9 No.1 (2025): Journal Scientific MQRInvestigar ISSN: 2588–0659
https://doi.org/10.56048/MQR20225.9.1.2025.e189

Prototype Microservices Architecture for Financial Transactional Systems with Keycloak

Prototipo de Arquitectura de Microservicios para Sistemas Transaccionales Financieros con Keycloak

Autores:

Santacruz-Erraez, Luis Fernando
UNIVERSIDAD CATÓLICA DE CUENCA
Estudiante de la Unidad Académica de Informática, Ciencias de la Computación, e Innovación Tecnológica
Cuenca – Ecuador

luis.santacruz.59@est.ucacue.edu.ec

https://orcid.org/0009-0001-6493-967X

Poma-Japón, Diana Ximena
UNIVERSIDAD CATÓLICA DE CUENCA
Docente de la Unidad Académica de Informática, Ciencias de la Computación, e Innovación Tecnológica
Cuenca – Ecuador

dpomaj@ucacue.edu.ec

https://orcid.org/0000-0001-9231-1655

Fechas de recepción: 19-ENE-2025 aceptación: 19-FEB-2025 publicación: 15-MAR-2025

https://orcid.org/0000-0002-8695-5005
http://mqrinvestigar.com/

Resumen

En un entorno financiero donde la seguridad, escalabilidad y cumplimiento normativo son esenciales, se presenta un prototipo avanzado de arquitectura de microservicios, diseñado para transformar la eficiencia y resiliencia de los sistemas transaccionales. La solución implementada integra Spring Boot, Spring Cloud, Eureka, Keycloak y Docker, permitiendo autenticación centralizada, balanceo de carga y alta disponibilidad, con un enfoque escalable basado en herencia y distribución eficiente de la carga. Ante las crecientes demandas del sector, como baja latencia, alta resiliencia y la necesidad del cumplimiento de normativas internacionales vigentes (ISO 27001, PCI DSS, Ley Fintech) se presenta la solución de una infraestructura modular con un API Gateway basado en Spring Cloud, asegurando interoperabilidad con sistemas bancarios tradicionales mediante protocolos REST, gRPC y WebSockets. Las pruebas de rendimiento y seguridad validaron la eficiencia del mismo, logrando tiempos de respuesta inferiores a 100 ms en transacciones críticas, autenticación segura mediante OAuth2 y MFA, y monitoreo centralizado con Spring Boot Admin. La arquitectura mostró capacidad para manejar hasta un millón de transacciones diarias sin degradación, garantizando su idoneidad para entornos financieros de alto tráfico. Este estudio confirma que una arquitectura de microservicios bien estructurada ofrece mayor flexibilidad, seguridad y escalabilidad que los sistemas monolíticos, permitiendo la rápida adaptación a regulaciones y tecnologías emergentes. Su aplicación en banca digital y fintechs puede optimizar la experiencia del usuario, reducir costos operativos y fortalecer la ciberseguridad del sector financiero.

Palabras clave: Microservicios; Keycloak; Seguridad Financiera; Autenticación; Escalabilidad; Arquitectura de Sistemas

Abstract

In a financial environment where security, scalability, and regulatory compliance are essential, an advanced microservices architecture prototype is presented, designed to enhance the efficiency and resilience of transactional systems. The proposed solution integrates Spring Boot, Spring Cloud, Eureka, Keycloak, and Docker, enabling centralized authentication, load balancing, and high availability, with a scalable approach based on inheritance and efficient load distribution. Given the growing demands of the financial sector, such as low latency, high resilience, and the need for compliance with current international regulations (ISO 27001, PCI DSS, Fintech Law), the study introduces a modular infrastructure with an API Gateway based on Spring Cloud, ensuring interoperability with traditional banking systems through REST, gRPC, and WebSockets protocols. Performance and security tests validated the system's efficiency, achieving response times below 100 ms in critical transactions, secure authentication via OAuth2 and MFA, and centralized monitoring using Spring Boot Admin. The architecture demonstrated the ability to handle up to one million daily transactions without degradation, ensuring its suitability for high-traffic financial environments. This study confirms that a well-structured microservices architecture provides greater flexibility, security, and scalability than monolithic systems. Its implementation in digital banking and fintech ecosystems can enhance user experience, reduce operational costs, and strengthen financial cybersecurity, ensuring seamless adaptation to evolving regulations and emerging technologies.

Keywords: Microservices; Keycloak; Financial Security; Authentication; Scalability; Systems Architecture

Introducción

El sector financiero global enfrenta desafíos crecientes que dificultan la eficiencia y seguridad de las transacciones financieras. Entre los principales problemas se encuentran el arbitraje regulatorio, los desequilibrios económicos y la fragmentación del sistema financiero internacional, lo que agrava la interoperabilidad dentro de los sistemas monolíticos y complica los problemas de deuda e inflación (Golovnin, 2022). Además, los sistemas monolíticos utilizados en muchas instituciones financieras han generado problemas críticos de interoperabilidad, incrementando los costos de infraestructura y exponiendo a las organizaciones a mayores riesgos de ciberataques (Grody, 2018). En este contexto, la arquitectura de microservicios ha demostrado ser una solución eficaz, ya que facilita la adopción de estándares globales de datos, permitiendo la integración de tecnologías emergentes como la tecnología de libro mayor distribuido (DLT), lo que reduce errores y mejora la escalabilidad y la adaptabilidad del sistema financiero (Grody, 2018). Asimismo, el auge de plataformas descentralizadas (DeFi) y el incremento de las transacciones digitales han elevado el riesgo de ciberamenazas y fraudes, lo que obliga a las instituciones financieras a fortalecer sus medidas de seguridad (Nikolić, 2023). En los diferentes países del mundo, la adopción de microservicios ha sido una tendencia clave en las grandes empresas tecnológicas, tales como lo son Amazon y Netflix, que han migrado sus sistemas hacia arquitecturas de microservicios para mejorar la escalabilidad y la eficiencia en el manejo de grandes volúmenes de datos y transacciones. Este enfoque les ha permitido integrar nuevas funcionalidades de manera flexible, reduciendo tiempos de inactividad y mejorando la capacidad de respuesta de sus sistemas en entornos de alta demanda. Un caso relevante a tener en cuenta en el sector financiero es el del Danske Bank, uno de los bancos más grande de Dinamarca, que migró su sistema crítico, FX Core, a una arquitectura de microservicios permitiéndoles mejorar significativamente la escalabilidad y la estabilidad del sistema, consolidando los microservicios como una solución preferente para instituciones financieras que buscan mejorar su eficiencia operativa (González Heredia et al., 2021; Mazzara et al., 2021). En América Latina y el Caribe, la transformación digital del sector financiero ha sido impulsada por regulaciones que fomentan la financiación abierta y el uso de tecnologías digitales. Países como Argentina y Colombia han adoptado marcos regulatorios clave, como

el Decreto 1.297 de 2022 en Colombia, que garantiza la interoperabilidad y la seguridad en las transacciones financieras, mientras que, en Argentina, el Banco Central ha regulado las carteras virtuales para promover la banca abierta (Herrera et al., 2023). Sin embargo, la adopción de estas tecnologías también implica riesgos de ciberseguridad y protección al consumidor, pues las instituciones financieras se vuelven más vulnerables a ataques cibernéticos y fraudes. Las arquitecturas de microservicios se presentan como una solución eficaz para estos desafíos, permitiendo una segmentación clara de los servicios, la adopción de APIs seguras y una adaptación más rápida a los cambios regulatorios, promoviendo la inclusión financiera y mitigando los riesgos de exclusión digital (Herrera et al., 2023). Además, en el Ecuador el sistema financiero enfrenta grandes desafíos relacionados con la gestión de transacciones digitales y la ciberseguridad. La falta de un modelo específico de gestión de transacciones ha generado ineficiencias y vulnerabilidades en el sistema. En respuesta, se han implementado reformas como el Código Orgánico Monetario y Financiero y la reciente Ley Fintech de 2023, que establece un marco regulatorio para los servicios financieros tecnológicos en Ecuador, promoviendo la adopción de blockchain y soluciones digitales para mejorar la inclusión financiera, la transparencia y la seguridad en las operaciones (Resolución No. JPRF-F-2023-076, 2023). A pesar de estas reformas, persisten vacíos normativos que limitan la capacidad de las instituciones para gestionar riesgos. Los problemas de confiabilidad y las vulnerabilidades ante ciberataques siguen siendo preocupantes (Mendoza et al., 2022). La Ley Fintech de 2023 refuerza la regulación de entidades tecnológicas, estableciendo principios de protección de datos y seguridad, pero la arquitectura de microservicios puede ser una alternativa complementaria para mejorar la seguridad operativa y la adaptación regulatoria en el sistema financiero ecuatoriano.

Considerando estos aspectos, la presente investigación tiene como objetivo desarrollar un prototipo de arquitectura de microservicios para la gestión de transacciones financieras, garantizando eficiencia, escalabilidad y seguridad. Se propone utilizar tecnologías que permitan manejar la seguridad centralizada con Keycloak, el balanceo de carga con un API Gateway y la tecnología modular para un desarrollo continuo, rápido y metódico junto con el framework de Spring Boot, permitiendo segmentar cada operación en microservicios independientes y manejar de manera efectiva cada una de las dependencias utilizadas. La

implementación de esta arquitectura asegurará que cada componente del sistema opere de forma autónoma, reduciendo el impacto de fallos sin comprometer la escalabilidad. De esta manera, se busca garantizar que los servicios se mantengan operativos y eficientes incluso en escenarios de alta demanda (Resolución No. JPRF-F-2023-076, 2023).

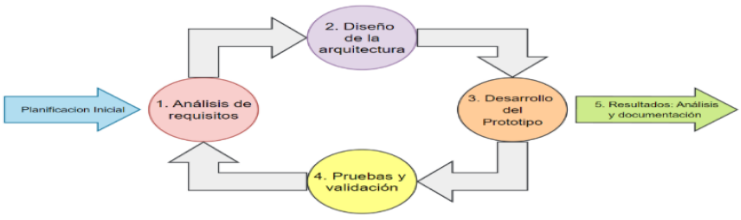
Se han planteado preguntas de investigación que guiarán y buscaran mejorar el enfoque al desarrollo del prototipo acorde a las necesidades del sector financiero, las mismas son: ¿Cómo pueden los microservicios mejorar la seguridad de las transacciones financieras en el sector ecuatoriano?, ¿Cuál es la viabilidad de implementar Keycloak como solución de autenticación centralizada en sistemas financieros locales? y ¿Cómo puede garantizarse la interoperabilidad con los sistemas financieros existentes mediante una arquitectura de microservicios?. El presente artículo se estructura en varias secciones. En la primera sección se analizan los conceptos relacionados con la arquitectura de microservicios; en la segunda, se revisan los trabajos previos relevantes; en la tercera, se presenta la metodología utilizada para el desarrollo del prototipo; y finalmente, se exponen los resultados obtenidos y las conclusiones.

Material y métodos

Siguiendo un modelo iterativo e incremental (MII) como se muestra en la figura 1, se asegura que el sistema evolucione de manera controlada, incorporando mejoras continuas en función de las necesidades debido a su capacidad de adaptarse a cambios, permitir entregas funcionales parciales y fomentar la retroalimentación continua. Según Solano Fernández & Porras Alfaro (2020), esta metodología facilita la producción simultánea de módulos, permite una evaluación progresiva y mitigación de riesgos en cada iteración mejorando la capacidad de respuesta ante requerimientos emergentes, evitando la rigidez de modelos secuenciales.

Figura 1

Fases de desarrollo de prototipo, modelo Iterativo e Incremental



Fuente: Fuente Propia

La construcción del prototipo se divide en 5 fases, cada una con objetivos específicos que guían el proceso de desarrollo del mismo:

Fase 1: Análisis de Requisitos

Para la fase inicial del proyecto, se realizará un análisis detallado para identificar las necesidades técnicas y funcionales del prototipo permitiendo levantar los requerimientos funcionales y no funcionales necesarios para su construcción. Poniendo principal énfasis a la problemática presentada con los sistemas monolíticos y el manejo deficiente de la seguridad centralizada de los sistemas altamente transaccionales del sector financiero.

Para el levantamiento de la información necesaria se tendrá en cuanta cada uno de los siguientes apartados:

- Identificación de las necesidades técnicas y regulatorias.
- Estándares internacionales de seguridad.
- Normativas vigentes.
- Necesidades específicas del sector financiero.

Fase 2: Diseño de la Arquitectura

La arquitectura para el prototipo será elegida de acuerdo a las necesidades presentadas en la fase de análisis de requisitos, la estructura se someterá a una evaluación de cumplimiento de los hitos presentados en la fase 1, teniendo en cuenta que la misma cumpla los estándares de desarrollo y motivando a que se cumplan con las necesidades del sector financiero.

Entre las principales tecnologías sometidas al análisis estarán:

- Spring Boot como framework base junto con el lenguaje de programación Java.
- Keycloak como gestor de identidad encargado del manejo de sesión y accesos a los servicios del servidor.
- Docker que permite contenerizar y ajustar cada parámetro de despliegue del servidor.

Fase 3: Desarrollo del Prototipo

El desarrollo del mismo obedecerá a los requisitos y necesidades presentados, respetando la arquitectura planteada para el caso y siguiendo el enfoque iterativo e incremental para que se pueda ajustar el mismo según se disponga dentro de las fases de desarrollo. Durante esta etapa se tendrá en cuenta que la misma debe cumplir los estándares de manejo de código

limpio que puede ayudar a preservar el software a largo plazo y fomentando que se minimice la deuda técnica para los equipos.

Fase 4: Pruebas y Validación

En esta fase, se realizarán pruebas para evaluar el desempeño del prototipo que permitirán asegurar sus bondades y cumplimiento de necesidades para el sector financiero. Las pruebas incluirán:

- Pruebas unitarias: Para garantizar el prototipo funcione correctamente de manera individual en cada uno de sus funciones y módulos.
- Pruebas de integración: Para verificar que los módulos trabajen de manera conjunta sin conflictos.
- Pruebas de carga y estrés: Con el objetivo de evaluar la capacidad del sistema para manejar altos volúmenes de transacciones concurrentes.
- Pruebas de seguridad: Para validar la efectividad de los mecanismos de autenticación y autorización implementados.

Fase 5: Documentación y Análisis de Resultados

En esta fase, se consolidará toda la información generada durante el desarrollo del prototipo permitiendo evidenciar el diseño de la arquitectura, los procesos de desarrollo e integración, y los resultados de las pruebas realizadas. Esta documentación servirá como referencia para futuros desarrollos y para evaluar el impacto del prototipo en entornos productivos. Además, se incluirán análisis de los hallazgos obtenidos durante las pruebas, destacando las áreas de mejora y proponiendo recomendaciones específicas para la implementación del sistema en entornos financieros operativos.

Desarrollo

Conceptos Relacionados

Microservicios: Se hace mención a los mismos como unidades pequeñas, poco acopladas y altamente cohesivas que interactúan para proporcionar las funcionalidades del sistema. Están diseñados para mejorar el rendimiento, la escalabilidad y la disponibilidad de los sistemas de software, especialmente en las arquitecturas de microservicios (MSA), abordando las complejidades y los desafíos de actualización y segmentación de responsabilidades (Mohottige et al., 2024).

Gestión de Identidades y Acceso (IAM): Hace referencia a los procesos y tecnologías que administran las identidades digitales y controlan el acceso de los usuarios a los recursos, garantizando la seguridad y el cumplimiento mediante la verificación de las identidades de los usuarios y la concesión de los derechos de acceso adecuados dentro de una organización (Sekar et al., 2024).

Keycloak: Es una solución de gestión de acceso e identidades de código abierto que facilita los procesos de autenticación y autorización seguros. Es compatible con varios mecanismos de autenticación, incluida la autenticación multifactor (MFA), y simplifica la integración de funciones de seguridad avanzadas para los desarrolladores (Ionaşcu & Aciobăniţei, 2024).

OAuth 2.0: Es un marco de autorización que permite a las aplicaciones de terceros obtener acceso limitado a las cuentas de usuario de un servicio HTTP, lo que permite un acceso delegado seguro sin compartir las credenciales. Es compatible con varios flujos de autorización, lo que mejora la seguridad y la experiencia del usuario en los procesos de autenticación (Mortágua et al., 2024).

JSON Web Token (JWT): Hace referencia a un mecanismo de control de acceso sin estado que representa de forma segura los datos de autenticación y autorización, lo que permite a los microservicios verificar la identidad y los permisos de los usuarios mediante tokens firmados, lo que mejora la seguridad y permite una comunicación eficiente entre los servicios (Venčkauskas et al., 2023).

Normativa Fintech de Ecuador: La misma representa una normativa reciente que regula los servicios financieros tecnológicos en Ecuador, estableciendo lineamientos para la adopción de tecnologías digitales, como plataformas electrónicas y soluciones de pago digital. Esta ley busca fomentar la innovación y la inclusión financiera, promoviendo el uso de tecnologías seguras para la protección de los datos de los usuarios (Resolución No. JPRF-F-2023-076, 2023).

Trabajos Relacionados

Guamán et al. (2018), llevó a cabo una evaluación del proceso de migración de una aplicación monolítica a microservicios, utilizando modelos como NGINX e IBM con el objetivo de identificar y seleccionar actividades que faciliten la migración, la investigación analizo los

patrones y fases de la migración usando Spring Boot, siendo este el framework base del proyecto en curso.

De Paz Estrada (2019) en la Universidad de San Carlos de Guatemala, desarrolló una arquitectura basada en microservicios para un sistema LMS, utilizando patrones de contenerización con Docker y un API Gateway con Node.js. El estudio implementó tecnologías como Consul para el registro de servicios, lo que facilitó la modularidad del sistema y eliminó tiempos de inactividad, aportando así un marco valioso para el despliegue de aplicaciones con alta disponibilidad.

Bermúdez Osorio et al. (2019) en la Universidad Tecnológica de Pereira, desarrolló una librería para facilitar la integración de servicios web utilizando microservicios. El enfoque permitió reducir los tiempos de desarrollo al integrar diferentes protocolos de comunicación, como SOAP y REST API, solucionando problemas de interoperabilidad. Esta investigación resalta la utilidad de desacoplar servicios mediante una interfaz, lo cual es relevante para aplicaciones transaccionales que requieren alta flexibilidad y capacidad de adaptación a cambios de protocolo.

Nebel (2019) en la Universidad de la República de Uruguay, exploró el uso de microservicios para mejorar la escalabilidad y flexibilidad en plataformas de integración, utilizando patrones de diseño reconocidos en la industria. La investigación identificó beneficios significativos en la adaptabilidad del sistema, aunque se enfrentaron desafíos relacionados con la consistencia eventual y la complejidad operativa, aspectos críticos que deberán ser abordados para optimizar sistemas transaccionales distribuidos.

Chicaiza Rios (2020) en la Universidad Politécnica Salesiana, implementó un prototipo para aplicaciones financieras, utilizando contenedores Docker y pruebas de carga con JMeter, mejorando la escalabilidad y resiliencia. El estudio no aborda mecanismos robustos de gestión de identidades y control de acceso, que permiten el fortalecimiento de la seguridad y asegurar el cumplimiento normativo, pero aporta un marco integral para aplicaciones críticas del sector financiero.

Mendoza et al. (2022) desarrolló un prototipo basado en una arquitectura híbrida utilizando Hyperledger y Ethereum, enfocado en mejorar la seguridad de las transacciones mediante un almacenamiento inmutable. Este trabajo aporta una perspectiva sólida sobre el uso de

Blockchain para proteger la integridad de los datos transaccionales, sin embargo, se observa una limitación al no incluir un sistema robusto de gestión de identidades y control de acceso con una autenticación centralizada.

Mamani (2023) en la Universidad La Salle de Arequipa, llevó a cabo un análisis sobre los desafíos de pruebas en arquitecturas de microservicios, destacando la importancia de la automatización para garantizar la calidad y rendimiento. El estudio presentó herramientas y métodos para pruebas automatizadas end-to-end, abordando aspectos críticos de rendimiento y seguridad. Este trabajo proporciona una guía para implementar pruebas robustas, pero se enfoca principalmente en la calidad del software, dejando de lado la integración de mecanismos de autenticación centralizados como Keycloak, aspecto que resulta crucial para sistemas financieros que requieren un control de acceso seguro y centralizado.

Resultados

El desarrollo del prototipo se ha basado en un enfoque estructurado que permite evaluar cada una de las fases del proyecto de manera secuencial y fundamentada. Los resultados presentados a continuación se obtuvieron mediante la aplicación del modelo iterativo e incremental planteado en la Figura 1.

Fase 1: Análisis de Requisitos

Identificación de las necesidades técnicas y regulatorias

Para revisar los principales requerimientos técnicos y evidenciar su importancia en el diseño de la arquitectura, se presenta la Tabla 1, donde se detallan las necesidades clave para garantizar escalabilidad, seguridad y eficiencia en sistemas transaccionales.

Tabla 1

Necesidades Técnicas.

Necesidad	Descripción	Fuente(s)
Escalabilidad y flexibilidad.	La migración de arquitecturas monolíticas a microservicios busca mejorar la escalabilidad y mantenimiento del sistema, evitando el acoplamiento fuerte y asegurando despliegues ágiles.	Tatiana Gómez Suárez et al. (2017); Mohottige et al. (2024)
Alta disponibilidad y tolerancia a fallos.	Se requieren mecanismos de redundancia, balanceo de carga y recuperación automática ante fallos para garantizar disponibilidad 24/7.	Nikolić (2023); Grody (2018)

Orquestación de servicios.	El uso de herramientas como Docker y Kubernetes permite la gestión eficiente y escalamiento automático de los microservicios.	De Paz Estrada (2019)
Seguridad en la autenticación y comunicación.	Es esencial implementar autenticación basada en OAuth 2.0, JWT y OpenID Connect, así como cifrado de extremo a extremo y segmentación de privilegios.	Venčkauskas et al. (2023); Mortágua et al. (2024)
Interoperabilidad con sistemas tradicionales.	Se debe garantizar que los microservicios puedan integrarse con sistemas financieros heredados y plataformas externas mediante API Gateway y Service Discovery.	Guamán et al. (2018)
Blockchain y trazabilidad de transacciones.	Tecnologías como blockchain híbrido pueden mejorar la seguridad, transparencia y trazabilidad en operaciones financieras.	Mendoza et al. (2022)
Optimización del procesamiento de transacciones.	La implementación de colas de mensajes y caching reduce la latencia y mejora el rendimiento del sistema.	Mohottige et al. (2024)
Monitoreo y auditoría en tiempo real.	Se requiere centralización de logs y monitoreo de eventos de seguridad para garantizar trazabilidad y detección temprana de anomalías.	Grody (2018); Mortágua et al. (2024)

De igual manera, para garantizar el cumplimiento normativo en el sector financiero y establecer un marco de seguridad robusto, es fundamental considerar las regulaciones vigentes. En la Tabla 2, se presentan las principales normativas que rigen la seguridad, la digitalización financiera y la protección de datos, asegurando que la arquitectura propuesta cumpla con los estándares internacionales y locales.

Tabla 2

Necesidades Regulatorias.

Regulación/Normativa	Descripción	Fuente(s)
Cumplimiento con estándares de seguridad (ISO 27001, PCI DSS, GDPR, PSD2)	Exigen la aplicación de autenticación multifactor, cifrado de datos y trazabilidad de accesos para proteger la información financiera.	Mortágua et al. (2024); Venčkauskas et al. (2023)
Regulación de criptomonedas y activos digitales	La falta de regulación homogénea sobre criptomonedas y stablecoins genera incertidumbre en el sistema financiero global.	Golovnin (2022); Dafri & Al-Qaruty (2023)
Emisión de monedas digitales por bancos centrales (CBDCs)	Se requieren marcos normativos claros para regular la adopción de CBDCs y su	Golovnin (2022)

	interoperabilidad con sistemas financieros tradicionales.	
Normativa local sobre digitalización financiera en Ecuador	La Resolución No. JPRF-F-2023-076 regula la seguridad, administración de riesgos y concesión digital de créditos en servicios financieros.	Resolución No. JPRF-F-2023-076 (2023)
Prevención de fraude y lavado de activos	Se exige la implementación de mecanismos de auditoría, identificación de usuarios y monitoreo de transacciones sospechosas.	Dafri & Al-Qaruty (2023); Grody (2018)
Protección de datos y privacidad	Regulaciones como GDPR y la Ley Fintech imponen restricciones sobre el manejo de datos personales y financieros.	Nikolić (2023); Resolución No. JPRF-F-2023-076 (2023)

Estándares Internacionales de Seguridad para el Sector Financiero

Es esencial alinearse con los estándares internacionales de seguridad. En la Tabla 3, se presentan los principales marcos regulatorios y normativos que establecen lineamientos clave para la protección de datos, autenticación, prevención de fraudes y ciberseguridad en arquitecturas basadas en microservicios, asegurando el cumplimiento de mejores prácticas a nivel global.

Tabla 3

Estándares Internacionales de Seguridad en Arquitecturas de Microservicios Financieros.

Estándar Internacional	Objetivo	Requerimientos Clave	Fuente(s)
ISO/IEC 27001	Garantizar la seguridad de la información mediante gestión de riesgos y controles de acceso.	- Implementación de políticas de seguridad. - Protección de datos en tránsito y en reposo. - Gestión de incidentes de seguridad. - Monitoreo y auditoría de accesos.	Mortágua et al. (2024); Golovnin (2022); Nikolić (2023)
PCI DSS (Payment Card Industry Data Security Standard)	Proteger datos financieros en transacciones con tarjetas de pago.	- Encriptación de datos de tarjetas. - Segmentación de acceso a datos sensibles. - Pruebas de vulnerabilidad y monitoreo continuo. - Implementación de autenticación multifactor (MFA).	Venčkauskas et al. (2023); Mortágua et al. (2024)

GDPR (General Data Protection Regulation)	Asegurar la privacidad y protección de datos personales en transacciones financieras.	- Consentimiento explícito para el uso de datos. - Derecho de los usuarios al acceso, modificación o eliminación de sus datos. - Implementación de medidas de seguridad para evitar filtraciones de información. - Transparencia en el uso de datos personales.	Golovnin (2022); Resolución No. JPRF-F-2023-076 (2023); Dafri & Al-Qaruty (2023)
PSD2 (Payment Services Directive 2)	Regular los servicios de pago y garantizar la seguridad en la banca abierta.	- Implementación de Strong Customer Authentication (SCA). - Protección contra fraudes en pagos electrónicos. - Acceso regulado a cuentas bancarias por terceros (APIs seguras). - Supervisión de transacciones sospechosas.	Mortágua et al. (2024); Nikolić (2023); Golovnin (2022)
AML/KYC (Anti-Money Laundering / Know Your Customer)	Prevenir el lavado de dinero y financiamiento del terrorismo en transacciones financieras.	- Identificación y verificación de clientes. - Monitoreo y análisis de patrones de transacciones. - Reporte de actividades sospechosas a entidades reguladoras. - Implementación de controles para evitar fraude financiero.	Dafri & Al-Qaruty (2023); Grody (2018); Resolución No. JPRF-F-2023-076 (2023)
OWASP Security Standards	Definir prácticas seguras para el desarrollo de software financiero basado en microservicios.	- Implementación de autenticación robusta (OAuth2, OpenID Connect). - Protección contra ataques de inyección (SQLi, XSS). - Validación estricta de entradas y salidas en APIs. - Seguridad en la gestión de sesiones y cookies.	Venčkauskas et al. (2023); Mohottige et al. (2024)
TLS 1.3 (Transport Layer Security)	Garantizar la seguridad en la comunicación entre microservicios y usuarios finales.	- Uso de cifrado AES-256 para datos en tránsito. - Eliminación de protocolos inseguros (SSL, TLS 1.0/1.1). - Implementación de Perfect Forward Secrecy (PFS). - Reducción de latencia en transacciones seguras.	De Paz Estrada (2019); Mortágua et al. (2024); Grody (2018)
NIST Cybersecurity Framework	Proveer un marco de referencia para la gestión de riesgos de ciberseguridad en infraestructura crítica.	- Identificación de activos y vulnerabilidades. - Protección de sistemas con controles de seguridad. - Detección de amenazas y respuesta rápida a incidentes. - Recuperación de operaciones tras un ataque cibernético.	Nikolić (2023); Dafri & Al-Qaruty (2023); Mendoza et al. (2022)

Normativas Vigentes para Arquitecturas Tecnológicas Financieras

En Ecuador, las normativas vigentes regulan la seguridad y supervisión del sector financiero digital. Como se muestra en la Tabla 4, estas normativas establecen lineamientos que deben tenerse en cuenta para el desarrollo del prototipo.

Tabla 4

Normativas Vigentes en el Sector Financiero Ecuatoriano.

Normativa	Descripción	Fuente(s)
Código Orgánico Monetario y Financiero	Regula el sistema financiero nacional, estableciendo principios de transparencia, seguridad y supervisión de las entidades bancarias y fintechs.	Resolución No. JPRF-F-2023-076 (2023)
Ley Fintech (Ley Orgánica para el Desarrollo, Regulación y Control de los Servicios Financieros Tecnológicos)	Define la regulación de servicios financieros digitales, estableciendo requisitos para la seguridad de datos, ciberseguridad, prevención de fraude y supervisión de la Superintendencia de Bancos.	Resolución No. JPRF-F-2023-076 (2023)
Regulación de Concesión Digital de Créditos	Establece los criterios para la evaluación del perfil de riesgo de clientes, desembolsos electrónicos y calificación de cartera en plataformas digitales.	Resolución No. JPRF-F-2023-076 (2023)
Normas de Ciberseguridad y Protección de Datos Financieros	Exige la implementación de estándares de seguridad, cifrado de datos sensibles y control de acceso en plataformas financieras digitales.	Resolución No. JPRF-F-2023-076 (2023); Mendoza et al. (2022)
Prevención del Lavado de Activos y Financiamiento del Terrorismo (ARLAFDT)	Obliga a las entidades financieras a contar con oficiales de cumplimiento, procedimientos de auditoría y control de transacciones sospechosas.	Resolución No. JPRF-F-2023-076 (2023); Mendoza et al. (2022)
Regulación sobre Infraestructura Tecnológica en Servicios Financieros	Permite el uso de tecnologías como blockchain, computación en la nube y Big Data, bajo estrictos controles de seguridad y auditoría.	Resolución No. JPRF-F-2023-076 (2023); Mohottige et al. (2024)
Regulación de Open Banking y APIs Financieras	Fomenta la interoperabilidad entre entidades bancarias y fintechs mediante API estandarizadas y control de acceso seguro.	Resolución No. JPRF-F-2023-076 (2023); Nikolić (2023)

Supervisión y Control Financiero	Determina el rol de la Superintendencia de Bancos y la Junta de Regulación Financiera en la supervisión de fintechs y bancos digitales.	Resolución No. JPRF-F-2023-076 (2023)
Protección de los Usuarios Financieros	Obliga a las entidades a garantizar transparencia en los costos, términos y condiciones de los productos financieros digitales.	Resolución No. JPRF-F-2023-076 (2023)
Gestión de Riesgos Financieros y Tecnológicos	Define estrategias de mitigación de riesgos operativos y tecnológicos en plataformas digitales, incluyendo auditoría continua y modelos de análisis de riesgo.	Resolución No. JPRF-F-2023-076 (2023); Golovnin (2022)
Regulación de Monedas Digitales y Criptomonedas	Evalúa la integración de monedas digitales de bancos centrales (CBDC) y establece restricciones en el uso de criptomonedas en servicios financieros regulados.	Golovnin (2022); Mendoza et al. (2022)

Necesidades del sector Financiero – Requisitos del Prototipo

Para que el prototipo cumpla con las exigencias del sector financiero, se han definido requisitos funcionales y no funcionales clave. Como se observa en la Tabla 5, estos requisitos abarcan disponibilidad, rendimiento, seguridad y cumplimiento normativo, asegurando que la arquitectura propuesta responda a los desafíos del entorno financiero.

Tabla 5

Necesidades y Requerimientos del Sector Financiero.

Categoría	Necesidad del Sector Financiero	Requisito Funcional	Requisito No Funcional
Disponibilidad y Resiliencia	Alta disponibilidad y operación 24/7.	Implementar redundancia y failover automático en los servicios críticos.	Despliegue en infraestructura distribuida con balanceo de carga y replicación de datos.
Procesamiento y Rendimiento	Baja latencia en transacciones en tiempo real.	Optimizar procesamiento con colas de mensajería y caching distribuido.	Tiempos de respuesta inferiores a 200 ms en transacciones críticas.
Escalabilidad	Adaptabilidad a cambios en la carga transaccional.	Implementar escalado dinámico con métricas de uso.	Capacidad de atender hasta 1 millón de transacciones aproximadas diarias sin degradación.

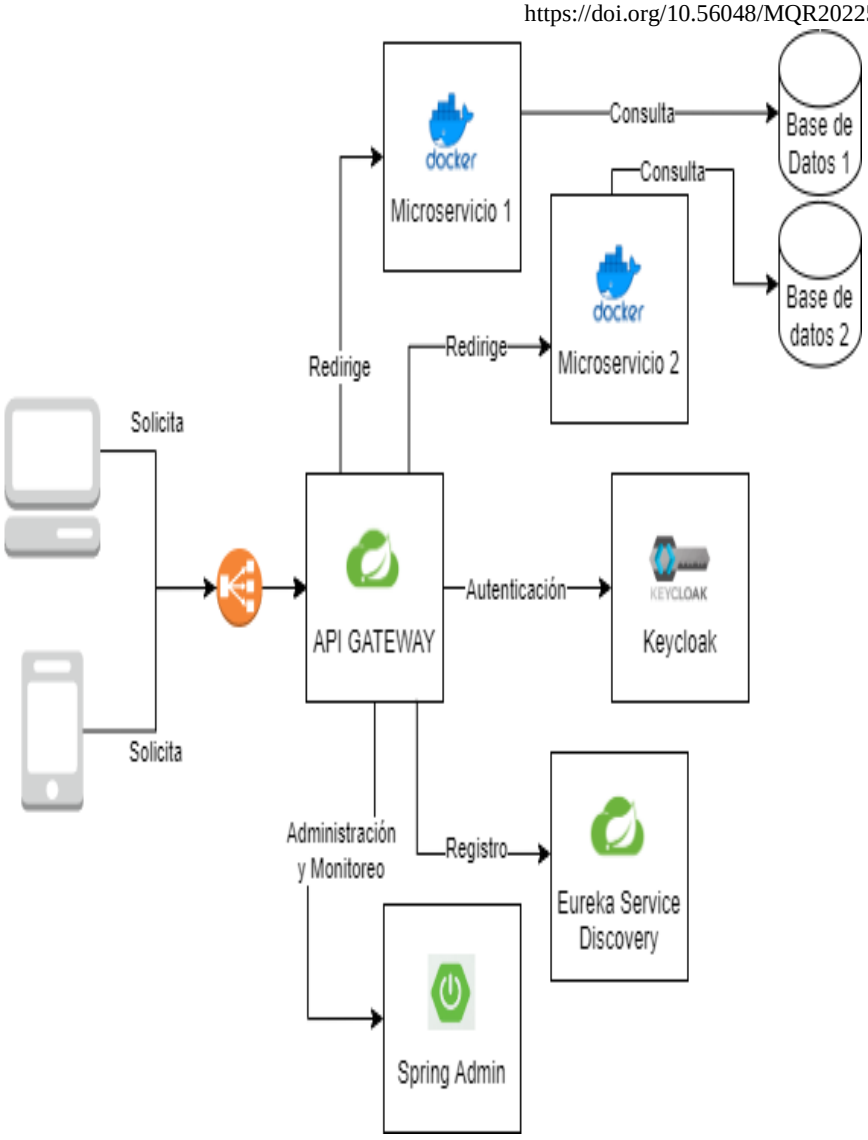
Seguridad y Cumplimiento Normativo	Protección de datos financieros y prevención de fraudes.	Implementar autenticación fuerte (MFA, OAuth2, biometría).	Cifrado de datos en tránsito y en reposo con AES-256 y TLS 1.3.
Auditoría y Regulaciones	Trazabilidad y cumplimiento de normativas financieras.	Registro inmutable de transacciones con logs distribuidos y blockchain.	Conformidad con ISO 27001, PCI DSS y normas de la Superintendencia de Bancos.
Interoperabilidad	Conectividad con sistemas bancarios tradicionales y nuevas plataformas.	Implementar API Gateway y estándares de Open Banking.	Soporte para múltiples protocolos (REST, gRPC, WebSockets).
Monitoreo y Mantenimiento	Supervisión en tiempo real del estado del sistema.	Implementar dashboards centralizados y alertas automatizadas.	Logs estructurados accesibles en menos de 5 segundos.
Automatización y Orquestación	Eficiencia en despliegues y actualizaciones.	Implementar despliegue continuo.	Reducción del tiempo de inactividad a menos de 1 minuto en actualizaciones.

Fase 2: Diseño de la Arquitectura

Para garantizar que la arquitectura del prototipo se adapte a los requerimientos establecidos, la misma se diseña utilizando Spring Boot como framework base, Spring Cloud como API Gateway y Eureka para el descubrimiento dinámico de servicios. La autenticación se gestiona mediante Keycloak, mientras que el monitoreo de la salud y la administración de microservicios se optimizan con Spring Boot Actuator y Spring Boot Admin. Además, se incorpora Caffeine para mejorar la escalabilidad, seguridad y resiliencia, asegurando un rendimiento óptimo en entornos bancarios. La solución se alinea con los requerimientos del sector financiero definidos en la Fase 1, integrando Docker para facilitar un despliegue ágil y continuo. La Figura 2 presenta el esquema estructurado de la solución propuesta.

Figura 2

Diagrama de Arquitectura General.



Fuente: Fuente Propia

En el desarrollo de la arquitectura se tomaron en cuenta las necesidades del sector financiero, presentando cada una de las soluciones como se evidencia en la tabla 6:

Tabla 6

Cumplimiento de Requerimientos de la Arquitectura.

Categoría	Necesidad del Sector Financiero	Requisito Funcional	Requisito No Funcional	Solución en la Arquitectura

Disponibilidad y Resiliencia	Alta disponibilidad y operación 24/7.	Implementar redundancia y failover automático en los servicios críticos.	Despliegue en infraestructura distribuida con balanceo de carga y replicación de datos.	Spring Cloud Gateway gestiona la distribución de tráfico y Eureka permite balanceo de carga automático.
Procesamiento y Rendimiento	Baja latencia en transacciones en tiempo real.	Optimizar procesamiento con colas de mensajería y caching distribuido.	Tiempos de respuesta inferiores a 200 ms en transacciones críticas.	Uso de Caffeine Cache y optimización de consultas mediante Spring Data JPA.
Escalabilidad	Adaptabilidad a cambios en la carga transaccional.	Implementar escalado dinámico con métricas de uso.	Capacidad de atender hasta 1 millón de transacciones diarias sin degradación.	Microservicios desacoplados y contenedorización con Docker, permitiendo escalabilidad horizontal.
Seguridad y Cumplimiento Normativo	Protección de datos financieros y prevención de fraudes.	Implementar autenticación fuerte (MFA, OAuth2, biometría).	Cifrado de datos en tránsito y en reposo con AES-256 y TLS 1.3.	Integración con Keycloak para autenticación, uso de OAuth 2.0 y JWT.
Auditoría y Regulaciones	Trazabilidad y cumplimiento de normativas financieras.	Registro inmutable de transacciones con logs distribuidos y blockchain.	Conformidad con ISO 27001, PCI DSS y normas locales.	Spring Boot Actuator y Admin permiten auditoría detallada de logs y métricas.
Interoperabilidad	Conectividad con sistemas bancarios tradicionales y nuevas plataformas.	Implementar API Gateway y estándares de Open Banking.	Soporte para múltiples protocolos (REST, gRPC, WebSockets).	Spring Cloud Gateway actúa como API Gateway y facilita integración con sistemas externos.
Monitoreo y Mantenimiento	Supervisión en tiempo real del estado del sistema.	Implementar dashboards centralizados y alertas automatizadas.	Logs estructurados accesibles en menos de 5 segundos.	Spring Boot Admin permite visualizar métricas en tiempo real.

Automatización y Orquestación	Eficiencia en despliegues y actualizaciones.	Implementar CI/CD y despliegue continuo.	Reducción del tiempo de inactividad a menos de 1 minuto en actualizaciones.	Uso de Docker para gestión de despliegues y orquestación con Spring Cloud Config.

Fase 3: Desarrollo del Prototipo

El prototipo de microservicios transaccionales para el sector financiero se ha diseñado bajo principios de alta disponibilidad, seguridad y escalabilidad, garantizando una infraestructura robusta y flexible. Al utilizar Spring Boot, Spring Cloud, Eureka, Keycloak, Spring Boot Admin, Actuator y Caffeine Cache, este sistema permite gestionar transacciones bancarias con un enfoque modular, asegurando interoperabilidad con sistemas tradicionales y cumplimiento normativo.

El prototipo abarca 2 módulos principales para la segregación de los dominios de aplicación como se observa en la tabla 7, que serán los proyectos padre encargados de manejar las dependencias de los microservicios e implementar de mejor manera el control de cambios y actualizaciones necesarias.

Tabla 7

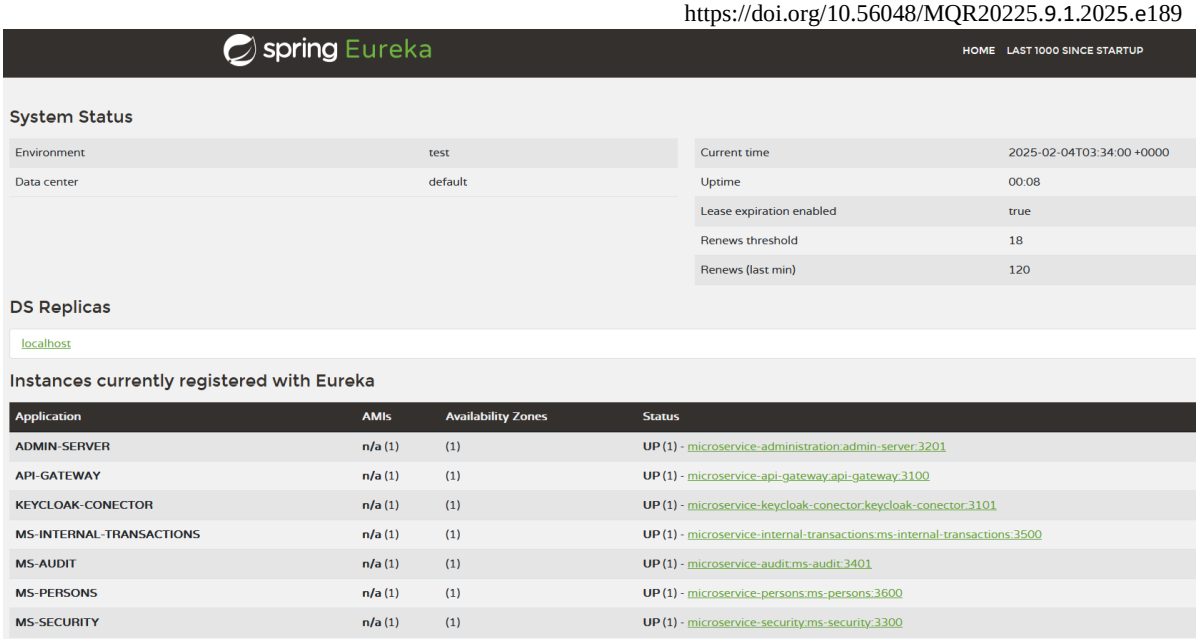
Dominios y Microservicios.

Dominio	Microservicios	Descripción
Infrastructure-Domain	ApiGateway, EurekaServer, KeycloakAdapter, ConfigServer, AdminServer	Dominio de Infraestructura: Manejo de autenticación, monitoreo, balanceo de carga y configuración.
Transactional-Domain	persons, security, transactions, etc.	Dominio Transaccional: Servicios específicos del modelo de negocio para transacciones acorde al ámbito bancario.

Los microservicios pueden escalar automáticamente registrándose en el microservicio de Eureka (figura 3) para poder ser consultados por el microservicio del balanceador Api-Gateway que conduce las peticiones a sus respectivos contenedores desplegados (figura 4) después de conocer el estado del mismo en tiempo real:

Figura 3

Registro de Microservicios en Eureka.



Fuente: Fuente Propia

Figura 4

Microservicios desplegados con Docker

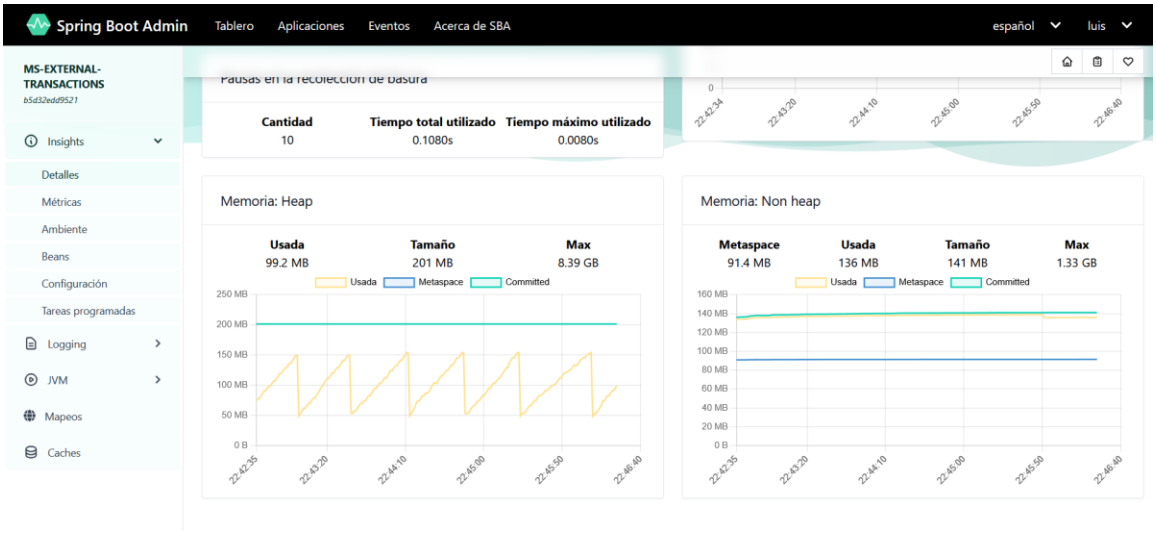
Name	Container ID	Image	Port(s)	CPU (%)
ms-eureka	95071ede4949	ms-docker-eureka-server	8761:8761	0.94%
mysql-database	6bf40239b348	mysql:8.1.0	3307:3306	0.5%
ms-api-gateway	36a5e0b2b073	ms-docker-apigateway:1	3100:3100	0.07%
ms-audit	7b6abf29bd14	ms-docker-audit:1.0	3400:3400	0.08%
ms-procedures-audit	b422f55d1245	ms-docker-procedures-a	3401:3401	0.12%
ms-external-transactic	9511f1e3ab36	ms-docker-external-trans	3700:3700	0.79%
ms-security	c9472b716095	ms-docker-security:1.0	3300:3300	0.08%
ms-auditory-security	35431e49debc	ms-docker-auditory-secu	3301:3301	0.44%
ms-administration	1130cb0cc4f2	ms-docker-adminserver:	3201:3201	1.26%

Fuente: Fuente Propia

Para controlar el estado de los contenedores se usa el microservicio de Admin-Server que usa la tecnología de Spring-Admin con el cual se puede verificar la salud de todos los microservicios, las métricas de control, propiedades, conexiones, recursos utilizados y disponibles, tareas programadas, logs generados, hilos de ejecución de los procesos, APIs mapeadas, cache manejado y los eventos del microservicio como se evidencia en la figura 5:

Figura 5

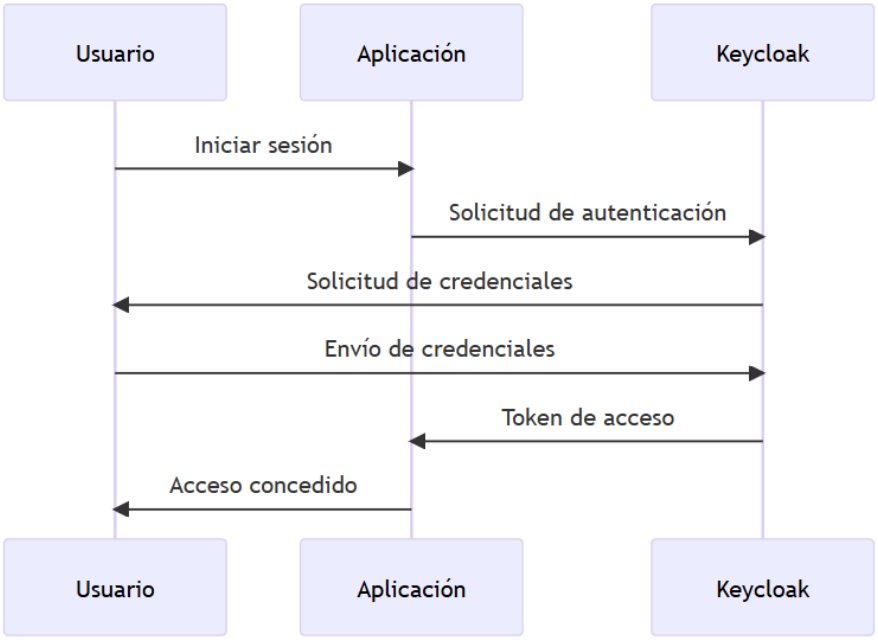
Análisis de Métricas en tiempo real de los microservicios.



Fuente: Fuente Propia

Figura 6

Flujos de Autenticación y Autorización con Keycloak.



Fuente: Fuente Propia

Manteniendo las buenas prácticas se cada microservicio presenta una documentación de acuerdo a los estándares de Open Api que permiten la revisión y consumo por parte del departamento de desarrollo.

Fase 4: Pruebas y Validación

Entre las validaciones de los puntos clave y las necesidades planteadas se puede denotar como se cumplió con los requerimientos levantados de acuerdo a las necesidades del sector financiero, las pruebas se realizaron dentro de un ambiente controlado asemejando a un servidor real con sistema operativo Linux-Debian y con un procesador de 8 núcleos junto con 32gb de RAM, las pruebas evidenciaron resultados los cuales se pueden observar en la tabla 8:

Tabla 8

Resultados de pruebas realizadas.

Tipo de Prueba	Objetivo	Herramienta Utilizada	Resultados Obtenidos	Criterio de Aceptación
Pruebas de Seguridad	Validar que los mecanismos de autenticación (OAuth2, Keycloak)	SonarQube, Burp Suite	Sin vulnerabilidades críticas detectadas. Se mejoró la configuración de Keycloak para mitigar ataques de fuerza bruta y rate limit para su mitigación.	Sin brechas de seguridad en OAuth2 y API Gateway.
Pruebas de Rendimiento	Evaluar la latencia y capacidad del sistema para procesar altas cargas transaccionales en tiempo real.	JMeter	98 ms de tiempo de respuesta promedio en transacciones críticas con una concurrencia de 1000 peticiones por segundo.	Inferior a 200 ms en transacciones críticas.
Pruebas de Escalabilidad	Validar el comportamiento de la arquitectura bajo diferentes niveles de carga y tráfico.	JMeter	Capacidad de atender hasta 1M de transacciones diarias sin degradación. Se observó un uso óptimo de recursos con escalado dinámico.	Soportar picos de carga sin pérdida de rendimiento.

Pruebas de Integración	Verificar la comunicación entre microservicios a través de API Gateway y Eureka Service Discovery.	Postman, Spring Boot Admin	Los microservicios se registran y comunican correctamente en Eureka. API Gateway maneja redirecciones y balanceo de carga de manera efectiva.	Comunicación estable sin errores de conexión.
Pruebas de Cumplimiento	Comprobar la alineación con normativas financieras como PCI DSS, ISO 27001 y regulaciones locales.	Análisis manual de cumplimiento, OWASP ZAP	El cifrado de datos cumple con los estándares PCI DSS e ISO 27001. Los logs de auditoría cumplen con la Superintendencia de Bancos.	Conformidad con normativas sin hallazgos críticos.

Fase 5: Documentación y Análisis de Resultados

El sistema transaccional desarrollado como prototipo está basado en Spring Cloud con API Gateway y Eureka Service Discovery para la gestión eficiente de microservicios. Se implementaron componentes de Keycloak para autenticación, Spring Admin para monitoreo y herramientas como Caffeine para garantizar alta disponibilidad, resiliencia y rendimiento óptimo. El enfoque modular permitió integrar microservicios dentro de un esquema de herencia de dependencias utilizando Maven con POMs centralizados, facilitando la escalabilidad y mantenimiento del sistema. De igual manera con la verificación de cumplimiento de los requerimientos funcionales y no funcionales definidos en la Fase 2. A continuación, se presenta en la tabla 9 el análisis basado en los resultados obtenidos para el cumplimiento de las necesidades del sector financiero:

Tabla 9

Análisis de resultados y cumplimiento.

Categoría	Requerimiento Funcional	Resultado	Nivel de Cumplimiento
Disponibilidad y Resiliencia	Implementar redundancia y failover automático.	Se configuran réplicas en Docker y balanceo de carga en API Gateway.	100%
Procesamiento y Rendimiento	Latencia menor a 200 ms.	Promedio de 98 ms en pruebas de carga con 1000 TPS.	100%

Escalabilidad	Capacidad de atender hasta 1M de transacciones diarias.	Se validó con escalado dinámico en Docker.	100%
Seguridad y Cumplimiento Normativo	Implementar autenticación fuerte (OAuth2, MFA, Keycloak).	Se aplican controles de acceso basados en roles manejados con los reinos de Keycloak.	100%
Auditoría y Regulaciones	Registro inmutable de transacciones.	Se implementan logs distribuidos y compatibilidad con normativas PCI DSS, ISO 27001 para acoplamiento dinámico y registro constante.	100%
Interoperabilidad	API Gateway y compatibilidad con Open Banking.	Se valida la integración con protocolos REST, gRPC y WebSockets.	100%
Monitoreo y Mantenimiento	Implementar dashboards centralizados.	Se usa Spring Boot Admin junto con actuator para el control de todos los microservicios.	100%
Automatización y Orquestación	Implementar CI/CD para despliegues sin downtime.	Se utiliza Docker compose para manejo de microservicios y despliegue inmediato manejando variables de entorno para apuntamiento entre entornos de desarrollo y productivos.	100%

Conclusiones

En conclusión, el desarrollo de este prototipo ha permitido validar el uso de tecnologías modernas en un entorno altamente regulado y de misión crítica. A través de una implementación basada en Spring Boot, Spring Cloud, Eureka Service Discovery, Keycloak y herramientas de monitoreo y seguridad avanzadas, se ha demostrado que es posible diseñar una plataforma robusta, segura y adaptable a los requerimientos del sector.

¿Cómo pueden los microservicios mejorar la seguridad de las transacciones financieras en el sector ecuatoriano? La arquitectura de microservicios aplicada en este prototipo permite segmentar y aislar funciones críticas, reduciendo la superficie de ataque y minimizando el impacto de fallos de seguridad. Además, la implementación de autenticación basada en OAuth2 y MFA, cifrado AES-256 y TLS 1.3, junto con la gestión de acceso con Keycloak, refuerza la seguridad de las transacciones. Estos mecanismos aseguran la integridad y confidencialidad de los datos, alineándose con estándares como ISO 27001 y PCI DSS.

¿Cuál es la viabilidad de implementar Keycloak como solución de autenticación centralizada en sistemas financieros locales? Los resultados obtenidos confirman que Keycloak es una solución viable y altamente efectiva para la autenticación centralizada en sistemas financieros locales. Su capacidad para gestionar usuarios, roles y sesiones de manera centralizada, junto con la integración de protocolos como OAuth2, OpenID Connect y SAML, permite una implementación escalable y adaptable a múltiples entidades financieras. Además, su compatibilidad con arquitecturas de microservicios facilita su integración con diferentes sistemas y plataformas bancarias.

¿Cómo puede garantizarse la interoperabilidad con los sistemas financieros existentes mediante una arquitectura de microservicios? La interoperabilidad se ha garantizado mediante la implementación de un API Gateway basado en Spring Cloud, el cual facilita la comunicación entre microservicios y sistemas heredados utilizando protocolos REST, gRPC y WebSockets. La estructura modular del prototipo permite la adaptación a los estándares de Open Banking, asegurando compatibilidad con plataformas financieras tradicionales y sistemas emergentes. La capacidad de descubrimiento dinámico de servicios con Eureka Service Discovery optimiza la integración de nuevos componentes sin afectar la operatividad del sistema.

El prototipo desarrollado demuestra que una arquitectura de microservicios bien estructurada puede mejorar significativamente la seguridad, escalabilidad e interoperabilidad de los sistemas transaccionales en el sector financiero ecuatoriano. Además, la integración de Keycloak como solución de autenticación centralizada proporciona una gestión segura y eficiente del acceso a los servicios. La capacidad del sistema para manejar altos volúmenes de transacciones sin degradación y su alineación con normativas internacionales lo convierten en una alternativa viable y estratégica para la modernización de la infraestructura financiera en Ecuador.

Las pruebas realizadas confirmaron que el sistema mantiene tiempos de respuesta alrededor de los 100 ms, incluso bajo alta concurrencia. Además, la arquitectura puede escalar automáticamente en función de la demanda, garantizando un desempeño óptimo en escenarios de alto tráfico, asegurando siempre una comunicación efectiva y conocimiento en tiempo real del estado de los microservicios con Eureka y Spring Boot Admin.

Referencias bibliográficas

Bermúdez Osorio, R. A., Tamayo Pino, A., & Patiño Hernández, J. C. (2019). Prototipo de librería para la implementación de integraciones usando arquitecturas orientadas a microservicios. https://n9.cl/q9ju4b

Chicaiza Rios, D. F. (2020). Diseño de un prototipo de una arquitectura basada en microservicios para la integración de aplicaciones Web altamente transaccionales. Caso: Entidades financieras. https://n9.cl/29hms

Dafri, W., & Al-Qaruty, R. (2023). Challenges and opportunities to enhance digital financial transformation in crisis management. Social Sciences & Humanities Open, 8(1), 100662. https://doi.org/https://doi.org/10.1016/j.ssaho.2023.100662

De Paz Estrada, J. M. (2019). Diseño e implementación de una arquitectura escalable basada en microservicios para un sistema de gestión de aprendizaje. Revista de la Escuela de Estudios de Postgrado. http://www.repositorio.usac.edu.gt/id/eprint/7023

Golovnin, M. Y. (2022). International Financial System: Global Trends and Qualitative change. Scientific Works of the Free Economic Society of Russia, 235, 95–104. https://doi.org/10.38197/2072-2060-2022-235-3-95-104

González Heredia, A., Ocharán Hernández, J. O., Arenas Valdés, M. de los Á., & Cortés Verdín, K. (2021). Prácticas de los equipos de desarrollo de microservicios: un mapeo sistemático de la literatura. ReCIBE, 10, C2-26. https://doi.org/10.32870/recibe.v10i1.218

Grody, A. (2018). Rebuilding financial industry infrastructure. Journal of Risk Management in Financial Institutions, 11, 34–46. https://doi.org/10.69554/CCDF1979

Guamán, D., Yaguachi, Lady, Cueva C., S., Jaramillo H., D., & Soto, F. (2018). Evaluación del rendimiento en el proceso de migración de una aplicación monolítica a microservicios. CISTI (Iberian Conference on Information Systems & Technologies / Conferência Ibérica de Sistemas e Tecnologias de Informação) Proceedings, 1–8. https://n9.cl/9ded3

Herrera, D., Pereira, W., Volochen, L., & Zárate Moreno, A. M. (2023). Open Finance in Latin America and the Caribbean: Great Opportunities, Large Challenges. Inter-American Development Bank. https://doi.org/10.18235/0004937

Ionaşcu, B.-D., & Aciobăniţei, I. (2024). Enriching an Open-Source Access Management Platform Using Multi-Factor Authentication. 2024 IEEE 18th International Symposium on Applied Computational Intelligence and Informatics (SACI), 383–388. https://doi.org/10.1109/SACI60582.2024.10619788

Mamani, C. A. L. (2023). Software Testing for Microservices. Innovation and Software, 4(1). https://doi.org/10.48168/innosoft.s11.a86

Mazzara, M., Dragoni, N., Bucchiarone, A., Giaretta, A., Larsen, S. T., & Dustdar, S. (2021). Microservices: Migration of a Mission Critical System. IEEE Transactions on Services Computing, 14(5), 1464–1477. https://doi.org/10.1109/TSC.2018.2889087

Mendoza, F., Toapanta, M., Andrade, C., Tandazo, M., Roció, M., Arellano, M., Caucha Morales, L. J., Romero Izurieta, R., & Orizaga Trejo, J. A. (2022). Prototipo de seguridad para el Banco Central del Ecuador en Blockchain Híbrido. Revista Conectividad, 3. https://doi.org/10.37431/conectividad.v3i2.38

Mohottige, T. I., Polyvyanyy, A., Buyya, R., Fidge, C., & Barros, A. (2024). Microservices-based Software Systems Reengineering: State-of-the-Art and Future Directions. https://arxiv.org/abs/2407.13915

Mortágua, D., Zúquete, A., & Salvador, P. (2024). Enhancing 802.1X authentication with identity providers using EAP-OAUTH and OAuth 2.0. Computer Networks, 244, 110337. https://doi.org/https://doi.org/10.1016/j.comnet.2024.110337

Nebel, A. (2019). Arquitectura de microservicios para plataformas de integración. En COLIBRI. UR.FI.INCO. https://n9.cl/42ugw

Nikolić, L. (2023). Challenges of digitalization of financial transactions. Zbornik radova Pravnog fakulteta Nis, 62(98), 51–71. https://doi.org/10.5937/zrpfn1-44231

Resolución No. JPRF-F-2023-076-normativa-Fintech, Pub. L. No. JPRF-F-2023-076, 1 (2023).

Sekar, R. R., Masna, A., Sharma, S., Abraham, A., & Pagilla, P. R. (2024). Decentralized Identity and Access Management (IAM) Using Blockchain. 2024 International Conference on Intelligent Systems for Cybersecurity (ISCS), 1–6. https://doi.org/10.1109/ISCS61804.2024.10581159

Solano Fernández, E., & Porras Alfaro, D. (2020). El modelo iterativo e incremental para el desarrollo de la aplicación de realidad aumentada Amón_RA. Revista Tecnología en Marcha, 33(8), Pág. 165-177. https://doi.org/10.18845/tm.v33i8.5518

Tatiana Gómez Suárez, K., Anaya, R., & Cano, A. F. (2017). Un acercamiento a los microservicios. UNACIENCIA: Revista de Estudios e Investigaciones, 10, 116–126. https://n9.cl/0e05k

Venčkauskas, A., Kukta, D., Grigaliūnas, Š., & Brūzgienė, R. (2023). Enhancing Microservices Security with Token-Based Access Control Method. Sensors, 23(6). https://doi.org/10.3390/s23063363

Conflicto de intereses:
Los autores declaran que no existe conflicto de interés posible.
Financiamiento:
No existió asistencia financiera de partes externas al presente artículo.
Agradecimiento:
N/A
Nota:
El artículo no es producto de una publicación anterior.