PROYECTO:

PLATAFORMA E-COMMERCE “SEEK SHOP”

CLIENTE:

SEEK

ARQUITECTO:

SETH KARIM LUIS MARTÍNEZ

MÉRIDA, YUCATÁN, MÉXICO A 15 DE ABRIL DE 2025

ÍNDICE

Resumen ejecutivo

Descripción general:

Seek Shop es una solución de comercio electrónico moderna, construida sobre una arquitectura de microservicios desplegada en Google Cloud Platform (GCP). El sistema está diseñado para ser altamente escalable, resiliente y seguro, permitiendo gestionar procesos críticos de una tienda en línea como autenticación de usuarios, catálogo de productos, pedidos, pagos, inventario y notificaciones, todo mediante una orquestación centralizada basada en flujos BPMN utilizando Camunda.

Objetivo del sistema:

Brindar una infraestructura robusta y flexible que permita la ejecución de operaciones clave del negocio de forma automatizada, observable y segura, integrando componentes en la nube y sistemas legados on-premise.

Usuarios objetivo:

- Clientes que interactúan mediante un portal web o aplicación móvil
- Administradores y operadores de negocio que monitorean procesos mediante dashboards
- Integradores técnicos que requieren servicios estandarizados con alta disponibilidad

Valor agregado de la solución:

- Separación clara de responsabilidades entre servicios.
- Orquestación de procesos de negocio completa mediante un Business Process Model and Notation (BPMN) como Camunda
- Gestión centralizada de configuración y secretos.
- Capacidad de integración con sistemas empresariales existentes (on-premise).
- Canal de comunicación confiable hacia usuarios móviles mediante Firebase.
- Pipelines de integración y despliegue continuo con infraestructura como código.
- Arquitectura observada y protegida en todos sus niveles.

Objetivos

La arquitectura de Seek Shop fue diseñada bajo principios de modularidad, escalabilidad, seguridad y resiliencia, con el objetivo de garantizar que el sistema sea mantenible, extensible y robusto frente a errores o crecimiento del negocio.

Objetivos principales:

1. Escalabilidad horizontal
    a. Permitir que cada microservicio pueda escalar de forma independiente, según la carga y necesidades del negocio.
    b. Uso de microservicios desacoplados.
    c. Despliegue en infraestructura cloud elástica (Cloud Run, GKE).
    d. Load Balancing automatizado para distribuir tráfico.
2. Resiliencia y tolerancia a fallos
    a. Garantizar que una falla puntual no impacte el funcionamiento del sistema completo.
    b. Implementación del patrón Circuit Breaker con Resilience4j.
    c. Retries automáticos y fallbacks.
    d. Flujos BPMN con manejo de errores y tareas compensatorias.
    e. Despliegue de servicios en redes privadas protegidas.
3. Seguridad
    a. Proteger datos, accesos y comunicaciones dentro y fuera del sistema.
    b. Autenticación centralizada con Google Identity Platform y JWT.
    c. Gestión de secretos con Secret Manager.
    d. Comunicación segura vía HTTPS y redes privadas (VPC).
    e. Control de salida a internet mediante Cloud NAT.
4. Orquestación explícita de procesos
    a. Modelar, automatizar y monitorear flujos de negocio con herramientas BPMN.
    b. Uso de Camunda como motor de procesos.
    c. Inicio y seguimiento de procesos desde un microservicio dedicado (servicio-compras).
    d. Representación visual y versionada de flujos de negocio.

5. Observabilidad
    a. Facilitar el monitoreo, diagnóstico y trazabilidad del sistema.
    b. Google Cloud Observability (Logging, Monitoring, Trace).
    c. Métricas y alertas por microservicio.
    d. Monitoreo de procesos BPMN y fallos de ejecución.
6. Automatización del ciclo de vida (CI/CD)
    a. Asegurar despliegues confiables, auditables y sin downtime.
    b. GitHub privado como repositorio central.
    c. Cloud Build para integración continua.
    d. Store Docker + despliegue automático en servicios cloud.
    e. Pipeline desacoplado por servicio, con ambiente dev/prod.
7. Compatibilidad con sistemas legados (híbrido cloud/on-premise)
    a. Integración segura con servicios existentes en infraestructura local.
    b. Servicio de facturación on-premise conectado vía VPN/API segura.
    c. Procesos BPMN diseñados para invocar servicios externos.

Arquitectura general

Descripción general

La solución está basada en una arquitectura de microservicios desplegada sobre GCP, donde cada componente cumple una responsabilidad de negocio bien definida y se comunica con otros mediante llamadas HTTP seguras o mediante procesos orquestados desde Camunda BPMN. El acceso externo se realiza a través de un API Gateway, y todas las configuraciones sensibles están centralizadas.

El sistema integra:

- Microservicios desacoplados por dominio
- Procesamiento orquestado y auditable con Camunda
- Componentes en red privada con control de salida a internet
- Servicios externos como Firebase y sistemas on-premise
- Un pipeline CI/CD completo basado en GitHub y Cloud Build

Diagrama de arquitectura

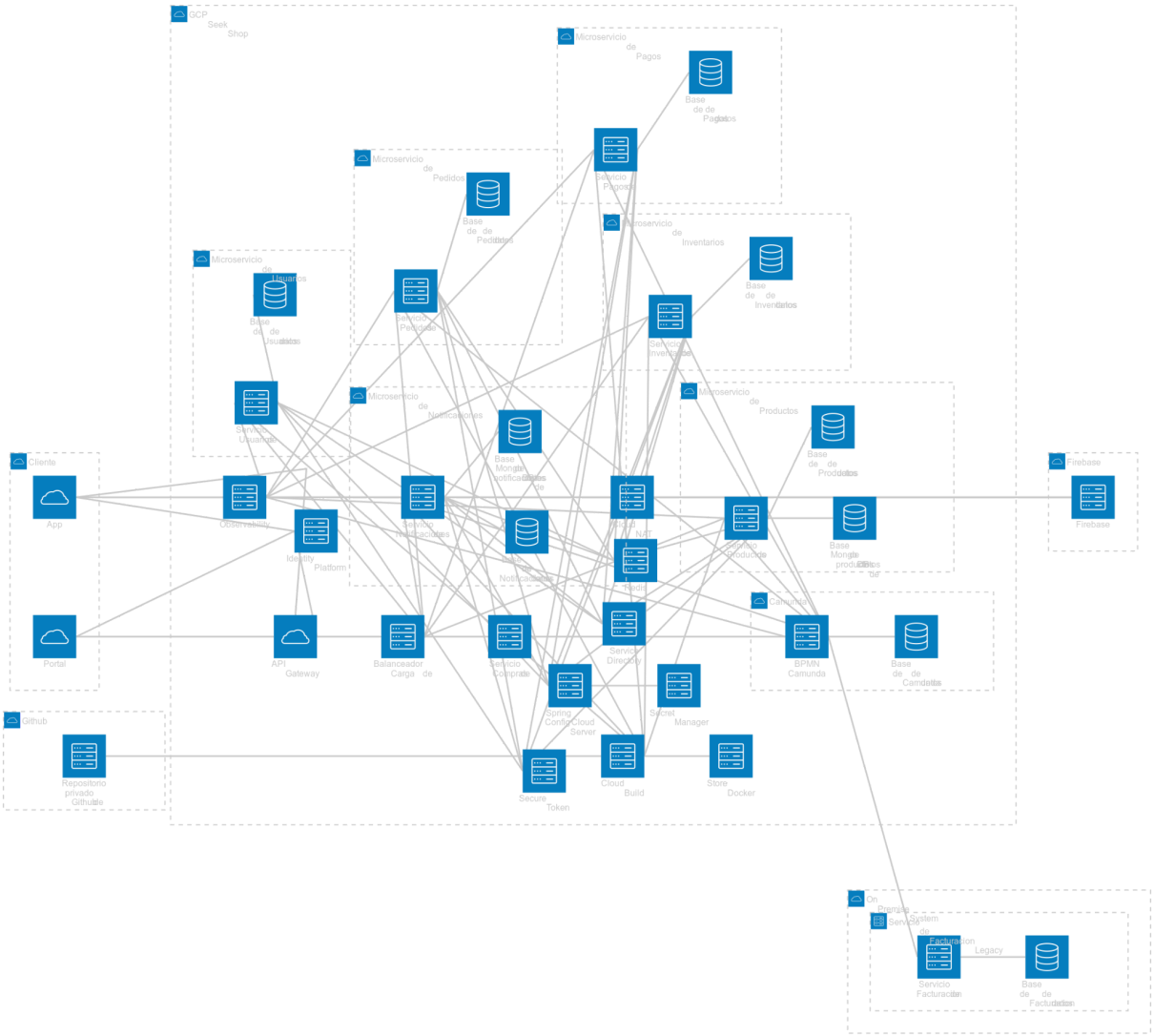

Diagrama 1 - Diagrama de arquitectura

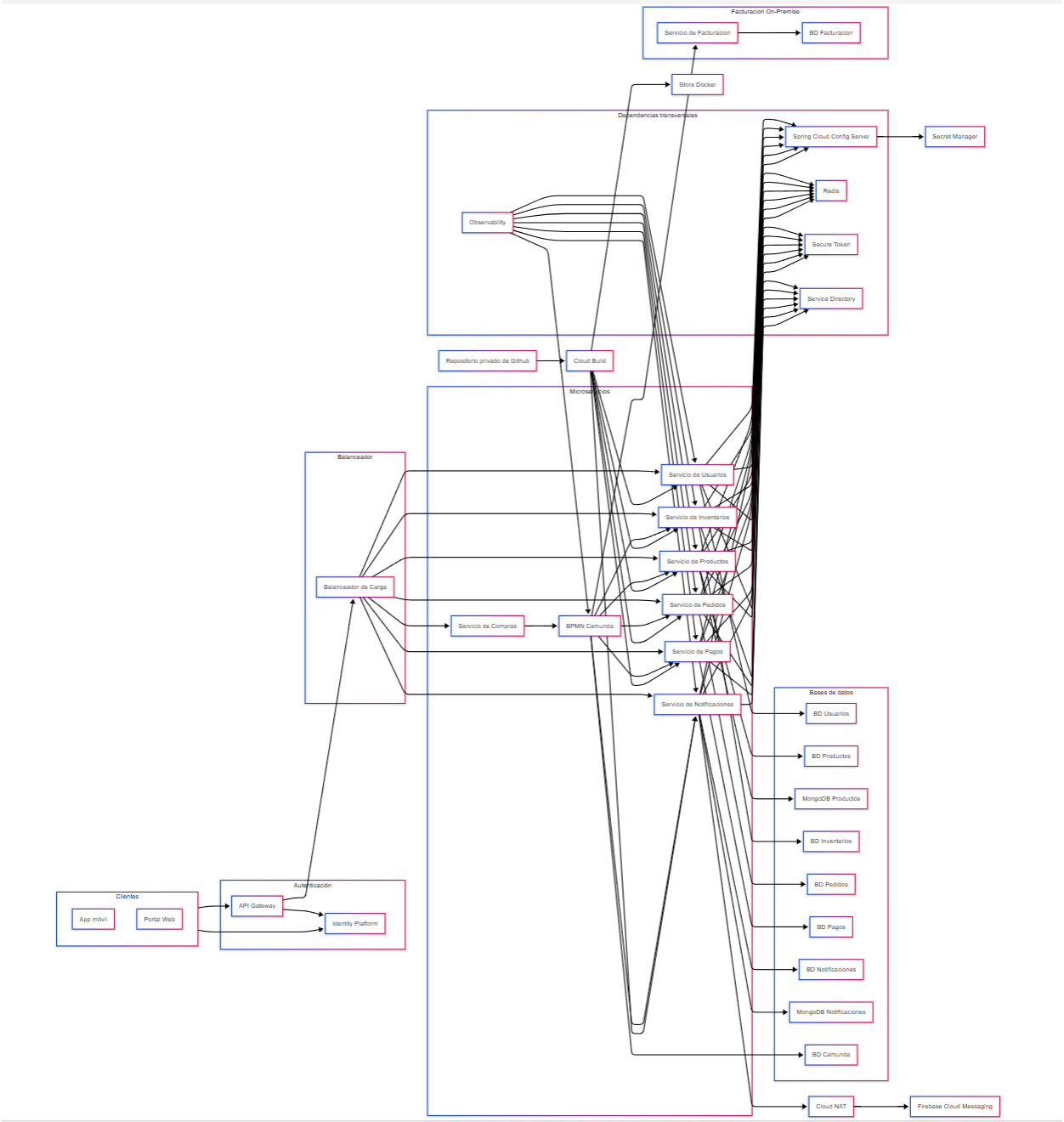

Diagrama 2 - Diagrama de arquitectura

Componentes principales

Github

Plataforma en la nube para hospedaje de código fuente y control de versiones basada en Git. Permite a los desarrolladores colaborar en tiempo real, realizar revisiones, integración continua (CI) y mantener un historial confiable de versiones. Se utiliza como repositorio central del código fuente de la solución, facilitando la automatización con Cloud Build para los despliegues en GCP.

Firebase

Plataforma de Google utilizada para funcionalidades en tiempo real como las notificaciones push en la aplicación móvil. Su integración con Firebase Cloud Messaging permite enviar mensajes a dispositivos Android e iOS de forma eficiente.

Sistema On Premise

Conjunto de sistemas legados del cliente, alojados en su infraestructura local. Estos sistemas siguen en operación y se integran parcialmente al ecosistema moderno mediante servicios como Camunda y la API de facturación.

Servicio de Facturación Legacy

Servicio alojado en infraestructura on-premise que incluye:

1. API de Facturación: expone funcionalidades de facturación del sistema legado.
2. Base de datos de Facturación: base de datos Oracle que almacena los registros históricos y operativos del sistema de facturación.

Seek Shop Portal Web

Aplicación web moderna construida en Angular que actúa como interfaz principal para los usuarios finales en plataformas de escritorio o navegadores móviles. Consume servicios mediante el API Gateway.

Seek Shop Aplicación Móvil

Aplicación móvil multiplataforma (Android/iOS) desarrollada en Flutter para ofrecer una experiencia de usuario adaptada a dispositivos móviles. Se comunica con la plataforma a través del API Gateway y recibe notificaciones mediante Firebase.

Google Cloud Platform

Infraestructura principal de la plataforma. Alojada en Google Cloud, combina servicios de red, seguridad, despliegue y monitoreo, facilitando escalabilidad y alta disponibilidad.

API Gateway

Punto de entrada único para clientes web y móviles. Permite definir rutas y aplicar autenticación mediante JWT, simplificando la gestión de seguridad y acceso a los servicios backend.

Identity Platform

Servicio de autenticación de usuarios gestionado. Permite implementar flujos seguros de login, emisión de tokens JWT y verificación de identidad de forma escalable.

Cloud Load Balancer

Distribuye el tráfico entrante de forma eficiente hacia los servicios backend, garantizando alta disponibilidad, balanceo geográfico y escalabilidad automática.

Cloud NAT

Proporciona conectividad saliente a Internet a los servicios internos sin exponer sus IPs públicas. En este caso, permite al microservicio de Notificaciones comunicarse con Firebase de manera segura.

Service Directory

Administra el descubrimiento y la referencia de servicios internos, facilitando su ubicación dentro de la VPC sin necesidad de configuraciones manuales o registros DNS externos.

Secret Manager

Permite almacenar y administrar secretos de forma segura. Se usa para guardar credenciales como claves API, contraseñas y tokens privados utilizados por los servicios.

Observability

Conjunto de herramientas (como Cloud Monitoring y Logging) para supervisar la salud, métricas y registros de la plataforma. Facilita el diagnóstico proactivo y la mejora del rendimiento de los microservicios y Camunda.

Cloud Build

Motor de CI/CD que compila, prueba y despliega automáticamente las aplicaciones alojadas en GitHub. Facilita integraciones y actualizaciones frecuentes sin intervención manual.

Artifact Registry

Repositorio central para imágenes de contenedores, bibliotecas y paquetes de dependencias. Se utiliza como almacén seguro para los artefactos generados por Cloud Build.

Seek Shop VPC

VPC (Virtual Private Cloud) que encapsula los microservicios desplegados en GCP, proporcionando un entorno seguro y privado. Contiene todos los servicios funcionales de la plataforma, separados por dominios.

Microservicio de Usuarios

Gestiona operaciones sobre usuarios como creación, autenticación y consulta de perfiles. Utiliza una base de datos PostgreSQL. Emplea Redis para la gestión de tokens activos.

Microservicio de Productos

Administra el catálogo de productos. Utiliza PostgreSQL para datos estructurados y MongoDB para información flexible o no estructurada.

Microservicio de Inventarios

Gestiona el stock disponible por producto. Usa PostgreSQL como base de datos relacional para garantizar integridad transaccional.

Microservicio de Pedidos

Orquesta la creación, seguimiento y actualización de órdenes de compra de los usuarios. Se apoya en PostgreSQL para trazabilidad y consistencia.

Microservicio de Pagos

Responsable del procesamiento y validación de pagos. Guarda los registros de pago en una base de datos PostgreSQL.

Microservicio de Notificaciones

Permite la creación y envío de notificaciones personalizadas a los usuarios. Utiliza PostgreSQL para control de envío y MongoDB para almacenamiento de eventos flexibles.

Microservicio de Compras

Servicio que proporciona una API para validar el payload de un proceso de compra de un producto. Inicia la orquestación del flujo de compra de un producto en Camunda.

BPMN

Modelo y Notación de Procesos de Negocio (BPMN). Lenguaje estándar de modelado de procesos de negocio. Utilizado para representar gráficamente flujos de decisión y acciones automatizadas.

Camunda

Motor de orquestación que ejecuta procesos BPMN definidos por el negocio. Desplegado como servicio Spring Boot. Se apoya en PostgreSQL para almacenar historial de procesos, estado de

ejecución y métricas. Orquesta microservicios como Productos, Inventarios, Pedidos y Notificaciones.

Redis

Almacenamiento en memoria usado para mejorar el rendimiento. En esta plataforma se emplea para guardar tokens de sesión activos, permitiendo invalidarlos antes de su expiración.

Spring Cloud Config Server

Gestor centralizado de configuraciones de microservicios. Permite tener consistencia en valores como URLs, timeouts, credenciales y parámetros operativos de cada componente, sin necesidad de modificar el código fuente.

Seguridad

La arquitectura de Seek Shop fue diseñada con un enfoque de seguridad robusto y multicapa, considerando tanto la protección de la infraestructura en Google Cloud Platform (GCP) como la seguridad en la comunicación entre servicios y usuarios externos.

Autenticación y Autorización

- Identity Platform de GCP es utilizado como proveedor de identidades. Permite autenticar usuarios mediante múltiples métodos (correo, redes sociales, etc.) y emite tokens JWT (JSON Web Token) válidos para acceder a los recursos protegidos de la plataforma.
- Los tokens JWT incluyen claims con información sobre el usuario y son firmados por Identity Platform. Estos tokens son validados por el API Gateway y los microservicios, permitiendo controlar el acceso.
- Los microservicios implementan autorización basada en roles y niveles, verificando los claims incluidos en el token para determinar el nivel de acceso del usuario.

Seguridad de red y aislamiento

- Toda la lógica de negocio está contenida dentro de una red privada llamada Seek Shop VPC, donde se alojan los microservicios, bases de datos y componentes internos.
- La VPC impide el acceso directo desde el exterior a los microservicios. Toda comunicación externa entra a través del Cloud Load Balancer y el API Gateway, que actúan como punto único de entrada.
- Cloud NAT se utiliza para permitir que los servicios internos puedan comunicarse hacia el exterior (por ejemplo, Firebase) sin exponer sus direcciones IP internas.

Comunicación segura

- Toda la comunicación entre el cliente (portal web y app móvil) y los servicios expuestos se realiza mediante HTTPS (TLS) para garantizar la confidencialidad e integridad de los datos transmitidos.
- Las llamadas internas entre microservicios también son autenticadas mediante validación del token JWT y gestionadas a través de una arquitectura segura.

Gestión de secretos

- Secret Manager se utiliza para almacenar credenciales, claves API, contraseñas y demás información sensible.
- El acceso a los secretos está restringido mediante políticas de IAM, y los microservicios acceden a sus secretos en tiempo de ejecución, evitando su exposición en el código o configuraciones embebidas.

Seguridad en la configuración y despliegue

- Spring Cloud Config Server permite centralizar y proteger la configuración sensible, además de controlar versiones de forma segura.
- Los entornos de desarrollo, pruebas y producción pueden mantener configuraciones separadas, minimizando riesgos de fugas.
- Las actualizaciones de código se realizan a través de Cloud Build y Artifact Registry, donde los paquetes y contenedores están firmados y versionados para asegurar integridad.

Monitorización y respuesta

- Se emplean servicios de Observabilidad (Cloud Logging, Monitoring, Error Reporting y Trace) para auditar accesos, registrar errores y medir el rendimiento.
- Esto permite la detección temprana de amenazas, identificación de patrones anómalos y mejora de la respuesta ante incidentes de seguridad.

Esta estrategia de seguridad integral asegura que tanto los datos de los usuarios como los servicios de la plataforma estén protegidos ante accesos no autorizados, fallos de configuración y amenazas externas.

Resiliencia y Tolerancia a Fallos

La arquitectura propuesta para la plataforma Seek Shop incorpora diversos mecanismos orientados a garantizar alta disponibilidad, resiliencia operativa y tolerancia a fallos. A continuación, se detallan los principales elementos implementados para cumplir con estos objetivos:

Balanceo de carga y escalamiento automático

La arquitectura utiliza Google Cloud Load Balancer, un servicio global y administrado, que distribuye automáticamente las solicitudes entrantes hacia los diferentes microservicios desplegados. Esta capa de balanceo de carga cumple dos funciones clave:

1. Alta disponibilidad: evita la concentración de tráfico en un único backend, distribuyéndolo uniformemente.
2. Escalabilidad automática: permite definir políticas de escalamiento horizontal (auto-scaling) para aumentar o reducir el número de instancias de microservicios con base en métricas como uso de CPU, latencia o número de peticiones concurrentes.

Esto garantiza un sistema que se adapta dinámicamente a las variaciones de tráfico, manteniendo un rendimiento óptimo incluso en situaciones de alta demanda.

Separación de responsabilidades (microservicios desacoplados)

Cada funcionalidad crítica del sistema (usuarios, productos, inventarios, pedidos, pagos, notificaciones, compras, etc.) está implementada como un microservicio independiente, lo que favorece:

- Aislamiento de fallos: un error en el microservicio de pagos, por ejemplo, no afecta el funcionamiento del servicio de inventarios o usuarios.
- Escalabilidad independiente: cada microservicio puede escalarse según su propia demanda sin impactar a otros.
- Despliegues desacoplados: facilita actualizaciones, pruebas y correcciones sin riesgo de afectar toda la plataforma.

Este enfoque garantiza que el sistema puede continuar funcionando incluso si uno o varios microservicios fallan, brindando una experiencia robusta y resiliente al usuario.

Orquestación de procesos resiliente (Camunda BPM)

Para flujos de negocio complejos, como el proceso de compra o pago, se emplea el motor de procesos Camunda BPM. Su incorporación permite:

1. Reanudación automática: si una tarea del flujo falla, Camunda conserva el estado del proceso y permite reanudar desde el punto de fallo sin reiniciar todo el flujo.
2. Visibilidad del estado: permite monitorear visualmente en qué paso se encuentra cada instancia de proceso.
3. Manejo flexible de errores: mediante eventos de error o límites de tiempo, se puede guiar el flujo hacia rutas alternativas o reintentos planificados.

Gracias a esta orquestación, la lógica de negocio distribuida entre microservicios se mantiene coherente, tolerante y recuperable en escenarios de falla.

Gestión de configuración centralizada y tolerante (Spring Cloud Config Server)

El uso de Spring Cloud Config Server permite centralizar la configuración de todos los microservicios, con los siguientes beneficios:

- Desacoplamiento de código y configuración: las propiedades específicas de entorno (como URLs de servicios o límites de concurrencia) no están embebidas en el código.
- Caché de configuración local: si el Config Server se vuelve temporalmente inalcanzable, cada microservicio puede continuar funcionando con la configuración en caché.
- Recarga dinámica: los cambios de configuración pueden propagarse sin necesidad de reiniciar los servicios.

Este componente mejora la flexibilidad operativa y evita interrupciones ante fallas en la gestión de configuraciones.

Redis para tolerancia en sesiones y tokens

Redis, empleado como un almacén de datos en memoria, cumple un rol clave en la resiliencia del sistema:

- Gestión de sesiones y tokens JWT: se utiliza para almacenar tokens de acceso válidos y sus metadatos, lo cual permite su verificación e invalidación inmediata en caso de compromisos de seguridad.
- Baja latencia y alta disponibilidad: Redis, al ser un sistema de almacenamiento en memoria, proporciona respuestas casi instantáneas, ideal para tareas como autenticación.
- Protección frente a fallos en autenticación: si un componente del flujo de login falla, Redis permite conservar el estado sin impactar a otros servicios.

Esto asegura un control seguro y eficiente sobre la sesión de usuario, sin convertir la autenticación en un punto único de fallo.

Conectividad externa resiliente mediante Cloud NAT

Para acceder a servicios externos como Firebase Cloud Messaging, se emplea Google Cloud NAT, lo que ofrece:

- Seguridad: las instancias internas no requieren direcciones IP públicas.
- Resiliencia en la red: mantiene conectividad hacia internet incluso ante cambios en la infraestructura o rotación de IPs externas.
- Aislamiento de tráfico: permite a los microservicios comunicarse con servicios externos sin abrir canales de entrada no deseados.

Esto evita fallos derivados de conectividad pública directa y mejora la disponibilidad al minimizar dependencias de red inseguras o inestables.

Resiliencia y tolerancia a fallos con Resilience4j y Camunda

La resiliencia es una propiedad fundamental en la arquitectura de Seek Shop. Está diseñada para tolerar fallos sin que estos afecten negativamente la experiencia del usuario o el funcionamiento global de la plataforma. Para garantizar esta capacidad, se han incorporado mecanismos en múltiples capas del sistema.

Uso de Resilience4j en microservicios

Cada microservicio implementa Resilience4j desde el inicio como biblioteca principal para los patrones de resiliencia técnica:

- Circuit Breaker: detiene temporalmente las llamadas a servicios fallidos para evitar sobrecarga y permitir su recuperación.
- Retry: reintenta automáticamente llamadas fallidas ante errores transitorios.
- Rate Limiter: limita el número de llamadas por unidad de tiempo para evitar picos de tráfico descontrolados.
- Bulkhead: aisla componentes del sistema, evitando que la saturación en uno afecte a los demás.

Estos patrones permiten que cada microservicio sea autónomo en el manejo de sus errores, sin depender de otros servicios para tolerar fallos técnicos.

Manejo de errores en procesos de negocio con Camunda

Camunda BPM, al ser responsable de los flujos de negocio, maneja errores de tipo lógico o de negocio:

- Retry Time Cycle: permite definir reintentos automáticos en tareas fallidas.
- Error Events: desvía el flujo hacia caminos alternativos ante errores esperados.
- Timer Events: pospone ejecuciones o define ventanas de reintento.

Esto garantiza la continuidad del proceso sin duplicar operaciones ni perder el contexto.

Separación de responsabilidades entre microservicios y Camunda

Componente	Tipo de error	Ejemplo	Responsable
Microservicio	Técnico (timeout, red)	Tiempo de espera excedido	Resilience4j
Camunda	Negocio/lógico	Pago rechazado por fondos	BPMN/Camunda

Los microservicios se encargan de la resiliencia operativa técnica, manejando errores comunes de infraestructura.

Camunda administra la resiliencia lógica y del proceso de negocio, con control sobre la continuidad de los flujos.

Observabilidad

La arquitectura de Seek Shop incorpora una estrategia integral de observabilidad, entendida no solo como la capacidad de recolectar métricas y logs, sino también como la habilidad para entender el estado interno del sistema y anticipar, detectar y resolver problemas en tiempo real. El objetivo de esta capa es brindar visibilidad completa del comportamiento de la plataforma, detectar anomalías, identificar cuellos de botella y facilitar la trazabilidad de errores a lo largo de todo el ecosistema distribuido.

Centralización de métricas, logs y trazas

Se integrarán servicios de observabilidad provistos por Google Cloud (como Cloud Logging, Cloud Monitoring y Cloud Trace) que permiten:

- Métricas personalizadas (CPU, memoria, número de solicitudes, latencia por endpoint).
- Logs estructurados provenientes de los microservicios usando Logback.
- Trazabilidad distribuida mediante el uso de trace IDs que permiten seguir una solicitud a través de múltiples microservicios, facilitando el diagnóstico de fallos complejos.

Todos los componentes reportarán al entorno centralizado de observabilidad de GCP, lo que permite unificar el análisis y alertamiento.

Etiquetado y correlación de eventos

Cada evento, log o métrica recolectada será enriquecida con etiquetas (tags) y metadatos, como:

- Nombre del microservicio
- Versión del despliegue
- Entorno (producción, staging, testing)
- Identificador de usuario o sesión (si aplica)
- Identificador de proceso (en caso de flujos Camunda)

Esto permite realizar búsquedas y dashboards altamente filtrables, detectar patrones y correlacionar errores con cambios de despliegue u otras métricas.

Trazabilidad de procesos de negocio (BPMN)

Camunda se integra con herramientas de observabilidad y expone métricas y trazas específicas para:

- Flujos iniciados / completados / con errores
- Estado de tareas de servicio y temporizadores
- Reintentos pendientes o fallidos

Esto facilita el análisis de procesos largos o asincrónicos y mejora la capacidad de auditar el comportamiento del sistema en torno a reglas de negocio complejas.

Estrategia de Calidad y Testing

Para garantizar que la plataforma Seek Shop cumpla con los estándares de calidad esperados y ofrezca una experiencia confiable al usuario final, se adopta una estrategia de testing integral. Esta estrategia está diseñada para ser preventiva, automatizada, continua y colaborativa, involucrando tanto a desarrolladores como a testers desde las primeras etapas del desarrollo.

Testing automatizado en múltiples niveles

La estrategia contempla pruebas automáticas en distintas capas del sistema:

- Pruebas unitarias (Unit Tests): Verifican el comportamiento de métodos o funciones individuales. Se aplican a componentes de negocio, servicios, controladores y utilerías usando frameworks como JUnit 5 y Mockito.
- Pruebas de integración (Integration Tests): Validan la interacción entre múltiples componentes: servicios, base de datos, y llamadas HTTP REST. Se utilizarán testcontainers y perfiles de Spring Boot específicos para pruebas.
- Pruebas de contratos (Contract Tests): Garantizan la compatibilidad entre consumidores y proveedores de APIs REST mediante herramientas como Spring Cloud Contract o PACT, previniendo errores en ambientes desacoplados.
- Pruebas end-to-end (E2E): Simulan flujos completos de negocio a través de la aplicación móvil, web y backend. Para estas pruebas se usarán herramientas como Postman, Selenium, Cypress, Appium o suites específicas de testing móvil multiplataforma.

Validación de procesos orquestados con BPMN

Los procesos definidos en Camunda BPM se validarán con pruebas específicas:

- Simulación de procesos completos con datos reales o mockeados.
- Verificación del flujo esperado mediante assertions en los modelos BPMN.
- Validación de reintentos automáticos, eventos de error y timers definidos en los flujos.

Esto permitirá asegurar la correcta ejecución de procesos complejos como compras, pagos y notificaciones.

Testing de resiliencia y tolerancia a fallos

Se desarrollarán escenarios para probar la resiliencia de los microservicios y del motor de procesos, incluyendo:

- Pruebas con fallas intencionales (servicios detenidos, bases de datos inactivas).
- Simulación de timeouts, errores de red y respuestas inválidas.
- Validación de recuperación mediante Resilience4j (Circuit Breaker, Retry, Fallbacks).

Estas pruebas permitirán confirmar que el sistema responde de manera robusta ante condiciones adversas.

Análisis estático de código y calidad técnica

Se integrarán herramientas como:

- SonarQube / SonarCloud para análisis de código, duplicaciones, bugs, code smells y cobertura.
- Checkstyle / PMD / SpotBugs para asegurar buenas prácticas y estilos consistentes.
- Estas herramientas se ejecutarán automáticamente en el pipeline de CI/CD.

Pruebas de rendimiento y carga

Se realizarán pruebas específicas para asegurar que el sistema es escalable y eficiente bajo demanda:

- Stress Testing: para encontrar el límite del sistema ante una carga extrema.
- Load Testing: para medir la capacidad bajo condiciones esperadas.
- Spike Testing: para simular picos de tráfico inesperados.

Se usarán herramientas como JMeter, k6 o Gatling, integradas con los entornos de pre-producción en GCP.

Pruebas de regresión automatizadas

Cada nuevo cambio o feature será validado contra un conjunto de pruebas automáticas de regresión, para asegurar que funcionalidades previas sigan funcionando correctamente. Esto se automatiza como parte del pipeline de integración continua (CI).

Integración con CI/CD

Todas las pruebas se ejecutarán de forma automática en cada commit o pull request, mediante Cloud Build en Google Cloud. Solo los builds que superen todos los criterios de calidad serán desplegados a entornos superiores.

Testeo manual exploratorio

Además del testing automatizado, se planificarán sesiones de testing exploratorio por parte del equipo de QA:

- Validación de experiencia de usuario (UX)
- Flujo general de uso
- Pruebas en dispositivos reales para la app móvil (cross-platform)

Ambientes de prueba dedicados

El equipo de desarrollo contará con entornos de integración y staging independientes, cada uno con su propia base de datos, configuración y métricas, permitiendo pruebas aisladas sin afectar producción.

Implementación de pruebas automatizadas con Selenium, Appium y Cypress

Con el objetivo de garantizar una cobertura funcional completa en la interfaz de usuario (UI), se incorporarán herramientas especializadas para pruebas automatizadas en las diferentes plataformas del sistema Seek Shop:

Cypress – Pruebas E2E para la aplicación web

Cypress será utilizado para automatizar las pruebas end-to-end (E2E) de la aplicación web (Portal Web Seek Shop) desarrollada con Angular. Se eligió Cypress por su facilidad de integración con Angular, su ejecución rápida y su capacidad para depurar errores visualmente.

Cobertura esperada:

- Flujo de login
- Visualización y compra de productos
- Estado de pedidos y pagos
- Interacción con formularios y validaciones

Integración:

- Las pruebas Cypress se ejecutarán en el pipeline de CI (Cloud Build) después del despliegue en ambiente de staging.
- Se usarán fixtures para simular datos en pruebas repetibles.

Appium – Pruebas automatizadas en la app móvil

Appium se empleará para validar la experiencia en la aplicación móvil multiplataforma (Android/iOS). Esta herramienta permite simular interacciones reales de usuario sobre dispositivos físicos o emulados.

Cobertura esperada:

- Navegación por la app
- Búsqueda y selección de productos
- Flujo de checkout
- Notificaciones push (Firebase)

Estrategia de ejecución:

- Las pruebas serán ejecutadas en un entorno de testing con emuladores (Android Studio / Xcode).
- Se integrarán en el CI/CD a través de scripts que levanten el emulador y ejecuten las pruebas.

Selenium – Validaciones complementarias en UI web

Selenium se utilizará para casos complementarios de pruebas E2E más personalizadas o cuando se necesite mayor flexibilidad de ejecución en distintos navegadores (Cross-browser testing).

Casos de uso:

- Validación en navegadores distintos (Chrome, Firefox, Safari)
- Accesibilidad (validaciones de focus, contraste, roles)
- Automatización de regresiones visuales

Ejecución:

- Integración con Selenium Grid o BrowserStack para pruebas paralelas
- Ejecución programada nocturna o bajo demanda tras un despliegue

Cada herramienta será utilizada estratégicamente en el contexto adecuado:

Herramienta	Plataforma	Tipo de prueba	Justificación principal
Cypress	Web (Angular)	End-to-end rápida	Rápido, visual, ideal para Angular
Appium	Móvil (Android/iOS)	Funcional E2E	Cross-platform, simula interacción real
Selenium	Web (Cross-browser)	E2E avanzada	Compatibilidad multi-navegador, accesibilidad

Web (Cross-browser) E2E avanzadaCompatibilidad multi-navegador, accesibilidad
Estas herramientas permitirán asegurar una experiencia consistente para el usuario final, validando los flujos críticos de extremo a extremo en todas las plataformas soportadas por Seek Shop.

Estrategia de CI/CD y Despliegue

La plataforma Seek Shop ha sido diseñada bajo un enfoque de integración y entrega continua (CI/CD), con el objetivo de reducir errores humanos, acortar ciclos de entrega, asegurar la calidad del software y habilitar despliegues repetibles y automatizados en los diferentes ambientes (desarrollo, staging y producción).

Herramientas y componentes utilizados

GitHub

Repositorio central del código fuente, gestionado mediante ramas (main, develop, feature/*, release/*, hotfix/*). Cada pull request activa automáticamente flujos de CI a través de integraciones con Cloud Build.

Cloud Build

Servicio de Google Cloud que ejecuta pipelines CI/CD definidos en archivos cloudbuild.yaml. Cada microservicio posee su propio pipeline que incluye pasos como:

- Descarga del código y resolución de dependencias

- Ejecución de pruebas unitarias y de integración
- Construcción de imágenes Docker
- Escaneo de seguridad (opcional)
- Push a Artifact Registry
- Despliegue a ambientes (dev, staging, prod)

Artifact Registry

Almacén centralizado y seguro de imágenes de contenedores construidas. Cada microservicio se empaqueta como una imagen Docker versionada y firmada, facilitando trazabilidad y rollback.

Cloud Run o Google Kubernetes Engine (según etapa)

El despliegue final de cada servicio puede realizarse sobre:

- Cloud Run en etapas tempranas para facilitar despliegues rápidos sin administración de infraestructura.
- GKE en etapas maduras que requieren mayor control sobre la orquestación, redes, y configuración avanzada.

Flujo CI/CD por microservicio

Cada microservicio sigue el siguiente ciclo de vida automatizado:

1. Desarrollador hace push o pull request
2. Cloud Build ejecuta pruebas, compila y construye imagen
3. Imagen es enviada a Artifact Registry
4. Despliegue automático a entorno (dev/staging/prod)
5. Validación post-deploy / smoke tests
6. Notificación del resultado (Slack/Email)

Estrategia de despliegue

La plataforma contempla una estrategia de despliegue progresivo basada en entornos diferenciados y flujos seguros de publicación:

Entorno de desarrollo (dev)

Despliegue inmediato tras commit a ramas feature/* o develop. Utilizado para validación rápida e integración con otros servicios.

Entorno de pruebas (staging)

Replica lo más fielmente posible la configuración de producción. Solo se despliega tras merge a release/*. Aquí se ejecutan:

- Pruebas E2E con Cypress, Selenium y Appium
- Pruebas de carga (opcional)
- Validación manual por QA

Entorno de producción (prod)

Despliegue controlado desde rama main, supervisado por equipo DevOps. Se puede utilizar:

- Canary deployment, exponiendo solo una fracción del tráfico a nuevas versiones.
- Blue/Green deployment, manteniendo la versión anterior activa durante un tiempo de transición.

Validaciones post-despliegue

Después de cada despliegue se activan tareas automáticas para validar la estabilidad del sistema:

- Pruebas de smoke testing automatizadas
- Verificación de estado del servicio (health checks)
- Métricas de latencia, errores y uso desde Google Cloud Monitoring
- Alertas en caso de fallos con Google Cloud Alerting o integraciones como Slack

Rollback y control de versiones

- Cada versión desplegada está asociada a un hash de commit y una imagen Docker específica.
- En caso de fallo crítico, es posible hacer rollback automático al último estado estable con un solo comando.
- Todas las configuraciones son versionadas a través del Config Server y gestionadas con Git.

Anexos

Repositorios

1. Diagramas

Flujo de compra de un producto

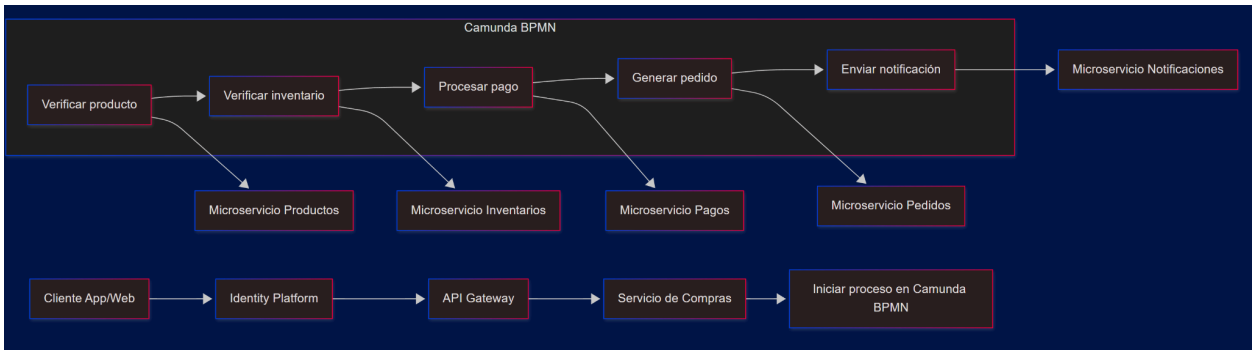

Diagrama de flujo de compra de un producto

Stack tecnológico

Backend: Microservicios

- Lenguaje: Java 17 (LTS)
- Framework principal: Spring Boot 3.2.x
- Arquitectura: Microservicios desacoplados con responsabilidad única
- Patrones usados:
    - RESTful API Design
    - API Gateway Pattern
    - Circuit Breaker (con Resilience4j)
    - Retry & Fallback (con Resilience4j)
    - Centralized Configuration Pattern
    - Externalized Secrets Pattern
    - Event-Driven Interaction (a futuro)
- Librerías y herramientas:
    - Spring Web para REST API
    - Spring Data JPA para acceso a bases de datos relacionales
    - Spring Data MongoDB para documentos NoSQL
    - Spring Security con JWT para autenticación
    - Spring Cloud Config (servidor central de configuración)
    - Resilience4j para circuit breakers, retries, bulkhead y rate limiter
    - MapStruct para conversión entre DTOs y entidades
    - Lombok para reducir boilerplate
    - Jakarta Validation para validación de datos de entrada

Persistencia de Datos

- PostgreSQL 15: Motor relacional principal usado por los microservicios de usuarios, productos, inventarios, pagos, pedidos y notificaciones.
- MongoDB 6.0: Motor NoSQL para datos no estructurados como logs de notificaciones y catálogos flexibles de productos.
- Oracle Database (on-premise): Sistema de facturación heredado; integración mediante REST y drivers compatibles con Java.
- Redis 7.x:
    - Usado como almacén en memoria.
    - Almacena tokens JWT activos y revocados.
    - Puede soportar caching de consultas de alta frecuencia en el futuro.

Frontend Web

- Framework: Angular 17
- Lenguaje: TypeScript
- Gestión de estado: RxJS + NgRx (si escala el proyecto)
- Estilo: Angular Material

- Consumo de APIs: HttpClient + JWT para autenticación
- Testing: Cypress

Aplicación Móvil

- Framework: Flutter 3.19.x
- Lenguaje: Dart
- Estrategia: Aplicación multiplataforma (Android + iOS)
- Consumo de APIs: http, dio o graphql_flutter (si aplica)
- Estado: Provider (con posible migración a Riverpod en versiones futuras)
- Notificaciones Push: Integración con Firebase Cloud Messaging
- Testing: flutter_test + integration_test

BPM y Orquestación

- Motor BPMN: Camunda Platform 7.21 (embedded)
- Librerías usadas:
    - camunda-spring-boot-starter para integración rápida
    - camunda-bpm-data para manejo de datos del proceso
- Persistencia: PostgreSQL (configurada desde el starter)
- Modelado BPMN:
    - Se usa el Camunda Modeler para diseñar diagramas BPMN con elementos como:
        - Service Task
        - Error Boundary Event
        - Timer Boundary Event
        - Retry Time Cycle
- Estrategia:
    - Camunda orquesta procesos complejos como el flujo de compra
    - Las tareas técnicas se delegan a microservicios especializados
    - Camunda maneja resiliencia lógica, errores de negocio y retoma del proceso

Google Cloud Platform (GCP)

- Compute: Google Cloud Run o GKE (como opciones de despliegue escalables)
- API Gateway: Control centralizado de acceso y enrutamiento
- Identity Platform: Autenticación y federación de usuarios (OAuth2, JWT)
- Secret Manager: Gestión segura de credenciales
- Cloud NAT: Salida controlada a internet para microservicios sin IP pública
- Cloud Load Balancing: Distribución de carga HTTP/S, TCP y SSL
- Cloud Build: Pipeline CI/CD
- Artifact Registry: Almacenamiento de imágenes Docker
- Service Directory: Descubrimiento de servicios
- Observability Stack:
    - Cloud Logging
    - Cloud Monitoring
    - Cloud Trace

    - Cloud Error Reporting

Testing

- Unit Tests: JUnit 5 + Mockito (Java)
- Integration Tests: Spring Boot Test + Testcontainers
- Contract Testing: Spring Cloud Contract o Pact (opcional)
- Frontend Testing:
    - Angular: Cypress
- Mobile Testing:
    - Flutter: Integration Test
    - Appium para pruebas funcionales multiplataforma
- E2E BPMN: Pruebas automatizadas de procesos usando Camunda Test Coverage

CI/CD

- Repositorios: GitHub (privado)
- Integración continua: GitHub Actions + Cloud Build
- Despliegue:
    - Build automático de imágenes Docker
    - Push a Artifact Registry
    - Deploy a Cloud Run o GKE con rollout progresivo
- Rollback: Versionado de imágenes y rollback en Cloud Build
- Análisis estático de código:
    - SonarQube (opcional)
    - Checkstyle/Detekt para Java/Kotlin

Buenas prácticas recomendadas para el equipo de desarrollo

Para asegurar la calidad, mantenibilidad y escalabilidad del sistema Seek Shop, el equipo de desarrollo deberá seguir las siguientes buenas prácticas:

1. Diseño orientado a responsabilidades únicas: Cada microservicio debe implementar una única responsabilidad claramente definida, evitando lógica de negocio duplicada y facilitando el escalamiento independiente.
2. Clean Code y principios SOLID: El código debe mantenerse legible, simple y extensible. Se fomenta el uso de patrones como Factory, Strategy y Adapter cuando sea necesario.
3. Pruebas automatizadas desde el primer día: Se deben implementar pruebas unitarias, de integración y end-to-end como parte del flujo de desarrollo. Ningún código deberá ser promovido sin cobertura mínima y validación automática.
4. Control de versiones y trabajo por ramas: Se trabajará bajo un modelo de ramas Git (feature, develop, main) y pull requests, asegurando revisiones y colaboración continua.
5. CI/CD y entrega continua: Todo cambio deberá pasar por pipelines automatizados (Cloud Build), que compilen, prueben, verifiquen y desplieguen de forma progresiva.
6. Documentación técnica viva: Cada servicio deberá incluir documentación técnica (README, Swagger/OpenAPI, diagramas), y mantenerse actualizada en cada iteración.

7. Fail fast & observabilidad: Los servicios deben fallar rápido en caso de error y enviar trazas, métricas y logs estructurados a los sistemas de observabilidad (Cloud Logging & Monitoring).
8. Seguridad por defecto: Todos los endpoints deben validarse con JWT. Nunca se deben incluir secretos en el código fuente; se gestionarán desde Secret Manager.
9. Evitar acoplamientos fuertes: La comunicación entre servicios debe ser siempre a través de interfaces bien definidas (REST o BPMN), evitando dependencias directas de implementación.
10. Cultura DevOps y mejora continua: Se fomentará la responsabilidad compartida sobre el ciclo de vida del software, incluyendo calidad, despliegue, monitoreo y soporte.

Equipo de desarrollo

Rol	Cantidad	Perfil	Responsabilidades
Líder técnico / Arquitecto de software	1		Diseño de arquitectura, decisiones tecnológicas, supervisión técnica, mentoring
Senior Backend Developer	3	Experto en Java, Spring Boot, JPA, REST, Camunda, Resilience4j, GCP	Implementación de microservicios, integración con BPMN, bases de datos.
Senior Frontend Developer	1	Angular, consumo de APIs REST, experiencia con JWT y buenas prácticas de UX	Desarrollo del portal web, integración con APIs, UI responsiva.
Senior Mobile Developer	1	Flutter, Firebase, consumo de APIs, experiencia multiplataforma	Desarrollo de la app móvil, integración con Firebase y backend.
Senior DevOps Engineer (GCP)	1	Terraform (opcional), GCP, Docker, GitHub Actions, Cloud Build, seguridad, monitoreo	Automatización CI/CD, infraestructura como código, seguridad, monitoreo.
Senior QA Automation Engineer	1	Selenium, Appium, Cypress, integración con CI, validación de APIs	Implementación de pruebas con Selenium, Appium, Cypress, pruebas funcionales

Analista de Negocio / Product Owner	1	Gestión ágil, planificación, comunicación con stakeholders	Definición de procesos BPMN, requisitos, validación funcional.
		Total	**9 personas (8 tiempo completo, 1 parcial)**

Estimación de tiempo

Con un desarrollo en modalidad full-time, ágil (Scrum), en sprints de 2 semanas:

Fase	**Duración estimada**
Planeación y diseño técnico	2 semanas
Configuración de infraestructura CI/CD + GCP	2 semanas
Desarrollo MVP backend (microservicios, BPMN, seguridad, resiliencia)	10 semanas
Desarrollo frontend web (Angular)	7 semanas
Desarrollo app móvil (Flutter)	8 semanas
Pruebas automáticas y QA	8 semanas (en paralelo)
Revisión, ajustes, documentación y despliegue	3 semanas
Total	**18 a 24 semanas (4 a 6 meses)**

Costos

Costos de personal

Rol	**Tarifa mensual**	**Cantidad**	**Meses**	**Total (MXN)**
Líder técnico	$90,000.00	1	6	$540,000.00
Senior Backend Developer	$60,000.00	3	6	$1,080,000.00
Senior Frontend Developer	$60,000.00	1	6	$360,000.00
Senior Mobile	$60,000.00	1	6	$360,000.00

Developer				
Senior DevOps Engineer	$65,000.00	1	6	$390,000.00
Senior QA Automation Engineer	$55,000.00	1	6	$330,000.00
Product Owner	$70,000.00	1	6	$420,000.00
			Total	**$3,480,000.00**