



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ingeniería de Sistemas e Informática

Escuela Profesional de Ingeniería de Sistemas

**Implementación de una aplicación web para la
visualización del consumo de APIs públicas en una
entidad financiera**

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el Título Profesional de Ingeniero de Sistemas

AUTOR

Yohan Javier MAMANI CUSI

ASESOR

Robert Elías ESPINOZA DOMÍNGUEZ

Lima, Perú

2026



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Mamani, Y. (2026). *Implementación de una aplicación web para la visualización del consumo de APIs públicas en una entidad financiera*. [Trabajo de suficiencia profesional de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática, Escuela Profesional de Ingeniería de Sistemas]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Yohan Javier Mamani Cusi
Tipo de documento de identidad	DNI
Número de documento de identidad	75002839
URL de ORCID	https://orcid.org/0009-0000-3959-4481
Datos de asesor	
Nombres y apellidos	Robert Elías Espinoza Domínguez
Tipo de documento de identidad	DNI
Número de documento de identidad	08136325
URL de ORCID	https://orcid.org/0000-0002-3456-8302
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Jaime Rubén Pariona Quispe
Tipo de documento	DNI
Número de documento de identidad	06220658
Miembro del jurado 1	
Nombres y apellidos	Walter Pedro Contreras Flores
Tipo de documento	DNI
Número de documento de identidad	07743259
Datos de investigación	
Línea de investigación	No aplica

Grupo de investigación	No aplica
Agencia de financiamiento	Sin financiamiento
Ubicación geográfica de la investigación	País: Perú Departamento: Lima Provincia: Lima
	Distrito: Cercado de Lima Jr. Carlos Amezaga No. 375 Universidad Nacional Mayor de San Marcos Latitud: -12.0564232 Longitud: -77.0843327
Año o rango de años en que se realizó la investigación	2025
URL de disciplinas OCDE	2.02.04 -- Ingeniería de sistemas y comunicaciones https://purl.org/pe-repo/ocde/ford#2.02.04



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
Escuela Profesional de Ingeniería de Sistemas

Acta Virtual de Sustentación
del Trabajo de Suficiencia Profesional

Siendo las 20:00 horas del día 31 de marzo del año 2026, se reunieron virtualmente los docentes designados como Miembros de Jurado del Trabajo de Suficiencia Profesional, presidido por el Lic. Pariona Quispe Jaime Rubén (Presidente), Lic. Contreras Flores Walter Pedro (Miembro) y el Lic. Espinoza Domínguez Robert Elías (Miembro Asesor), usando la plataforma Meet (<http://meet.google.com/wcc-jfha-tms>), para la sustentación virtual del Trabajo de Suficiencia Profesional intitulado: **“IMPLEMENTACIÓN DE UNA APLICACIÓN WEB PARA LA VISUALIZACIÓN DEL CONSUMO DE APIS PÚBLICAS EN UNA ENTIDAD FINANCIERA”**, por el Bachiller Mamani Cusi Yohan Javier; para obtener el Título Profesional de Ingeniero de Sistemas.

Acto seguido de la exposición del Trabajo de Suficiencia Profesional, el Presidente invitó al Bachiller a dar las respuestas a las preguntas establecidas por los miembros del Jurado.

El Bachiller en el curso de sus intervenciones demostró pleno dominio del tema, al responder con acierto y fluidez a las observaciones y preguntas formuladas por los señores miembros del Jurado.

Finalmente habiéndose efectuado la calificación correspondiente por los miembros del Jurado, el Bachiller obtuvo la nota de **17 (DIECISIETE)**.

A continuación el Presidente de Jurados el Lic. Pariona Quispe Jaime Rubén, declara al Bachiller **Ingeniero de Sistemas**.

Siendo las 20:50 horas, se levantó la sesión.

Presidente

Lic. Pariona Quispe Jaime Rubén

Miembro

Lic. Contreras Flores Walter Pedro

Miembro Asesor

Lic. Espinoza Domínguez Robert Elías



UNIVERSIDAD NACIONAL MAYOR DE

SAN MARCOS

Universidad del Perú, Decana de América

CERTIFICADO DE SIMILITUD

Yo, Robert Elías Espinoza Dominguez en mi condición de asesor acreditado mediante Resolución Decanal N° 001268-2025-D-FISI/UNMSM y OFICIO N° 0018-PTTSP-FISI-2025 del Trabajo de Suficiencia Profesional, cuyo título es Implementación de una aplicación web para la visualización del consumo de APIs públicas en una entidad financiera, presentado por el bachiller Mamani Cusi Yohan Javier para optar el Título Profesional de Ingeniero de Sistemas, CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante la plataforma PIDS, el documento evaluado cuenta con el 12% de similitud, nivel PERMITIDO para continuar con los trámites correspondientes y para su **publicación en el repositorio institucional.**

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del **TITULO PROFESIONAL DE INGENIERO DE SISTEMAS.**

Lima, 4 de junio de 2026

Firma del Asesor:

Nombres y Apellidos del asesor: Robert Elías Espinoza Dominguez

DNI: 08136325

FICHA CATALOGRÁFICA

Implementación de una aplicación web para la visualización del consumo de APIs públicas en una entidad financiera

AUTOR: MAMANI CUSI, YOHAN JAVIER

ASESOR: ESPINOZA DOMÍNGUEZ, ROBERT ELÍAS

LIMA – PERÚ, 2026

TÍTULO PROFESIONAL: Título Profesional de Ingeniero de Sistemas

ÁREA / PROGRAMA / LÍNEA DE INVESTIGACIÓN:

Ingenierías / Tecnología de Información y Comunicación / Ingeniería de Sistemas

PREGRADO: Universidad Nacional Mayor de San Marcos - Facultad de Ingeniería de Sistemas e Informática - Escuela Profesional de Ingeniería de Sistemas

Formato 21 x 29.7 cm

Páginas: x, 78

DEDICATORIA

Dedico este trabajo a mi familia que me apoyó en todo momento para cumplir mis metas y objetivos. Gracias por todo el apoyo y consejos para seguir mejorando como persona y profesional.

AGRADECIMIENTOS

Agradezco a mi asesor Robert Elías Espinoza Domínguez, gracias a su guía logramos completar el desarrollo del presente trabajo. Gracias por su compromiso, consejos y paciencia en este proceso.

Agradezco también a la comisión de titulación por darnos la oportunidad de obtener el título profesional y continuar con nuestras metas profesionales.

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS
Implementación de una aplicación web para la visualización del consumo de APIs
públicas en una entidad financiera

Autor: Mamani Cusi, Yohan Javier

Asesor: Espinoza Domínguez, Robert Elías

Título: Trabajo de Suficiencia Profesional para optar por el Título Profesional de Ingeniero de Sistemas

Fecha: Enero del 2026

RESUMEN

En el presente trabajo describe la implementación de una aplicación web para la visualización del consumo de Interfaces de Programación de Aplicaciones (APIs) públicas en una entidad financiera bajo el enfoque de Open Banking. El objetivo fue cubrir la necesidad de los responsables de cada API en visualizar el consumo y costos de estas, que se encuentran en el API Management de Open Banking. De esta información, cada responsable de las APIs podrá tomar acción según las métricas y costos que se muestren en la aplicación web.

La implementación de esta aplicación integró distintos componentes Azure y se utilizó el marco de trabajo SCRUM, mejorando la organización y la calidad del proyecto. La implementación del proyecto fue bajo los lineamientos del enfoque Open Banking realizada en Reino Unido.

El resultado de esta implementación mejoró la visibilidad y control del consumo de cada API. Permitió a los responsables de las APIs reducir los tiempos de análisis, facilitó la toma de decisiones y optimizar sus recursos y costos.

Palabras Claves: Aplicación web, Azure, Open Banking.

MAJOR NATIONAL UNIVERSITY OF SAN MARCOS
FACULTY OF SYSTEMS AND INFORMATIC ENGINEERING
PROFESSIONAL SCHOOL OF SYSTEMS ENGINEERING

**Implementation of a web application for viewing public API consumption in a
financial institution**

Author: Mamani Cusi, Yohan Javier

Advisor: Espinoza Domínguez, Robert Elías

Title: Work of Professional Sufficiency to obtain the Professional Title of Systems Engineer.

Date: January 2026

ABSTRACT

This work describes the implementation of a web application to visualize the consumption of public Application Programming Interfaces (APIs) in a financial institution under the Open Banking approach. The objective was to meet the need for the responsables of each API to visualize their consumption and costs in Open Banking API Management. Based on this information, each responsible can take action according to the statistics and costs displayed in the web application.

The implementation of this application integrated different Azure components and used the SCRUM framework, improving the organization and quality of the project. The project was implemented under the guidelines of the Open Banking approach carried out in the United Kingdom.

The result of this implementation improved the visibility and control of each API's consumption. It allowed responsables to reduce analysis times, facilitated decision-making, and optimized their resources and costs.

Keywords: Web application, Azure, Open Banking.

TABLA DE CONTENIDO

DEDICATORIA	iii
AGRADECIMIENTOS	iv
ÍNDICE DE TABLAS	ix
ÍNDICE DE FIGURAS.....	x
INTRODUCCIÓN	1
CAPÍTULO I - TRAYECTORIA PROFESIONAL	3
CAPÍTULO II - CONTEXTO EN EL QUE SE DESARROLLA LA EXPERIENCIA..	8
2.1. EMPRESA - ACTIVIDAD QUE REALIZA	8
2.2. VISIÓN	8
2.3. MISIÓN.....	8
2.4. ORGANIZACIÓN DE LA EMPRESA	8
2.5. ÁREA, CARGO Y FUNCIONES DESEMPEÑADAS	9
2.6. EXPERIENCIA PROFESIONAL REALIZADA EN LA ORGANIZACIÓN	9
CAPÍTULO III - ACTIVIDADES DESARROLLADAS	11
3.1. SITUACIÓN PROBLEMÁTICA	11
3.1.1. DEFINICIÓN DEL PROBLEMA	12
3.1.2. PROBLEMAS ESPECÍFICOS.....	12
3.2. SOLUCIÓN.....	12
3.2.1. OBJETIVOS.....	12
3.2.2. ALCANCE	13
3.2.3. ETAPAS Y METODOLOGÍA.....	14
3.2.4. FUNDAMENTOS UTILIZADOS	15
3.2.5. IMPLEMENTACIÓN DE LAS ÁREAS, PROCESOS, SISTEMAS Y BUENAS PRÁCTICAS	26
3.3. EVALUACIÓN.....	64
3.3.1. EVALUACIÓN ECONÓMICA / EVALUACIÓN COSTO-BENEFICIO..	64

3.3.2. INTERPRETACIÓN DEL VAR Y DEL TIR	65
3.3.3. BENEFICIOS DEL PROYECTO	66
CAPÍTULO IV - REFLEXIÓN CRÍTICA DE LA EXPERIENCIA.....	67
CAPÍTULO V - CONCLUSIONES Y RECOMENDACIONES	68
5.1. CONCLUSIONES.....	68
5.2. RECOMENDACIONES	68
5.3. FUENTES DE INFORMACIÓN.....	69
5.4. GLOSARIO.....	71
ANEXOS	77
ANEXO 1: Script de creación de tablas de base de datos	77

ÍNDICE DE TABLAS

Tabla 1 <i>Experiencia laboral</i>	3
Tabla 2 <i>Educación Académica</i>	6
Tabla 3 <i>Cursos realizados</i>	6
Tabla 4 <i>Certificaciones</i>	7
Tabla 5 <i>Idiomas estudiados por el autor</i>	7
Tabla 6 <i>Etapas del desarrollo del proyecto</i>	14
Tabla 7 <i>Principales proveedores de servicios cloud del 2025</i>	18
Tabla 8 <i>Ceremonias Scrum</i>	24
Tabla 9 <i>Roles involucrados en el proyecto</i>	26
Tabla 10 <i>Requerimientos iniciales del sistema.</i>	28
Tabla 11 <i>Ejemplo del resultado de la consulta de métricas</i>	32
Tabla 12 <i>Costos de infraestructura</i>	33
Tabla 13 <i>Detalle de los casos de uso</i>	38
Tabla 14 <i>Especificación de caso de uso: CU01 - Consultar métricas del dashboard</i> ..	40
Tabla 15 <i>Especificación de caso de uso: CU02 - Gestionar permisos por API</i>	41
Tabla 16 <i>Especificación de caso de uso: CU03 - Consultar inventario de APIs</i>	43
Tabla 17 <i>Especificación de caso de uso: CU04 - Gestionar costos de infraestructura</i>	44
Tabla 18 <i>Especificación de caso de uso: CU05 - Consultar costos mensuales por API</i>	45
Tabla 19 <i>Tipos de issues</i>	45
Tabla 20 <i>Product Backlog con los issues iniciales</i>	46
Tabla 21 <i>Estados de Issues</i>	48
Tabla 22 <i>Tecnologías utilizadas</i>	50
Tabla 23 <i>Roles del sistema</i>	50
Tabla 24 <i>Actividades realizadas en los sprints</i>	58
Tabla 25 <i>Costos del proyecto</i>	64
Tabla 26 <i>Cálculos de VAN</i>	66

ÍNDICE DE FIGURAS

Figura 1 <i>Organigrama de la empresa financiera</i>	9
Figura 2 <i>Sistema Open Banking</i>	16
Figura 3 <i>Patrón de diseño Observer</i>	19
Figura 4 <i>Flujo de autenticación SSO</i>	23
Figura 5 <i>Estructura de Scrum Framework</i>	25
Figura 6 <i>Consulta para calcular las métricas</i>	29
Figura 7 <i>Modelo de negocio</i>	36
Figura 8 <i>Arquitectura del sistema</i>	37
Figura 9 <i>Diagrama de casos de uso</i>	39
Figura 10 <i>Diagrama de caso de uso: CU01 - Consultar métricas del dashboard</i>	40
Figura 11 <i>Diagrama de caso de uso: CU02 - Gestionar permisos por API</i>	41
Figura 12 <i>Diagrama de caso de uso: CU03 - Consultar inventario de APIs</i>	42
Figura 13 <i>Diagrama de caso de uso: CU04 - Gestionar costos de infraestructura</i>	43
Figura 14 <i>Diagrama de caso de uso: CU05 - Consultar costos mensuales por API</i>	44
Figura 15 <i>Herramienta Board de Jira</i>	48
Figura 16 <i>Reuniones de Daily Scrum</i>	49
Figura 17 <i>Diagrama de base de datos</i>	52
Figura 18 <i>Ejecución de una consulta en KQL</i>	53
Figura 19 <i>Proceso de entrega continua</i>	54
Figura 20 <i>Archivo de configuración del pipeline</i>	55
Figura 21 <i>Plataforma de Dynatrace</i>	56
Figura 22 <i>Plataforma de Kibana</i>	57
Figura 23 <i>Prototipo del módulo Dashboard</i>	60
Figura 24 <i>Prototipo del módulo Inventarios de Apis</i>	61
Figura 25 <i>Prototipo del Mantenimiento de costos de infraestructura</i>	62
Figura 26 <i>Prototipo del Listado de costos por API</i>	63

INTRODUCCIÓN

En el sector financiero las empresas buscan sobresalir mediante innovación y tecnología. Anteriormente el sector financiero, sobre todo en el sector bancario, era muy cuidadoso con la información que maneja debido a que es información sensible. Sin embargo, para generar un valor agregado hacia al usuario y revolucionar el sistema financiero nace el modelo Open Banking.

Este modelo se basa en el principio fundamental de que el cliente bancario es dueño de su propia información personal, por lo cual debería ser libre de compartir su información siempre y cuando el mismo cliente lo autorice. De igual forma si el cliente decide revocar el permiso, se dejaría de compartir la información. Open Banking revolucionó todo el sector financiero promoviendo la innovación tecnológica y una transformación cultural. Desde ese enfoque las empresas bancarias buscan integrarse y asociarse con otras empresas para brindar un valor agregado y mejores productos a los clientes.

En la empresa donde se desarrolló el presente proyecto, tomó como base el sistema financiero del Reino Unido que adaptó el modelo de Open Banking permitiendo compartir la información de los clientes bancarios con otras instituciones a través de APIs que funcionan como puentes para la comunicación entre sistemas.

Nuestro proyecto Open Banking expone las APIs a través de un API Management en Azure Cloud. Si el cliente acepta el consentimiento de compartir su información, empresas terceras pueden comunicarse con estas APIs y obtener la información del cliente. Cada API expone los endpoints que define dentro de su archivo OpenAPI, luego redirige la petición hacia el core bancario. Las métricas de estas solicitudes que pasan por el API Management se obtienen desde Logs Analytics. Esta información es útil para los responsables de cada API, quienes son los responsables de su gestión, ya que facilita la toma de decisiones, así como el monitoreo y control de sus APIs, finalmente es importante para el cálculo del costos y presupuesto anual que cada responsable de API maneja dentro del banco.

A continuación, se presenta la estructura del presente trabajo de suficiencia profesional:

- En el capítulo I, se describe la trayectoria profesional del autor de este trabajo de suficiencia profesional. Donde se evidencia la experiencia que se ganó como profesional a lo largo del tiempo.
- En el capítulo II, se describe el contexto de la empresa donde se realizó el presente proyecto y funciones realizadas en el cargo.
- En el capítulo III, se describe el desarrollo de la aplicación web y la arquitectura de integración con otros componentes cloud realizada.
- En el capítulo IV, se describe la reflexión crítica de la experiencia del autor.
- En el capítulo V, se describen las conclusiones y recomendaciones del presente trabajo de suficiencia profesional.

CAPÍTULO I - TRAYECTORIA PROFESIONAL

El autor trabaja como desarrollador de software, con más de 6 años de experiencia en la creación y análisis de aplicaciones. Se ha especializado como desarrollador de backend con Java, pero también ha trabajado en frontend y en aplicaciones móviles.

Actualmente desarrolla sus labores en Tata Consultancy Services en un equipo ágil para el desarrollo de aplicaciones en una empresa financiera del Perú.

Anteriormente el autor trabajó para empresas de diferentes tamaños e industrias, como son empresas del sector documental, sector de gestión fiscal y principalmente en el sector bancario. Los detalles se describen en la Tabla 1.

Tabla 1
Experiencia laboral

Empresa/Periodo	Descripción de actividades realizadas
Tata Consultancy Services (Septiembre 2025 - Actualidad)	Cargo: Developer Backend Actividades: – Desarrollar un servicio que funciona como Producer y trabaja con tópicos de Kafka. – Mapear campos debido a la integración con otros servicios bancarios. – Realizar seguimiento del proceso de entrega continua. – Actualizar documentación del sistema.
Baufest (Junio 2024 - Septiembre 2025)	Cargo: Developer Backend Actividades: – Desarrollo de una aplicación web para crear un dashboard de un sistema Open Banking. – Realizar la integración con servicios cloud. – Apoyo con la renovación de certificados y secretos de componentes Azure. – Realizar documentación técnica del proyecto. – Soporte en el despliegue continuo.

Logros:

- Cumplir eficientemente con los estándares de desarrollo de la institución financiera.

Zoluxiones**(Enero 2022 – Junio 2024)****Cargo:** Ingeniero de Software Senior**Actividades:**

- Análisis de los requerimientos de usuario.
- Participar en las reuniones con el usuario final.
- Desarrollo de nuevos módulos para un sistema de gestión de clientes para una entidad financiera.
- Dar soporte en los pases de producción.
- Analizar e investigar los incidentes o errores que se presenten en el ambiente de producción.
- Revisar los cambios realizados por los miembros del equipo para que se cumpla con los objetivos de Code Review.
- Actualizar documentación del sistema.

Logros:

- Ser líder técnico del equipo de soporte a producción.

TCI**(Julio 2021 – Enero 2022)****Cargo:** Analista programador Java**Actividades:**

- Diseñar la arquitectura de un sistema para la gestión de importaciones y exportaciones.
- Implementar la arquitectura basada en microservicios y colas.
- Implementar el módulo de seguridad del sistema.
- Proponer ideas para mejorar el desarrollo del sistema.
- Apoyar con el desarrollo de módulos en el desarrollo frontend.
- Implementar el microservicio de envío de correos.

Logros:

- Implementación de colas basada en mensajería con RabbitMQ.

EXACT**Cargo:** Analista programador TI**(Noviembre 2020 – Julio 2021)****Actividades:**

- Análisis de requerimientos de usuario.
- Proponer ideas para optimizar el desarrollo del sistema.
- Desarrollar de servicios backend, frontend y aplicativos móviles.
- Implementar un ecosistema de Spring Cloud.
- Documentar manuales de usuario de los sistemas.
- Investigar y proponer tecnologías para optimizar el desarrollo de sistemas.

Logros:

- Implementar un ecosistema de Spring Cloud.
- Implementar un sistema móvil con un módulo de localización.
- Implementar test unitarios para mejorar la calidad de código.

EXACT**Cargo:** Practicante TI**(Febrero 2019 – Noviembre 2020)****Actividades:**

- Desarrollar pequeños módulos de servicios backend y frontend.
- Implementar un ecosistema de Spring Cloud.
- Documentar manuales de usuario de los sistemas.
- Participar en las ceremonias Scrum.

Logros:

- Lograr el ascenso para pasar de practicante a analista.

Nota. Fuente: Elaboración propia

La educación académica del autor se detalla en la Tabla 2.

Tabla 2
Educación Académica

Período	Educación Académica
Marzo 2015 – Agosto 2020	Grado Académico de Bachiller en Ingeniería de Sistemas Escuela Académico Profesional de Ingeniería de Sistemas Facultad de Ingeniería de Sistemas e Informática Universidad Nacional Mayor de San Marcos

Nota. Fuente: Elaboración propia

Los cursos realizados por el autor se muestran en la Tabla 3.

Tabla 3
Cursos realizados

Fechas	Cursos
Julio 2025	Curso de Microsoft Azure Fundamentals AZ-900 Udemy
Agosto 2023	React: De cero a experto Udemy
Febrero 2021	Desarrollo Móvil: Flutter - De Cero a experto Udemy
Enero 2020	Desarrollo frontend con Angular Udemy
Julio 2019	Microservicios con Spring Cloud Udemy

Nota. Fuente: Elaboración propia

Las certificaciones logradas por el autor se muestran en la Tabla 4:

Tabla 4
Certificaciones

Fechas	Certificación/Institución
Julio 2025	Microsoft Azure fundamentals Microsoft Azure
Junio 2025	Scrum fundamentals certified Scrum Study

Nota. Fuente: Elaboración propia

Los estudios de idiomas realizados por el autor se listan en la Tabla 5:

Tabla 5
Idiomas estudiados por el autor

Periodo	Idioma	Nivel	Institución
Noviembre 2021 - Septiembre 2022	Inglés	Avanzado	Británico
Abril 2021 - Septiembre 2021	Inglés	Intermedio	Británico
Enero 2020 - Marzo 2021	Inglés	Básico	Británico

Nota. Fuente: Elaboración propia

CAPÍTULO II - CONTEXTO EN EL QUE SE DESARROLLA LA EXPERIENCIA

2.1. EMPRESA - ACTIVIDAD QUE REALIZA

La empresa del presente proyecto pertenece al sector financiero en Perú por más de 120 años, principalmente en el sector de banca y seguros. Actualmente es una de las empresas con mayor relevancia en el mercado y cuenta con más de 10 millones de clientes en el 2024.

La empresa tiene más de 400 agencias en todo el Perú y 5000 agentes para aumentar la cobertura de sus operaciones. Adicionalmente cuenta con más de 15 mil colaboradores, siendo una de las empresas más grande a nivel nacional dentro del sector financiero.

La empresa invierte en tecnología para mejorar los productos y la experiencia que brinda a los clientes. Tiene alianzas estratégicas con empresas como Microsoft para trabajar con recursos cloud. La empresa invierte también en tecnologías modernas como IA para asistentes virtuales y buscadores inteligentes.

2.2. VISIÓN

La empresa tiene como visión ser el banco líder en el sector, brindando productos y servicios mediante un sistema financiero innovador.

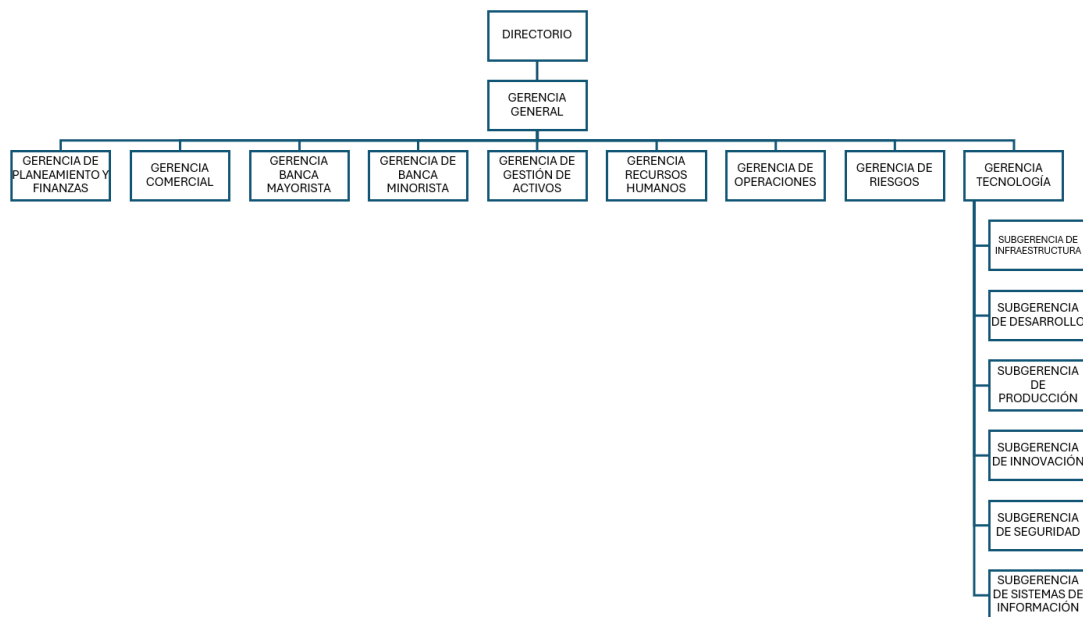
2.3. MISIÓN

La empresa tiene como objetivo brindar productos y servicios financieros para asistir a sus clientes en la mejora de su calidad de vida y situación financiera.

2.4. ORGANIZACIÓN DE LA EMPRESA

La organización de la empresa donde se realizó el presente trabajo se representa en la Figura 1.

Figura 1
Organigrama de la empresa financiera



Nota. Fuente: Elaboración propia

2.5. ÁREA, CARGO Y FUNCIONES DESEMPEÑADAS

El proyecto actual se llevó a cabo en el área de arquitectura de Open Banking, que pertenece a la Subgerencia de Innovación. En este proyecto, trabajó como desarrollador backend senior y su tarea principal era crear una aplicación web que conecta con el Log Analytics para mostrar la información de las APIs de Open Banking. Adicionalmente se cumplieron las funciones de soporte en la entrega continua a producción, análisis de requerimientos de usuario, renovación de certificados de servicios cloud y soporte a producción.

2.6. EXPERIENCIA PROFESIONAL REALIZADA EN LA ORGANIZACIÓN

Durante el tiempo que se trabajó en el proyecto, se completaron metas exitosamente permitiendo completar y desplegar la solución en producción. Adicionalmente el trabajo fue exigente debido a los fuertes lineamientos, estándares y normas que maneja la institución financiera.

El cargo que se desempeñó en la organización fue de nivel senior durante un año y se cumplió las siguientes funciones:

- Desarrollar los módulos backend cumpliendo con los estándares de calidad y buenas prácticas.
- Apoyo en el desarrollo de módulos frontend.
- Soporte en el despliegue continuo de los pases a producción.
- Análisis de los requerimientos de usuario.
- Diseñar la arquitectura del sistema.
- Tomar decisiones técnicas.
- Cumplir los lineamientos de la empresa.
- Realizar la documentación técnica del sistema.
- Revisión de código de otros miembros del equipo.
- Cumplir el rol de backup del líder técnico.

CAPÍTULO III - ACTIVIDADES DESARROLLADAS

3.1. SITUACIÓN PROBLEMÁTICA

El sector bancario es un mercado muy competitivo donde se busca sobresalir mediante innovación para dar mejores productos y servicios a los clientes. Este sector paso de ser una banca cerrada, donde se controlaba toda la información del cliente y restringía acceso a cualquier empresa tercera, a pasar a un sistema abierto mediante el sistema Open Banking promoviendo compartir la información del cliente hacia terceros autorizados, debido a que solo pueden acceder a la información solo si el cliente brinda su consentimiento.

Open Banking es un sistema que expone APIs de la entidad financiera mediante un API Managment para que empresas terceras puedan obtener o enviar información de forma segura del cliente bancario. En este contexto, las APIs públicas se refieren a las APIs expuestas por la entidad financiera para ser consumidas por terceros autorizados. Cada API de la entidad financiera tiene un responsable de velar por el correcto funcionamiento de esta, por lo cual debe saber las métricas que tiene su API como son la cantidad de peticiones, el tiempo de respuesta y la cantidad de peticiones exitosas y fallidas. Sin embargo, antes del presente trabajo, el responsable de la API no podía visualizar la información a menos que solicite acceso al API Managment de Open Banking y le asignen un grupo de red específico, pero este proceso ya no es permitido por el banco por temas de seguridad.

Adicionalmente tener un API desplegado en el API Managment tiene un costo de infraestructura, así como costos asociados a la cantidad de peticiones realizadas a la API. Toda esta información es necesaria para que el responsable de la API pueda estimar un presupuesto anual y realizar el cobro a los consumidores que realizan peticiones hacia su API. No se contaba con reportes de esta información y el cálculo del costo total de las APIs públicas expuestas por la entidad financiera era realizado de forma manual por el Cloud Engineer y el Product Owner. Al ser un flujo Manual, este era propenso a errores y dependía de intervención humana que representaba un costo operativo adicional.

3.1.1. DEFINICIÓN DEL PROBLEMA

Carencia de mecanismos que permitan visualizar el consumo de las APIs públicas de la entidad financiera.

3.1.2. PROBLEMAS ESPECÍFICOS

- Ausencia de métricas para el análisis del consumo de las APIs públicas de la entidad financiera.
- Carencia de una base de datos que permita almacenar los costos de infraestructura y el consumo de las APIs públicas de la entidad financiera.
- Carencia de una arquitectura que integre la aplicación web con servicios cloud.
- Existencia de procesos manuales para el cálculo de los costos de las APIs públicas que generan riesgos de errores humanos.
- Falta de reportes para optimizar la toma de decisiones de los responsables de las APIs públicas de la entidad financiera.

3.2. SOLUCIÓN

Se planteó la implementación de una aplicación web para la visualización del consumo de APIs públicas de la entidad financiera con la finalidad de optimizar el análisis de las métricas y mejorar la gestión de costos por parte de los responsables de las APIs.

3.2.1. OBJETIVOS

Objetivo general

- Implementar una aplicación web para la visualización del consumo de APIs públicas que permita optimizar el tiempo de análisis de las métricas y mejorar la gestión de costos en la entidad financiera.

Objetivos específicos

- Definir métricas para el análisis el consumo de las APIs públicas de la entidad financiera.
- Diseñar e implementar la base de datos que permita almacenar los costos de infraestructura y el consumo de las APIs públicas de la entidad financiera.

- Diseñar e implementar una arquitectura que integre la aplicación web con servicios cloud.
- Automatizar los cálculos de los costos de las APIs públicas de la entidad financiera.
- Generar reportes para optimizar la toma de decisiones de los responsables de las APIs públicas de la entidad financiera.

3.2.2. ALCANCE

Alcance Funcional

La solución del presente trabajo va a considerar las siguientes funcionalidades:

- Listar métricas de cada API desde el Log Analytics.
- Listar la información detallada de las APIs.
- Mantenimiento mensual de los costos de infraestructura.
- Listado de los costos mensuales por API.
- Exportar a Excel los costos mensuales por API.

Alcance Técnico

La solución del presente trabajo utilizamos las siguientes tecnologías:

- Spring Webflux
- Java 17
- Azure SQL Database
- API Management
- Log Analytics
- Key Vault
- Web App
- Angular
- Microsoft Entra ID
- Jenkins
- Dynatrace
- Kibana

Alcance Organizacional

La solución del presente trabajo cubre el proceso de análisis de las métricas de cada API que está desplegada en el API Management de Open Banking y la gestión de costos por los responsables de cada API de la entidad financiera.

3.2.3. ETAPAS Y METODOLOGÍA

Dentro del presente se utilizó la metodología ágil Scrum para el desarrollo de la aplicación web. Se dividió en el desarrollo en 12 sprint de dos semanas cada una. Dentro de esas semanas se destinaba dos días para realizar las siguientes ceremonias de Scrum:

- Sprint Planning
- Daily Scrum
- Sprint Review
- Sprint Retrospective

La implementación de la aplicación web comprende las siguientes etapas de la tabla 6.

Tabla 6

Etapas del desarrollo del proyecto

Etapas	Actividades
Análisis	– Recabar información de los requerimientos mediante reuniones con el usuario. Se levantó toda la información y los diferentes casos involucrados dentro del proceso.
Diseño	– Se construye el Product Backlog con los requerimientos priorizados por el Product Owner. – Se elaboró el diseño de la solución para posteriormente presentarlo al usuario y también al equipo de arquitectura para su aprobación. – En esta etapa se diseñó el diagrama de arquitectura. – Se realizó la planificación de los módulos a desarrollar.

	– Se diseñan los diagramas de base de datos y se definen las tecnologías a usar. – Realizar la integración y configuración con los recursos cloud.
Desarrollo del sistema	– Implementar los módulos de desarrollo según la planificación realizada. – Implementar nuevos requerimientos. – Realizar pruebas en los diferentes ambientes: Desarrollo, Certificación y producción.
Soporte a producción	– Realizar las pruebas en el ambiente de producción. – Dar soporte a producción. – Realizar la documentación final

Nota. Fuente: Elaboración propia

3.2.4. FUNDAMENTOS UTILIZADOS

Fundamentos Teóricos

Open Banking

El sistema Open Banking permite la transferencia de datos financieros de los clientes bancarios con fintechs y otras empresas terceras. Open Banking describe un modelo general en el que los intermediarios financieros comparten voluntariamente o por una regulación establecida, el acceso a los datos de sus clientes con otras empresas bancarias o financieras (Babina, 2025).

En agosto de 2016, la Autoridad de Competencia y Mercados (CMA) publicó una regulación que exigía a los bancos más grandes e importantes del Reino Unido que permitieran a los startups con licencia el acceso a los datos financieros de los clientes bancarios. Esta iniciativa tenía como objetivo mejorar la competencia, productividad y la innovación en el sector de los servicios financieros, permitiendo a los clientes bancarios compartir sus datos financieros con empresas terceras de forma segura. Este modelo se implementó de manera oficial en el Reino Unido en enero de 2018 (Akyildirim, 2025).

El sistema Open Banking es importante para las empresas que quieren ser competitivas, innovadoras y dar mejores productos y servicios a sus clientes. Si una

empresa tercera tiene el consentimiento del cliente del cual quiere obtener sus datos, podrá realizar una petición a las APIs expuestas del sistema bancario. Permitiendo la interoperabilidad en un sistema de transferencia seguro.

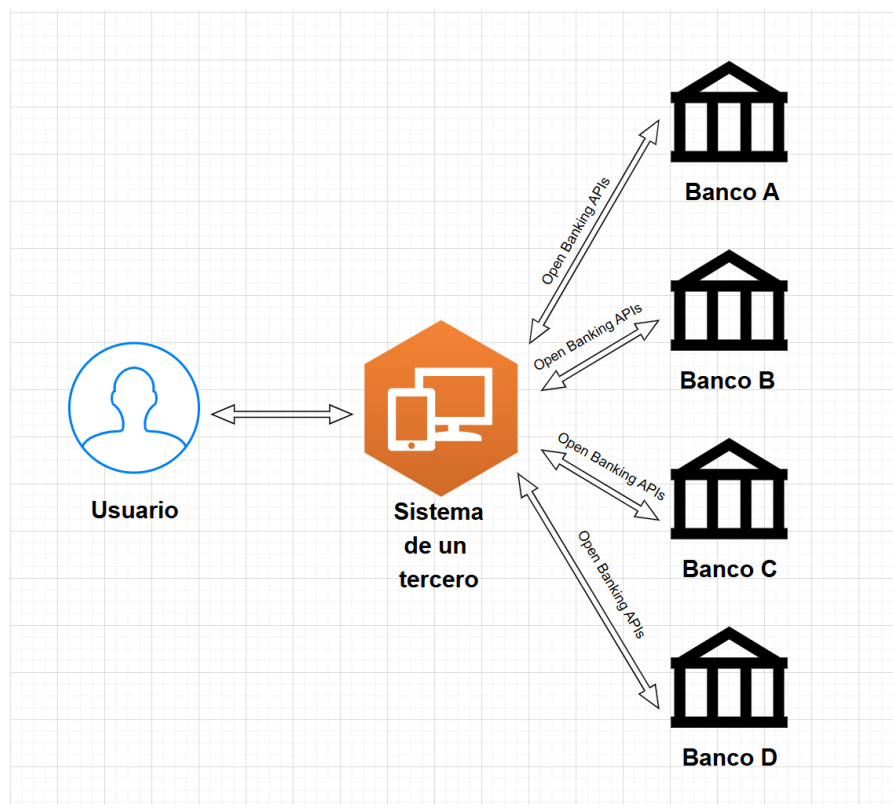
Ejemplos de servicios con Open Banking:

- Permitir conectar con bancos para iniciar pagos
- Validar cuentas bancarias
- Transferencias
- Préstamos
- Herramientas de finanzas personales
- Seguros, inversiones y pensiones
- Pagos e-commerce

El sistema Open Banking se representa gráficamente en la figura 2.

Figura 2

Sistema Open Banking



Nota. Fuente: Elaboración propia

API

Las APIs son herramientas que permite intercambiar información entre dos o más sistemas informáticos a través de una red. Las API en un sistema Open Banking son expuestas por el proveedor de datos por medio de endpoints que permiten a los consumidores interactuar con la base de datos del proveedor y prestar servicios financieros en nombre de los clientes (Babina, 2025).

Las APIs permiten agilizar el proceso de integración de aplicaciones, crear nuevos productos y permitir que los datos y servicios se puedan reutilizar y acceder globalmente. Con el creciente uso y dependencia de las API en sistemas informáticos, las organizaciones deben administrarlas como recursos importantes de primera categoría a lo largo de su ciclo de vida (Microsoft Azure, 2025).

API son las siglas de Interfaz de Programación que permite la interacción entre dos componentes mediante peticiones y respuestas. La API define los recursos, endpoints, datos aceptados, respuestas y mecanismos de seguridad.

Cloud Computing

La computación en la nube es el uso de servicios informáticos, como servidores, almacenamiento, bases de datos, redes, software, análisis e inteligencia. La organizaciones pueden hacer uso de estos servicios por medio de internet, lo cual brinda una innovación más rápida y recursos flexibles. En este tipo de tecnología solo se paga por los servicios en la nube que usa, lo que permite reducir los costos operativos y de infraestructura, permitiendo escalar a medida que cambian las necesidades de las organizaciones (Microsoft Azure, s.f.).

Cloud computing es importante para las organizaciones debido que permite la reducción de costos, porque ya no se preocupa por el hardware o infraestructura en un ambiente físico, sino se paga bajo el enfoque de pago por uso (pay-as-you-go). Otro aspecto importante es la seguridad porque ofrecen estrictos estándares de seguridad, backup y recuperación antes de desastres.

Existen muchos proveedores de servicios cloud, los principales proveedores del 2025 se muestran en la tabla 7.

Tabla 7

Principales proveedores de servicios cloud del 2025

Características	AWS	Azure	GCP
Cuota de mercado (primer trimestre de 2025)	29%	22%	12%
Regiones globales	36	60	42
Zonas de disponibilidad	114	126	127
Services Offered	200+	200+	100+

Nota. Fuente: Slingerland (2025)

Fintech

Las empresas de tecnología financiera (Fintech) han dado un giro importante en el sector financiero, revolucionando el sistema bancario y apoyando la expansión de nuevos modelos de negocio, procesos y productos financieros. Las empresas Fintech trabajan en un contexto de constante evolución, determinado por avances tecnológicos, complejidad regulatoria y alta competitividad (Ferilli, 2024).

Fintech es la unión de dos conceptos entre finanzas y tecnología. Las empresas buscan innovar o transformar el proceso financiero donde los consumidores y empresas interactúan.

Fundamentos Técnicos

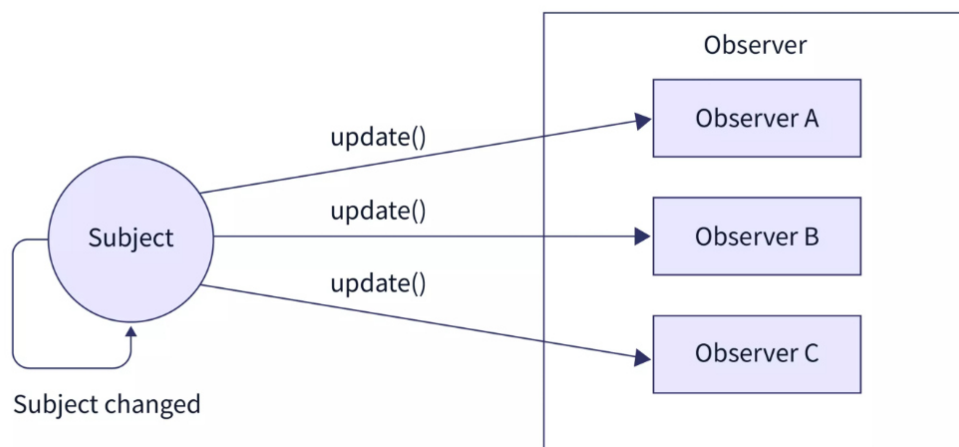
Programación reactiva

La Programación Reactiva es un enfoque basado en eventos que permite el procesamiento asíncrono y no bloqueante de datos. Este enfoque modela eventos como flujos de datos observables donde los consumidores podrán observar y reaccionar sobre ellos. La programación reactiva está basada en una arquitectura de eventos utilizando el patrón Observer, donde los componentes se comunican a través de flujos de mensajes asincrónicos, lo que impulsa reducir el acoplamiento y permite escalar independientemente los componentes del sistema. Este enfoque trabaja bien con las arquitecturas de microservicios, lo que facilita el desarrollo de aplicaciones altamente escalables y mejorar la tolerancia a fallos (Malhotra, 2024).

La programación es muy importante porque permite muchos procesos de manera simultánea sin bloquear los hilos de ejecución. Por lo cual al no ser bloqueante aprovecha mejor los recursos. Este paradigma de programación se base en el patrón de diseño Observer que consiste que un objeto observado notifica mediante eventos a los observadores por un cambio de estado. En la figura 3 encontraremos el diseño del patrón Observer.

Figura 3

Patrón de diseño Observer



Nota. Fuente: Somani (2022)

Contrato OpenAPI

Es un documento estándar que ayuda a describir las funcionalidades que tiene una API REST sin importar el lenguaje de programación utilizado en la API. Permite la documentación de las funcionalidades, endpoints, parámetros y modelos de datos. Es muy importante en la interacción de APIs debido a que facilita el entendimiento sin la necesidad de acceder al código fuente. Adicionalmente es útil cuando se tenga diseñado primero el documento, se puede utilizar para generar el código base de un proyecto, esto se denomina Contract-First. Sin embargo, el documento también se puede generar a partir del código del API, esto se denomina Code-First.

Entrega continua (CD)

La entrega continua es un proceso automatizado para lanzar nuevas versiones de un sistema. Cuando un cambio es aprobado se lanza automáticamente un flujo que compila, prueba y prepara la actualización. Sin embargo, cuando se debe tomar la decisión de desplegar cada cambio en un entorno de producción se necesita intervención humana porque esta decisión la toma el desarrollador (Amazon Web Services, s.f.).

La entrega continua es una práctica de desarrollo de software donde los desarrolladores crean o suben nuevas versiones de software para desplegarlo a producción en cualquier momento. Para lograr este modelo de entrega constante, implica crear ambientes pre productivos para realizar pruebas previas al despliegue en producción. Los nuevos cambios realizadas en una solución de entrega continua se implementan automáticamente en un entorno de pruebas automáticas, que garantizan la calidad en la validación de código y comprueban cualquier número de errores e incoherencias. Una vez que el código pasa todas las pruebas de cada ambiente, la entrega continua requiere intervención humana para autorizar el despliegue de los cambios en el ambiente de producción (Susnjara, 2024).

Contract-First

La metodología Contract-First se basa en los contratos de cada API donde primero creamos un documento de diseño (o definición de contrato) que describe formalmente los endpoints de las APIs que vamos a implementar, antes de escribir código de programación o cualquier prototipo real. Esta metodología es útil para detallar claramente la estructura de la futura API antes de implementarla. Dada una descripción formal de la API mediante un archivo yaml, los consumidores de nuestra API tendrán el contexto y podrán empezar a crear APIs para probar con servidores simulados generados, si así lo desean, no necesitan que la API esté desplegada para que puedan comenzar con su desarrollo. Por lo tanto, el desarrollo basado en contratos incrementa la independencia de los equipos involucrados dentro de un proyecto. En la fase de diseño, la definición de la interfaz puede servir como artefacto de diseño compartido entre equipos involucrados, de modo que se convierte en un contrato mutuamente acordado entre los implementadores y los usuarios de los endpoints (Eros, 2021).

Esta metodología fue muy importante debido a que nos permitió primero definir los distintos recursos que va a necesitar nuestro servicio. Luego cuando comenzamos con el desarrollo, utilizamos el contrato creado para generar la base de nuestro proyecto.

API Management

API Management es una plataforma cloud para administrar las API en todos los entornos. Como plataforma como servicio, API Management permite el ciclo de vida completo de la API. API Management funciona como una puerta de enlace, un plano de administración y un portal para desarrolladores, con características diseñadas especialmente para realizar la gestión (Microsoft Azure, s.f.).

Es muy importante que una organización use esta plataforma porque permite tener una mayor gobernanza y control sobre las APIs que expone. Permite una gran cantidad de peticiones y también el balanceo de carga para evitar sobrecarga a una sola instancia del API. En un contexto permite el análisis de métricas y control de costos de cada API.

Log Analytics

Log Analytics de Azure es una herramienta para ejecutar consultas de los registros guardados en el almacén de registros de Azure Monitor (Microsoft Azure, s.f.).

Log Analytics es un recurso cloud que permite la gestión de los registros y métricas generados por otros recursos de Azure. Para obtener la información y métricas desde el Azure monitor se puede usar su propio lenguaje Kusto Query Language (KQL).

Azure SQL Database

Es una base de datos administrado en la nube. Es una plataforma como servicio (PaaS) completamente administrado que se encarga de la mayoría de las funciones de administración de bases de datos, como las de actualizar, aplicar revisiones, crear copias backup de seguridad sin necesidad de intervención humana. Azure SQL Database utiliza la versión estable más reciente del motor de base de datos de Microsoft SQL Server (Microsoft Azure, s.f.).

Azure Key Vault

Azure Key Vault es un servicio de seguridad en la nube para almacenar y proteger secretos de forma segura. Un secreto es cualquier información sensible que desea controlar estrechamente el acceso, como claves de API, contraseñas, certificados o claves criptográficas. El servicio Key Vault acepta dos tipos de contenedores: almacenes y grupos de módulos de seguridad de hardware administrados (Microsoft Azure, s.f.).

En el presente trabajo cumplió un rol importante para almacenar las credenciales para solicitar información desde las aplicaciones web hacia los demás componentes dentro de la arquitectura de la solución. Las aplicaciones web tenían dentro de sus variables de entorno la URL hacía el Key Vault.

Microsoft Entra ID

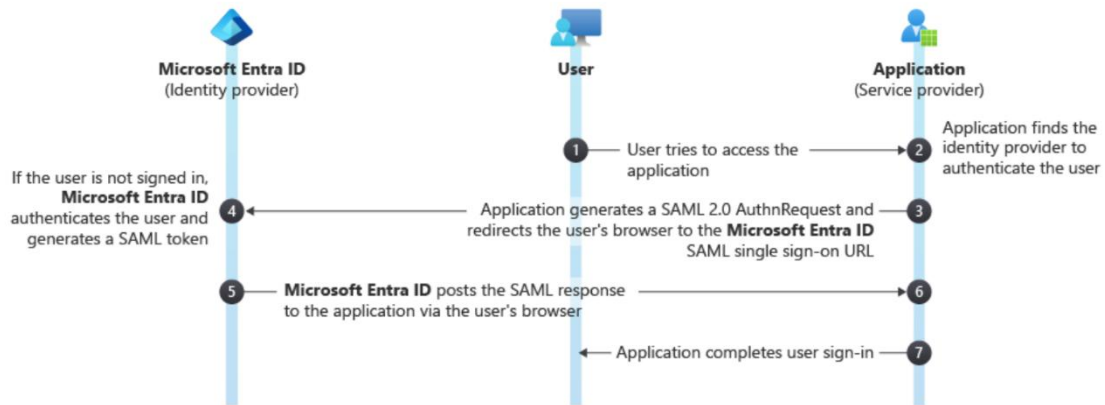
Es un servicio de Azure Cloud para la administración de identidades y acceso basado en la nube. Provee mecanismos de identidad, autenticación, políticas y protección esencial para proteger a los usuarios, dispositivos, aplicaciones y recursos (Microsoft Azure, s.f.).

Este servicio es muy importante porque permite el inicio de sesión único (SSO), por lo cual los usuarios de una empresa solo van a necesitar usar una sola credencial para acceder a múltiples sistemas, aplicaciones y servicios.

El flujo de autenticación SSO cuando se integra con Microsoft Entra ID se muestra en la figura 4.

Figura 4

Flujo de autenticación SSO



Nota. Fuente: Microsoft Azure

Fundamentos Metodológicos

Scrum

Es una metodología de desarrollo ágil que se basa en procesos iterativos llamados sprint, que se utilizan en el desarrollo de software. Scrum es un marco de trabajo ágil que permite entregar valor al cliente de forma continua durante todo el desarrollo del proyecto. Su principal objetivo es satisfacer las necesidades del cliente impulsando una cultura de comunicación abierta, responsabilidad compartida y mejora continua. El proceso de desarrollo del sistema comienza con los requerimientos iniciales del usuario, seguida de la creación de una lista de características ordenadas por prioridad (Product Backlog) que el propietario del producto desea de acuerdo con los requerimientos del cliente (Zhmai, 2022).

En esta metodología contiene ceremonias para llevar un mejor control de los incrementos que se realizan en cada sprint. Estas ceremonias se representan en la tabla 8.

Tabla 8
Ceremonias Scrum

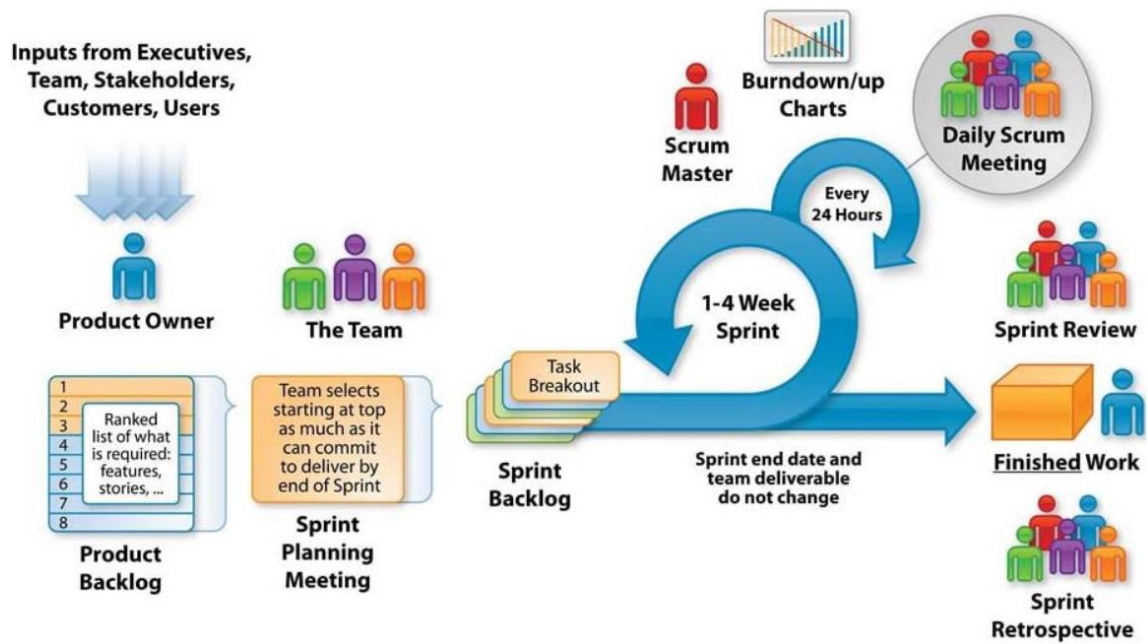
Ceremonia	Actividades
Sprint Planning	– Se analizan las actividades que se realizan en el siguiente sprint. – Se escogen las actividades desde el sprint backlog. – Se realizan cada inicio de sprint.
Daily Scrum	– Se informa las actividades que se están trabajando. – Se notifica si existe algún bloqueo para tomar medidas de acuerdo con la prioridad. – Se realizan diariamente.
Sprint Review	– Se revisa las actividades que se comprometieron en el sprint. – Se analiza que actividades quedaron pendientes para pasarlos a un siguiente sprint. – Se realiza al finalizar el sprint.
Sprint Retrospective	– Se evalúa que cosas salieron bien para repetirlas y continuar haciéndolas. – Se evalúa que cosas salieron mal para evitar repetir el error. – Se realiza al finalizar el sprint.

Nota. Fuente: Elaboración propia

En esta metodología nos brinda una estructura con etapas desde el análisis y diseño de los requerimientos hasta entregar el producto final hacía el cliente. La estructura se representa en la figura 5.

Figura 5

Estructura de Scrum Framework



Nota. Fuente: Zhmai (2022)

3.2.5. IMPLEMENTACIÓN DE LAS ÁREAS, PROCESOS, SISTEMAS Y BUENAS PRÁCTICAS

La metodología en el presente proyecto es Scrum. Por lo cual cada miembro del equipo cuenta un rol importante en la construcción del proyecto. Estos roles se representan en la tabla 9.

Tabla 9

Roles involucrados en el proyecto

Rol	Cantidad de recursos	Función
Product Owner	1	– Define los requerimientos del sistema. – Elaboración y priorización del producto backlog.
Scrum Master	1	– Encargado de las ceremonias SCRUM. – Gestiona y verifica las historias cumplan con todos los puntos necesarios de la actividad a realizar. – Gestiona los avances y bloqueos en las actividades del equipo.
Líder técnico	1	– Lidera al equipo técnico en el desarrollo de sus actividades. – Resolver dudas técnicas que aparezcan durante el desarrollo. – Apoyar en incidentes o bloqueos técnicos.
Desarrollador Frontend	1	– Encargado del desarrollo frontend del sistema.
Desarrollador Backend	2	– Encargado del desarrollo backend del sistema. – Apoyo en la elaboración de la arquitectura del sistema.

		– Soporte en la configuración de componentes cloud.
Quality Analyst (QA)/Tester	1	– Realizar pruebas a los avances de los desarrolladores. – Realizar un plan de pruebas.
Cloud Engineer	1	– Realizar la configuración de los componentes cloud. – Validar la estabilidad operativa. – Apoyo en la elaboración de la arquitectura del sistema.
Analista de seguridad	1	– Validar que el desarrollo o configuración de componentes no involucre un riesgo de seguridad.

Nota. Fuente: Elaboración propia

El autor del presente proyecto tuvo el rol de desarrollador backend.

A. ETAPAS DE DESARROLLO

Para el presente trabajo se definió 4 etapas según lo descrito en la tabla 6. A continuación detallaremos las actividades realizadas en cada etapa del proyecto:

A.1 Análisis

En etapa se realizaron las reuniones con los stakeholders. Los involucrados en las reuniones fueron el Product Owner, el Líder Técnico y el Cloud Engineer. A partir de estas sesiones se pudo lograr los requerimientos iniciales del sistema. El listado de estos requerimientos se muestra en la tabla 10.

Tabla 10

Requerimientos iniciales del sistema.

ID	Requerimiento	Descripción
OPR001	Iniciar sesión	El usuario debe ingresar con las mismas credenciales que manejan dentro de la organización.
OPR002	Mostrar un dashboard	Crear un dashboard para mostrar en gráficos de barras las métricas de las APIs.
OPR003	Mostrar información de las APIs	Mostrar la información registrada en el core bancario de cada API.
OPR004	Mantenimiento de owner de las APIs	Mantenimiento para asignar APIs a un usuario para que cumpla el rol de Owner.
OPR005	Mantenimiento de costos	Mantenimiento mensual de los costos de infraestructura.
OPR006	Listar los costos de APIs mensualmente	Listar los costos de APIs según el número de peticiones y el costo de infraestructura.
OPR007	Exportar los costos de APIs	Exportar en un Excel el valor de los costos de APIs mensualmente.

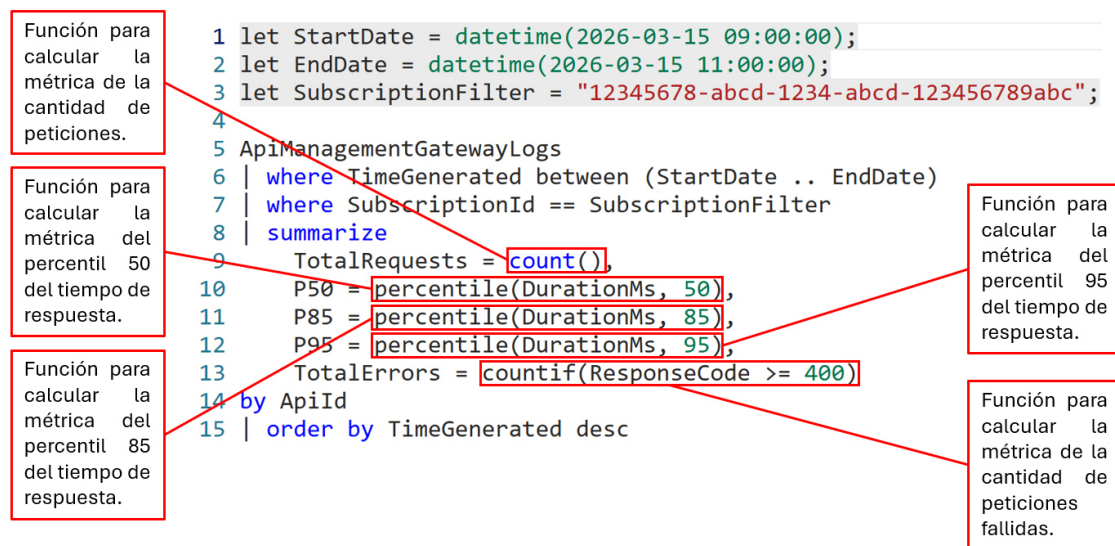
Nota. Fuente: Elaboración propia.

A.1.1 métricas

En esta etapa se definen las métricas, las cuales se obtienen a partir de los registros obtenidos desde el Log Analytics mediante consultas en KQL. En la figura 6 se muestra la consulta utilizada, en la cual se resalta las funciones utilizadas para calcular las métricas.

Figura 6

Consulta para calcular las métricas



Nota. Fuente: Elaboración propia

Métrica 1: Cantidad total de peticiones

Esta métrica indica la cantidad de peticiones realizadas a una API durante un periodo determinado. Esto permite visualizar el nivel de uso de la API por parte de los consumidores.

La fórmula para obtener el total de peticiones es:

$$Total\ de\ peticiones = \sum_{i=1}^n 1$$

Donde:

- n: número total de peticiones.

La fórmula se implementa mediante la función `count()` en la consulta en KQL mostrada en la figura 6.

Métrica 2: Percentil 50 del tiempo de respuesta

Esta métrica indica la mediana del tiempo de la respuesta y representa el valor por debajo del cual se encuentra el 50% de los tiempos de respuesta de las solicitudes. Permite visualizar el tiempo de respuesta más representativo de la API ya que no es afectado por solicitudes con tiempos de respuesta muy altos.

Para calcular el percentil 50 primero se debe ordenar la lista de los tiempos de respuesta de menor a mayor. Posteriormente se debe aplicar la siguiente formula:

$$P_{50} = 0.50 (n+1)$$

Donde:

- n: número de peticiones
- P_{50} : Posición del percentil en la lista de los tiempos de respuesta ordenada.

Si el valor de la posición es un valor entero, se utiliza esa posición para ubicar el valor del percentil 50 en la lista de los tiempos de respuesta, pero si es un decimal se debe realizar una interpolación.

El cálculo de este percentil se realiza mediante la función `percentile(DurationMs, 50)` en la consulta en KQL mostrada en la figura 6.

Métrica 3: Percentil 85 del tiempo de respuesta

Esta métrica indica el valor por debajo del cual se encuentran 85% de los tiempos de respuesta de las solicitudes. Permite mostrar el tiempo de respuesta que experimenta la mayoría de los usuarios y nos ayuda a analizar la calidad del servicio. Para calcular el percentil 85 primero se debe ordenar la lista de los tiempos de respuesta de menor a mayor. Posteriormente se debe aplicar la siguiente formula:

$$P_{85} = 0.85 (n+1)$$

Donde:

- n: número de peticiones
- P_{85} : Posición del percentil en la lista de los tiempos de respuesta ordenada.

Si el valor de la posición es un valor entero, se utiliza esa posición para ubicar el valor del percentil 85 en la lista de los tiempos de respuesta, pero si es un decimal se debe realizar una interpolación.

El cálculo de este percentil se realiza mediante la función `percentile(DurationMs, 85)` en la consulta en KQL mostrada en la figura 6.

Métrica 4: Percentil 95 del tiempo de respuesta

Esta métrica indica el valor por debajo del cual se encuentran 95% de los tiempos de respuesta de las solicitudes. Permite analizar el comportamiento del API en escenarios cercanos al peor caso detectando demoras elevadas. Para calcular el percentil 95 primero se debe ordenar la lista de los tiempos de respuesta de menor a mayor. Posteriormente se debe aplicar la siguiente formula:

$$P_{95} = 0.95 (n+1)$$

Donde:

- n: número de peticiones
- P_{95} : Posición del percentil en la lista de los tiempos de respuesta ordenada.

Si el valor de la posición es un valor entero, se utiliza esa posición para ubicar el valor del percentil 95 en la lista de los tiempos de respuesta, pero si es un decimal se debe realizar una interpolación.

El cálculo de este percentil se realiza mediante la función `percentile(DurationMs, 95)` en la consulta en KQL mostrada en la figura 6.

Métrica 5: Cantidad de solicitudes fallidas

Esta métrica indica la cantidad de peticiones que no fueron procesadas correctamente hacia el API y devuelven un código de error. Los códigos en las respuestas HTTP son las siguientes:

- 4xx: Errores del cliente como pueden ser solicitud incorrectas o falta de autenticación.
- 5xx: Errores del servidor como pueden ser errores en el API o inestabilidad del ambiente.

Por lo tanto, para el cálculo de esta métrica solo se consideran las solicitudes con código de respuesta mayor o igual a 400.

La fórmula para obtener la cantidad de solicitudes fallidas es:

$$\text{Cantidad de solicitudes fallidas} = \sum_{i=1}^n f(i)$$

Donde:

$$f(i) = \begin{cases} 1, & \text{si código de respuesta} \geq 400 \\ 0, & \text{si código de respuesta} < 400 \end{cases}$$

Y:

- n: número de peticiones

El cálculo de este percentil se realiza mediante la función `countif(ResponseCode >= 400)` en la consulta en KQL mostrada en la figura 6.

En la tabla 11 se muestra el ejemplo de los resultados obtenidos de la consulta en KQL, donde se muestran las métricas para un conjunto de APIs en un intervalo de tiempo.

Tabla 11

Ejemplo del resultado de la consulta de métricas

API	Total de peticiones	P50 tiempo de respuesta (ms)	P85 tiempo de respuesta (ms)	P95 tiempo de respuesta (ms)	Solicitudes fallidas
api-credit-v1	15,420	185	290	410	63
api-debit-v1	12,310	210	340	520	51
api-approval-v1	21,580	275	420	610	96
api-servicing-v1	18,940	240	370	540	72
api-clientes-v1	9,870	195	300	430	23
api-exchange-v1	16,120	225	360	510	48
api-bussiness-v1	8,540	175	260	370	17
api-transfer-v1	25,630	150	230	350	12
api-account-v1	7,950	190	280	420	26
api-user-v1	6,780	180	250	360	19

Nota. Fuente: Elaboración propia

A.1.2 Cálculo de los costos

En esta etapa de análisis se definen las fórmulas y variables para el cálculo de los costos totales de las APIs, los cuales se obtienen considerando dos costos principales: El costo total de infraestructura y el costo total del consumo de la API.

En la tabla 12 se detallan los costos de infraestructura. Estos costos son ingresados al sistema por el usuario con rol Admin y se encuentran en la base de datos.

Tabla 12
Costos de infraestructura

Costo	Descripción
Certificados	Representa al costo de certificados digitales de seguridad.
Soporte	Representa al costo de las actividades de monitoreo, mantenimiento y solución de incidentes.
Infraestructura_dev	Representa el costo de infraestructura en el ambiente de desarrollo según la cantidad de peticiones como puede ser el valor del servicio API Managment y Log Analytics.
Infraestructura_qa	Representa el costo de infraestructura en el ambiente de certificación según la cantidad de peticiones como puede ser el valor del servicio API Managment y Log Analytics.
Infraestructura_prod	Representa el costo de infraestructura en el ambiente de producción según la cantidad de peticiones como puede ser el valor del servicio API Managment y Log Analytics.

Nota. Fuente: Elaboración propia

La fórmula para calcular el costo total de infraestructura es

$$\text{costo total de infraestructura} = \frac{\text{costo de certificados} + \text{costo de soporte}}{n_{APIs}}$$

Donde:

- n_{APIs} : Es el número total de APIs desplegadas en el APIM Managment.

Para calcular el costo total del consumo de APIs se debe sumar los costos de consumo de APIS por ambiente.

La fórmula para calcular el costo del consumo del ambiente de desarrollo:

$$cost_APIs_{dev} = \frac{Petición\ por\ API_{dev} \times Infraestructura_{dev}}{Petición\ totales_{dev}}$$

Donde:

- *Petición por API_{dev}*: Son la cantidad de peticiones por API en el ambiente de desarrollo.
- *Infraestructura_{dev}*: Es el costo por infraestructura según la cantidad de peticiones en el ambiente de desarrollo. Este valor está almacenado en la base de datos.
- *Petición totales_{dev}*: Es la suma de todas las peticiones de las APIs en el ambiente de desarrollo.

La fórmula para calcular el costo del consumo del ambiente de certificación:

$$cost_APIs_{cert} = \frac{Petición\ por\ API_{cert} \times Infraestructura_{qa}}{Petición\ totales_{cert}}$$

Donde:

- *Petición por API_{cert}*: Son la cantidad de peticiones por API en el ambiente de certificación.
- *Infraestructura_{qa}*: Es el costo por infraestructura según la cantidad de peticiones en el ambiente de certificación. Este valor está almacenado en la base de datos.
- *Petición totales_{cert}*: Es la suma de todas las peticiones de las APIs en el ambiente de certificación.

La fórmula para calcular el costo del consumo del ambiente de producción:

$$cost_APIs_{prod} = \frac{Petición\ por\ API_{prod} \times Infraestructura_{prod}}{Petición\ totales_{prod}}$$

Donde:

- *Peticiones por API_{prod}*: Son la cantidad de peticiones por API en el ambiente de producción.
- *Infraestructura_{prod}*: Es el costo por infraestructura según la cantidad de peticiones en el ambiente de producción. Este valor está almacenado en la base de datos.
- *Peticiones totales_{prod}*: Es la suma de todas las peticiones de las APIs en el ambiente de producción.

La fórmula para calcular el costo total del consumo de las APIs es la siguiente:

$$\text{Costo total del consumo} = \text{cost_APIs}_{dev} + \text{cost_APIs}_{cert} + \text{cost_APIs}_{prod}$$

Donde:

- *cost_APIs_{dev}*: Es el costo del consumo del ambiente de desarrollo previamente calculado.
- *cost_APIs_{cert}*: Es el costo del consumo del ambiente de certificación previamente calculado.
- *cost_APIs_{prod}*: Es el costo del consumo del ambiente de producción previamente calculado.

La fórmula para calcular el costo total del API es la siguiente:

$$\text{Costo total por API}$$

$$= \text{costo total de infraestructura} + \text{Costo total del consumo}$$

Donde:

- *costo total de infraestructura*: Es el costo total de infraestructura previamente calculado.
- *Costo total del consumo*: Es el costo total del consumo de las APIs previamente calculado.

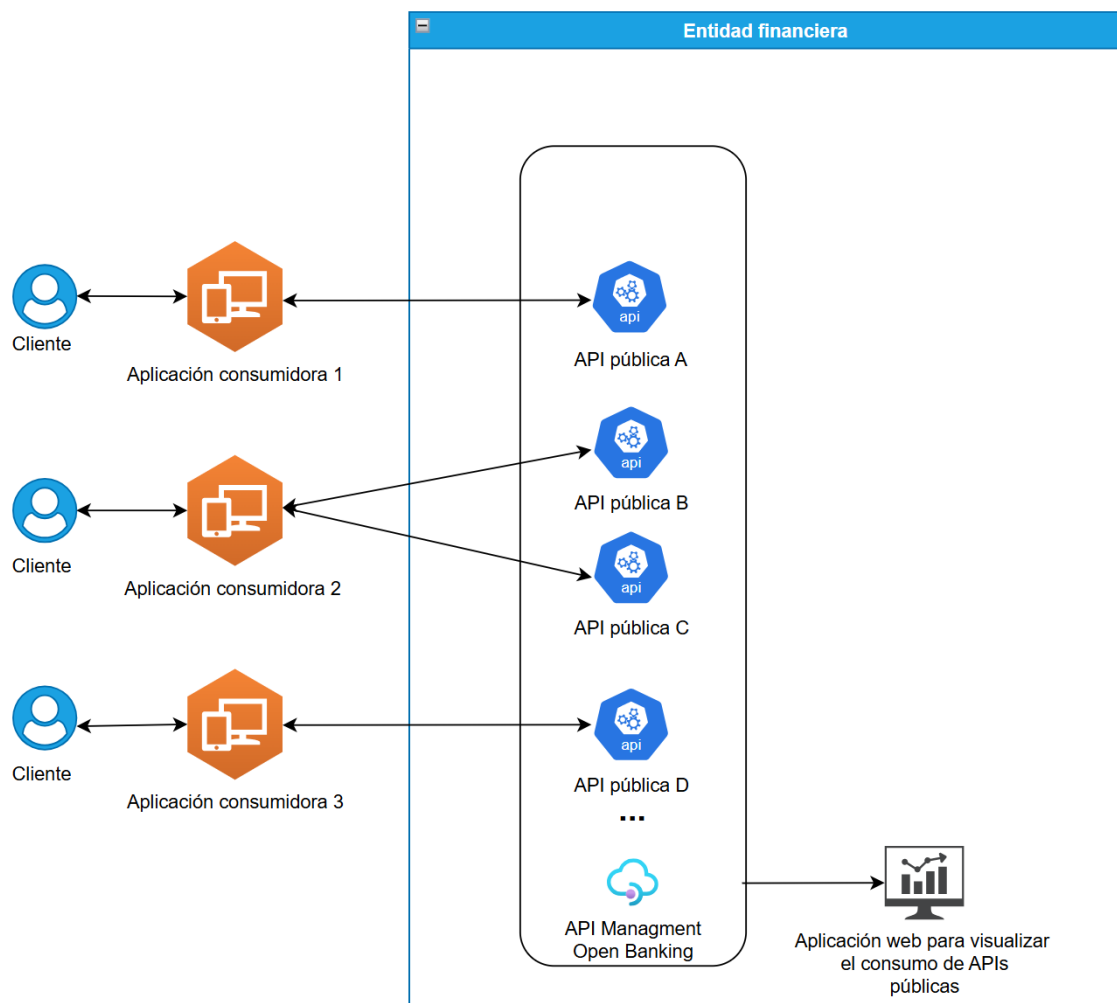
A.2 Diseño

A.2.1 Modelo de negocio

En esta etapa del proyecto se realizó la documentación del proyecto. Se definió el modelo de negocio del sistema, la cual se muestra en la figura 7. En esta figura podemos visualizar como el modelo Open Banking expone APIs públicas para que terceros autorizados puedan consumirlas y generen nuevas aplicaciones y servicios. Adicionalmente como se relaciona con la aplicación web del presente trabajo para optimizar el tiempo de análisis y mejorar la gestión de costos de las APIs públicas de la entidad financiera.

Figura 7

Modelo de negocio

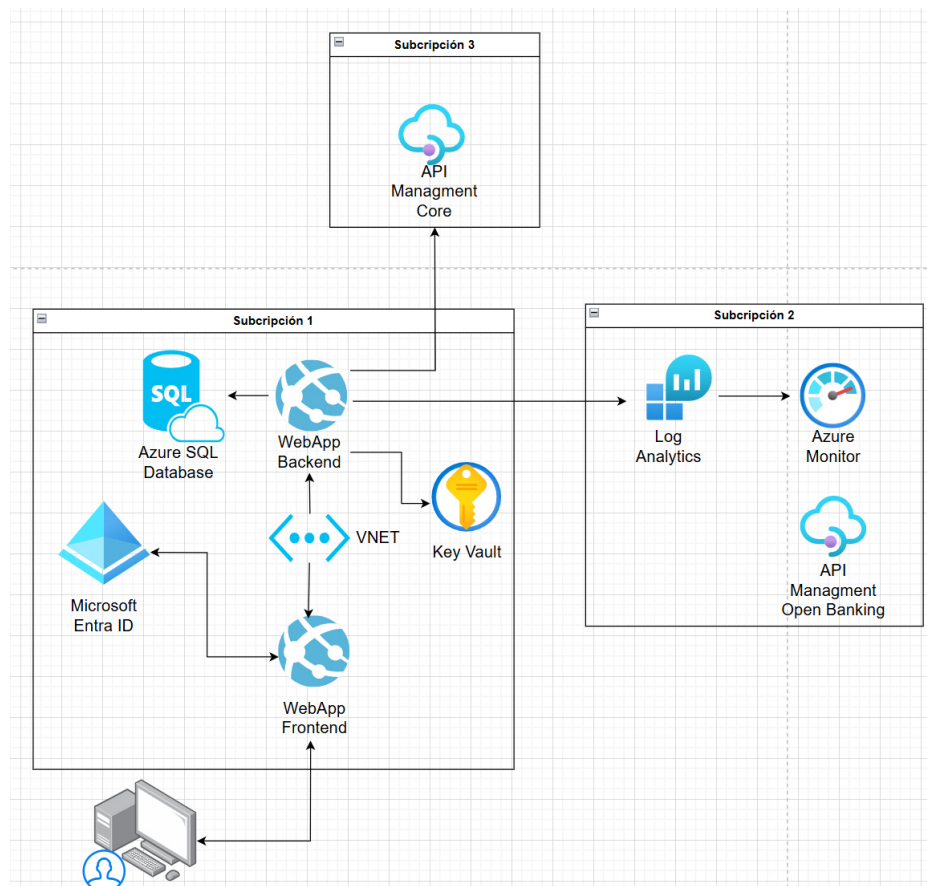


Nota. Fuente: Elaboración propia

A.2.2 Arquitectura del sistema

En esta etapa también se diseñó la arquitectura del sistema, la cual se muestra en la figura 8. En esta figura se muestra cómo se relaciona la solución implementada con otros servicios de la organización. En esta figura se muestra como el usuario utiliza el servicio de Microsoft Entra ID para autenticarse y poder interactuar con el WebApp del frontend, luego a través de un Vnet se relaciona con el WebApp del backend. Este WebApp utiliza el recurso Key Vault para almacenar información sensible y también utiliza el servicio Azure SQL Database para almacenar la información de las APIs. Este WebApp se relaciona con el API Managment Core para obtener las APIs registradas en la organización y se relaciona con el Log Analytics para obtener la información de las peticiones que se realizaron por medio del API Management de Open Banking.

Figura 8
Arquitectura del sistema



Nota. Fuente: Elaboración propia

A.2.3 Casos de uso

En esta etapa se definieron los casos de uso. Los actores definidos para nuestros casos de uso son los siguientes:

- **Administrador:** Gestiona los accesos para visualizar las APIs públicas y realiza el mantenimiento de los costos mensuales de infraestructura.
- **Responsable del API:** Consulta las métricas y costos de sus APIs públicas.
- **Visor (Gerencia):** Consulta las métricas y costos de todas las APIs públicas en la entidad financiera.

Los casos de uso abarcan las principales funcionalidades implementadas en el presente trabajo. En la tabla 13 se muestra el detalle de los casos de uso:

Tabla 13

Detalle de los casos de uso

Código	Casos de uso	Actores	Descripción
CU01	Consultar métricas en dashboard	Administrador, Visor y Responsable del API	Los actores consultan las métricas de las APIs en el dashboard. Sin embargo, el Responsable del API solo puede consultar las métricas de sus APIs mientras los demás actores pueden consultar de todas las APIs.
CU02	Gestionar permisos por API	Administrador	El administrador gestiona los permisos para consultar las métricas de cada API.
CU03	Consultar inventario de APIs	Administrador y Visor	Los actores consultan el inventario de APIs donde se mostrará información detallada del API.
CU04	Gestionar costos de infraestructura	Administrador	El administrador realiza el mantenimiento de los costos mensuales de infraestructura.

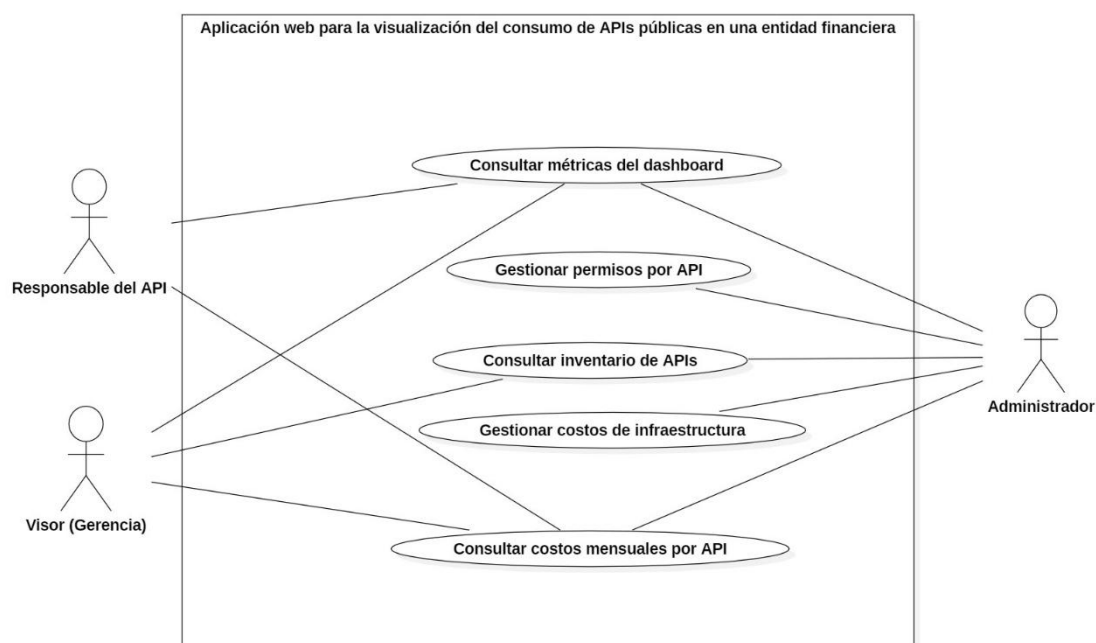
CU05	Consultar costos mensuales por API	Administrador, Visor y Responsable del API solo Responsable del API puede visualizar los costos de sus APIs mientras los demás actores pueden consultar todas las APIs.
------	------------------------------------	---

Nota. Fuente: Elaboración propia

En la figura 9 se muestra el diagrama de caso de uso.

Figura 9

Diagrama de casos de uso

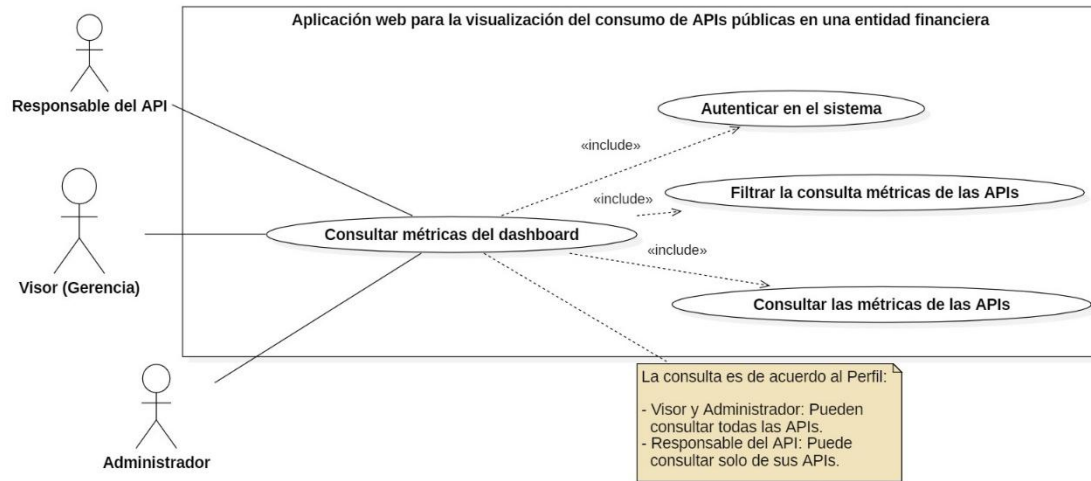


Nota. Fuente: Elaboración propia

En la figura 10 se muestra el diagrama del caso de uso CU01 y en la tabla 14 se muestra la especificación del caso de uso.

Figura 10

Diagrama de caso de uso: CU01 - Consultar métricas del dashboard



Nota. Fuente: Elaboración propia

Tabla 14

Especificación de caso de uso: CU01 - Consultar métricas del dashboard

Caso de uso	Consultar métricas del dashboard
Descripción	Permite consultar y visualizar las métricas de las APIs públicas en el dashboard de la aplicación web.
Actores	– Responsable del API – Visor – Administrador
Precondiciones	Los responsables del API deben ser asociados a sus APIs
Flujo principal	1. El usuario debe autenticarse en el sistema. 2. El usuario debe ingresar al módulo del dashboard.

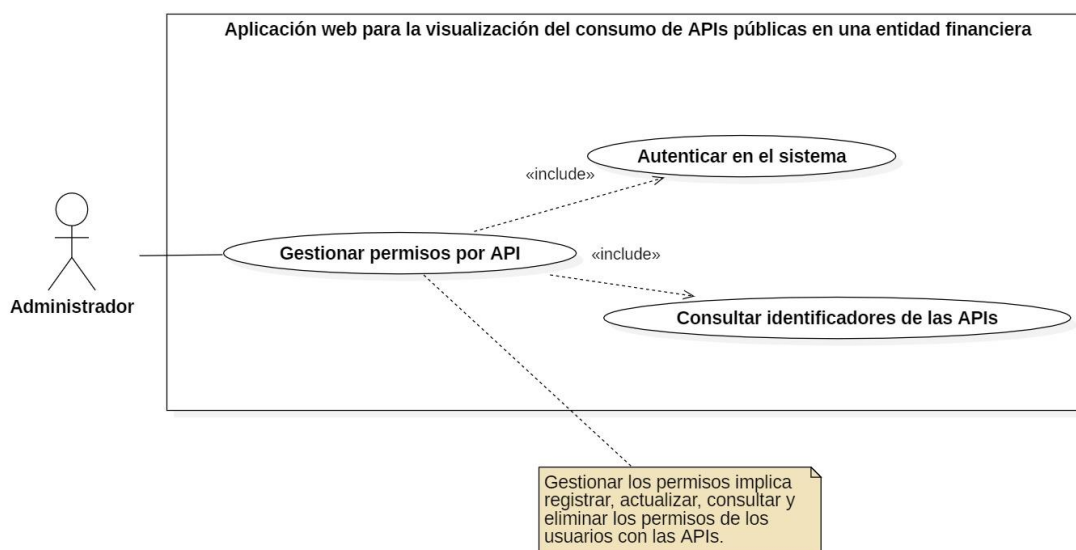
3. El usuario debe ingresar los filtros para realizar la consulta.
4. El usuario realiza la consulta de las APIs públicas.

Nota. Fuente: Elaboración propia

En la figura 11 se muestra el diagrama del caso de uso CU02 y en la tabla 15 se muestra la especificación del caso de uso.

Figura 11

Diagrama de caso de uso: CU02 - Gestionar permisos por API



Nota. Fuente: Elaboración propia

Tabla 15

Especificación de caso de uso: CU02 - Gestionar permisos por API

Caso de uso	Gestionar permisos por API
Descripción	Permite gestionar los permisos de los usuarios para que puedan visualizarlas sus APIs.

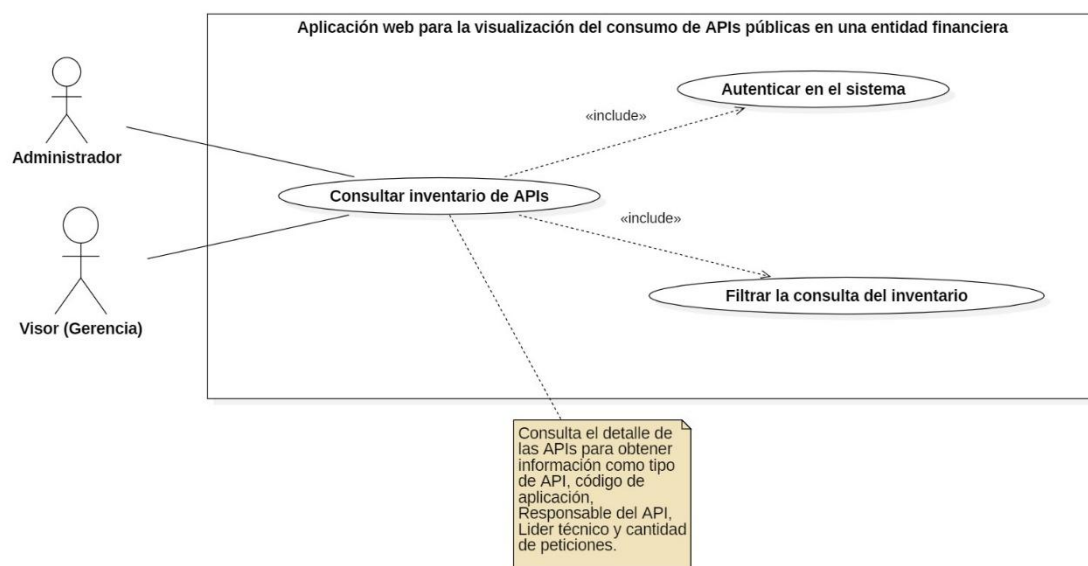
Actores	– Administrador
Flujo principal	1. El Administrador debe autenticarse en el sistema.
	2. El Administrador consulta los identificadores de las APIs.
	3. El Administrador registrar los permisos para que los usuarios con perfil Responsable de APIs puedan visualizar sus APIs en el sistema.

Nota. Fuente: Elaboración propia

En la figura 12 se muestra el diagrama del caso de uso CU03 y en la tabla 16 se muestra la especificación del caso de uso.

Figura 12

Diagrama de caso de uso: CU03 - Consultar inventario de APIs



Nota. Fuente: Elaboración propia

Tabla 16

Especificación de caso de uso: CU03 - Consultar inventario de APIs

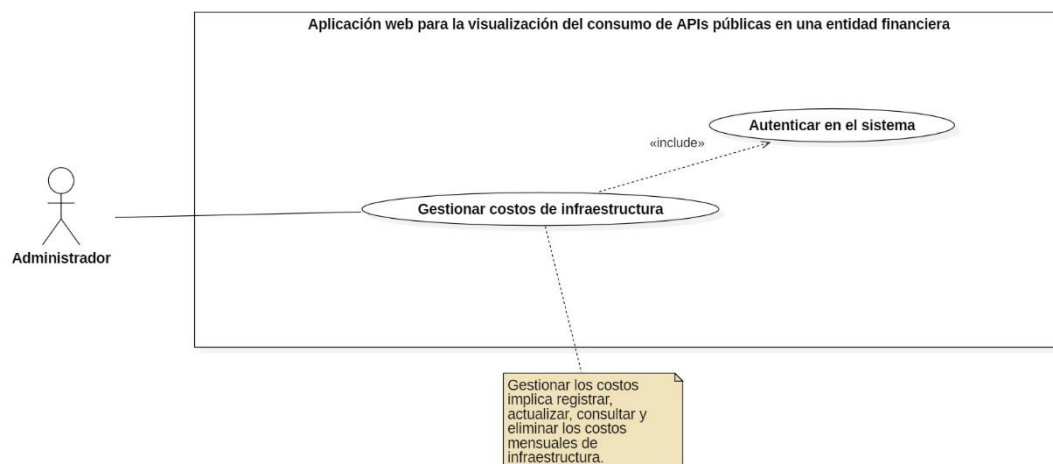
Caso de uso	Consultar inventario de APIs
Descripción	Permite consultar la información de las APIs que tuvieron peticiones en el mes seleccionado.
Actores	– Administrador y Visor
Flujo principal	1. El usuario debe autenticarse en el sistema. 2. El usuario filtra las APIs por el mes en el módulo de inventario de APIs. 3. El usuario consulta el detalle de las APIs.

Nota. Fuente: Elaboración propia

En la figura 13 se muestra el diagrama del caso de uso CU04 y en la tabla 17 se muestra la especificación del caso de uso.

Figura 13

Diagrama de caso de uso: CU04 - Gestionar costos de infraestructura



Nota. Fuente: Elaboración propia

Tabla 17

Especificación de caso de uso: CU04 - Gestionar costos de infraestructura

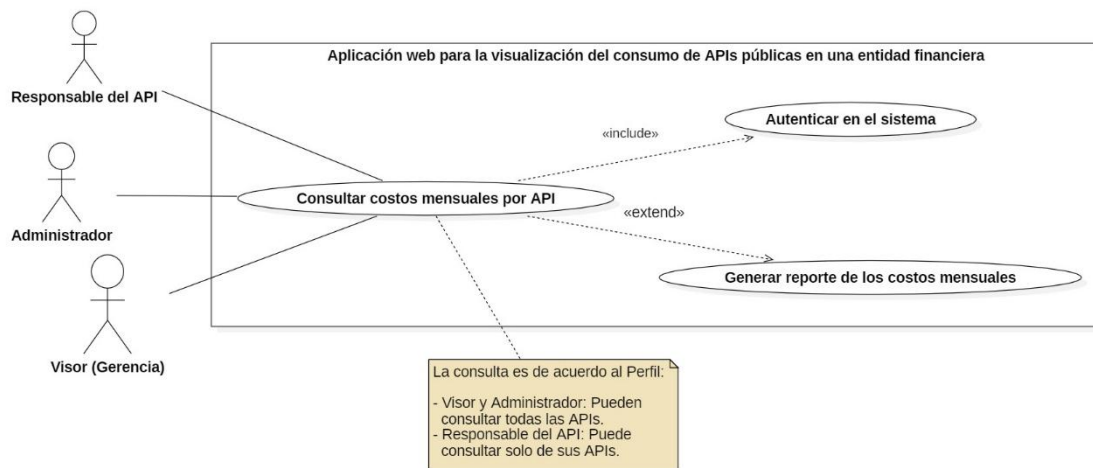
Caso de uso	Gestionar costos de infraestructura
Descripción	Permite gestionar los costos de infraestructura para obtener el costo total mensual del API.
Actores	– Administrador
Flujo principal	1. El usuario debe autenticarse en el sistema. 2. El usuario lista los costos de infraestructura. 3. El usuario gestiona los costos de infraestructura. Implica registrar, actualizar, consultar y remover los costos.

Nota. Fuente: Elaboración propia

En la figura 14 se muestra el diagrama del caso de uso CU05 y en la tabla 18 se muestra la especificación del caso de uso.

Figura 14

Diagrama de caso de uso: CU05 - Consultar costos mensuales por API



Nota. Fuente: Elaboración propia

Tabla 18*Especificación de caso de uso: CU05 - Consultar costos mensuales por API*

Caso de uso	Consultar costos mensuales por API
Descripción	Permite consultar, visualizar y exportar reporte de los costos mensuales de la API.
Actores	– Administrador, Visor y Responsable del API.
Precondiciones	Se debe registrar los costos de infraestructura del mes seleccionado.
Flujo principal	1. El usuario debe autenticarse en el sistema. 2. El usuario filtra los costos por mes. 3. El usuario consulta los costos totales por mes. 4. El usuario genera el reporte de los costos por mes.

Nota. Fuente: Elaboración propia

A.2.4 Gestión de tareas y seguimiento del proyecto

En la etapa de diseño se definieron los tipos de issues que se van a utilizar para plasmar las actividades de acuerdo con los requerimientos del sistema. Estos tipos de issues se representan a la tabla 19.

Tabla 19*Tipos de issues*

Tipo	Descripción
Historia de usuario (HU)	Este tipo de issue es usado cuando la actividad va a generar valor al usuario. Por ejemplo, puede ser agregar un nuevo módulo al sistema.
Tarea	Este tipo de issue es usuario cuando la actividad no genera un valor al usuario. Por

	ejemplo, puede ser la configuración de algún componente.
Request	Este tipo de issue es usado cuando se necesita realizar una actividad adicional pero que no será desplegado a producción. Por ejemplo, puede ser revisar los logs de un componente para revisar algún incidente.
Spike	Este tipo de issue es usado cuando se debe investigar para aprender o validar. Por ejemplo, investigar una nueva tecnología a utilizar.

Nota. Fuente: Elaboración propia

En este proyecto usamos la metodología Scrum, por lo cual se diseñó el Product Backlog con los issues iniciales de acuerdo con los requerimientos iniciales priorizados por el Product Owner. Los Issues iniciales que se definieron en el proyecto se encuentran en la tabla 20.

Tabla 20
Product Backlog con los issues iniciales

ID	Tipo de Issue	Título	Descripción
OPI001	Request	Realizar documentación inicial	Se debe realizar la documentación inicial del sistema.
OPI002	Spike	Integración con el Log Analytics	Se debe investigar cómo se debe integrar con el Log Analytics.
OPI002	Tarea	Configurar los componentes cloud	Se debe configurar los componentes cloud que se necesitan dentro de la arquitectura del sistema.
OPI003	Tarea	Configurar la integración con Microsoft Entra ID	Se debe configurar los grupos de seguridad para que los usuarios ingresen con sus cuentas de la

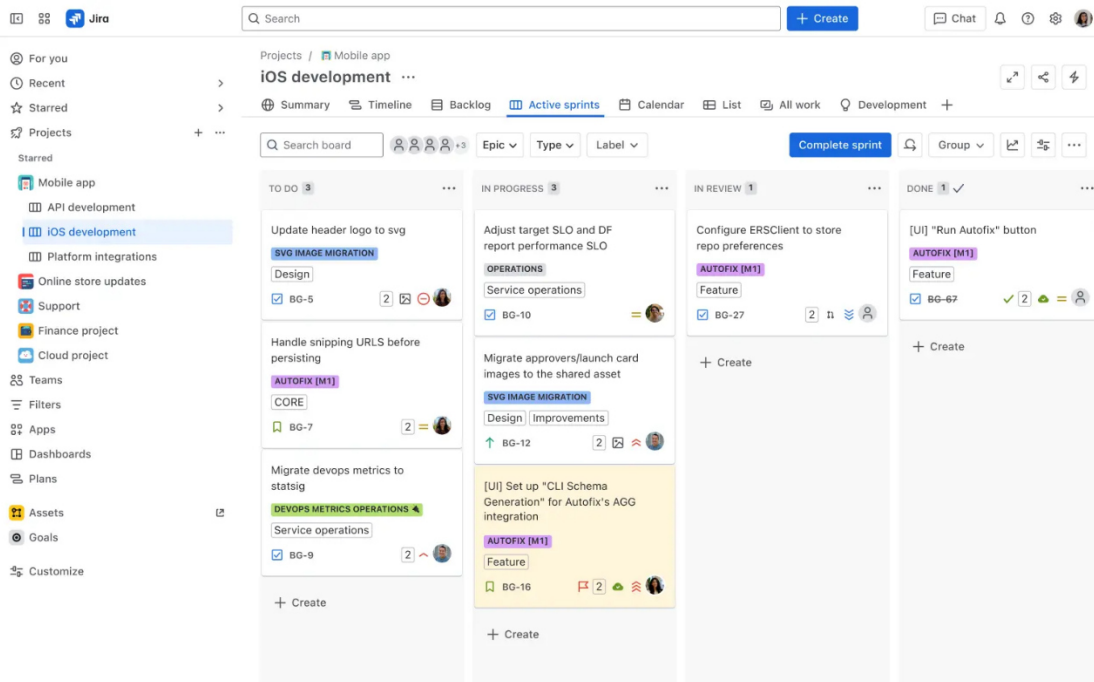
			organización. Adicionalmente se debe asignar los roles correspondientes.
OPI004	HU	Iniciar sesión	Se debe agregar el módulo de iniciar sesión.
OPI005	HU	Implementar al dashboard para la visualización de métricas	Se debe mostrar las métricas de cada API en gráficos de barras. Esta información viene desde el Log Analytics.
OPI006	HU	Implementar módulo de inventario de APIs	Este módulo debe mostrar la información detallada que viene desde el Core Bancario.
OPI007	HU	Módulo de mantenimiento de Owner	Este módulo permite el asignar APIs a los owners de acuerdo con su email.
OPI008	HU	Implementar mantenimiento de costos	Este módulo permite la gestión de costos de infraestructura.
OPI009	HU	Implementar listado de información de costos por API	Este módulo realiza el cálculo de los APIs de acuerdo con el número de peticiones y a los costos de infraestructura.
OPI010	HU	Exportar costos por API	Esta funcionalidad exporta en un Excel los costos de infraestructura y el cálculo de los costos por API
OP011	Request	Elaborar manual de usuario	Realiza el manual de usuario con las funcionalidades implementadas.

Nota. Fuente: Elaboración propia

Para realizar un mejor control del avance de cada issue, se usó la herramienta Board que nos proporciona la plataforma Jira. En la figura 15 se muestra la herramienta Board.

Figura 15

Herramienta Board de Jira



Nota. Fuente: Jira

Los estados de issues se muestran en la tabla 21.

Tabla 21

Estados de Issues

Estado	Descripción
Abierto	Estado de creación del issue.
Análisis y diseño	Estado para el proceso de entender el requerimiento a realizar.
Implementación	Estado para el proceso de ejecución del desarrollo, implementación o configuración que requiere la tarea.
Implementación done	Estado para indicar que terminó la etapa de implementación.

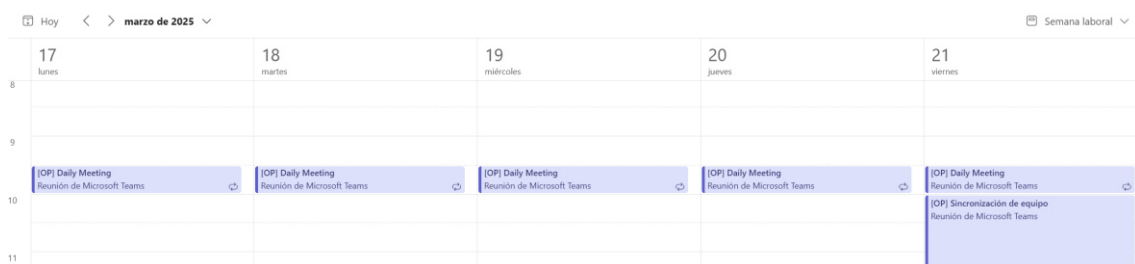
Bloqueado	Estado para indicar que se presentan bloqueos para completar la actividad.
Pruebas QA	Estado para realizar las pruebas de la implementación.
Pruebas QA done	Estado para indicar que terminaron que se culminaron las pruebas.
Gestión a producción	Estado para gestionar el pase a producción.
Producción en proceso	Estado para indicar que el issue está pasando a producción.
Resuelto	Estado para indicar que se completó el issue.

Nota. Fuente: Elaboración propia

Para llegar un mejor control de cómo va el avance del equipo, también se definieron Daily Scrum diarias para mostrar el avance de cada uno o mencionar si existe algún bloqueo. Las reuniones híbridas se agendaban todos los días al inicio del horario laboral como se muestra en la figura 16 utilizando la plataforma de Microsoft Teams.

Figura 16

Reuniones de Daily Scrum



Nota. Fuente: Microsoft Teams

A.3 Desarrollo del sistema

Esta etapa inicia con la documentación técnica para posteriormente comenzar con el desarrollo del sistema.

Se documentaron las tecnologías a utilizar tanto para la implementación local como para otro ambiente. Las tecnologías se muestran en la tabla 22.

Tabla 22

Tecnologías utilizadas

Categoría	Tecnologías
Frontend	JavaScript/Angular
Backend	Java 17/Spring WebFlux
Base de datos	SQL Server/Azure SQL Database
Componentes cloud	Azure/WebApp, Key Vault, API Management y Log Analytics
Seguridad	Microsoft Entra ID
Control de versiones	GitHub

Nota. Fuente: Elaboración propia

Se documentaron los roles y los permisos que tienen cada uno de los usuarios dentro del sistema. Los roles se muestran en la tabla 23.

Tabla 23

Roles del sistema

Roles	Descripción	Permisos	Jerarquía
Owner	Es el responsable de cada API	– Tiene acceso al dashboard pero solo a visualizar la información de su API asignada. – Tiene acceso al módulo de costos, pero solo	Baja

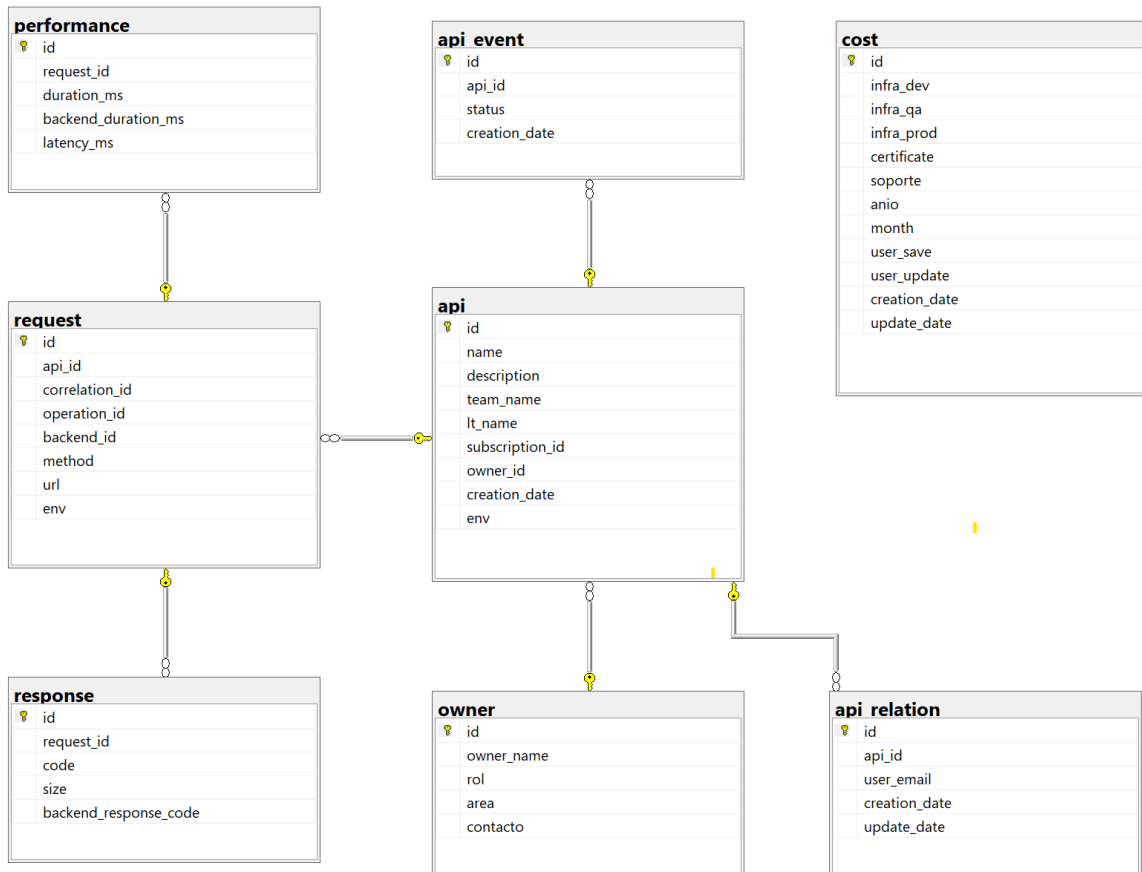
		puede visualizar la información de su API.	
Visor	Puede visualizar la información de todas las APIs pero no puede realizar mantenimientos dentro de la aplicación web.	– Tiene acceso al dashboard y puedo visualizar la información de todas las APIs. – Tiene acceso al módulo de inventario de APIs. – Puede visualizar los costos por APIs.	Media
Admin	Puede visualizar la información de todas las APIs y también puede realizar mantenimientos dentro de la aplicación web.	– Tiene acceso al dashboard y puedo visualizar la información de todas las APIs. – Tiene acceso al módulo de inventario de APIs. – Tiene acceso al módulo de mantenimiento de Owner por API. – Tiene acceso al módulo de mantenimiento de costos. – Tiene acceso al módulo de los costos por APIs.	Alta

Nota. Fuente: Elaboración propia

Se documentó el diagrama de base de datos, el cual se muestra en la figura 17. Los scripts utilizados para la creación de tablas se encuentran en el Anexo 1.

Figura 17

Diagrama de base de datos



Nota. Fuente: Elaboración propia.

En la figura 17 se muestran las siguientes tablas:

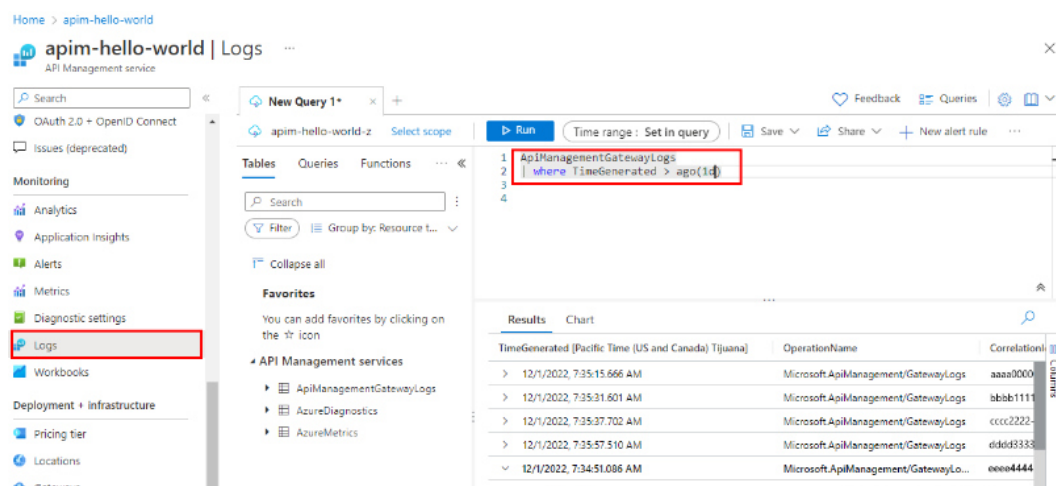
- La tabla api la cual contiene la información detallada de las APIS desplegadas en el API Managment.
- La tabla api_relation que permite relacionar al usuario autenticado con las APIs que puede visualizar.
- La tabla owner donde se almacena la información de los responsables de cada API.
- La tabla request donde se almacena la información de las peticiones realizadas a cada API registrada en la tabla api.

- La tabla cost donde se almacena la información de los costos mensuales de infraestructura.
- Las tablas response y performance que almacenan las métricas de las peticiones registradas en la tabla request.
- La tabla event almacena los estados asociados al proceso de registro de cada API.

Esta etapa también se investigó la integración con el componente de Log Analytics. Para obtener esta información generamos una consulta en KQL el cual filtra la información necesaria para nuestro requerimiento. En la figura 18 se muestra la ejecución de una consulta en Log Analytics de Microsoft Azure.

Figura 18

Ejecución de una consulta en KQL



Nota. Fuente: Microsoft Azure

En el presente trabajo se definen los siguientes 3 ambientes para realizar la implementación:

- Desarrollo: Ambiente para que los desarrolladores puedan realizar sus pruebas y se comienza a realizar las pruebas iniciales del QA.
- Certificación: Ambiente para que el QA pueda realizar sus pruebas simulando al ambiente de producción.
- Producción: Ambiente donde el sistema web se encuentra desplegado para el uso del usuario final.

Para desplegar nuestros cambios a producción debíamos realizar un proceso y cumplir con los estándares de la organización. Cuando teníamos una tarea o historia de usuario, debíamos crear la branch feature para realizar el desarrollo en esa rama. Luego se debe desplegar desde la rama al ambiente de desarrollo para hacer las pruebas iniciales en ese ambiente.

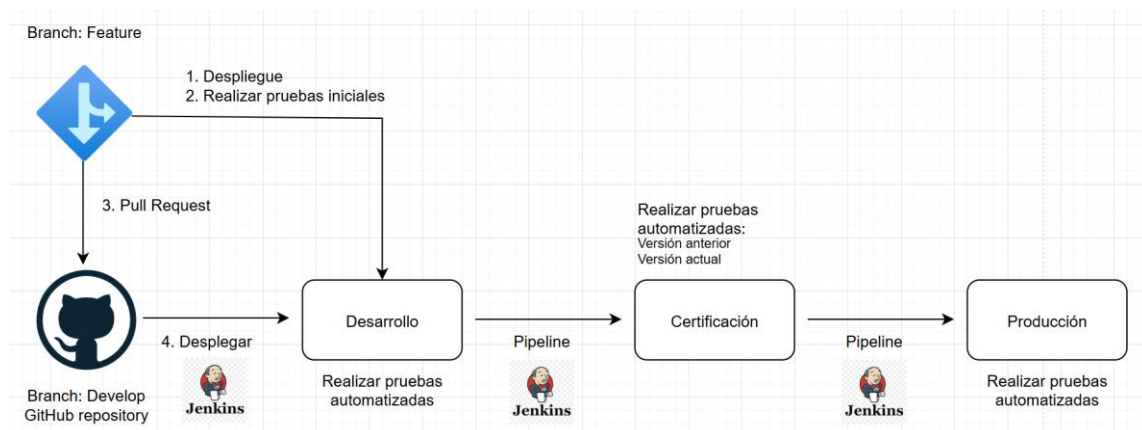
Cuando se terminen las pruebas iniciales, podemos realizar los pull request hacia la rama develop y nuevamente desplegar nuestros cambios a desarrollo. Luego se debe desplegar los cambios al ambiente de certificación mediante los pipelines de Jenkins y posteriormente dar soporte a la entrega continua para el despliegue a producción.

Cuando el desarrollo llega al ambiente de certificación, el QA del equipo debe realizar las pruebas automatizadas, rendimiento y seguridad. El QA puede devolver el issue hacia las etapas anteriores en caso lo encuentre necesario.

El proceso de entrega continua para desplegar los cambios hacia producción se muestra en la figura 19.

Figura 19

Proceso de entrega continua

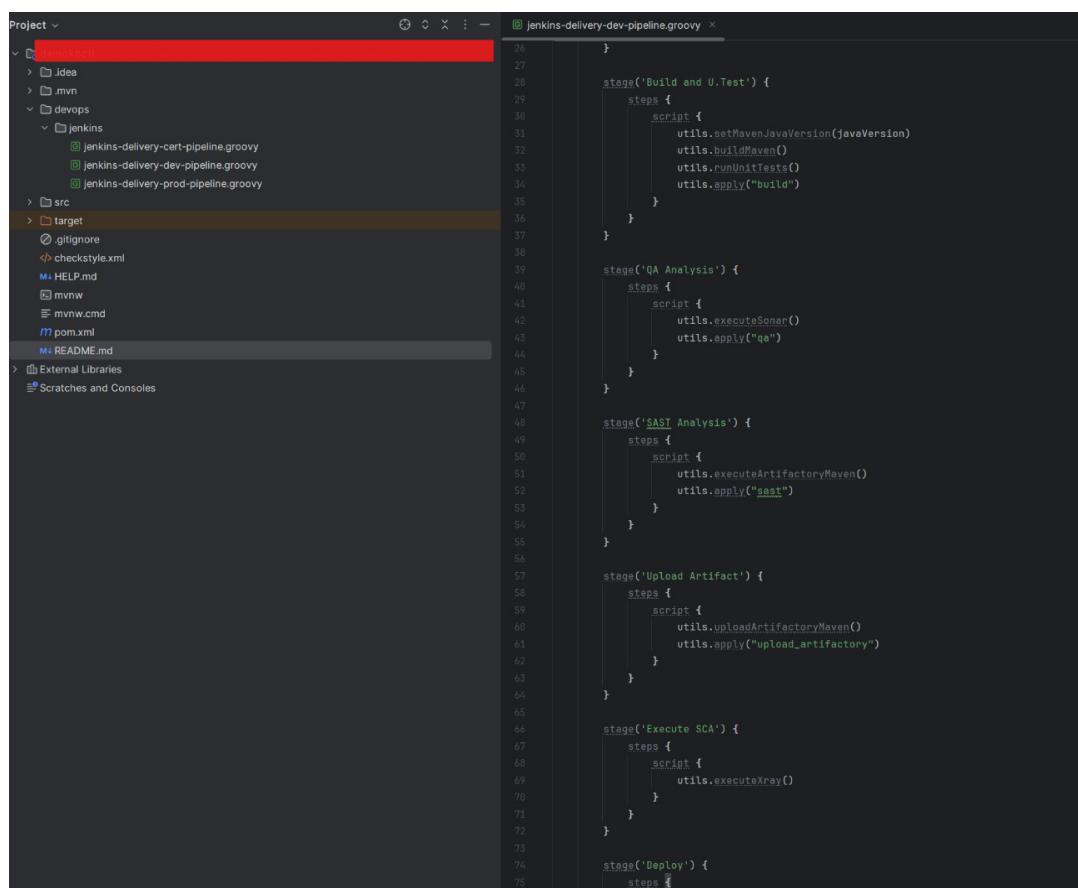


Nota. Fuente: Elaboración propia

Los pipelines ejecutan los scanners de Xray, Sonar y Fortify antes de desplegar el sistema en cada ambiente. Estos scanners evalúan las vulnerabilidades de seguridad y los estándares de calidad en el desarrollo realizado. Los archivos de configuración de los pipelines para desplegar a determinado ambiente se muestran en la figura 20. En esta figura se muestran las etapas que se ejecutan cuando se lanza el pipeline. Iniciando con la construcción del proyecto, luego continúa con los scanners y finalmente se despliega la nueva versión del proyecto.

Figura 20

Archivo de configuración del pipeline



Nota. Fuente: Elaboración propia

Al final la etapa de implementación se completaron todas las interfaces del sistema, pero las cuales no se pueden incluir en el presente trabajo debido a que contienen información confidencial de la entidad financiera. Sin embargo, en una sección posterior del presente trabajo se muestran los prototipos del sistema que permiten visualizar el diseño de las interfaces desarrolladas.

A.4 Soporte a producción

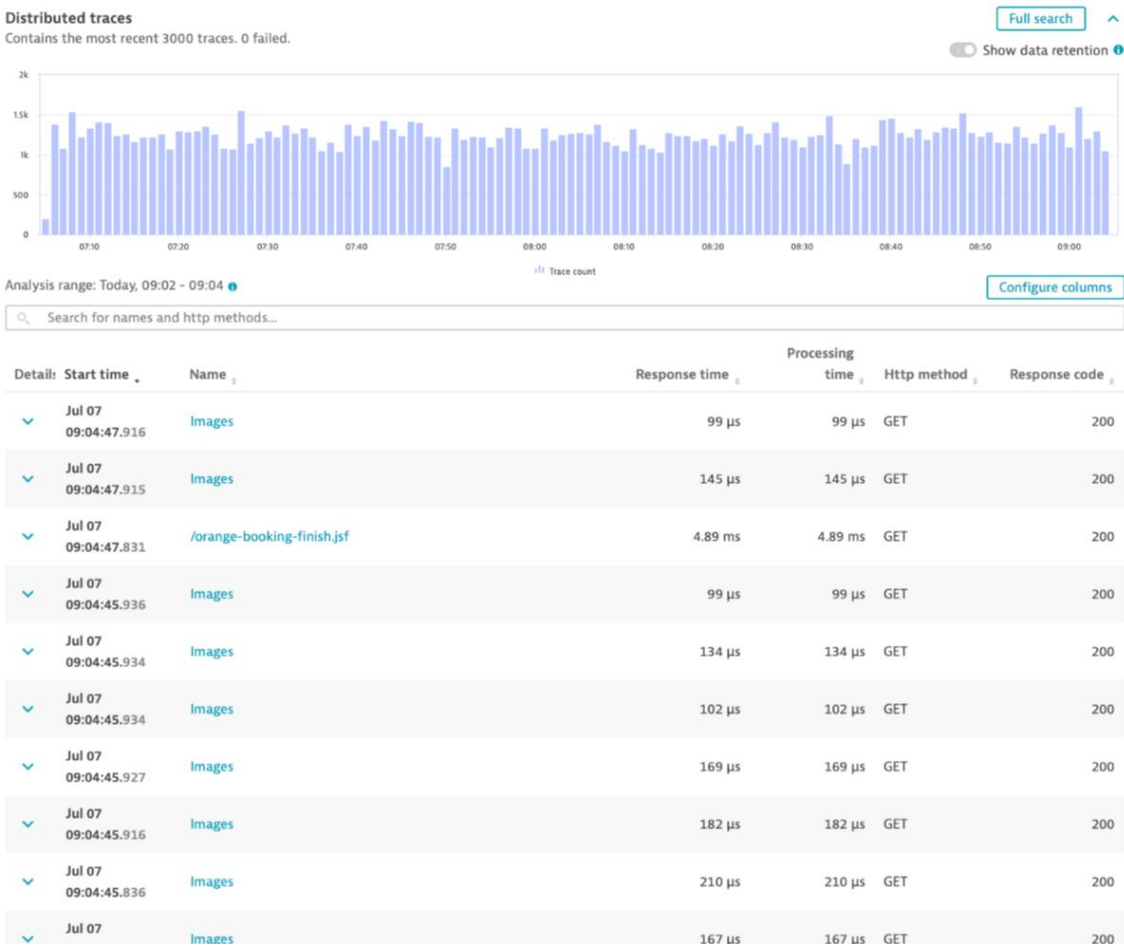
En esta etapa se da soporte a los incidentes que aparecen en producción. Para esto si en caso se requiera, nos reunimos con los involucrados que reportaron el incidente en producción para entender y replicar la acción realizada en este ambiente.

Posteriormente nosotros nos apoyábamos en las herramientas de Dynatrace y Kibana para analizar el incidente.

Dynatrace es una plataforma que permite a las organizaciones monitorear y analizar de cada aplicación. En el presente proyecto nos ayudó a visualizar los endpoints que fallaron de nuestro servicio. Cuando ubicamos el endpoint, también nos da la información necesaria para identificar en que parte del flujo falló. En la figura 21 se muestra la plataforma Dynatrace, donde se visualizan las peticiones realizadas a una API.

Figura 21

Plataforma de Dynatrace

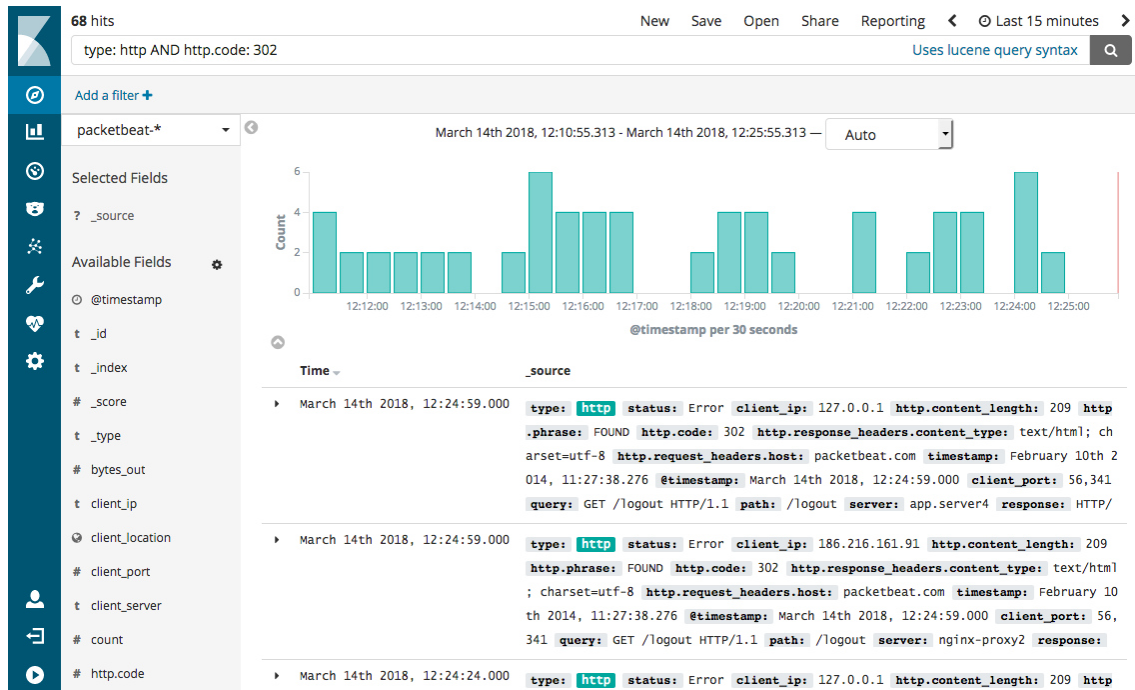


Nota. Fuente: Dynatrace

Para continuar la revisión de los incidentes, en caso necesite revisar los logs del servicio, utilizamos Kibana porque tiene una interfaz amigable y nos permite obtener los logs en base a filtros. En la figura 22 podemos visualizar la plataforma de kibana

Figura 22

Plataforma de Kibana



Nota. Fuente: Elastic

B. SPRINTS

Las 4 etapas mencionadas se dividieron en 12 sprints. Las actividades realizadas en los sprints se detallan en la tabla 24.

Tabla 24
Actividades realizadas en los sprints

Tipo de Issue	Título	Descripción	Sprint
Request	Analizar los requerimientos	Se debe analizar los requerimientos iniciales del proyecto.	1
Request	Coordinar con los equipos para la integración	Se debe coordinar con los equipos para la integración de componentes con la aplicación web.	1
Request	Realizar documentación inicial	Se debe realizar la documentación inicial del sistema	1
Request	Diseñar la arquitectura del sistema	Se debe diseñar la arquitectura inicial del sistema.	2
Request	Diseñar el diagrama de base de datos	Se debe diseñar el diagrama de base de datos de acuerdo con los requerimientos iniciales.	2
Spike	Investigar las tecnologías a usar	Se debe investigar las tecnologías que se utilizarán para la implementación del proyecto.	2
Request	Configurar los componentes cloud	Se debe configurar los componentes para el despliegue del desarrollo.	3
Spike	Investigar el framework backend	Se debe investigar el framework backend utilizado dentro de la entidad financiera.	3
Spike	Investigar el framework frontend	Se debe investigar el framework frontend utilizado dentro de la entidad financiera.	3
Spike	Investigar la integración con el Log Analytics	Se debe investigar cómo se debe integrar con el Log Analytics.	4
Request	Configurar la integración con Microsoft Entra ID	Se debe configurar la autenticación del sistema en base a Microsoft Entra ID.	4
Request	Realizar la configuración inicial del servicio backend	Se debe realizar la configuración inicial del servicio backend.	4
Request	Realizar la configuración inicial del servicio frontend	Se debe realizar la configuración inicial del servicio frontend.	5
Request	Crear los scripts de la base de datos	Se debe crear los scripts para la creación de tablas en base al diagrama de base de datos.	5
Request	Crear script KQL para obtener data del Log Analytics	Se debe crear un Script KQL para obtener información de las APIs desde Log Analytics.	5
Request	Iniciar sesión	Se debe agregar el módulo de iniciar sesión.	6
Request	Realizar la configuración para obtener la data desde Log Analytics	Se debe integrar el Log Analytics con el servicio backend.	6

Request	Agregar la configuración para obtener data de la base de datos	Se debe integrar y obtener data desde la base de datos.	6
Request	Desarrollar la interfaz del dashboard	Se debe mostrar los gráficos de barras del dashboard en el servicio frontend.	7
Request	Realizar pruebas unitarias del servicio backend	Se debe realizar las pruebas unitarias a la lógica agregada al servicio backend para obtener data desde el Log Analytics y base de datos.	7
Request	Agregar pruebas unitarias en el servicio frontend	Se debe agregar las pruebas unitarias del módulo del dashboard.	7
Request	Integrar el servicio backend con el frontend	Se debe obtener la data del Log Analytics en el frontend desde el servicio backend.	8
Request	Realizar pruebas de integración del servicio	Se debe realizar pruebas de integración para validar el correcto funcionamiento del dashboard.	8
HU	Implementar el dashboard para la visualización de métricas	Se debe implementar el dashboard para mostrar las métricas de las APIs públicas.	9
Spike	Investigar como obtener información de las APIs desde el core bancario	Se debe mostrar la información detallada de cada API pública del sistema Open Banking.	9
Request	Integrar con el APIM Managment del core bancario	Se debe obtener información del API Managment para obtener la información detallada del core bancario.	9
Request	Desarrollar el módulo de inventario de APIs	Se debe mostrar la información de las APIs en el servicio frontend.	10
Request	Obtener información del API Managment del core bancario	Se debe listar desde el servicio backend la información de las APIs del Core Bancario.	10
HU	Implementar módulo de inventario de APIs	Se debe implementar el módulo de inventario de APIs para mostrar la información detallada de cada API.	10
Request	Desarrollar el mantenimiento de costos	Se debe desarrollar el módulo para listar, actualizar, eliminar y crear los costos mensuales de las APIs.	11
Request	Desarrollar la interfaz del mantenimiento de costos	Se debe desarrollar la interfaz del mantenimiento de costos en el servicio frontend.	11
HU	Implementar mantenimiento de costos	Se debe implementar el módulo para el mantenimiento de costos.	11
Request	Desarrollar la interfaz para mostrar los costos por API	Desarrollar la interfaz para mostrar una tabla los costos por API mensualmente.	11
Request	Listar la información de los costos calculados por API	Desarrollar el listado de los costos por API en el servicio backend.	12
Request	Listar la información de los costos por API en Excel	Desarrollar la lógica para construir un Excel en el servicio backend.	12
HU	Implementar listado de información de costos por API	Se debe implementar módulo para mostrar los costos mensuales por API y exportar Excel de los costos.	12
Request	Elaborar manual de usuario	Se debe documentar las funciones finales del sistema para nuevos usuarios.	12

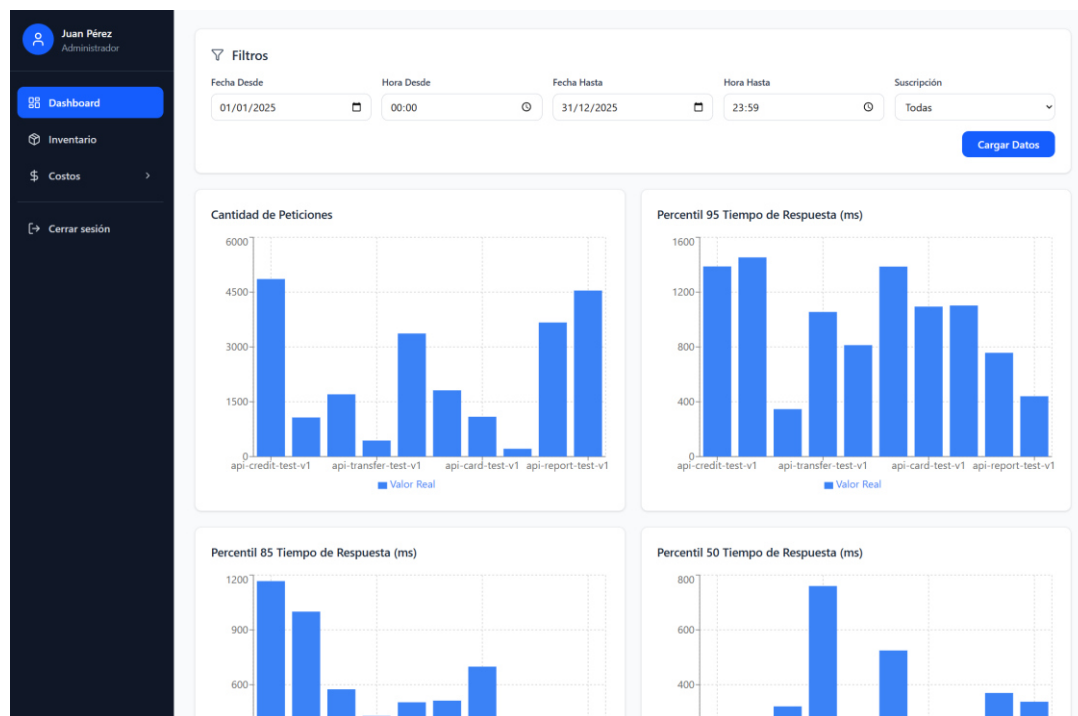
Nota. Fuente: Elaboración propia

C. PROTOTIPOS

En la figura 23 se muestra el prototipo del módulo Dashboard, donde se muestran filtros de fechas para realizar la búsqueda en determinado periodo. Este filtro solo puede realizar la búsqueda hasta 3 meses antes de la fecha. Adicionalmente se muestra el filtro de la subscripción, una subscripción representa a un consumidor de la API y permite visualizar las métricas de un consumidor. Todos los roles del sistema tienen acceso a este módulo. Pero el rol Owner solo puede ver las métricas de sus APIs, mientras que los roles del sistema Visor y Admin pueden ver las métricas de todas las APIs. El rol Owner son para los responsables de las APIs que pueden tener un perfil técnico o no, por lo cual se muestra un gráfico fácil de entender para los usuarios de sistema. En este módulo se muestra la cantidad de peticiones de las APIs para identificar que APIs tienen mayor demanda y que APIs no están siendo usadas. Adicionalmente se muestran gráficos de barras para visualizar el tiempo de respuestas de las APIs, para identificar si tienen un correcto performance o presentan demoras. También se puede ver la cantidad de respuestas fallidas, permitiendo identificar las APIs caídas o con errores.

Figura 23

Prototipo del módulo Dashboard



Nota. Fuente: Elaboración propia

En la figura 24 se muestra el prototipo del módulo de inventario de APIs donde se muestra la información de todas las APIs desplegadas que recibieron peticiones en el mes seleccionado. Los roles del sistema que tienen permiso para acceder a este módulo es Visor y Admin. Este módulo permite conocer y obtener más información de cada API y asignar responsabilidades. Este módulo tiene como objetivo saber con quién contactarse en caso se requiera una pronta atención ante un incidente.

Figura 24
Prototipo del módulo Inventarios de Apis

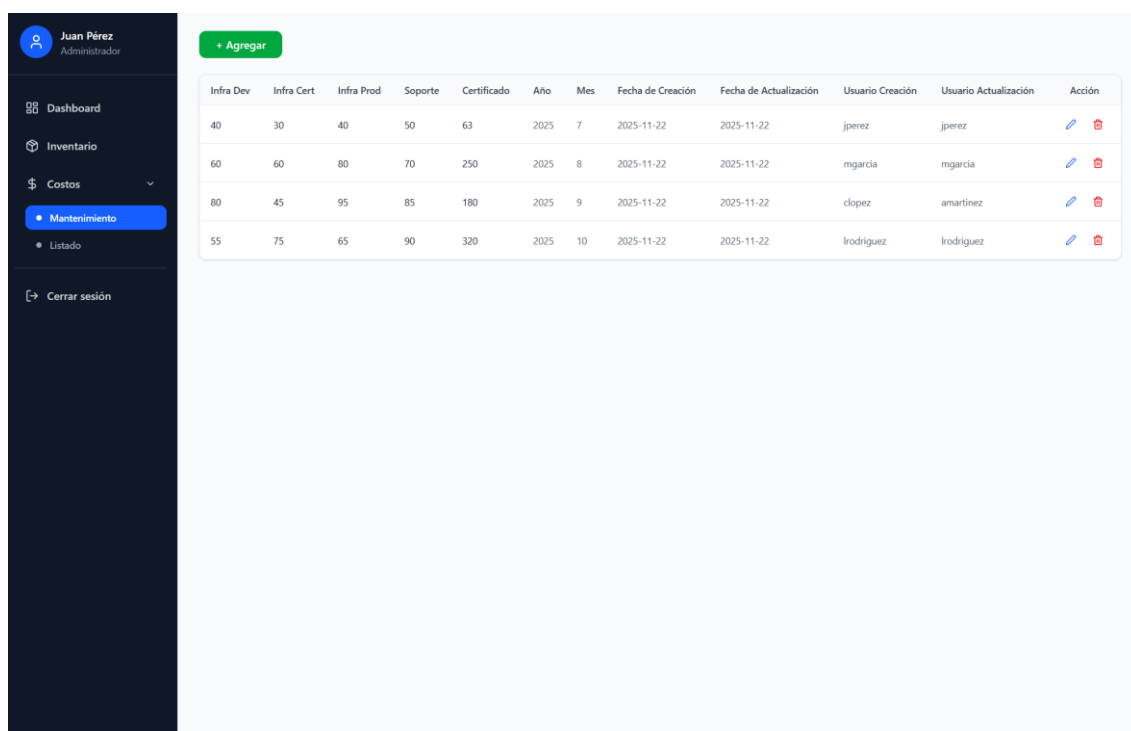
N°	Nombre del API	Identificador del API	Tipo de API	Código de Aplicación	Responsable	LT	Cantidad de Peticiones
1	api-credit-test-v1	API-CR-001	REST	APP-CR-2024	Juan Pérez	Roberto Vargas	1,427
2	api-debit-test-v1	API-DB-002	REST	APP-DB-2024	María García	Patricia Morales	4,744
3	api-account-test-v1	API-AC-003	GraphQL	APP-AC-2024	Carlos López	Fernando Castro	3,203
4	api-transfer-test-v1	API-TR-004	REST	APP-TR-2024	Ana Martínez	Gabriela Ruiz	373
5	api-payment-test-v1	API-PY-005	REST	APP-PY-2024	Luis Rodríguez	Miguel Ángel Silva	3,247
6	api-loan-test-v1	API-LN-006	SOAP	APP-LN-2024	Sofía Hernández	Andrea Navarro	3,796
7	api-card-test-v1	API-CD-007	REST	APP-CD-2024	Diego Torres	Ricardo Mendoza	4,205
8	api-user-test-v1	API-US-008	GraphQL	APP-US-2024	Laura Ramírez	Valeria Ortiz	4,972
9	api-balance-test-v1	API-BL-009	REST	APP-BL-2024	Pedro Sánchez	Javier Paredes	1,253
10	api-report-test-v1	API-RP-010	REST	APP-RP-2024	Carmen Flores	Daniela Reyes	3,456

Nota. Fuente: Elaboración propia

En la figura 25 se muestra el prototipo del Mantenimiento de costos de infraestructura dentro del módulo de Costos donde se muestra la información de los costos de infraestructura registrados por mes. Los costos se listan de manera ascendente de acuerdo con los campos año y mes. Al lado derecho se muestra las opciones para actualizar o remover los costos. En la parte superior se muestra el botón “Agregar” para registrar un nuevo costo de infraestructura para determinado mes. Esta información será útil para hacer el cálculo total del costo por API. El rol Admin es el único que tiene acceso a esta funcionalidad.

Figura 25

Prototipo del Mantenimiento de costos de infraestructura



Infra Dev	Infra Cert	Infra Prod	Soporte	Certificado	Año	Mes	Fecha de Creación	Fecha de Actualización	Usuario Creación	Usuario Actualización	Acción
40	30	40	50	63	2025	7	2025-11-22	2025-11-22	jperez	jperez	 
60	60	80	70	250	2025	8	2025-11-22	2025-11-22	mgarcia	mgarcia	 
80	45	95	85	180	2025	9	2025-11-22	2025-11-22	clopez	amartinez	 
55	75	65	90	320	2025	10	2025-11-22	2025-11-22	lrodriguez	lrodriguez	 

Nota. Fuente: Elaboración propia

En la figura 26 se muestra el prototipo del Listado de costos por API dentro del módulo de Costos. Esta información consiste en el costo total de infraestructura más el costo total por el consumo de las APIs en los ambientes de desarrollo, certificación y producción, dan como resultado el costo total por API del mes seleccionado en el filtro de la parte superior. El botón “Cargar datos” permite listar los costos una vez

seleccionado el filtro del periodo. A partir de este listado de costos se puede generar un reporte en Excel cuando se presiona el botón exportar. Todos los roles del sistema tienen acceso a esta funcionalidad, pero los Owners solo pueden visualizar de sus APIs. Este módulo permite a los responsables de cada API identificar cuanto es el costo de mantener su API desplegada y adicionalmente realizar los cobros a los consumidores en base al número de peticiones. Para dividir el costo entre los consumidores, el responsable de la API puede utilizar el filtro de subscripción en el dashboard y de esta forma realizar un cobro en proporción a la cantidad de peticiones por consumidor.

Figura 26
Prototipo del Listado de costos por API

N°	Nombre de la API	Cantidad de Peticiones	% de Uso Dev	% de Uso Cert	% de Uso Prod	Costo x Infra	Costo x Consumo Dev	Costo x Consumo Cert	Costo x Consumo Prod	Costo Total
1	api-credit-test-v1	1,506	33%	34%	60%	\$541.16	\$362.73	\$258.14	\$499.82	\$1,661.85
2	api-debit-test-v1	6,668	29%	26%	49%	\$2,476.57	\$768.21	\$605.06	\$1,029.78	\$4,879.62
3	api-account-test-v1	1,995	33%	37%	55%	\$973.65	\$213.03	\$421.53	\$1,089.27	\$2,697.48
4	api-transfer-test-v1	4,706	17%	36%	46%	\$2,318.8	\$748.4	\$234.81	\$600.69	\$3,902.7
5	api-payment-test-v1	4,753	19%	23%	30%	\$1,629.41	\$777.08	\$705.54	\$1,189.72	\$4,301.75
6	api-loan-test-v1	1,528	16%	33%	37%	\$722.37	\$542.1	\$737.52	\$1,008.12	\$3,010.11
7	api-card-test-v1	6,143	16%	22%	51%	\$2,416.76	\$317.78	\$737.75	\$700.79	\$4,173.08
8	api-user-test-v1	2,220	27%	22%	55%	\$2,160.92	\$548.45	\$220.47	\$1,186.04	\$4,115.88
9	api-balance-test-v1	4,880	33%	40%	33%	\$1,457.8	\$328.54	\$679.46	\$302.12	\$2,767.92
10	api-report-test-v1	7,531	21%	27%	52%	\$678.81	\$636.71	\$536.99	\$382.86	\$2,235.37

Nota. Fuente: Elaboración propia

Dado que el código fuente de la aplicación web es información confidencial de la entidad financiera, no se puede compartir en este informe. Sin embargo, se muestran los procesos de la implementación y prototipos de la aplicación, a fin de evidenciar el diseño, implementación y funcionamiento del sistema.

3.3. EVALUACIÓN

3.3.1. EVALUACIÓN ECONÓMICA / EVALUACIÓN COSTO-BENEFICIO

En esta sección evaluaremos los costos del proyecto en relación con los sueldos y tiempos de los miembros del equipo. El Costo fue estimado considerando que la jornada laboral en promedio son 160 horas mensuales, pero hay una asignación parcial de los miembros del equipo debido que participan simultáneamente en otros proyectos del equipo. Estos costos se muestran en la tabla 25.

Tabla 25

Costos del proyecto

Puesto	Sueldo mensual (S/)	Costo x hora (S/)	Horas dedicadas por mes	Meses	Costo total x Mes (S/)
Product Owner	8,000	50.00	40	6	12,000
Scrum Máster	6,000	37.50	20	4	3,000
Líder Técnico	6,000	37.50	80	6	18,000
Desarrollador Frontend Senior	6,000	37.50	120	4	18,000
Desarrollador Backend Senior	6,000	37.50	120	4	18,000
Desarrollador Backend Semi Senior	4,000	25	80	4	8,000
QA	5,000	31.25	80	4	10,000
Analista de seguridad	7,000	43.75	20	4	3,500

Cloud Engineer	7000	43.75	20	5	4,375
Costo total					94,875

Nota. Fuente: Elaboración propia

Dentro de los costos que vamos a calcular, no se va a considerar los costos de los equipos debido a que el proveedor nos lo proporciona. Adicionalmente los costos de certificados y de componentes cloud no se consideran debido a que ya se contaban dentro de la empresa y no están siendo proporcionados para el presente trabajo.

El objetivo del proyecto es reducir los tiempos que los responsables de cada API dedican para el análisis de las métricas. Adicionalmente el tiempo ahorrado para realizar los cálculos de los costos y generar los reportes.

En base a estos objetivos, los beneficios son los siguientes:

- Ahorro del tiempo de un Cloud Engineer en brindar el acceso a los responsables de las APIs para revisar las métricas de sus APIs. Este proceso ya no es permitido por la organización por temas de seguridad, dando como solución el dashboard del presente proyecto. El tiempo estimado del Cloud Engineer en brindar el acceso son de 4 horas por responsable de cada API, el ahorro inicial son 4,900 soles.
- Reduce el tiempo de análisis de métricas por parte de los responsables de las APIs y mejora la toma de decisiones. Considerando que cada responsable destina aproximadamente un día u 8 horas laborales al mes para realizar el análisis, pero con el presente proyecto reduce a 4 horas laborales al mes. Si existen 28 responsables de APIs en Open Banking, el ahorro anual sería de 67,200 soles.
- El tiempo del Product Owner y el Cloud Engineer para generar los reportes es de 3 días en un mes. Por lo cual tomando como base sus sueldos mensuales, el ahorro anual de ambos sería 27,000 soles.

3.3.2. INTERPRETACIÓN DEL VAR Y DEL TIR

Según lo costos mencionados con respecto a los costos por miembros de equipo, tenemos un ahorro inicial de 4,900 soles, un ahorro anual de 94,200 soles y un costo final del proyecto es 94,875 soles. El ahorro inicial se agregará al primer año del proyecto y no es un flujo anual, adicionalmente que se tendrá flujos de 94,200 soles anuales durante 4 años a una tasa de 10 % anual, se detalla los cálculos del VAR y TIR en la tabla 26.

Tabla 26

Cálculos de VAN

Año	Flujo en soles	Flujo descontado en soles
0	-94,875	-94,875
1	99,100	90,091
2	94,200	77,851
Total		73,067 (VAN)

Nota. Fuente: Elaboración propia

Se obtuvo un VAN de 73,067 soles con una tasa de descuento de 10%. Esto indica que el proyecto es rentable.

Adicionalmente el TIR es de 80% que es mayor a la tasa de descuento del 10% por lo cual confirma que el proyecto es confiable y viable.

3.3.3. BENEFICIOS DEL PROYECTO

Adicionalmente a los beneficios económicos, también se obtuvieron beneficios dentro del proceso de Open Banking como lo son:

- Reducir el tiempo de análisis de las métricas por parte de los responsables de las APIs. Debido que lo podrán visualizar mejor en el dashboard de la aplicación web.
- Reducir el tiempo de búsqueda de información de la API, debido a que nosotros mostramos la información detallada por APIs que viene desde el core bancario.
- Llevar un mejor control de los costos de infraestructura. Debido que antes solo lo manejan en Excel.
- Elaborar reportes de costos más rápidos debido que el mismo sistema genera el reporte.
- Reducir el error humano en el proceso de cálculo de costos por API debido que ahora es realizado por el sistema.

CAPÍTULO IV - REFLEXIÓN CRÍTICA DE LA EXPERIENCIA

El autor del presente proyecto ocupó el puesto de Desarrollador Backend Senior, su participación fue desde el inicio de la implementación del proyecto. En el cual enfrentó distintos retos debido a los altos estándares de seguridad y calidad que conlleva un sistema dentro de una empresa financiera. El autor se preocupó por desarrollar con buenas prácticas y al tener un cargo senior, también fue responsable en revisar el código de los demás desarrolladores. Se preocupó por tener un código limpio, ordenado y evitar el code smell.

Durante la implementación del proyecto, se obtuvo los siguientes conocimientos:

- El proceso de entrega continua y las herramientas utilizadas como Jenkins, SonarQube, Fortify, Artifactory y Xray.
- La Configuración de los recursos cloud utilizados.
- La programación reactiva en el desarrollo del servicio backend.

El gran reto dentro del proyecto fue el proceso de despliegue a producción, porque la entidad financiera tiene subprocesos que hacen demorar los pases a producción. Sin embargo, son subprocesos importantes como:

- La revisión de los estándares de calidad y seguridad
- La revisión del equipo de QA
- Cumplir con el formato estándar de la historia de usuario.
- Desplegar y redespargar nuestro desarrollo en el ambiente de certificación.
- Ejecutar scanner de seguridad y calidad.
- Revisión por parte del equipo de Devops para verificar que el pase cumple con todos los requisitos necesarios.
- La configuración de componentes cloud requiere de pases a producción que necesitan detallar las instrucciones de manera cuidadosa.
- Requiere de varias aprobaciones para completar la reserva de pase a producción.

El presente trabajo me hizo reflexionar en la gran importancia de Open Banking en las entidades financieras. Compartir la información con otras empresas abre puertas a la innovación mediante nuevos productos y servicios donde el más beneficiado es el mismo cliente. Otro punto importante en Open Banking es que brinda un canal seguro para compartirla información con otras empresas.

CAPÍTULO V - CONCLUSIONES Y RECOMENDACIONES

En este capítulo se detallará las conclusiones y recomendaciones que obtuvimos al desarrollar la aplicación web para visualizar las métricas de las APIs públicas desplegadas en el API Managment de Open Banking. Podemos concluir que se alcanzaron los objetivos del proyecto.

5.1. CONCLUSIONES

- Se implementó una aplicación web para la visualización del consumo de APIs públicas que permitió optimizar el tiempo de análisis de las métricas y mejoró la gestión de costos en la entidad financiera.
- Se definió las métricas para el análisis del consumo de las APIs públicas de la entidad financiera.
- Se diseño e implementó una base de datos que permitió almacenar los costos de infraestructura y el consumo de las APIs públicas de la entidad financiera.
- Se implementó una arquitectura que integró la aplicación web con servicios cloud.
- Se automatizó el proceso del cálculo de los costos de las APIs públicas de la entidad financiera.
- Se generó reportes que permitieron optimizar la toma de decisiones de los responsables de las APIs públicas de la entidad financiera.

5.2. RECOMENDACIONES

- La información respaldada por el Log Analytics tiene una duración de 3 meses. Por lo cual limita en el rango de búsqueda en la aplicación web. Como recomendación se debería guardar la información en la base de datos de forma periódica para tener una mayor duración en la búsqueda de información de cada API en el dashboard.
- Se podría agregar una funcionalidad para enviar alertas o notificaciones a los responsables de las APIs y de esta forma están informado cuando hay un incidente o informar alguna anomalía.

5.3. FUENTES DE INFORMACIÓN

Amazon Web Services. (s.f.) ¿Qué es la entrega continua?
<https://aws.amazon.com/es/devops/continuous-delivery/>

Akyildirim, E. (2025) Global perspectives on open banking: Regulatory impacts and market response <https://doi.org/10.1016/j.intfin.2025.102159>

Babina, T. (2025). Customer data access and fintech entry: Early evidence from open banking. *Journal of Financial Economics*. <https://doi.org/10.1016/j.jfineco.2024.103950>

Dynatrace (s.f.) Service analysis <https://docs.dynatrace.com/docs/observe/applications-and-microservices/services/analysis/service-analysis-new>

Elastic (s.f.) Kibana queries and filters
<https://www.elastic.co/docs/reference/beats/packetbeat/kibana-queries-filters>

Eros, B. (2021) Contract-First Development Using RestAssured and OpenAPI
<https://hazelcast.com/blog/contract-first-development-using-restassured-and-openapi>

Ferilli, G. B. (2024). Fintech governance and performance: Implications for banking and financial stability. *Research in International Business and Finance*.
<https://doi.org/10.1016/j.ribaf.2024.102349>

Malhotra, N. (2024) Reactive programming
<https://www.valuelabs.com/resources/whitepapers/reactive-programming>

Microsoft Azure (s.f.). Conceptos básicos de Azure Key Vault.
<https://learn.microsoft.com/es-es/azure/key-vault/general/basic-concepts>

Microsoft Azure (s.f.). Monitor API Management <https://learn.microsoft.com/en-us/azure/api-management/monitor-api-management>

Microsoft Azure (s.f.). Protocolo SAML de inicio de sesión único
<https://learn.microsoft.com/es-es/entra/identity-platform/single-sign-on-saml-protocol>

Microsoft Azure (s.f.). ¿Qué es Azure SQL Database? <https://learn.microsoft.com/es-es/azure/azure-sql/database/sql-database-paas-overview?view=azuresql>

Microsoft Azure (s.f.). ¿Qué es la Azure API Management? <https://learn.microsoft.com/es-es/azure/api-management/api-management-key-concepts>

Microsoft Azure (s.f.). ¿Qué es Microsoft Entra? <https://learn.microsoft.com/es-es/entra/fundamentals/what-is-entra>

Microsoft Azure (s.f.). What is cloud computing? <https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-cloud-computing>

Slingerland, C. (2025) 21+ Top Cloud Service Providers Globally In 2025 <https://www.cloudzero.com/blog/cloud-service-providers>

Somani, S. (2022) Observer Design Pattern <https://www.scaler.com/topics/design-patterns/observer-design-pattern>

Susnjara, S. (2024) ¿Qué es la implementación continua? <https://www.ibm.com/es-es/think/topics/continuous-deployment>

Zhmai, A. (2022) Stages of building and implementing the scrum methodology. *Economics and Society*. <https://doi.org/10.32782/2524-0072/2022-42-86>

5.4. GLOSARIO

Acoplamiento:

Es el nivel de relación entre componentes y por lo cual en un sistema se busca tener bajo acoplamiento para facilitar el mantenimiento, agilizar el desarrollo y aumentar la reutilización de código.

Angular:

Es el framework para el desarrollo frontend basado en el lenguaje de programación typescript y permite la creación de aplicaciones web.

API:

Es una interfaz de programación de aplicaciones para la comunicación en sistemas mediante estándares y protocolos.

API Management:

Recurso de la plataforma cloud Azure que permite publicar y gestionar las APIs de una organización.

API REST:

Es una API que usa protocolos HTTP para el intercambio de información mediante mensajes JSON.

Autoridad de Competencia y Mercados (CMA):

Es el organismo regulador del Reino Unido que plantea las bases de la competencia y regulación de mercados, incluido el sector financiero.

Azure:

Es la plataforma cloud que brinda distintos recursos en la nube para la gestión de aplicaciones y sistemas.

Azure Monitor:

Servicio de Azure para la recopilación y almacenamiento de métricas para realizar el análisis de recursos.

Azure SQL Database:

Servicio que funciona como una base de datos relacional dentro de la plataforma Azure.

Balance de carga:

Es la tecnología que permite distribuir la cantidad de peticiones a cada servicio. Esto evita la sobrecarga de un servicio.

Branch:

Es una rama que contiene una versión de código independiente para desarrollar cambios de una forma aislada.

Calidad de código:

Es el nivel que el código tiene para medir su legibilidad, eficiencia y seguridad. Se debe aplicar buenas prácticas de desarrollo para tener un nivel alto de calidad de código.

Claves criptográficas:

Son claves de seguridad con algún algoritmo de cifrado para proteger información sensible.

Cloud Engineer:

Es el encargado de gestionar y configurar todos los recursos cloud que se necesite en la implementación de la solución.

Code Review:

Es el proceso de revisión de código con la finalidad de encontrar errores o mejorar la calidad de código. Es proceso es necesario en caso se necesite la aprobación de un nuevo cambio de código.

Code smell:

Son malas prácticas de desarrollo que puede generar problemas en el mantenimiento del sistema.

Consentimiento:

Es la autorización que brinda un usuario para el uso de sus datos en un sistema informático.

Control de versiones:

Es el sistema que permite gestionar todos los cambios realizados en el código fuente de un sistema.

Core bancario:

Son el conjunto de servicios centrales que gestionan las operaciones principales en una entidad financiera.

Dashboard:

Es un sistema que muestra métricas e indicadores en base a gráficos.

Daily Scrum:

Es la reunión diaria para que los miembros del equipo den el estatus de sus actividades en un marco ágil Scrum.

Despliegue continuo:

Es el proceso automatizado para desplegar los cambios de un sistema a producción sin intervención humana.

Dynatrace:

Es la herramienta para realizar seguimiento y monitoreo de una aplicación. Permite visualizar las trazas de cada solicitud de un API.

Endpoints:

Es una dirección URL para realizar solicitudes a una API.

Entrega continua:

Es el proceso automatizado para que los cambios pasen la pruebas en cada ambiente. Pero para desplegar en producción se requiere intervención humana.

Escalamiento:

Es la característica de un sistema que indica que puede crecer a nivel de carga de trabajo de forma eficiente sin perder rendimiento.

Fintechs:

Empresas que brindan servicios financieros y usan tecnologías para innovar procesos.

Fortify:

Es la herramienta para análisis de seguridad de código y nos da el detalle de cada observación encontrada.

GitHub:

Es un repositorio para gestionar el código fuente de una aplicación y facilita el control de versiones.

Gobernanza:

Conjunto de políticas que regulan la gestión de sistemas informáticos en una organización.

Inteligencia Artificial (IA):

Es el campo de la informática para el desarrollo de sistemas que simulan el comportamiento humano para realizar tareas.

Interoperabilidad:

Es la capacidad de comunicación entre sistemas que permite lograr objetivos comunes.

Issues:

Es el registro de un trabajo en Jira para realizar seguimiento dentro de un proyecto. Representa la actividad que está realizando un miembro del equipo.

Jenkins:

Plataforma para automatizar el despliegue de los cambios a producción.

Kafka:

Es una tecnología para la transmisión de eventos mediante un canal de comunicación para procesar datos en tiempo real.

Kibana:

Es una herramienta para la visualización y análisis de datos principalmente logs.

Kusto Query Language (KQL):

Es el lenguaje para realizar consultas en los recursos de Azure.

Microservicio:

Es la arquitectura de un sistema para dividir la lógica en distintos servicios independientes.

OpenAPI:

Es un documento estándar para describir los endpoints de una API.

Platform as a Service (PaaS):

Es un modelo de computación en la nube que brinda recursos a organizaciones sin administrar la infraestructura.

Pipelines

Conjunto de datos consecutivos para el despliegue de un sistema de manera continua y automatizada.

Product Backlog:

Es una lista priorizada de todas las actividades a realizar para la construcción de un sistema con la metodología Scrum.

Quality Analyst (QA):

Es la persona responsable de ejecutar pruebas del sistema para validar que cumplan los requerimientos definidos dentro de cada issue.

RabbitMQ:

Es una tecnología para enviar los mensajes de manera asíncrona entre aplicaciones por medio de colas y tópicos.

Sonar:

Es la herramienta que permite gestionar la calidad de código, detectar code smell y mostrar el porcentaje de la cobertura de las pruebas unitarias.

Sobrecarga:

Es un problema que ocurre cuando hay muchas peticiones hacia un servicio y este no tenga la capacidad para manejar esa cantidad de peticiones.

Sprint Planning:

Es la ceremonia de Scrum para definir las tareas del siguiente sprint.

Sprint Review:

Es la ceremonia de Scrum para revisar el estatus de las actividades comprometidas dentro del sprint.

Sprint Retrospective:

Es la ceremonia de Scrum para que el equipo comparta su retroalimentación de las actividades y proponer mejoras para los siguientes sprint.

Spring Cloud:

Conjunto de herramientas dentro del ecosistema spring para el desarrollo de una arquitectura basada en microservicios.

Spring WebFlux

Es un framework para desarrollar aplicaciones asíncronas con el lenguaje de programación Java.

Stakeholders:

Son las personas involucradas para desarrollo de un sistema.

Startups:

Son empresas nuevas que generalmente utilizan la tecnología para tener modelos de negocios innovadores.

Test unitarios:

Son pruebas escritas para validar pequeñas partes de código de forma individual.

Tópicos:

Son canales para enviar y recibir mensajes de manera asíncrona.

Web App:

Servicio de Azure para alojar y desplegar una aplicación. No necesita la administración de servidores.

Xray

Herramienta utilizada para validar las vulnerabilidades de las dependencias de un sistema.

ANEXOS

ANEXO 1: Script de creación de tablas de base de datos

```
CREATE DATABASE ApiManagementDBtest;

USE ApiManagementDB;

CREATE TABLE dbo.owner (
    id BIGINT PRIMARY KEY,
    owner_name VARCHAR(100) NOT NULL,
    rol VARCHAR(50) NOT NULL,
    area VARCHAR(100) NOT NULL,
    contacto VARCHAR(100) NOT NULL
);

CREATE TABLE dbo.api (
    id BIGINT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description NVARCHAR(MAX),
    team_name VARCHAR(100),
    it_name VARCHAR(100),
    subscription_id VARCHAR(100),
    owner_id BIGINT,
    creation_date DATETIME2,
    env VARCHAR(50),
    FOREIGN KEY (owner_id) REFERENCES dbo.owner(id)
);

CREATE TABLE dbo.request (
    id BIGINT PRIMARY KEY,
    api_id BIGINT NOT NULL,
    correlation_id VARCHAR(100) NOT NULL,
    operation_id VARCHAR(100) NOT NULL,
    backend_id VARCHAR(100) NOT NULL,
    method VARCHAR(20) NOT NULL,
    url VARCHAR(255) NOT NULL,
    env VARCHAR(50) NOT NULL,
    FOREIGN KEY (api_id) REFERENCES dbo.api(id)
);

CREATE TABLE dbo.response (
    id BIGINT PRIMARY KEY,
    request_id BIGINT NOT NULL,
    code VARCHAR(10) NOT NULL,
    size INT NOT NULL,
    backend_response_code VARCHAR(10) NOT NULL,
    FOREIGN KEY (request_id) REFERENCES dbo.request(id)
);
```

```

CREATE TABLE dbo.performance (
    id BIGINT PRIMARY KEY,
    request_id BIGINT NOT NULL,
    duration_ms INT NOT NULL,
    backend_duration_ms INT NOT NULL,
    latency_ms INT NOT NULL,
    FOREIGN KEY (request_id) REFERENCES dbo.request(id)
);

-- Tabla: api_event
CREATE TABLE dbo.api_event (
    id BIGINT PRIMARY KEY,
    api_id BIGINT NOT NULL,
    status VARCHAR(50),
    creation_date DATETIME2,
    FOREIGN KEY (api_id) REFERENCES dbo.api(id)
);

CREATE TABLE dbo.cost (
    id BIGINT PRIMARY KEY,
    infra_dev DECIMAL(10,2),
    infra_qa DECIMAL(10,2),
    infra_prod DECIMAL(10,2),
    certificate DECIMAL(10,2),
    soporte DECIMAL(10,2),
    anio INT,
    month INT,
    user_save VARCHAR(100),
    user_update VARCHAR(100),
    creation_date DATETIME2,
    update_date DATETIME2
);

CREATE TABLE dbo.api_relation (
    id BIGINT PRIMARY KEY,
    api_id BIGINT NOT NULL,
    user_email VARCHAR(100),
    creation_date DATETIME2,
    update_date DATETIME2,
    FOREIGN KEY (api_id) REFERENCES dbo.api(id)
);

```
