UPN
UNIVERSIDAD PRIVADA DEL NORTE

FACULTAD DE INGENIERÍA

Carrera de Ingeniería de Sistemas Computacionales

“DISEÑO E IMPLEMENTACIÓN DE MICROSERVICIOS EN JAVA CON SPRING BOOT PARA LA GESTIÓN DE TRANSACCIONES BANCARIAS EN LA EMPRESA SOCIUS PERÚ SAC, 2024”

Trabajo de suficiencia profesional para optar al título profesional de:

Ingeniero de Sistemas Computacionales

Autores:

Camila Tamara Cruz Diaz

Saul David Ramos Pacheco

Asesora:

Dra. Ing. Laura Sofía Bazán Díaz

https://orcid.org/0000-0001-6377-8328

Lima - **Perú**

2025

Informe de Similitud

Dedicatoria

En primer lugar, a Dios, por haberme dado la fortaleza y ayudado a terminar este proyecto de vida.

A mis pacientes padres, que han hecho posible todo lo que he conseguido, por brindarme todo su amor, comprensión, consejos y motivación a lo largo de este trayecto.

Camila

En primer lugar, a Dios, por brindarme la posibilidad de cumplir uno de mis más grandes deseos.

A mi familia, por su amor puro, seguridad y respaldo ininterrumpido. A mis padres, que con su empeño y entrega me orientaron en este trayecto y me incentivaron a continuar.

Saul

Agradecimiento

A los docentes de la carrera de Ingeniería de Sistemas Computacionales de la "Universidad Privada del Norte" por habernos brindado la orientación correcta durante nuestros estudios.

Un especial agradecimiento a la Dra. Ing. Laura Sofia Bazán Díaz por la valiosa orientación que ha proporcionado en la construcción de este estudio.

Y así mismo nuestras familias que nos han brindado apoyo incondicional, ayuda y cooperación en el desarrollo de esta investigación.

Índice de tablas

Índice de Figuras

RESUMEN EJECUTIVO

En la compañía se presentaba un problema relacionado con los tiempos de respuesta inicial vinculados a los cambios de divisa para la gestión de transacciones bancarias. Por ello, se buscó la creación e implementación de microservicios en Java con Spring Boot para solucionar estos problemas críticos. El cambio de divisa suele ser lento debido a la complejidad e integración del código. Al migrar a microservicios, esta funcionalidad se transforma en un servicio autónomo, mejorado con Spring Boot para conservar tasas de cambio constantes y patrones asíncronos como la implementación de Kafka para acelerar el procesamiento.

Además, el procesamiento y filtrado de divisas se facilitó al implementar un microservicio especializado que verifica y transforma divisas de forma eficaz, incorporando reglas de negocio y validaciones en tiempo real.

El uso de Spring Cloud simplificó la administración de configuración, resistencia y balanceo de carga, mientras que herramientas como Spring Boot Actuator posibilitaron la supervisión y mejora del rendimiento. Este enfoque incrementó la escalabilidad, disminuyó los tiempos de respuesta y garantizó un procesamiento de las transacciones bancarias más sólido y seguro.

CAPÍTULO I. INTRODUCCIÓN

SOCIUS PERÚ S.A.C. es una empresa dedicada a complementar equipos TI con una visión estratégica y práctica siempre a la vanguardia en la tecnología, acompañan y dan soluciones a los desafíos de diversos clientes agregando valor a sus negocios adaptando procesos digitales en el sector de banca, seguros y afines.

La empresa posee un grupo de expertos con más de dos décadas de trayectoria en la implementación y adopción de arquitecturas y plataformas tecnológicas en compañías del sector financiero, seguros y previsional (Socius Perú S.A.C, 2025).

En su evolución, involucró un desarrollo sostenido y se consolidó como un proveedor confiable para su extensa variedad de clientes industriales, con un método de aprendizaje constante, práctico y escalable que se ajusta a las demandas empresariales.

Se dedica a Core Banking, Finanzas, Seguros, Medios de pago, Pago a empresas, entre otras áreas, destacando por sus características de calidad de servicio, confiabilidad, cumplimiento y eficiencia.

A continuación, en la figura 1 se muestra la misión y visión de la empresa.

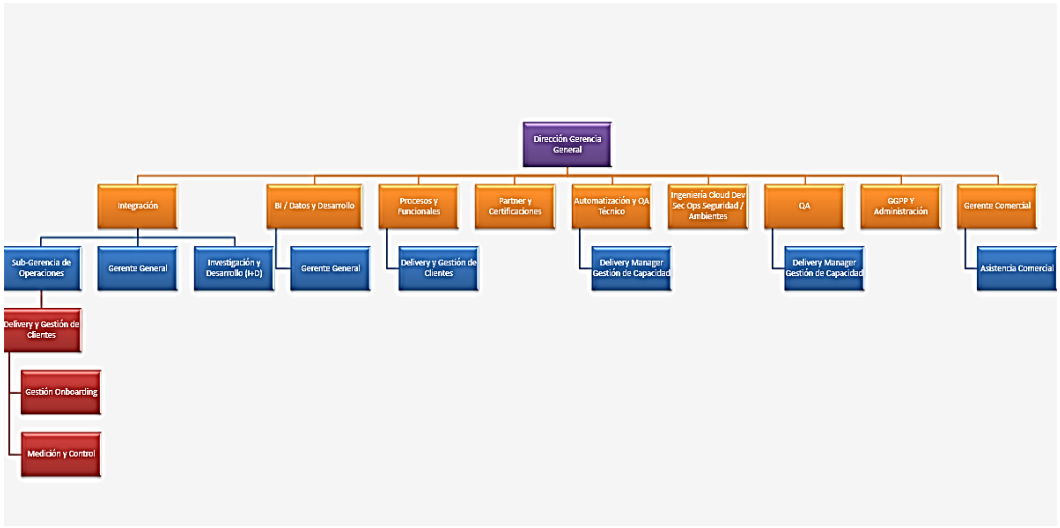
Figura 1
Misión y Visión de la empresa

Nota. Fuente: Socius Perú S.A.C. (2025).

En SOCIUS PERÚ S.A.C. se proporciona una amplia gama de servicios técnicos para los sectores de finanzas, seguros y pensiones, incluido el desarrollo e integración de los sistemas bancarios: (1) Desarrollo e integración de sistemas bancarios, (2) Implementación de la arquitectura de microservicio, (3) Modernización de sistemas heredados, (4) Asesoramiento sobre la conversión digital, y (5) Implementación de seguridad y cumplimiento normativo en transacciones digitales.

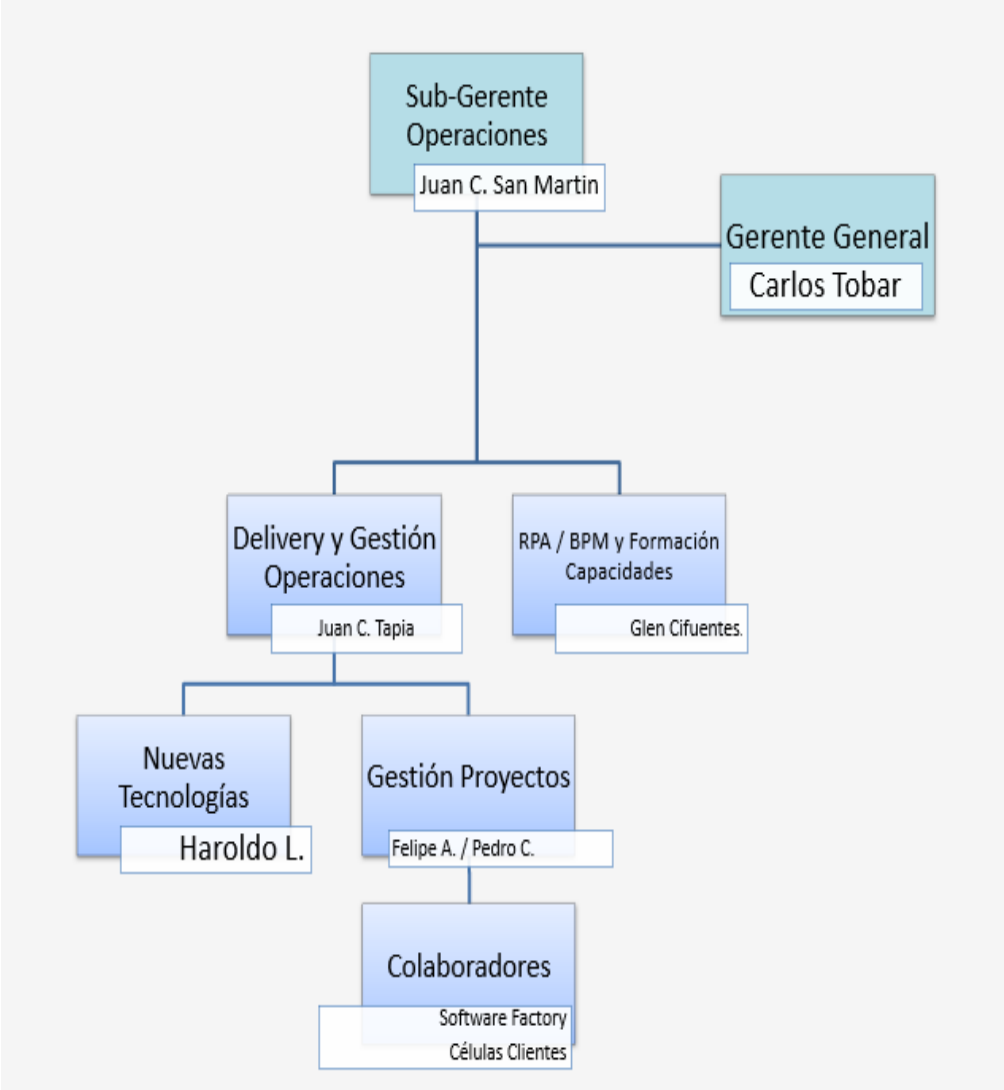
En la Figura 2, se muestra el organigrama de la organización, destacando su estructura jerárquica y funcional. Se incluyen áreas clave como Dirección, Gerencia General, Integración, Subgerencias de Operaciones, Desarrollo de BI/Datos, Procesos y Funcionales, Partner y Certificaciones, Automatización y QA, entre otras. Cada sección define roles específicos para garantizar la eficiencia operativa y el cumplimiento de los objetivos estratégicos de la empresa.

Figura 2
Organigrama de la empresa



La Figura 3, presenta un cuadro de entidades que destacan las estructuras jerárquicas y funcionales. Incorpora áreas fundamentales como la administración general, la sugestión, la entrega y administración operativa, RPA/BPM, formación competitiva, tecnología emergente y diversos departamentos operativos como la administración de proyectos. Cada sección define roles específicos para garantizar la eficiencia y el cumplimiento de los objetivos estratégicos de la Compañía y asignar responsabilidades claras a los gerentes y empleados en áreas como el software de fábrica y las celdas de los clientes.

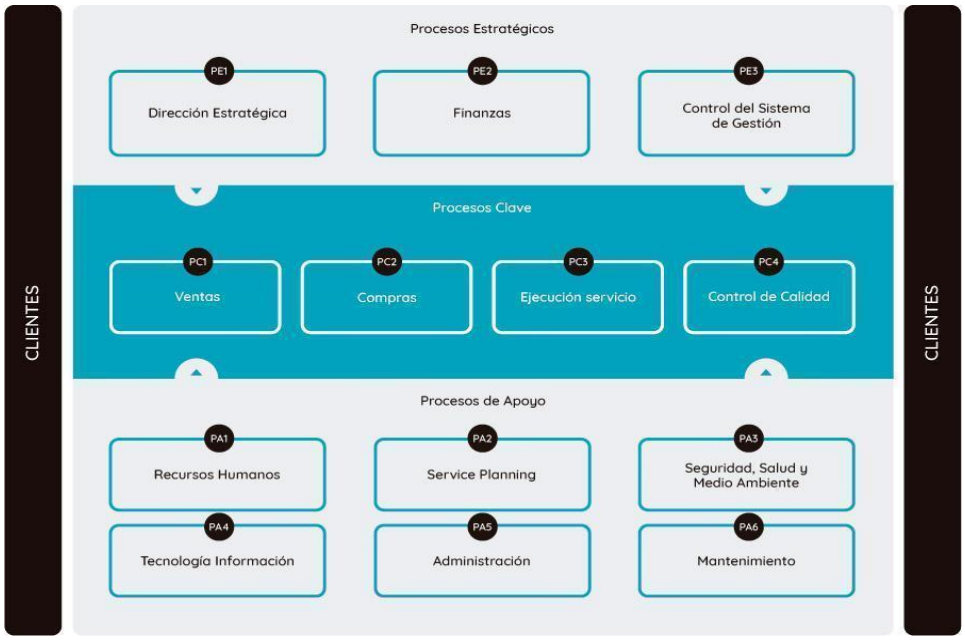
Figura 3
Organización del Área de Operaciones



Nota. Fuente: Socius Perú S.A.C. (2025).

A continuación, en la Figura 4 se muestra el mapa de procesos de la empresa, estructurado en procesos estratégicos, clave y de apoyo, destacando áreas como Dirección Estratégica, Ventas, Compras, Ejecución del Servicio, Recursos Humanos, y en la Tabla 1 se muestran los principales clientes de Socius.

Figura 4
Mapa de procesos



Nota. Fuente: Socius Perú S.A.C. (2025).

Tabla 1
Principales Clientes Socius

ÍTEM	CLIENTES	RUBRO	SECTOR	PAÍS
1	Banco Santander, S.A.	Banco	Financiero	España / América Latina
2	Banco de Crédito e Inversiones (BCI)	Banco	Financiero	Perú
3	AFP Provida S.A. (MetLife)	Fondos de Pensiones	Seguros	Chile
4	BUPA Chile S.A.	Salud	Salud	Reino Unido / Chile

Nota: Esta tabla muestra los principales clientes de Socius, organizados por razón social, rubro, sector y país, evidenciando su alcance en diversas industrias y regiones.

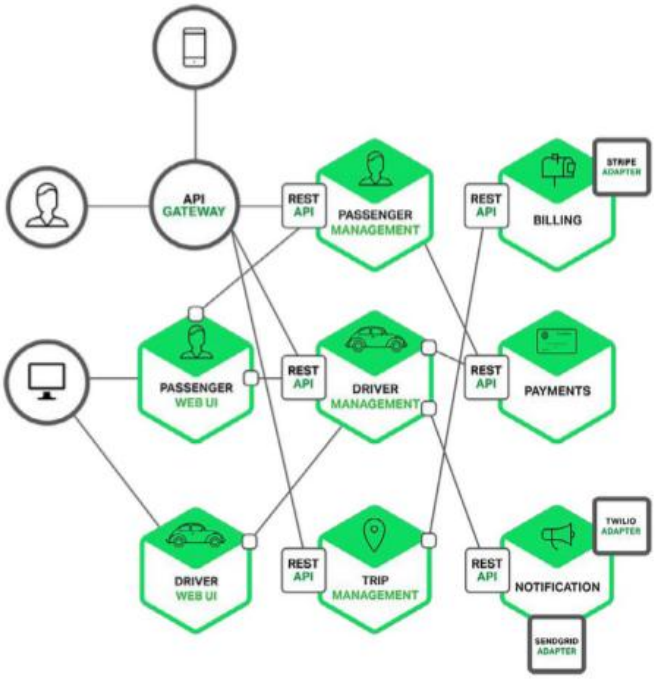
CAPÍTULO II. MARCO TEÓRICO

Durante la elaboración del marco teórico, se tomó en cuenta un saber práctico crucial vinculado al diseño e implementación de microservicios Java con botas de resorte utilizadas en la administración de transacciones bancarias. Esta información abarca 1) la arquitectura de microservicio, 2) la gestión y renovación de estructuras de datos, y 3) la seguridad y gestión de usuarios.

La **Arquitectura de Microservicios**, comprende las entidades independientes que operan de manera autónoma en infraestructuras orientadas a proporcionar plataformas como servicio (PAAS) o en su propio sistema. Se lleva a cabo la comunicación entre servicios mediante llamadas de red para fortalecer la separación y prevenir el acoplamiento. Se pone el producto por encima del proyecto, en el que los equipos generarán soluciones a lo largo del ciclo de vida. Los microservicios proporcionan variedad tecnológica, facilitando la solución de problemas particulares y fomentando la experimentación, al mismo tiempo que proponen la implementación de limitaciones técnicas (Contreras, 2018).

A continuación, cada servicio producido poseerá su propia arquitectura vinculado con otros, como se muestra en la Figura 5.

Figura 5
Arquitectura de microservicios



Nota. Fuente: Chris Richardson (2023).

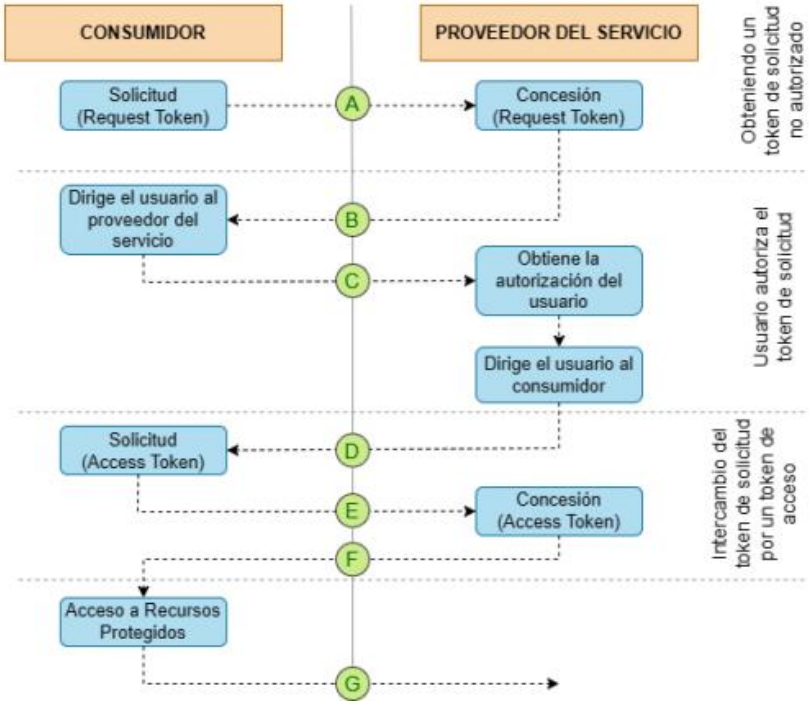
La **Gestión y actualización de estructuras de datos**, así como la mejora constante en los procesos de negocios, es una demanda permanente, particularmente en el desarrollo de software, donde nuevas tecnologías surgen constantemente. Uno de los avances fundamentales es la virtualización de ambientes de desarrollo, que mejora el tiempo de los programadores al suprimir la administración manual de sistemas operativos en cada proyecto. En este escenario, Docker emerge como una de las soluciones más destacadas, facilitando la estandarización y aceleración de los procesos de desarrollo, disminuyendo recursos y potenciando la eficacia dentro de la organización. (González, 2017)

La **seguridad y administración de usuarios** en arquitecturas de microservicios son comunes en muchas empresas que buscan dividir las funcionalidades en el desarrollo de software. Sin embargo, presentan retos,

especialmente en términos de seguridad. OAuth es una opción para la autenticación y autorización en este entorno, aunque tiene debilidades que requieren atención. Las amenazas y consideraciones de seguridad están detalladas en el documento RFC-6819. Este trabajo planea, diseña, desarrolla y ejecuta un entorno de microservicios que utiliza OAuth 2.0, analizando y configurando la seguridad según RFC-6819 (Andrés & Chiriboga, 2023).

A continuación, se lleva a cabo el **procedimiento de autenticación y autorización** en microservicios a través de la transacción de tokens. El cliente pide un Request Token, el cual se reemplaza por un Access Token después de que el usuario haya autorizado. El Token de Acceso confirmado facilita el acceso a los recursos protegidos, como se muestra en la Figura 6.

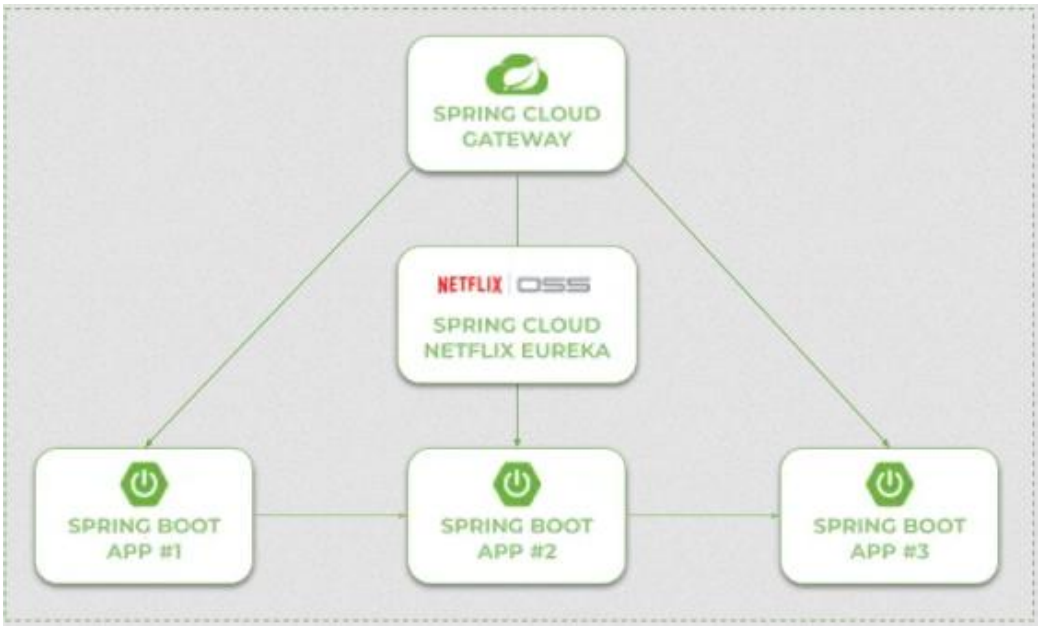
Figura 6
Flujo de Autorización de Token



Nota. Fuente: Andrés & Chiriboga (2023).

Spring Boot, el progreso de las tecnologías web es acelerado, y los marcos de trabajo son fundamentales en este progreso. Spring Boot, una ampliación de Spring Framework ha sobresalido como uno de los mejores instrumentos para la creación de aplicaciones web en Java. Su principal ventaja es la facilidad para crear microservicios, que son usados por grandes empresas como Netflix (Figura 7), Amazon y Google. Este trabajo de fin de máster investiga y analiza Spring Boot como un framework para el desarrollo de aplicaciones web empresariales, abordando su influencia en las tecnologías web, sus ventajas, debilidades y características. Además, se ha desarrollado una aplicación de gestión de eventos utilizando un modelo de microservicios para demostrar las cualidades de Spring Boot (Ramírez, 2020).

Figura 7
Arquitectura de Microservicios con Spring Cloud Gateway y Netflix Eureka

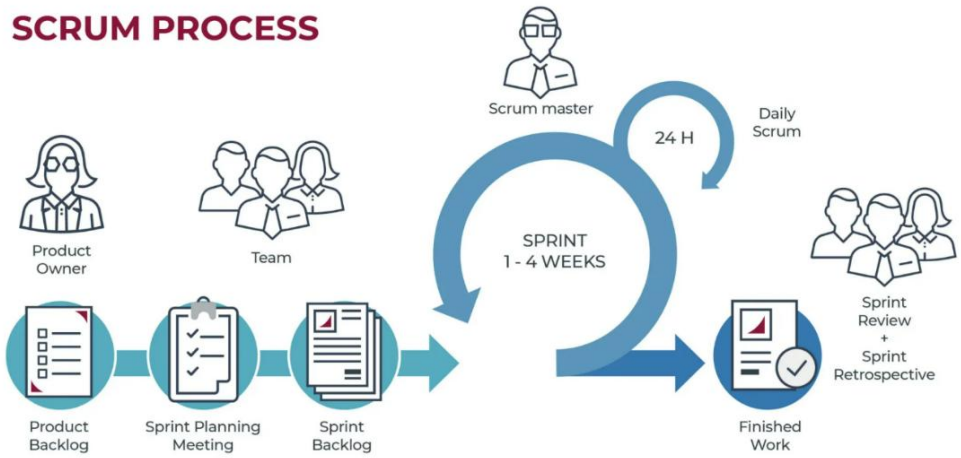


Nota. Fuente: Spring Microservicios (2005).

Scrum, como una metodología ágil de desarrollo de software, es un marco de trabajo que prioriza el trabajo en equipo, la autonomía y la colaboración,

enfocándose en entregas incrementales y adaptativas, basado en la transparencia, inspección y adaptación, Scrum permite controlar el progreso del software mientras el cliente define prioridades y el equipo se autoorganiza para alcanzar los objetivos. A pesar de que fomenta una arquitectura emergente, se aconseja establecer una arquitectura inicial (Sprint 0) para orientar el proyecto (Figura 8), particularmente en contextos complejos. Scrum es eficaz en la asignación de tareas, monitoreo de progresos y modificaciones constantes, lo que lo convierte en perfecto para proyectos con tiempos breves y modificaciones constantes, disminuyendo gastos y potenciando la competitividad, incluso en escenarios especializados como aplicaciones geográficas o proyectos tecnológicos (Velasco, 2021).

Figura 8
Metodología Scrum para el desarrollo de software

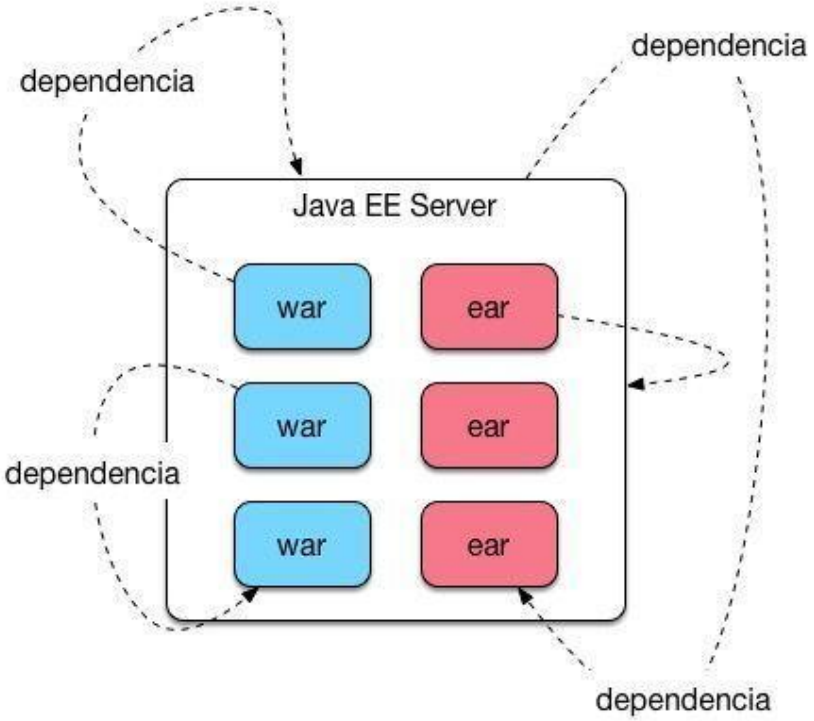


Nota. Fuente: Initium Software (2005).

Java sin duda es uno de los lenguajes de programación más populares en el mundo con su enfoque en la programación orientada a objetos. Además, cuenta con una gran cantidad de bibliotecas que simplifican la creación de aplicaciones potentes y escalables, ya sea para escritorio, web o dispositivos móviles (Figura 9). Su flexibilidad y robustez lo han convertido en una herramienta esencial para

desarrolladores en todo el mundo. Su sintaxis sencilla, comparable a C++, y la capacidad de ejecutarse en cualquier sistema operativo gracias a la JVM, lo convierten en una opción all inclusive. Aunque enfrenta desafíos de compatibilidad entre versiones, Java sigue siendo crucial en el desarrollo de sistemas complejos, aplicaciones web (usando el modelo MVC) y applets, manteniendo su relevancia en la period genuine de tecnologías emergentes (Moreno, 2015).

Figura 9
Estructura de despliegue y dependencias en un servidor Java EE



Nota. Fuente: Arquitectura java (2016).

La **base de datos (BD)**, se asemeja a la construcción de un hogar, puesto que necesita planos (gráficos) que orienten su evolución y faciliten futuras alteraciones. Es crucial que los programadores entiendan el dominio del conocimiento del cliente para obtener información pertinente y prevenir incertidumbres, reflejando todo en una especificación de requisitos. La fase de

diseño conlleva la creación de un modelo teórico y visual, una estructura lógica y una estructura física (aplicación en SQL), ajustándose a los diferentes Sistemas de Gestión de Bases de Datos (SGBD), independientemente de si sistemas monolíticos o de servidor. Es principal elaborar documentos y utilizar lenguas estándar para asegurar la transparencia del proyecto(Gómez, 2012).

GitHub, iniciar un proyecto de software requiere una estrategia que incluya repositorios en la nube con control de versiones, registro de incidencias, colaboración en equipo y versionamiento de código. GitHub, con millones de desarrolladores y repositorios, es la plataforma líder para compartir y reutilizar componentes de software. No obstante, hallar repositorios valiosos representa un reto debido al enorme volumen de datos. Para tratar este asunto, se sugiere OntoGitHubSearch, un modelo que emplea una ontología basada en la arquitectura de software y el procesamiento de lenguaje natural para categorizar repositorios de acuerdo con las demandas del cliente. Se realizó un análisis del modelo con programadores y arquitectos, evidenciando su efectividad a través de indicadores. Adicionalmente, se ha analizado GitHub a través de métodos de aprendizaje automático y análisis de redes sociales para detectar repositorios y grupos de desarrolladores, a pesar de que estos métodos no siempre toman en cuenta los intereses particulares del usuario. El modelo sugerido incluye una capa de búsqueda que emplea la API de GitHub, y una capa de resultados que organiza y categoriza los repositorios para su reaprovechamiento en proyectos nuevos o ya existentes (Ordóñez & Ordóñez, 2021).

SonarQube, la adopción de instrumentos como SonarQube para el análisis de la Deuda Técnica está en aumento rápido. Sin restricción, las empresas del distrito tenían dudas sobre la eficacia de las normas sugeridas por SonarQube. Por ello, se llevó a cabo un análisis empírico en 21 proyectos de código abierto para valorar su vínculo con la tendencia a fallos. Mediante el uso del algoritmo SZZ y la comparación de siete modelos de aprendizaje automático, se descubrió que únicamente 25 de las 202 reglas de SonarQube para Java muestran una reducida tendencia a fallos. Además, las anomalías categorizadas como "bugs" por SonarQube a menudo no están vinculadas a fallos, lo que indica que su modelo predictivo es ineficaz. Se aconseja que las compañías examinen meticulosamente las normativas que implementan (Lenarduzzi & Lomio, 2020).

Postman Client, es la plataforma líder de API a nivel global. Las características de Postman facilitan cada etapa de la elaboración de una API y aceleran la colaboración para contribuir a la creación de API más eficientes y veloces. Postman, lanzado en 2012, es un cliente destinado a la evaluación de API. Actualmente se utiliza principalmente para realizar solicitudes API REST de forma sencilla, esta administra nuestra API y la documenta (Monción, 2023).

Maven, es una infraestructura que puede ayudarle en la automatización de la construcción. Ayuda a los desarrolladores y coordinadores de construcción a integrar una amplia gama de requisitos de construcción y estándares en un solo sistema que es lo suficientemente flexible para su personalización (IBM, 2021).

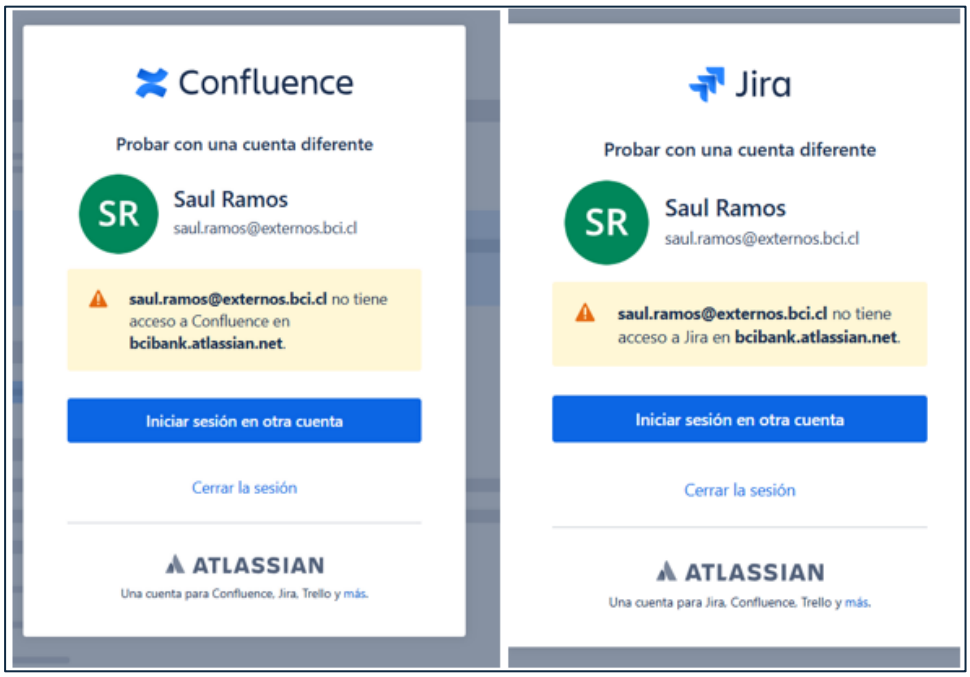
Git, es un sistema de gestión de versiones y gracias a ello los desarrolladores pueden recuperar todas sus versiones anteriores. Pero Git no solo permite eso,

abarca otras muchas posibilidades. Permite llevar en paralelo varias versiones del mismo software, por ejemplo, cuando un desarrollador está trabajando en una nueva funcionalidad, pero esta no debe ser integrada en el software final, también sirve como documentación completa (Dauzon, 2018).

Openshift, es una PaaS de Red Hat, hay tres versiones diferentes: OpenShift Origin, OpenShift Online y OpenShift Enterprise. OpenShift Origin, la versión de código abierto y gratuito de OpenShift, es el proyecto inicial para las otras dos versiones. Está en GitHub y se lanzó bajo una licencia de Apache 2. Todos los cambios en la base de código pasan por el repositorio público, tanto para Red Hat como para desarrolladores externos. Si se desea utilizar esta versión, se deberá instalar en una infraestructura propia (Rojas, 2019).

Es crucial destacar que, dentro de las **limitaciones**, surgieron retos importantes vinculados con la coordinación de horarios entre Perú y otras naciones, lo que dificultaba la coordinación del equipo de desarrollo. Además, la administración de permisos en los repositorios y las configuraciones de los equipos (Figura 10), que necesitaban satisfacer los rigurosos requisitos de seguridad establecidos por el banco, constituyeron barreras adicionales. El proceso de elaborar cuentas con los permisos correspondientes también resultó ser un procedimiento complejo que requirió una organización detallada. Para superar estas limitaciones, para la implementación y gestión de contenedores. Estas herramientas facilitaron el desarrollo de una solución eficiente, segura y escalable, optimizando las operaciones bancarias y garantizando la integridad del sistema de acuerdo con los criterios requeridos por el banco.

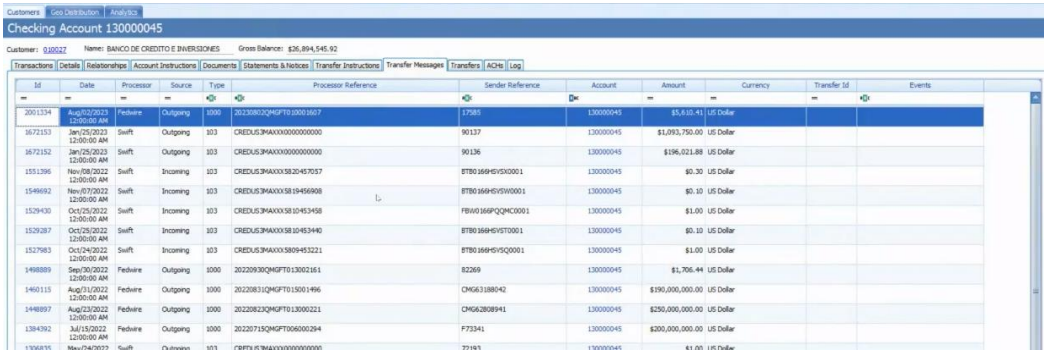
Figura 10
Acceso denegado



CAPÍTULO III. DESCRIPCIÓN DE LA EXPERIENCIA

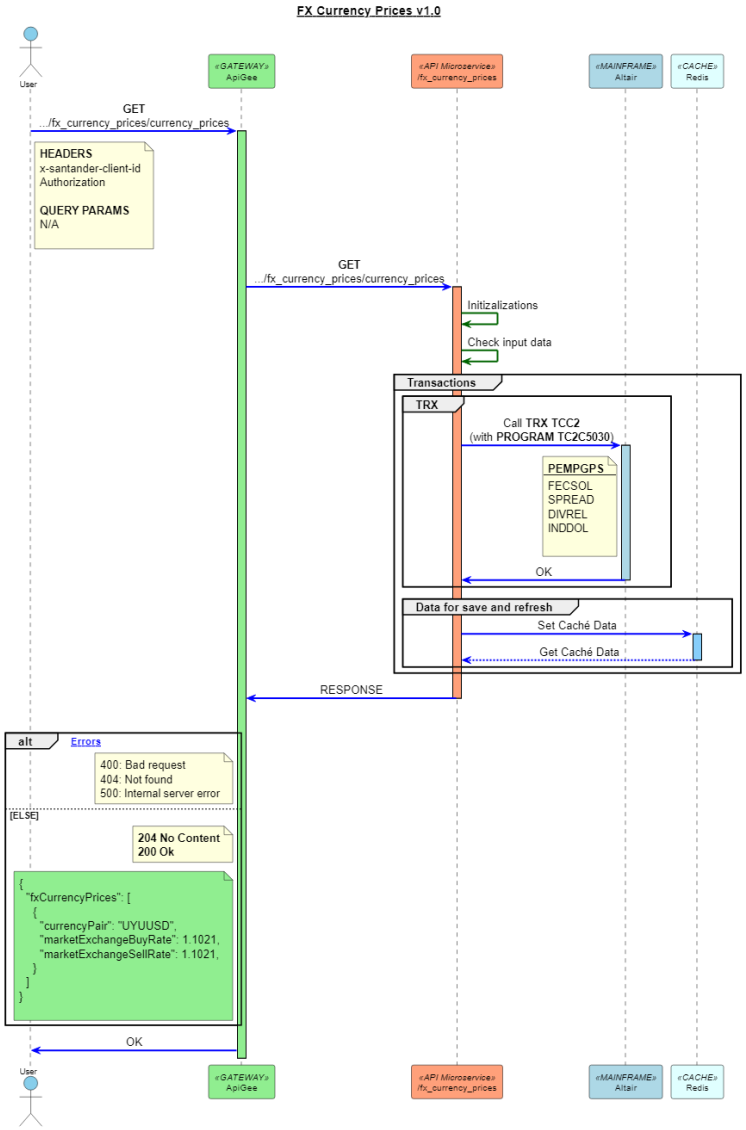
El proyecto se inició identificando el problema de la antigüedad de los datos sobre los precios de divisas, lo que afectaba la precisión de las operaciones financieras y la toma de decisiones en tiempo real. Además, los métodos internos de actualización y consulta de estos valores no eran eficientes, causando demoras e inconsistencias en la información (Figura 11). Para tratar este problema, se detectó la necesidad de perfeccionar los procedimientos de consulta y filtrado de divisas (Figura 12), incrementando así la eficacia del sistema. Como respuesta, en Spring Boot se creó un microservicio con Java y Maven, empleando la librería Darwin. Este microservicio ofrece la posibilidad de filtrar por moneda y examinar todas las existentes en tiempo real, mejorando la exactitud y agilidad en la administración de datos cambiarios.

Figura 11
Proceso de Divisa Antiguo



Nota. Fuente: Socius Perú S.A.C. (2025).

Figura 12
Proceso optimizado Divisas



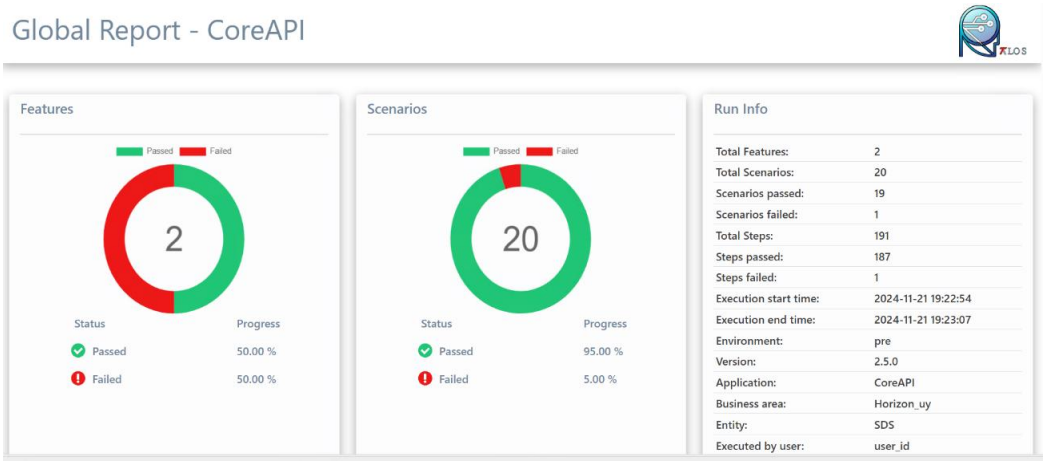
Nota. Fuente: Socius Perú S.A.C. (2025).

A comienzos del año 2023 realizamos nuestro ingreso a la empresa Socius Perú SAC, postulando a través de la bolsa de trabajo de la universidad que ya contaba con convenio en ese entonces. El proceso de ingreso ocurrió al recibir una llamada del reclutador, solicitando una entrevista con ellos y programando el encuentro. Nos sometieron a un proceso de selección donde nos cuestionaron acerca de nuestro saber teórico y nos llevaron a una entrevista personal; después de 2

semanas, recibimos un mensaje de WhatsApp informándonos de que habíamos sido seleccionados. Al comenzar nuestras tareas en la compañía, desempeñamos roles como desarrollador de integraciones inicialmente. Algunas de estas tareas incluían la creación de componentes de software a medida para diferentes clientes, la integración de sistemas a través de APIs, la administración y documentación de archivos técnicos, entre otros. Más adelante, hemos comenzado a interactuar con diferentes clientes con proyectos para colaborar directamente, como por ejemplo bancos y compañías de telecomunicaciones.

Juntamente con el superior Claudio Saravia, supervisor líder técnico, y Glen Cifuentes, el coordinador de proyecto, nos dedicamos a crear un microservicio para mejorar la respuesta en la búsqueda de divisas, dado que el sistema mostraba retrasos en el filtrado, lo que impactaba el tiempo de respuesta para los clientes en el proceso de cambio de divisa. Además, Socius Perú S.A.C. puso en marcha validaciones en el microservicio con el equipo de QA con el objetivo de optimizar la calidad y eficacia del sistema (Figura 13).

Figura 13
Pruebas de QA

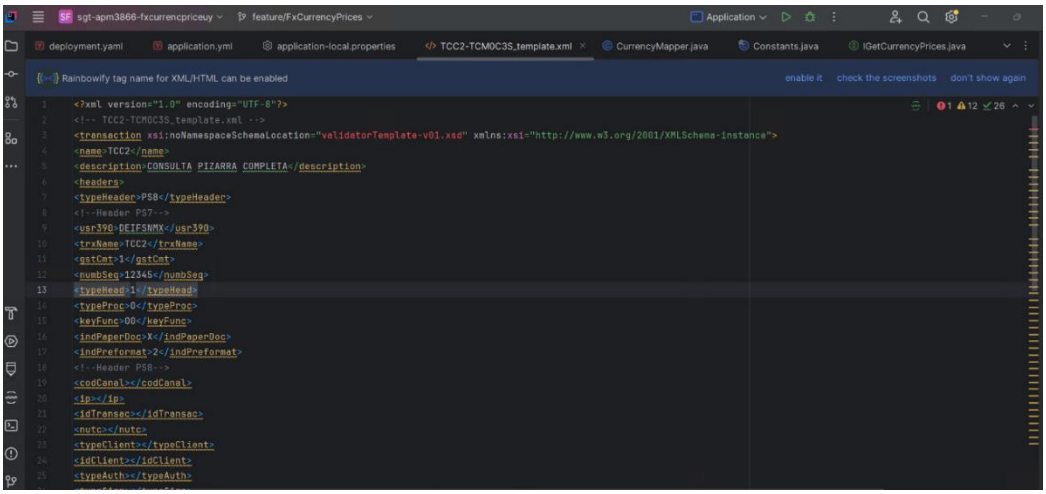


Nota. Fuente: Socius Perú S.A.C. (2025).

El proyecto se llevó a cabo en un inicio para determinar las metas a lograr con los microservicios (MS), incrementar el tiempo de respuesta en el cambio de divisas y perfeccionar el manejo de filtros de divisas en distintos sectores de la compañía. De esta manera queríamos cubrir las necesidades de cambio de divisas y brindar soluciones en tiempo real a las áreas correspondientes.

Para este propósito, se desarrolló un MS que puede encontrar la divisa correcta del usuario en la base de datos utilizando la transacción "TCC2-TCM0C3S" (Figura 14). Se requiere un extracto de datos para aplicar el filtro de divisa.

Figura 14
Template de transacción TCC2-TCM0C3S



Nota. Fuente: Socius Perú S.A.C. (2025).

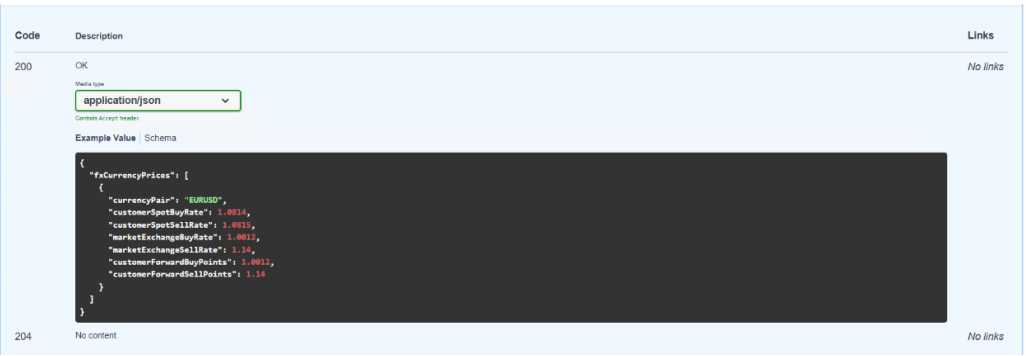
Los objetivos planteados fueron: 1) Identificar los problemas de tiempo de respuesta inicial relacionados con el cambio de divisa en el sistema heredado del banco. 2) Desarrollar MS utilizando Spring Boot, Java, Maven y bibliotecas Darwin basadas en la transacción TCC2-TCM0C3S.

Además, para mejorar la capacidad de respuesta del proyecto monetario, se modificó la estructura de respuesta a través de la configuración de Swagger (Figura 15). El proyecto se desarrolló en Spring Boot utilizando Java y Maven siguiendo una estructura de paquetes (Figura 16) que permite la aplicación de un patrón en capas.

Estas capas fueron seleccionadas porque contenían datos importantes sobre la compra de divisas para el área de ventas, tanto en la aplicación móvil como en la web. Posteriormente, se realizó la creación del Token JWT (Figura 17) para hacer más seguro el acceso. Además, se implementó una característica que determina de manera automática la fecha del día, lo que facilita la representación del cambio de divisas en los diferentes países.

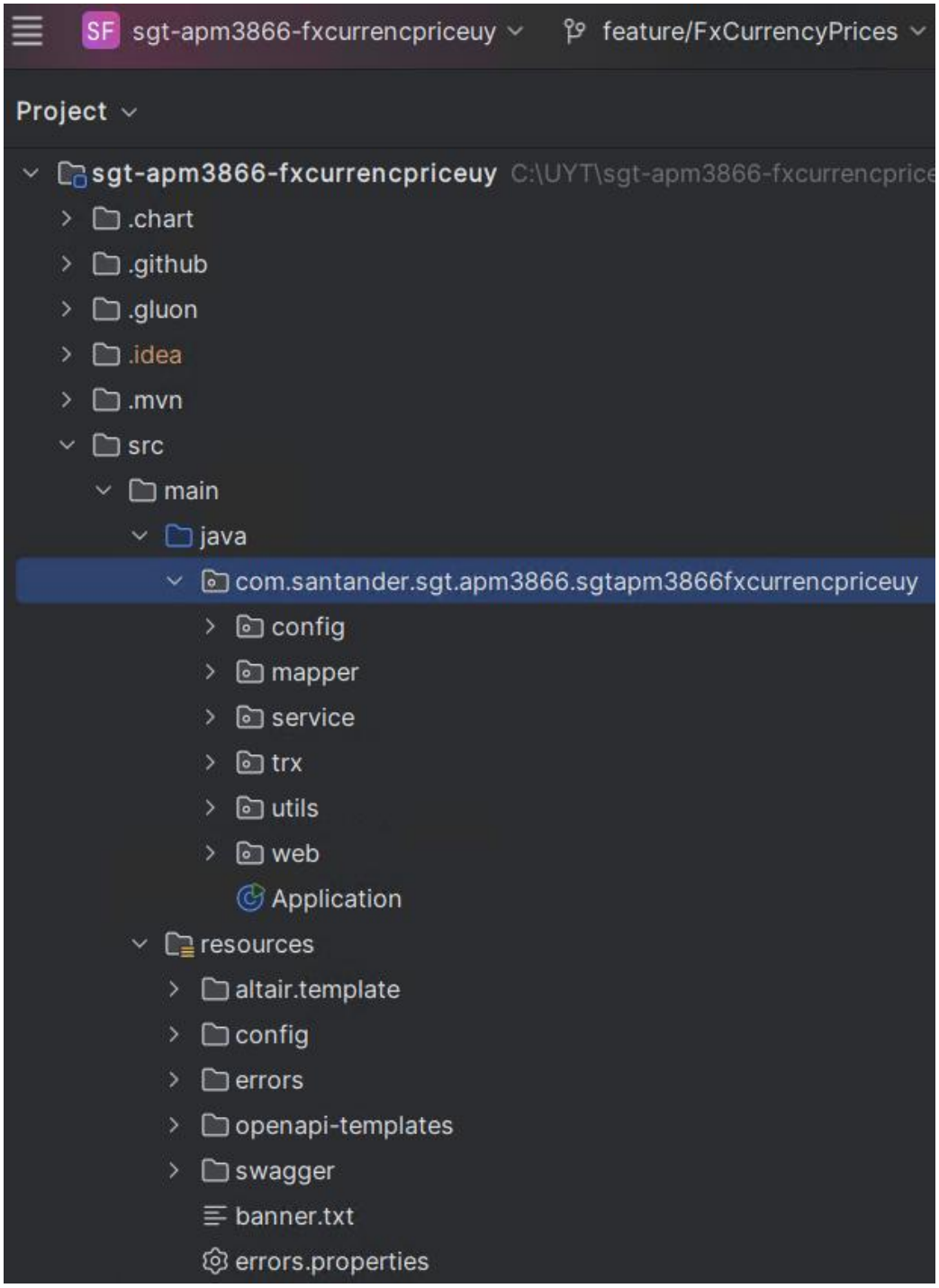
Así, el usuario podrá efectuar cambios de divisas de manera más ágil y eficaz, eligiendo la divisa del país pertinente. La información se presenta en forma de vista en Gluon (Figura 18). Por esta razón, se instauró un microservicio (MS) con el objetivo de mejorar el proceso de intercambio de moneda en el banco.

Figura 15
Estructura de respuesta Swagger



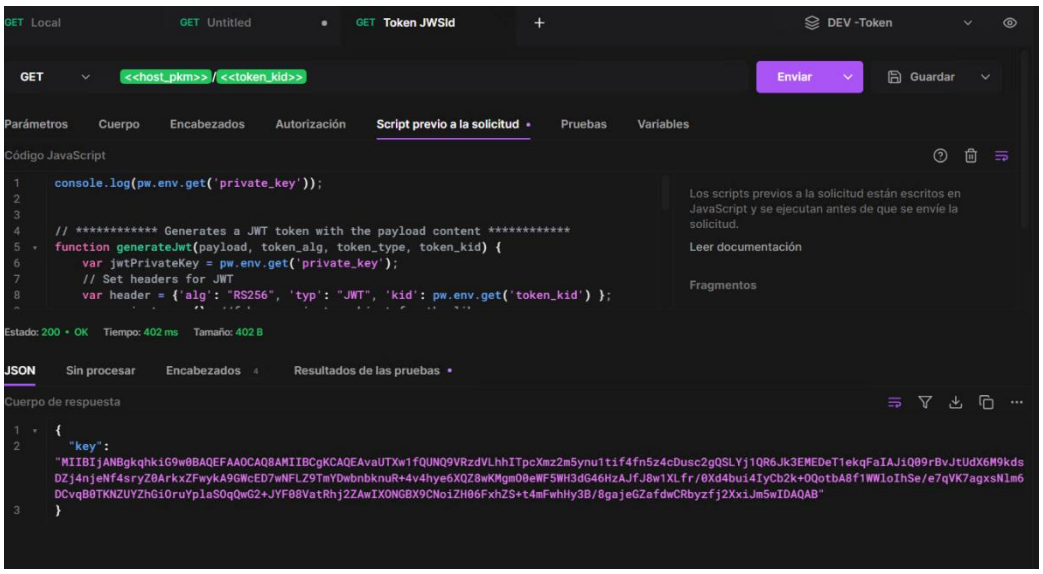
Nota. Fuente: Socius Perú S.A.C. (2025).

Figura 16
Estructura de patrón por capas



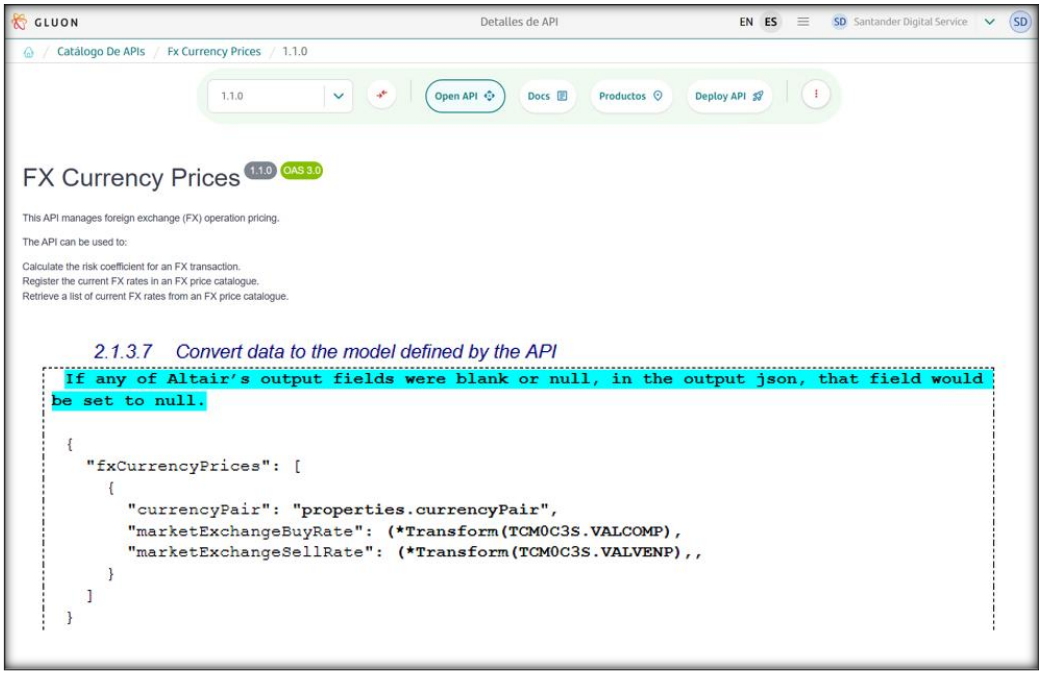
Nota. Fuente: Socius Perú S.A.C. (2025).

Figura 17
Creación del token JWT



Nota. Fuente: Socius Perú S.A.C. (2025).

Figura 18
Vista en Gluon



Nota. Fuente: Socius Perú S.A.C. (2025).

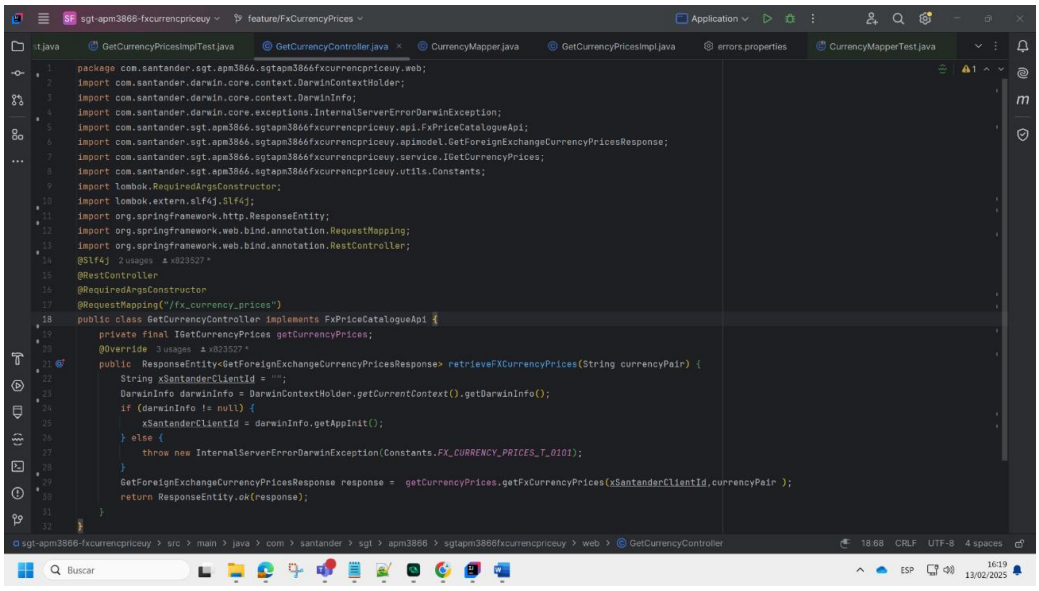
La primera versión del proyecto consistió en las siguientes estrategias de desarrollo: una aplicación desarrollada en Spring Boot con Java, Maven y la librería

Darwin (Figura 19). En este desarrollo, Spring Boot representa la sección del backend, donde se implementan las funciones, transacciones, condicionales, entre otros.

Es importante destacar que, al realizar este microservicio, se recibe los resultados en Postman, el cual presenta todas las monedas que fueron procesadas en la transacción. Además, se ha incorporado un filtro para asistirle en la elección de la moneda adecuada, lo cual resulta particularmente beneficioso al registrar diversas monedas.

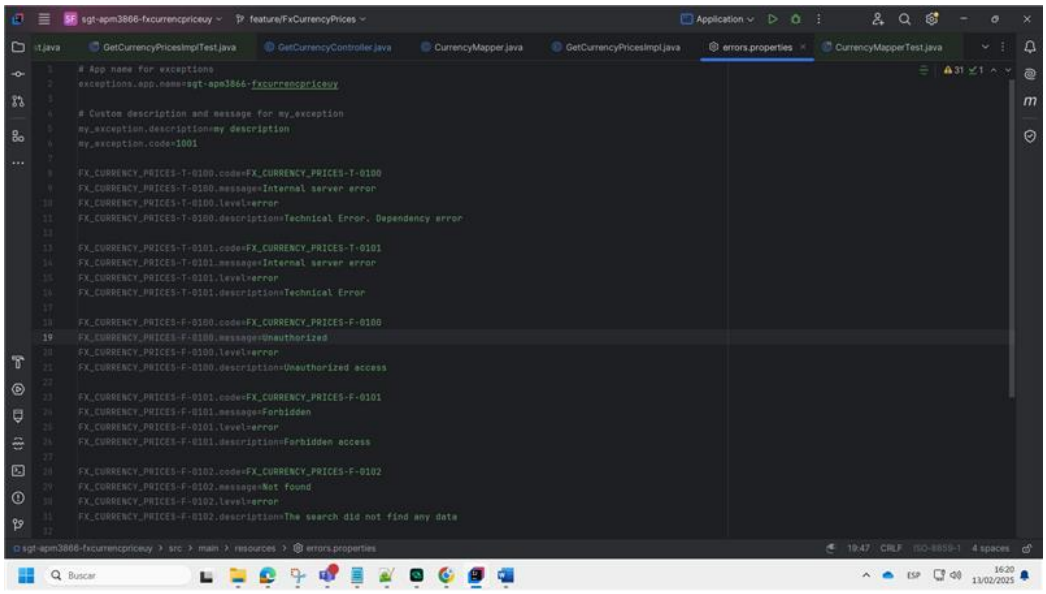
Además, se han eliminado las duplicaciones de monedas para mejorar los resultados. El liderazgo técnico reconoció la necesidad de incluir la validación de entrada y la generación de respuestas de error (Figura 20).

Figura 19
Spring Boot



Nota. Fuente: Socius Perú S.A.C. (2025).

Figura 20
Respuestas de error



Nota. Fuente: Socius Perú S.A.C. (2025).

Adicionalmente, se muestra la implementación de un filtro (Figura 21), que permite al usuario ingresar la divisa para buscar la moneda que necesita para su compra, ya sea mediante la web o la app. Además, el usuario podrá ver la tabla completa de las divisas que tengan transacciones y, mediante un parámetro indicado, podrá hacer uso del filtro de búsqueda de divisas.

Figura 21
Resultado general y buscador

Nota. Fuente: Socius Perú S.A.C. (2025).

CAPÍTULO IV. RESULTADOS

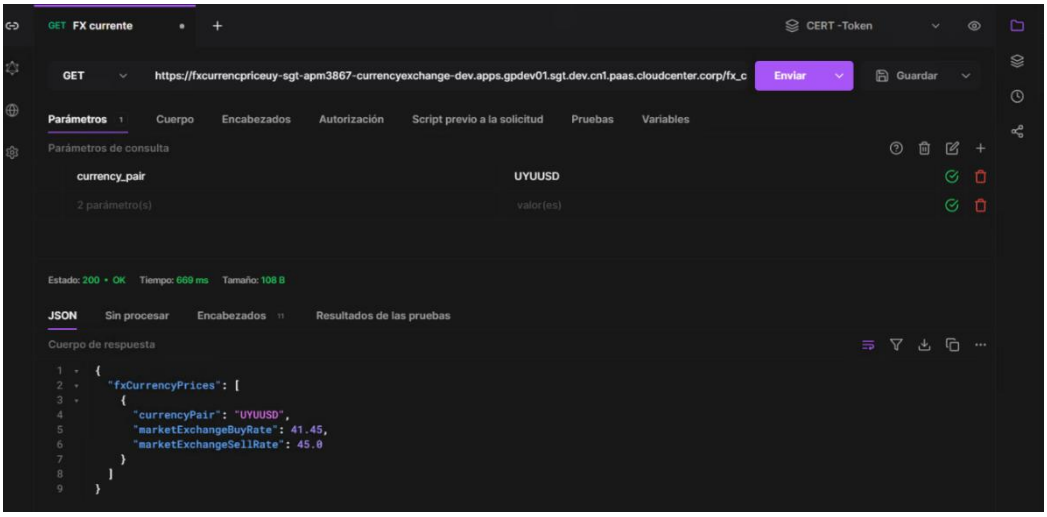
A continuación, se muestran los resultados obtenidos en relación con los objetivos específicos planteados:

Para el primer objetivo, que consiste en identificar los problemas de tiempo de respuesta inicial relacionados con el cambio de divisa en el sistema heredado del banco, se logró determinar que el sistema anterior presentaba demoras significativas en la consulta y procesamiento de divisas por lo cual, mediante el uso del Alfa de Cronbach, se evaluó la fiabilidad de los datos y se obtuvo un valor de 0.724, lo que indica un nivel de confiabilidad aceptable en la medición de los tiempos de respuesta.

Para el segundo objetivo, que consiste en desarrollar microservicios utilizando Spring Boot, Java, Maven y bibliotecas Darwin basadas en la transacción TCC2-TCM0C3S, se implementó un microservicio (MS) que mejora el filtrado y procesamiento de divisas, desplegado en Docker el cual, para perfeccionar la representación y gestión de las transacciones, se emplearon las tablas Código Divisa y Mercado de Cambio - Tasa de Compra.

El microservicio posibilita una filtración más eficaz de los datos proporcionados en la transacción, mostrando la divisa del país a través de un filtro (Figura 22).

Figura 22
Filtro divisa de dólar

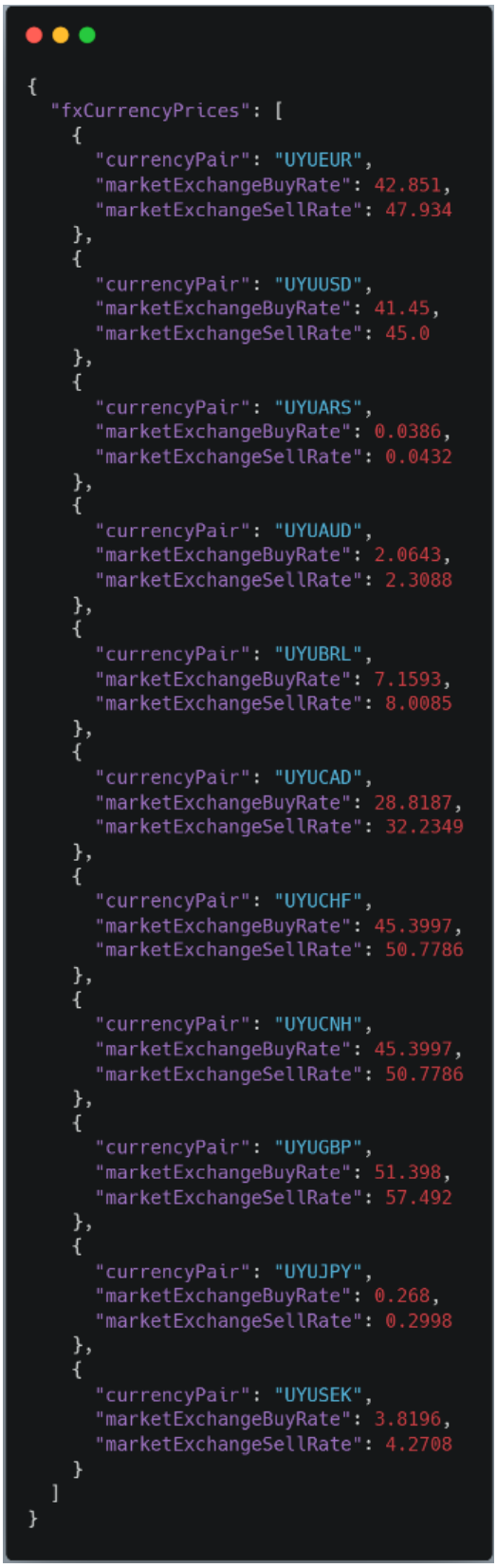


Nota. Fuente: Socius Perú S.A.C. (2025).

En esta parte se presentan todas las divisas involucradas en la transacción, así como el número de divisas de diversos países que gestiona el sistema (Figura 23).

Estos hallazgos demuestran que la puesta en marcha de la solución basada en microservicios facilita la optimización de la administración de divisas, consiguiendo un rendimiento y fiabilidad superior del sistema.

Figura 23
Lista divisas




```
{
  "fxCurrencyPrices": [
    {
      "currencyPair": "UYUEUR",
      "marketExchangeBuyRate": 42.851,
      "marketExchangeSellRate": 47.934
    },
    {
      "currencyPair": "UYUUSD",
      "marketExchangeBuyRate": 41.45,
      "marketExchangeSellRate": 45.0
    },
    {
      "currencyPair": "UYUARS",
      "marketExchangeBuyRate": 0.0386,
      "marketExchangeSellRate": 0.0432
    },
    {
      "currencyPair": "UYUAUD",
      "marketExchangeBuyRate": 2.0643,
      "marketExchangeSellRate": 2.3088
    },
    {
      "currencyPair": "UYUBRL",
      "marketExchangeBuyRate": 7.1593,
      "marketExchangeSellRate": 8.0085
    },
    {
      "currencyPair": "UYUCAD",
      "marketExchangeBuyRate": 28.8187,
      "marketExchangeSellRate": 32.2349
    },
    {
      "currencyPair": "UYUCHF",
      "marketExchangeBuyRate": 45.3997,
      "marketExchangeSellRate": 50.7786
    },
    {
      "currencyPair": "UYUCNH",
      "marketExchangeBuyRate": 45.3997,
      "marketExchangeSellRate": 50.7786
    },
    {
      "currencyPair": "UYUGBP",
      "marketExchangeBuyRate": 51.398,
      "marketExchangeSellRate": 57.492
    },
    {
      "currencyPair": "UYUJPY",
      "marketExchangeBuyRate": 0.268,
      "marketExchangeSellRate": 0.2998
    },
    {
      "currencyPair": "UYUSEK",
      "marketExchangeBuyRate": 3.8196,
      "marketExchangeSellRate": 4.2708
    }
  ]
}
```

Los resultados que se presentan a continuación son los obtenidos del Anexo 1, donde a través del caso Cronbach se pudo determinar que la fiabilidad de un examen o instrumento se relaciona con la consistencia de las notas obtenidas por las mismas personas en distintas situaciones o con diferentes grupos de reactivos equivalentes (Vega, 2019).

Para evaluar la confiabilidad de esta investigación, se utilizó el método del Alfa de Cronbach (figura 24). Según Quero (2010) , el coeficiente α de Cronbach permitió a los investigadores medir la consistencia interna de un instrumento basado en una escala Likert o cualquier otra escala de opciones múltiples. A lo largo del tiempo, han surgido diversas modificaciones a las fórmulas de Kuder y Richardson, sin embargo, el estadígrafo α de Cronbach ha sido el más ampliamente aceptado. En este caso, el cálculo obtenido arrojó un valor de 0.724, lo que indicó un nivel de confiabilidad aceptable en la medición de los ítems evaluados.

Figura 24
Confiabilidad

Items	I	II	II	IV	V	VI	VII	VIII	SUMA DE ITEMS
Sujetos									
Glen Cifuentes (1)	4	4	4	4	4	4	4	4	32
Claudio Saravia (2)	4	3	4	4	4	3	4	4	30
Lukas Muñoz (3)	3	3	3	3	3	3	3	3	24
Victor Martinez (4)	3	4	4	3	4	4	4	3	29
Ignacio Puentes (5)	4	3	3	3	3	3	3	3	25
Juan Ramirez (6)	4	3	2	3	2	3	3	4	24
Ariel Ponce (7)	4	3	3	3	3	3	3	3	25
Agustin Hidalgo (8)	4	4	3	3	4	4	4	4	30
VARP	0.25	0.3594	0.3594	0.5	0.3594	0.25	0.3594	0.25	7.3594

VAR = VARIANZA

K	8
K-1	7
SUMA VAR ITEMS	2.697
VAR TOTAL	7.3594

K = CANTIDAD DE ITEMS

ALFA	0.724

CAPÍTULO V. CONCLUSIONES Y RECOMENDACIONES

Conclusiones

Al concluir el proyecto se obtuvieron las siguientes conclusiones:

Para el objetivo de identificar los problemas de tiempo de respuesta inicial relacionados con el cambio de divisa en el sistema heredado del banco, el principal problema se originó debido a que el sistema anterior presentaba demoras significativas en el procesamiento y consulta de divisas, para ello se evaluó la fiabilidad de los datos mediante el uso del Alfa de Cronbach obteniendo un nivel 0.724 aceptable en la medición de los tiempos de respuesta.

Para el objetivo de desarrollar microservicios utilizando Spring Boot, Java, Maven y bibliotecas Darwin basadas en la transacción TCC2-TCM0C3S, se vio la necesidad de mejorar el filtrado y procesamiento de divisas siendo desplegadas en Docker mediante la implementación de microservicios.

Seguidamente, se exponen las siguientes lecciones adquiridas durante la ejecución del proyecto:

El proyecto evidenció la relevancia de construir una arquitectura de microservicios bien organizada y desacoplada, lo que facilitó la escala de cada componente de forma autónoma de acuerdo con la demanda, aplicando principios de Domain-Driven Design (DDD) y estableciendo las restricciones entre microservicios para prevenir dependencias innecesarias y de esa manera simplificar el mantenimiento.

La experiencia en seguridad y protección de datos sensibles a través de sistemas de seguridad garantizó que únicamente usuarios y clientes autorizados pudieran acceder a los microservicios establecidos, salvaguardando la información delicada de las transacciones bancarias.

La capacidad para optimizar el rendimiento y la eficiencia del sistema mediante el uso de balanceadores de carga mejoró significativamente el rendimiento del sistema, reduciendo los tiempos de respuesta y la carga en las bases de datos.

Recomendaciones

Es necesario diseñar microservicios altamente vinculados y débilmente acoplados, de esta forma se segmentan las funcionalidades en microservicios pequeños y especializados con el objetivo de simplificar el mantenimiento, la flexibilidad y la actualización autónoma de cada servicio.

Se recomienda establecer mecanismos de resiliencia y tolerancia a fallos mediante herramientas como Spring Cloud Circuit Breaker para gestionar las eventuales interrupciones en las llamadas entre los servicios. Además, implementa reintentos y tiempos de espera para ciclos de errores, garantizando que el sistema continúa operando incluso si uno o varios microservicios fallan, optimizando la disponibilidad y la experiencia del usuario.

Se recomienda monitorear y administrar el desempeño de los microservicios, incorporando herramientas como Grafana para visualizar las

métricas en tiempo real y Elasticsearch o Kibana para el análisis de registros, lo que permite detectar errores y así mejorar el rendimiento del sistema.

REFERENCIAS

Andrés, P., & Chiriboga, R. (2023). *Autenticación de microservicios basados en OAuth*. https://openaccess.uoc.edu/handle/10609/147247

Contreras, D. A. B. (2018). Arquitectura de Microservicios. *Tecnología Investigación y Academia*, *6*(1), 36–46. https://revistas.udistrital.edu.co/index.php/tia/article/view/9687

Dauzon. (2018). *Git: Controle la gestión de sus versiones (conceptos, utilización y casos ... - Samuel Dauzon - Google Libros*. https://books.google.es/books?hl=es&lr=lang_es&id=kAP-5cbnmPYC&oi=fnd&pg=PA13&dq=Conceptos+de+git&ots=2mKZdhhKbp&sig=9S6Vhu1Nl66hB7U1_8yYiAR-Oug#v=onepage&q&f=false

Genaro Moreno Beltrán. (2015). *Boletín Científico :: UAEH*. https://repository.uaeh.edu.mx/revistas/index.php/xikua/article/download/332/4434?inline=1

González. (2017). *UNIVERSIDAD POLITÉCNICA DE SINALOA PROGRAMA ACADÉMICO DE INGENIERÍA EN INFORMÁTICA*.

IBM. (2021). *Desarrollo de proyectos de Apache Maven - Documentación de IBM*. https://www.ibm.com/docs/es/radfws/9.6.1?topic=SSRTLW_9.6.1/com.ibm.aries.osgi.doc/topics/maven_intro.html

Lenarduzzi, V., & Lomio, F. (2020). Are SonarQube Rules Inducing Bugs? *SANER 2020 - Proceedings of the 2020 IEEE 27th International Conference on Software Analysis, Evolution, and Reengineering*, 501–511. https://doi.org/10.1109/SANER48275.2020.9054821

Monción. (2023). *UNIVERSIDAD POLITÉCNICA DE CARTAGENA*.

Ordóñez, H., & Ordóñez, C. (2021). Recuperación y clasificación de arquitecturas software en GitHub para reutilización, soportado por ontologías. *Revista Científica*, *41*(41), 242–251. https://doi.org/10.14483/23448350.17644

Óscar Gómez. (2012). *Bases de datos Versión 1.1 Óscar Gómez.*

Quero. (2010). Confiabilidad y coeficiente Alpha de Cronbach. *Telos*. https://www.academia.edu/6906740/Confiabilidad_y_coeficiente_Alpha_de_Cronbach

Ramírez. (2020). *Estudio del framework Spring, Spring Boot y microservicios*. https://ebuah.uah.es/dspace/handle/10017/45107

Rojas. (2019). *UNIVERSIDAD TÉCNICA DEL NORTE BIBLIOTECA UNIVERSITARIA*.

Socius Perú S.A.C. (2025). *SOCIUS - Arquitectura y Adopción Tecnológica*. Pagina Web. https://sociuscorp.com/

Vega. (2019). Validez y confiabilidad de la escala de sentido de coherencia en estudiantes de grado de enfermería de una universidad española. *Gaceta Sanitaria*, *33*(4), 310–316. https://doi.org/10.1016/J.GACETA.2018.02.009

Vinicio Estrada-Velasco, M. (2021). Revisión Sistemática de la Metodología Scrum para el Desarrollo de Software. *Dominio de Las Ciencias, ISSN-e 2477-8818, Vol. 7, Nº. Extra 4, 2021 (Ejemplar Dedicado a: AGOSTO ESPECIAL), Pág. 54*, *7*(4), 54. https://doi.org/10.23857/dc.v7i4.2429

ANEXOS

Anexo 1: Matriz de Operacionalización

Tabla 2

Evaluación de microservicios en transacciones bancaria

VARIABLE	DIMENSIONES	INDICADORES	N.°	DESCRIPCIÓN	ESCALA 1	ESCALA 2	ESCALA 3	ESCALA 4	ESCALA 5
Implementación de microservicios para la gestión de transacciones bancarias.	**Eficiencia**	Reducción de tiempos de respuesta	1	¿De qué manera los microservicios pueden reducir los tiempos de respuesta en las transacciones bancarias?	Muy ineficiente	Deficiente	Regular	Bueno	Excelente
			2	¿Qué factores del modelo bancario actual limitan la velocidad de respuesta y cómo la arquitectura de microservicios contribuye a superarlos?	Nada eficiente	Poco eficiente	Regular	Eficiente	Muy eficiente
	Seguridad y confiabilidad	Protección de datos en transacciones	3	¿Cómo garantizar la seguridad y confiabilidad de las transacciones a través de una arquitectura de microservicios?	Nada seguro	Poco seguro	Regular	Seguro	Muy seguro
			4	¿Qué protocolos y estándares de seguridad son indispensables para proteger microservicios expuestos en entornos bancarios y minimizar riesgos de accesos no autorizados?	Sin protocolos	Pocos protocolos	Regular	Seguridad adecuada	Seguridad robusta
	Integración	Conexión entre sistemas internos y externos	5	¿Cómo mejorar la integración entre sistemas internos y externos de los bancos utilizando microservicios?	Muy deficiente	Deficiente	Regular	Buena	Excelente
			6	¿Cómo se integran los microservicios bancarios con sistemas de terceros para ofrecer servicios financieros más amplios y mejorar la experiencia del usuario?	Sin integración	Baja integración	Regular	Buena integración	Integración óptima
	Metodologías y herramientas	Uso de tecnologías para desarrollo y despliegue	7	¿Qué herramientas y metodologías son más efectivas para el desarrollo y despliegue de microservicios en proyectos bancarios?	Nada efectivo	Poco efectivo	Regular	Efectivo	Muy efectivo
			8	¿De qué forma se aplican las prácticas de DevOps para automatizar pruebas, monitorear servicios y garantizar un despliegue continuo confiable de los microservicios bancarios?	Sin automatización	Automatización deficiente	Regular	Automatización eficiente	Automatización avanzada