

# Arquitectura de Integracion

**CONFIDENCIAL**

**[Fecha 23/09/2024]**

**Versión [1.0]**

Contenido

**CONFIDENCIAL.....1**

**1. DISEÑAR UNA ARQUITECTURA DE INTEGRACIÓN DE ALTO NIVEL .....3**

**1.1. CONTEXTO .....3**

**1.2. CONTENEDORES .....3**

**1.3. COMPONENTES .....4**

**2. ESPECIFICAR PATRONES DE INTEGRACIÓN Y TECNOLOGÍAS A UTILIZAR.....5**

**3. ABORDAR REQUISITOS DE SEGURIDAD, CUMPLIMIENTO DE NORMATIVO, LEY ORGÁNICA DE PROTECCIÓN DE DATOS PERSONALES .....6**

**4. PROPONER ESTRATEGIA PARA GARANTIZAR ALTA DISPONIBILIDAD Y RECUPERACIÓN ANTE DESASTRES.....6**

**5. PROPONER ESTRATEGIA DE INTEGRACIÓN MULTICORE.....7**

**6. DELINEAR UN ENFOQUE PARA LA GESTIÓN DE IDENTIDAD Y ACCESO EN TODOS LOS SISTEMAS .....7**

**7. DISEÑAR UNA ESTRATEGIA DE API INTERNAS Y EXTERNAS, BAJO ESTÁNDARES DE LA INDUSTRIA RESPECTO A LA MENSAJERÍA .....8**

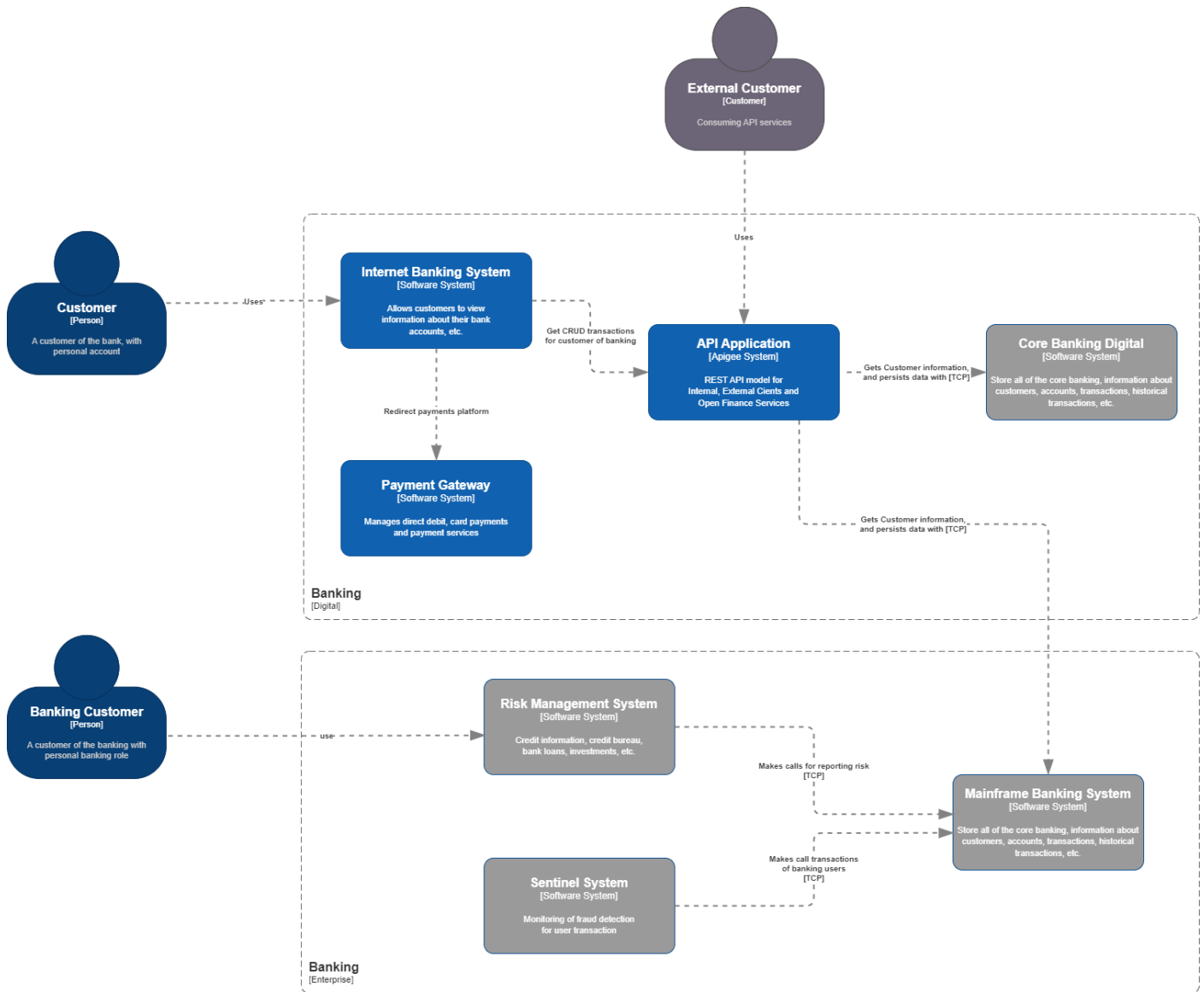
**8. PROPONER UN MODELO DE GOBIERNO DE APIS MICROSERVICIOS .....10**

**9. PROPONER UN PLAN DE MIGRACIÓN GRADUAL QUE MINIMICE EL RIESGO OPERATIVO .....11**

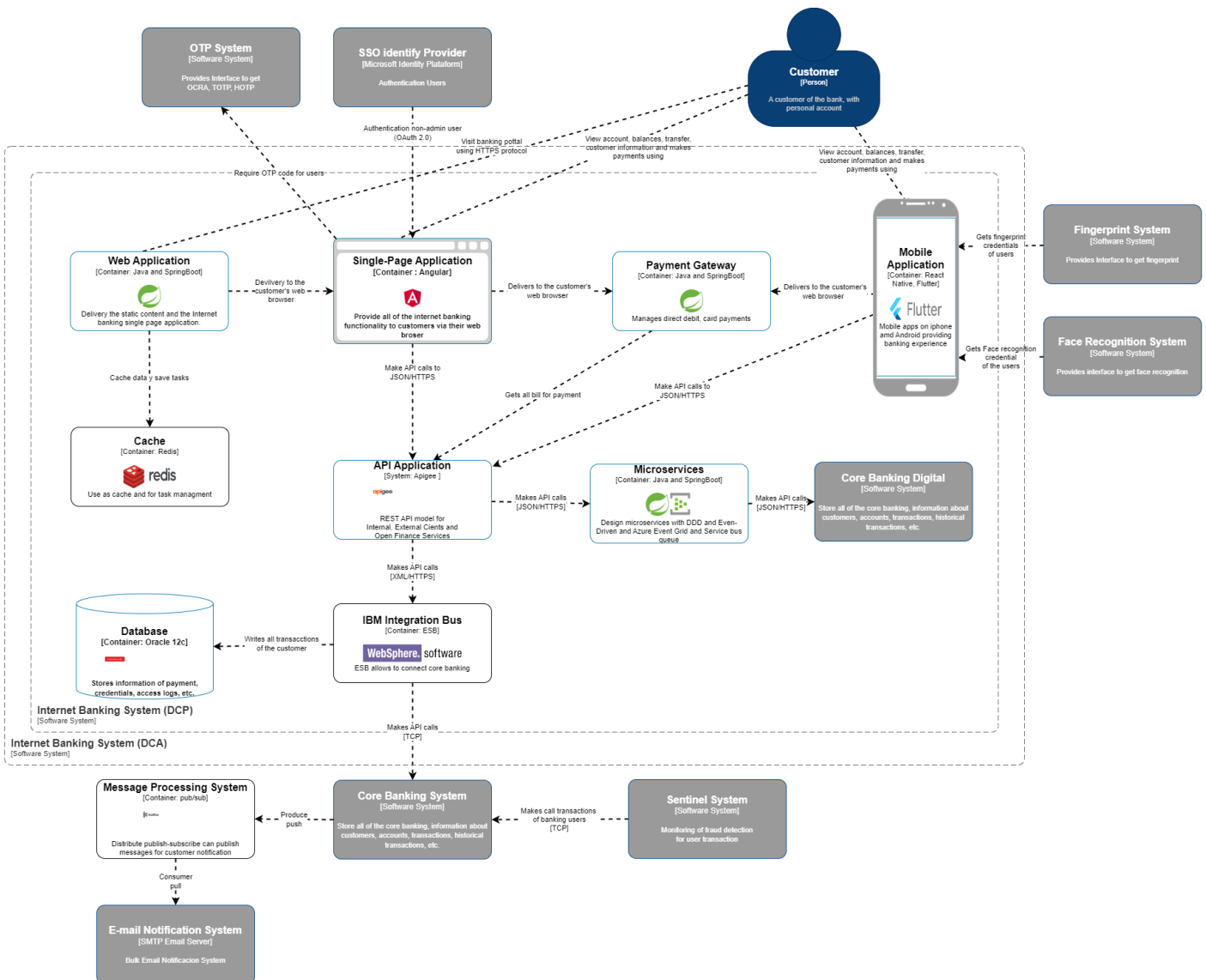
## 1. Diseñar una arquitectura de integración de alto nivel

Se realizó el modelado c4 en base al scope de la solución en base a los siguientes diagramas.

### 1.1. Contexto



### 1.2. Contenedores



### 1.3. Componentes

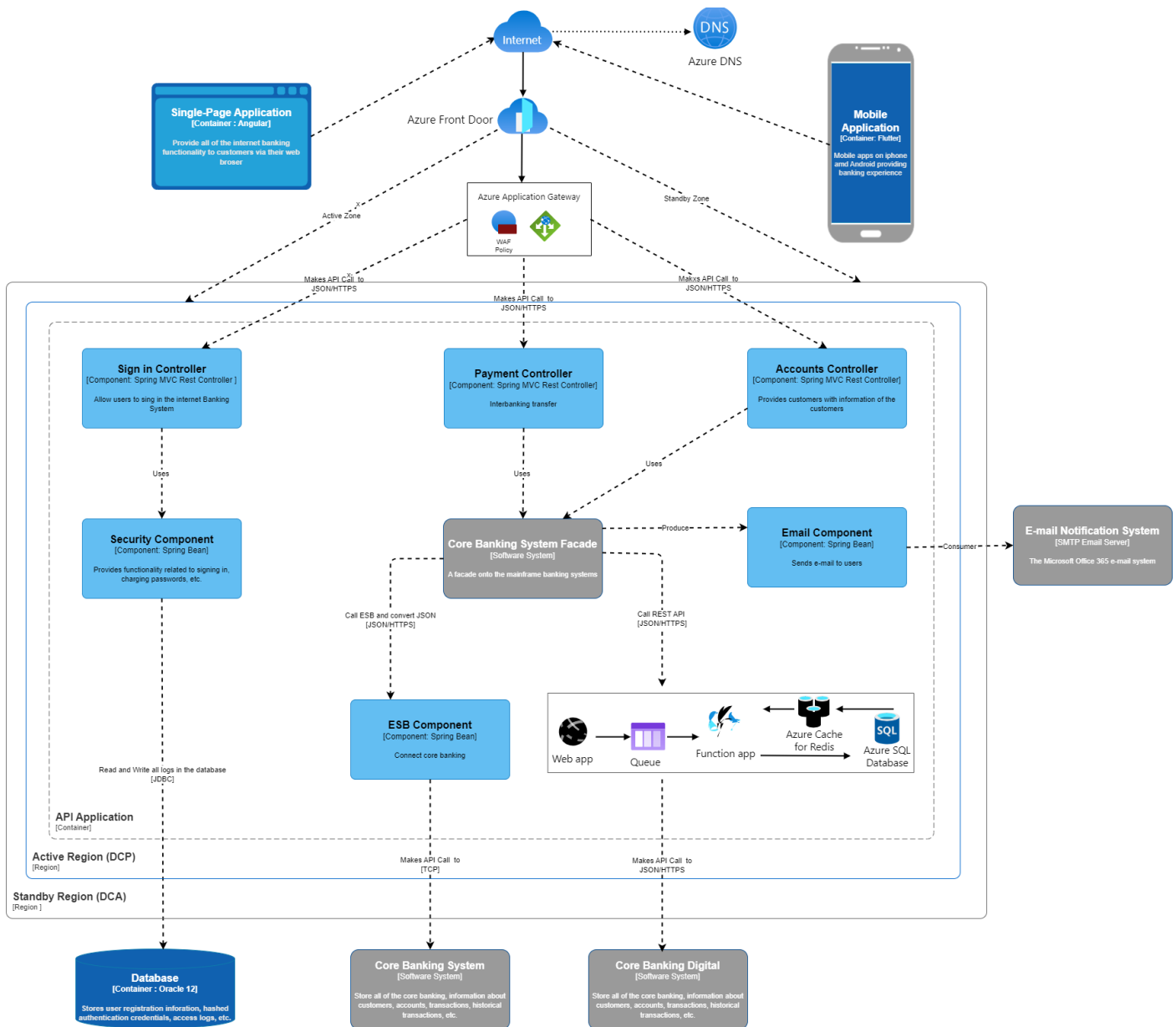
Elaborado por:

[Franklin Gallardo]

Documento Técnico

Fecha: 23/09/2024

Versión V 1.0



## 2. Especificar patrones de integración y tecnologías a utilizar

Se implementaron los siguientes patrones de integración:

- Message Channel
- Point-to-Point Channel
- Routing messages
- Transformation Message
- Composed Message
- Message Transformation

Elaborado por:

[Franklin Gallardo]

Documento Técnico  
Fecha: 23/09/2024  
Versión V 1.0

- Message Endpoint
- Async Message
- Message Routing
- Publish-Subscribe Channel

Se utilizaron las siguientes tecnologías.

- **IBM Integration Bus.** – Para integración con el core bancario tradicional.
- **Apigee.** - Se integra a servicios expuestos por el bus y a los microservicios que se conectan al core bancario digital, permite exponer los servicios internos y externos.
- **Apache Kafka.** - Sistema mensajería pub/sub para notificaciones al cliente.
- **App Services.** – Microservicios con un enfoque DDD de modelado de software y la utilización de contenedores.
- **Event Grid Service Bus.** – Permite el manejo de mensajería en Azure
- **Azure Functions.** - Funciones serverless que es una IaC.

### 3. Abordar requisitos de seguridad, cumplimiento de normativo, ley orgánica de protección de datos personales

- Normas generales para las instituciones del sistema financiero (<https://www.superbancos.gob.ec/bancos/codificacion-de-normas-de-la-sb-libro-uno-sistema-financiero/>), las principales:
  - Se requiere doble método de autenticación un MFA, el primero usuario y clave, el segundo puede ser biométrico, reconocimiento facial, OTP, etc.
  - La información sensible no debe salir del territorio ecuatoriano, esta información son las identificaciones, nombres, etc.
- La norma ISO/IEC 27001, normativa reconocida internacionalmente para la gestión de la seguridad de la información. Permite establecer, implementar, mantener y mejorar continuamente un sistema de gestión de la seguridad de la información (SGSI)., lo importante de la normativa:
  - Protección de datos
  - Cumplimiento del regulatorio legal y requerimientos como GDPR
  - Gestión de riesgo enfocado a reducir vulnerabilidades

### 4. Proponer estrategia para garantizar alta disponibilidad y recuperación ante desastres

- Se recomienda utilizar en on-prem DCA y DCP
- En la nube se requiere utiliza Azure Front Door que es un CDN que ofrece alto rendimiento, escalabilidad y experiencias de usuario seguras para su contenido y aplicaciones, ya que es un servicio global que permite distribuir requerimientos entre regiones, como se detalla en el diagrama de componentes.

A nivel de DR:

Elaborado por:

[Franklin Gallardo]

Documento Técnicos  
Fecha: 23/09/2024  
Versión V 1.0

Se requiere definir un HA con:

- **Redundancia de Componentes Críticos** como servidores implementando load balancing, discos y almacenamientos implementando RAID o almacenamiento distribuidos mediante un NAS
- **Balanceo de carga** Implementar balanceadores de carga de nivel de red (DNS) y de nivel de aplicación (como NGINX o HAProxy).
- **Replicación de base de datos** SQL server con replicación nativa y oracle con data guard
- **Monitoreo continuo** con Nagios, Prometheus, o Zabbix que supervisen constantemente el estado de los sistemas y alerten sobre posibles fallos

Recuperación ante Desastres (Disaster Recovery - DR)

- Tener definido los objetivos
  - **RPO (Recovery Point Objective):** El máximo tiempo de datos que la empresa está dispuesta a perder. A menor RPO, más frecuente debe ser la replicación de datos.
  - **RTO (Recovery Time Objective):** El tiempo máximo en que los sistemas deben estar operativos tras un desastre.
- Copia de Seguridad (Backups)
- Sitios de Recuperación (DR Sites)

## 5. Proponer estrategia de integración multicore

Existen varias estrategias en las que se pueden implementar a nivel de multicore, todo depende del proceso y la carga del mismo.

- Apigee es un API Manager enriquecido, el cual permite diseñar, desarrollar, publicar, gestionar tráfico, implementar niveles de seguridad, monetizar, gracias a sus deferentes componentes como: Traffic Managment, Security, Mediation, Extension.
- Un ESB como IBM Integration Bus permite realizar la orquestación, transformación y ruteo, lo cual permite realizar la integración de manera sencilla con tramas TCP directamente con el CORE
- Dependiendo del escenario se puede utilizar la implementación de colas de mensaje, para integración con programación asincrónica.
- Transacciones en línea que se realizan llamando a COMTI TCP/IP permite realizar un wrapper de los servicios desarrollados en COBOL y que puedan ser consumidos desde cualquier lenguaje de programación
- Para procesos pesados en donde se requiere procesar millones de transacciones diarias, se utilizan archivos TXT en formato zip para la transferencia de información entre los diferentes cores, es una manera sencilla de transmitir grandes volúmenes de información de manera sencilla, este esquema se utiliza para reportes de la super intendencia de bancos y seguros como reporte de tarjetas de créditos S21, S22. Etc.

## 6. Delinear un enfoque para la gestión de identidad y acceso en todos los sistemas

Elaborado por:

[Franklin Gallardo]

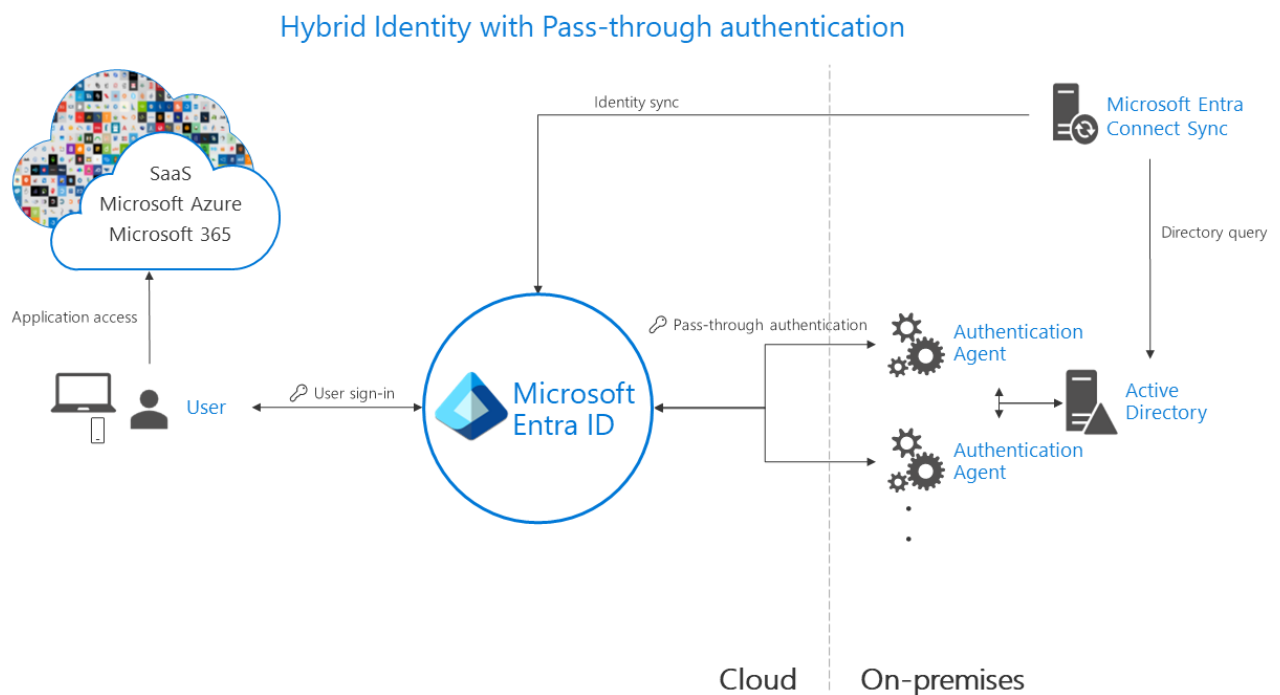
Documento Técnicos  
Fecha: 23/09/2024  
Versión V 1.0

Se lo puede realizar con Hybrid identity with Pass-through authentication, utilizando dos agentes para redundancia

Con Microsoft Entra ID podemos tener:

- **Autenticación y Autorización** con Single Sign-On (SSO), Autenticación multifactor (MFA), Autenticación basada en riesgos.
- **Gestión de Identidades** de usuarios y grupos, Federación de identidades, Sincronización con Active Directory On-Premise.
- **Gobierno de Identidades y Accesos** con políticas de acceso condicional, Privileged Identity Management (PIM)
- **Aplicaciones Empresariales** acceso a la Galería de aplicaciones preintegradas con SaaS, autenticación con OpenID Connect y OAuth 2.0, Acceso seguro a aplicaciones locales.
- **Protección contra Amenazas** gracias a la Identidad protegida, Supervisión continua.


Un esquema sería el siguiente.



## 7. Diseñar una estrategia de API internas y externas, bajo estándares de la industria respecto a la mensajería

Con Apigee se puede establecer tanto API internas como externas, se requiere definir los ambientes en los cuales se va publicar las apis, casos de uso en escenarios para publicar.





<> Develop

Specs

API Proxies

Shared Flows

Offline Trace

Publish

Analyze

Admin

Help

Feedback

Proxies

Environment

prod

All

test

NAME	STATUS
afln-afiliaciones-inscritos	
ciam-tecnologia-usuarios	
cols-medicamentos-prescripcion	
cols-supermercados-bonos	
cols-afiliaciones-empresas	
sise-empleabilidad-vacantes	
saftc-tecnologia-mensajes	

## Estándares

- **Políticas de Seguridad:** Aplicar autenticación basada en tokens (OAuth, JWT) o claves API para garantizar que solo los servicios autorizados puedan interactuar con las APIs.
- **Rate Limiting y Throttling:** Definir límites en las solicitudes de API para evitar la sobrecarga de servicios internos.
- **Monitorización y Análisis:** Utilizar Apigee para monitorizar el uso de las APIs, detectar problemas de rendimiento y realizar análisis de patrones de uso.
- **Orquestación y Transformación:** Apigee puede orquestar la comunicación entre varios servicios internos y transformar las solicitudes/respuestas para asegurar compatibilidad entre sistemas.

## APIs Internas

- **Conexión de microservicios:** Apigee puede ayudar a gestionar la comunicación entre microservicios internos, proporcionando políticas para el control de tráfico, caching, y control de errores.
- **Automatización de procesos internos:** Por ejemplo, APIs que exponen servicios internos de recursos humanos, gestión de inventario, o sistemas de facturación que son utilizados por las aplicaciones internas.

## APIs Externas

- **Exposición pública:** Las APIs externas están accesibles desde Internet y generalmente requieren controles de seguridad más estrictos para proteger los datos y sistemas.
- **Autenticación y autorización robustas:** Es fundamental implementar autenticación sólida (OAuth 2.0, OpenID Connect) para identificar y controlar el acceso de los usuarios externos.

- **Escalabilidad:** Las APIs externas suelen tener una alta demanda, por lo que deben estar diseñadas para escalar automáticamente y manejar grandes volúmenes de tráfico.
- **Documentación pública:** Al estar dirigidas a desarrolladores externos, es importante proporcionar una documentación clara y herramientas de soporte (como portales de desarrolladores) para facilitar su adopción.
- **Control y gobernanza:** Las APIs públicas requieren políticas más detalladas para la limitación de tasas, acuerdos de nivel de servicio (SLAs), y monitoreo para evitar abusos o mal uso.

Característica	APIs Internas	APIs Externas
Accesibilidad	Acceso limitado a la red interna o VPN.	Expuestas públicamente a través de internet.
Seguridad	Autenticación menos compleja (dentro de la organización).	Autenticación robusta (OAuth, claves API, TLS).
Escalabilidad	Normalmente con menor demanda de tráfico.	Escalable para manejar grandes volúmenes de solicitudes.
Documentación	Documentación interna limitada para equipos específicos.	Documentación pública detallada con soporte para desarrolladores externos.
Control de Acceso	Control estricto de acceso a servicios internos.	Requiere autorización y validación de acceso público.
Ciclo de vida	Cambios rápidos y menor dependencia externa.	Cambios controlados y versiones para minimizar el impacto externo.

## 8. Proponer un modelo de gobierno de APIs microservicios

Que se entiende por un **modelo de gobierno de APIs en un entorno de microservicios** establece las pautas, políticas y procedimientos necesarios para gestionar de manera efectiva el ciclo de vida de las APIs que exponen los microservicios.

Que se debe tener en cuenta

- Descentralización con Control Centralizado
- Autonomía de los Microservicios
- Estándares y Normativas Comunes

Modelo de Gobierno de APIs para Microservicios

- **Política de Diseño de APIs** se debe considerar el diseño basado en contratos, Consistencia en la nomenclatura y estructura en los métodos HTTP (GET, POST, PUT, DELETE), Documentación con Swagger/OpenAPI, definir una estrategia clara de versionado utilizar versionado en la URL (`/v1/resource`) o mediante encabezados.

- **Seguridad** utilizar autenticación y autorización e implementar mecanismos robustos de autenticación como OAuth 2.0, JWT, o mTLS (mutual TLS), políticas de acceso condicional, cifrado de extremo a extremo, gestión de secretos integrando herramientas como HashiCorp Vault, AWS Secrets Manager, o Azure Key Vault.
- **Estandarización de Protocolos de Comunicación** utilizando REST vs. gRPC definir cuándo es adecuado utilizar APIs REST o gRPC según el caso de uso, política de Serialización estándar como JSON o Protocol Buffers para garantizar la interoperabilidad entre microservicios y consumidores, manejo de errores: Establecer convenciones para la gestión de errores y códigos de estado HTTP
- **Monitoreo y Trazabilidad** utilizando logging centralizado recoja registros de todos los microservicios, como ELK Stack (Elasticsearch, Logstash, Kibana) o Splunk, para facilitar la depuración y auditoría, Tracing distribuido trazabilidad como OpenTelemetry para implementar rastreo distribuido que permita seguir el flujo de las solicitudes a través de los distintos microservicios, monitoreo de rendimiento Utilizar herramientas de monitoreo como Prometheus, Grafana, o Datadog, Alertas y métricas de salud (latencia alta, tiempo de inactividad, etc).
- **Gestión del Ciclo de Vida de APIs** con la Creación y publicación de APIs, Versionado y Deprecación, Gestión del tráfico y QoS (Calidad de Servicio)
- **Automatización y DevOps** Integración continua y entrega continua (CI/CD) con Jenkins, GitLab CI, o Azure DevOps, Testing automatizado, Despliegue de microservicios despliegue como Blue-Green Deployments o Canary Releases
- **Gobernanza del Acceso** definir roles y permisos Definir una estructura de roles y permisos que controle quién puede desarrollar, modificar y consumir las APIs, Delegación de acceso, Auditoría y conformidad
- **Portal de Desarrolladores y Autogestión** con Portal de APIs, Políticas de adopción y soporte.

#### Beneficios de un Modelo de Gobierno de APIs para Microservicios

- **Consistencia:** Asegura que todas las APIs y microservicios sigan estándares comunes, facilitando la interoperabilidad.
- **Escalabilidad:** Facilita la adición de nuevos microservicios y APIs a medida que la organización crece, sin generar desorden.
- **Seguridad:** Proporciona un marco claro para proteger las APIs contra ataques y accesos no autorizados.
- **Calidad y Confiabilidad:** Asegura que las APIs se mantengan confiables, bien documentadas y compatibles con el tiempo.

## 9. Proponer un plan de migración gradual que minimice el riesgo operativo

Es un modelo de lanzamiento de aplicaciones que **transfiere poco a poco el tráfico de usuarios de cierta versión anterior de una aplicación o microservicio a una versión nueva casi idéntica, cuando ambas se encuentran en producción**

La **versión anterior se denomina entorno azul, mientras que la versión nueva se conoce como entorno verde**. Cuando el tráfico de producción se transfiere por completo del entorno azul al verde, la versión azul puede

conservarse en caso de que sea necesaria una restauración, o puede extraerse de la producción y actualizarse para convertirse en la plantilla a partir de la cual se realizará la próxima actualización.

