eliminan en el quinto paso.

Finalmente, el cuarto paso elimina los datos del día anterior de PAGE_VISITS, consolidados en otras tablas.

Este proceso mantiene tablas con pocos datos para facilitar la generación de estadísticas. Estas estadísticas se cargan y almacenan en caché en el backend para su reutilización, optimizando el rendimiento y evitando la sobrecarga de la base de datos.

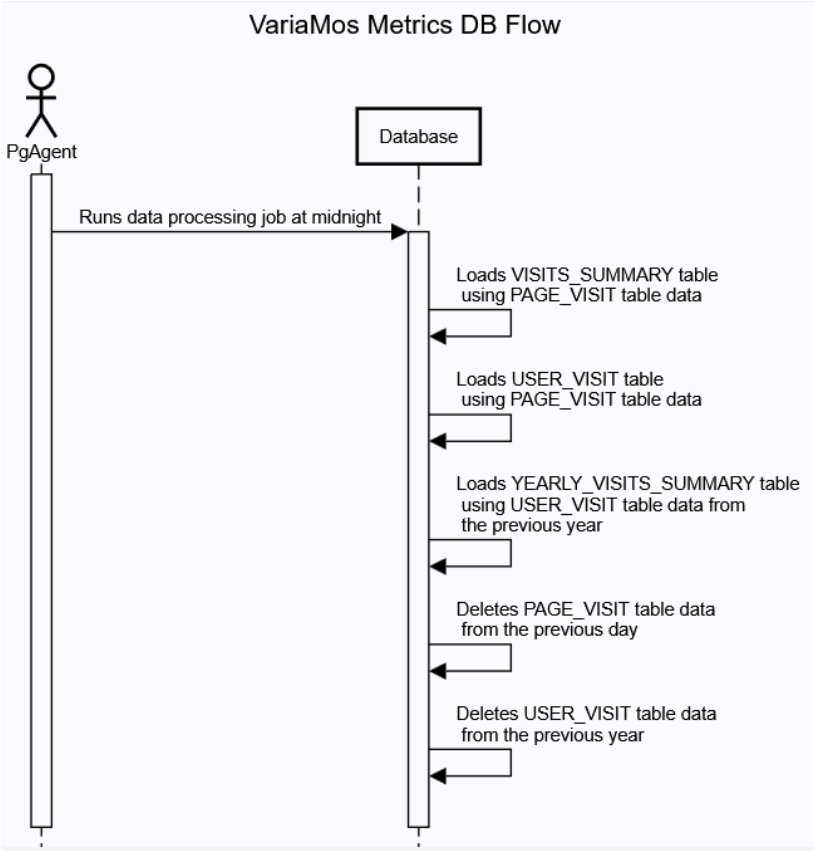

Fig. 14. Página de métricas.

Monitoreo de Micro-servicios

La Figura 15 muestra la página de monitoreo de microservicios, que presenta una lista de los contenedores Docker registrados en el servidor. Esta interfaz de usuario proporciona a los administradores la capacidad de supervisar el estado de cada microservicio individual, como el estado en el que está el microservicio y cuánto tiempo lleva en dicho estado. Además, a través de esta interfaz, los administradores pueden realizar una variedad de acciones en los contenedores, como iniciar un contenedor que actualmente está detenido, reiniciar un contenedor que ya se está ejecutando o detener completamente un contenedor que está en funcionamiento.

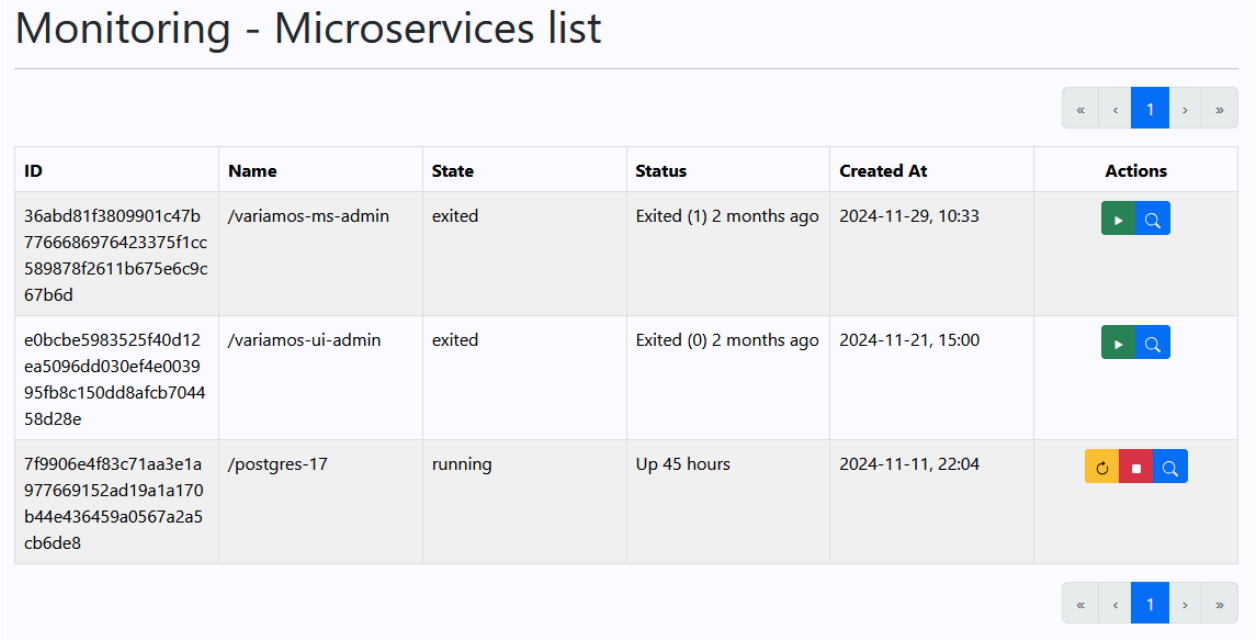

Fig. 15. Página de monitoreo - Listado de microservicios.

Además de las acciones que se pueden ejecutar en un microservicios, en la Figura 16 se muestra que la interfaz también proporciona una función de visualización de logs en tiempo real. Esto se logra mediante una conexión WebSocket, que facilita la recepción de actualizaciones en tiempo real de los logs del microservicio. Esta funcionalidad es particularmente valiosa para la depuración y el seguimiento de los errores que pueda estar experimentando la aplicación, así como para identificar el motivo de la caída de un microservicio.

Para garantizar un rendimiento fluido y evitar la congelación o ralentización de la aplicación web debido a un uso elevado de recursos, los registros se almacenan en un búfer en el frontend. Este búfer tiene una capacidad máxima configurable, expresada en bytes, lo que proporciona un mecanismo para administrar el consumo de recursos de la aplicación web. Al aprovechar la visualización de registros en tiempo real y la gestión eficiente del almacenamiento en búfer, los administradores y desarrolladores pueden obtener información valiosa sobre el comportamiento y el rendimiento de sus microservicios, lo que permite una depuración y optimización eficaces.

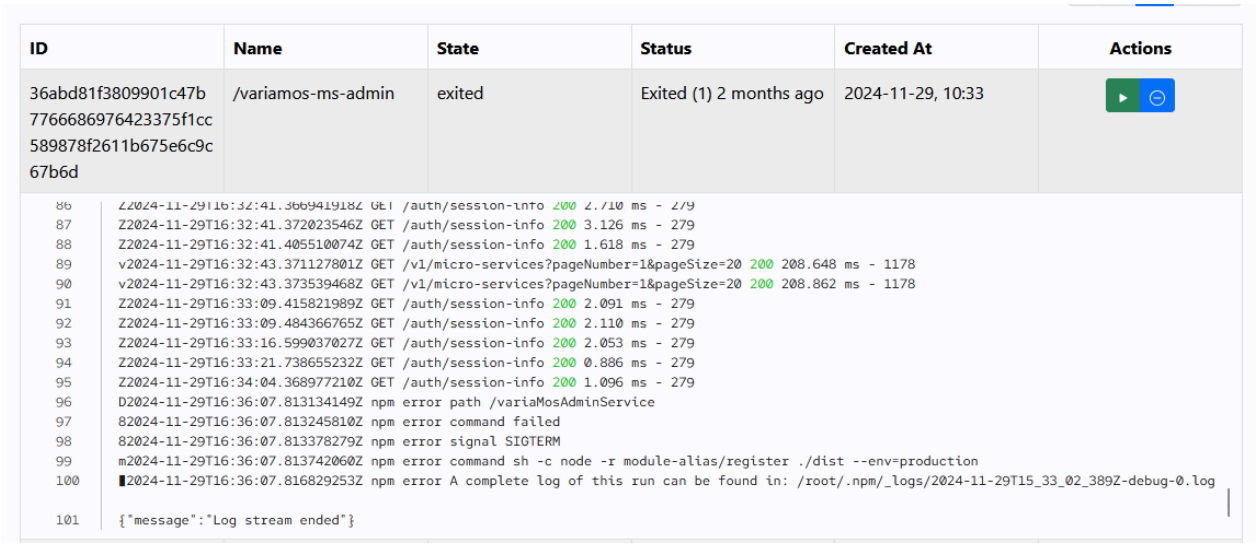

Fig. 16. Visualización de logs en la página de monitoreo.

VI. CONCLUSIONES Y RECOMENDACIONES

La aplicación ha sido mejorada en varios aspectos para aumentar su seguridad, facilitar la gestión de usuarios, roles y permisos, y crear herramientas que permitan gestionar el funcionamiento de la aplicación.

Con respecto a la seguridad, se implementaron la autenticación mediante Google (previamente existente) y usuario/contraseña, considerando el cifrado de contraseñas. Además, se perfeccionó la gestión de usuarios, roles y permisos, otorgando a los administradores un control más preciso sobre el acceso a la aplicación.

Además, se han creado nuevas herramientas que permiten a los administradores gestionar el funcionamiento de la aplicación. Estas herramientas incluyen un panel de control que muestra información sobre el uso de la aplicación, así como herramientas para consultar los datos estadísticos que se recolectan.

Estos cambios han mejorado significativamente la seguridad y la facilidad de gestión de la aplicación, lo que la convierte en una herramienta más valiosa para los administradores.

Recomendaciones y trabajos futuros

Los elementos identificados para el desarrollo futuro de la aplicación incluyen la adopción de microfrontends, que facilitarán la reutilización de componentes en todo el sistema y permitirán el desarrollo independiente de funcionalidades, de manera similar a como lo han hecho los microservicios hasta ahora.

Además, se recomienda la creación de librerías de componentes que puedan ser reutilizadas en los diferentes microservicios, siguiendo el patrón de web components. Esto hará que estos componentes sean agnósticos respecto al framework o librería que se esté utilizando para desarrollar los microfrontends, aumentando su compatibilidad y versatilidad.

Dado que se están utilizando microservicios, es crucial generar un módulo que contenga las clases del dominio que sean reutilizables en toda la aplicación. Esto contribuirá a una mayor estandarización de los microservicios mediante la definición de interfaces, que actuarán como los contratos que regirán la comunicación entre los microservicios y los microfrontends.

Esta modularización y estandarización no solo mejorarán la eficiencia del desarrollo y mantenimiento, sino que también promoverán la coherencia y la interoperabilidad en todo el sistema.

REFERENCIAS

[1] R. Mazo et al., "VariaMos - A microservices-based web application to engineer product lines and self-adaptive systems". [Enlace: https://variamos.com/].

[2] R. Mazo et al., "VariaMos: an extensible tool for product line engineering", Proceedings of the XIX International Software Product Line Conference, 2015.

[3] P. Clements y L. Northrop, Software Product Lines: Practices and Patterns, Addison-Wesley, 2002.

[4] R. C. Martin, Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2008.

[5] R. Fielding, Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000.

[6] D. Hardt, The OAuth 2.0 Authorization Framework. RFC 6749, 2012.

[7] M. Jones, J. Bradley, and N. Sakimura, JSON Web Token (JWT). RFC 7519, 2015.

[8] NPM, "Bcrypt for Node.js". [Enlace: https://www.npmjs.com/package/bcrypt].

[9] PostgreSQL Documentation, "JSON Functions and Operators". [Enlace: https://www.postgresql.org/docs/current/functions-json.html].

[10] React Documentation, "Introduction to React". [Enlace: https://reactjs.org/docs/getting-started.html].

[11] Bootstrap Documentation, "Bootstrap Overview". [Enlace: https://getbootstrap.com/docs/5.3/getting-started/introduction/].

[12] M. Fowler, "Continuous Integration," martinfowler.com, 2006. [Enlace: https://martinfowler.com/articles/continuousIntegration.html].

[13] GitHub Documentation, "GitHub Actions Documentation," GitHub, 2020. [Enlace: https://docs.github.com/en/actions].

[14] K. Schwaber and J. Sutherland, The Scrum Guide: The Definitive Guide to Scrum. Scrum Alliance, 2017.

[15] M. Cohn, Succeeding with Agile: Software Development Using Scrum. Addison-Wesley, 2009.

[16] R. Kruchten, The Rational Unified Process: An Introduction, 3rd ed., Addison-Wesley, 2003.

[17] J. F. Rumbaugh, Unified Modeling Language Reference Manual. Addison-Wesley, 2005.

[18] GitHub Documentation, "GitHub Features". [Enlace: https://docs.github.com/en].