VISION DE ARQUITECTURA

Descripción del Problema

ABCall busca posicionarse como un jugador clave en el segmento de pequeñas y medianas empresas que requieren servicios de atención al cliente externalizados. El principal problema que enfrenta es la falta de infraestructura y capacidad técnica en estas empresas para manejar eficazmente la gestión de llamadas y otros canales de comunicación de atención al cliente. El objetivo del proyecto es diseñar un sistema integral que permita gestionar de manera eficiente las comunicaciones entrantes y salientes, integrando funcionalidades avanzadas como analítica predictiva e inteligencia artificial generativa para automatizar y mejorar la experiencia del usuario.

Objetivos del Proyecto

- **Automatización y Eficiencia**: Desarrollar un sistema que permita gestionar hasta 500 llamadas por minuto y automatizar la mayor parte del proceso de gestión de PQRs a través de diferentes canales, como teléfono, correo electrónico y chatbot.
- **Analítica Avanzada**: Implementar servicios de analítica predictiva y de IA generativa para mejorar la resolución de incidentes y optimizar la atención al cliente.
- **Alta Disponibilidad y Escalabilidad**: Asegurar que el sistema esté disponible 24/7 con alta capacidad de respuesta, capaz de manejar picos de demanda sin degradar la calidad del servicio.
- **Facilidad de Integración**: Proveer mecanismos de integración basados en APIs REST y GraphQL para la actualización en tiempo real de la información del cliente y sus servicios.

Riesgos Identificados

- **Riesgo de Escalabilidad**: La posibilidad de que el sistema no maneje eficientemente los picos de llamadas podría afectar la calidad del servicio y la satisfacción del cliente.
- **Dependencia Tecnológica**: El éxito del proyecto depende de la integración eficaz de tecnologías avanzadas como la analítica predictiva y la IA generativa, lo que puede representar un desafío técnico significativo.
- **Confidencialidad y Seguridad**: El manejo de grandes volúmenes de datos sensibles requiere robustas medidas de seguridad para evitar brechas de seguridad y garantizar la integridad de los datos.

Restricciones de Negocio y Tecnología

- **Restricciones de Tiempo:** El proyecto debe ser completado en 16 semanas, con un prototipo funcional listo al final de este periodo para poder asegurar la inversión de riesgo.
- **Restricciones Presupuestarias:** El equipo de desarrollo está limitado a 4 personas para la arquitectura y 4 para el desarrollo, lo que requiere una optimización máxima de los recursos disponibles.
- **Tecnología de Despliegue:** El sistema se desplegará en la nube, aprovechando las capacidades de escalabilidad y alta disponibilidad de los servicios cloud.

Esfuerzo Estimado

Estimación de Esfuerzo y Actividades (8 Semanas, Equipo de 4, 16 horas/persona/semana)

# EDT	**Nombre**	**Esfuerzo (Semanas)**
Gerencia del Proyecto		
1.1	Crear Acta de Constitución del Proyecto	0.5
1.2	Crear el Documento de Alcance del Proyecto	1
1.3	Gestión del Proyecto y Reportes	0.5 (distribuido en las demás fases)
Diseño		
2.1	Diseño de la Arquitectura	2
2.1.1	Definir la arquitectura del sistema	1
2.1.2	Seleccionar patrones de diseño	0.5
2.1.3	Diseñar la base de datos	0.5
2.2	Diseño de la Interfaz de Usuario	1
2.2.1	Diseñar la interfaz de usuario (UI)	0.5

2.2.2	Crear prototipos de la interfaz de usuario	0.25
2.2.3	Realizar pruebas de usabilidad	0.25
2.3	Diseño de las Pruebas	0.5
2.3.1	Definir la estrategia de pruebas	0.25
2.3.2	Documentar los casos de prueba	0.25
Implementación		
3.1	Desarrollo del Software	4
3.1.1	Implementar las historias de usuario	3
3.1.2	Realizar pruebas unitarias	0.5
3.1.3	Integrar el código de forma continua	0.5
3.2	Revisiones de Código	1
3.2.1	Realizar revisiones de código	1
3.3	Control de Versiones	0.5 (distribuido en las demás fases)
Pruebas		
4.1	Pruebas de Integración	1
4.1.1	Probar la integración entre componentes	1
4.2	Pruebas de Sistema	0.5
4.2.1	Probar el sistema completo	0.5
4.3	Pruebas de Aceptación del Usuario	0.5
4.3.1	Probar el sistema con usuarios finales	0.5
Despliegue		
5.1	Preparación del Entorno de Producción	0.5
5.2	Despliegue del Software	0.25
5.3	Pruebas de Humo	0.25

- **Diseño de Arquitectura**: Se estima un esfuerzo de 8 semanas para diseñar la arquitectura del sistema, incluyendo la definición de modelos, patrones de diseño y experimentos de arquitectura.

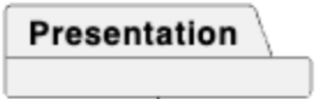
- **Desarrollo del Prototipo**: Se estima un esfuerzo adicional de 8 semanas para desarrollar el prototipo que debe incluir las funcionalidades clave definidas en el plan del proyecto.

Modelos Arquitectónicos

Convenciones en la utilización de los modelos:

Los diagramas descritos posteriormente fueron realizados siguiendo la convención UML 2.0. A continuación se deja la descripción de cada componente utilizado.

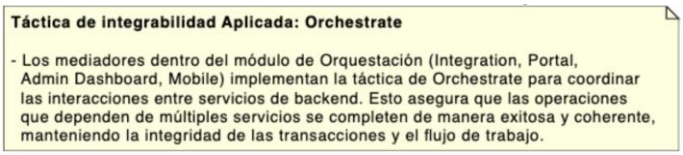
Paquete: Su función es agrupar elementos del modelo relacionados entre sí. En este caso, cada paquete representa un conjunto de clases, interfaces y otros elementos que colaboran para proporcionar una funcionalidad específica dentro del sistema.



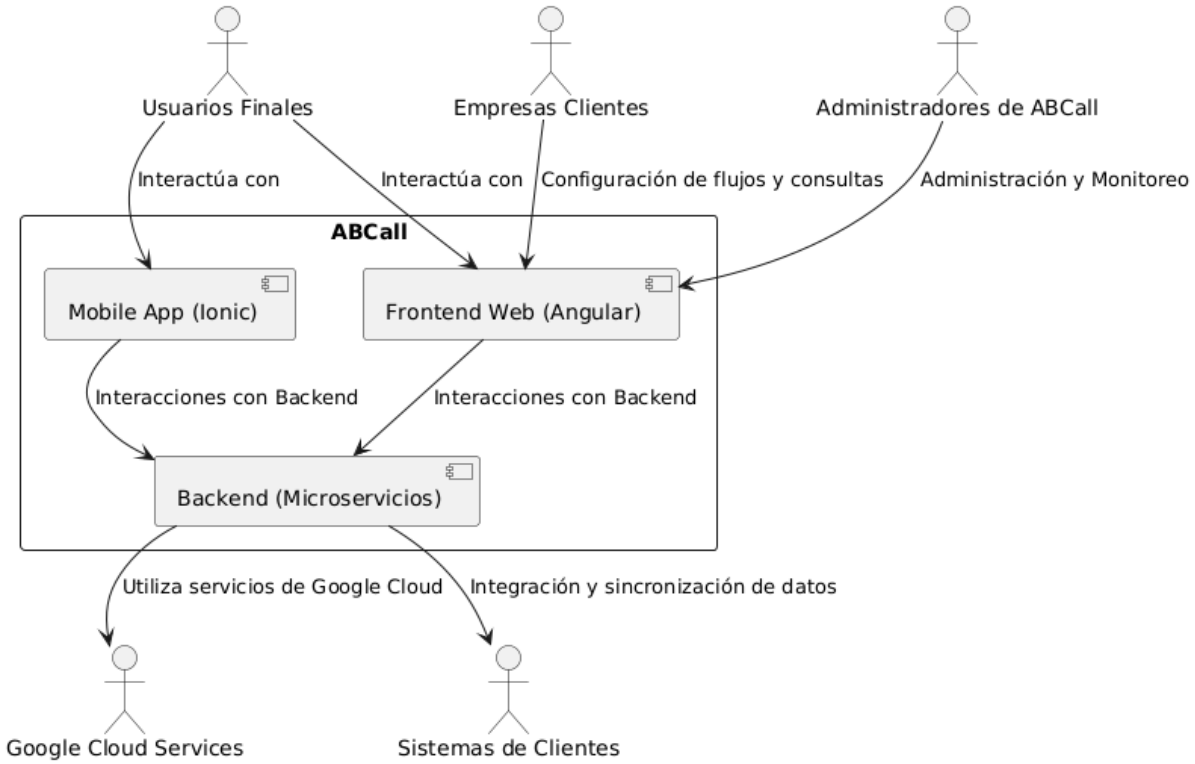
Componente: Representa una parte modular y reemplazable de un sistema que encapsula su contenido y cuya manifestación es reemplazable dentro de su entorno.

Asociación: Representa una relación estructural entre dos o más clasificadores, como clases, interfaces o componentes.

Nota: Tarjeta usada con el fin de dar una explicación acerca de algo relacionado con algún elemento del modelo o del modelo en general.



Modelo de Contexto:

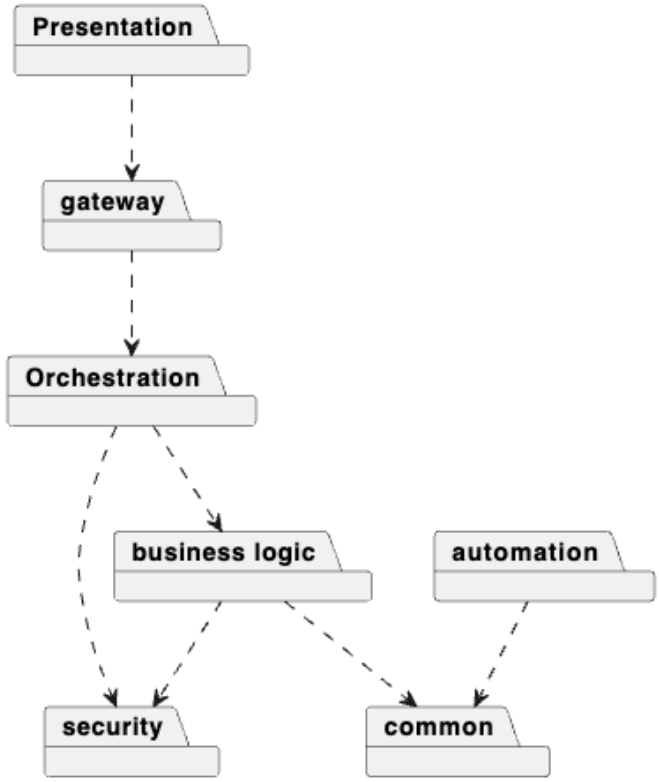


Punto de vista de Desarrollo:

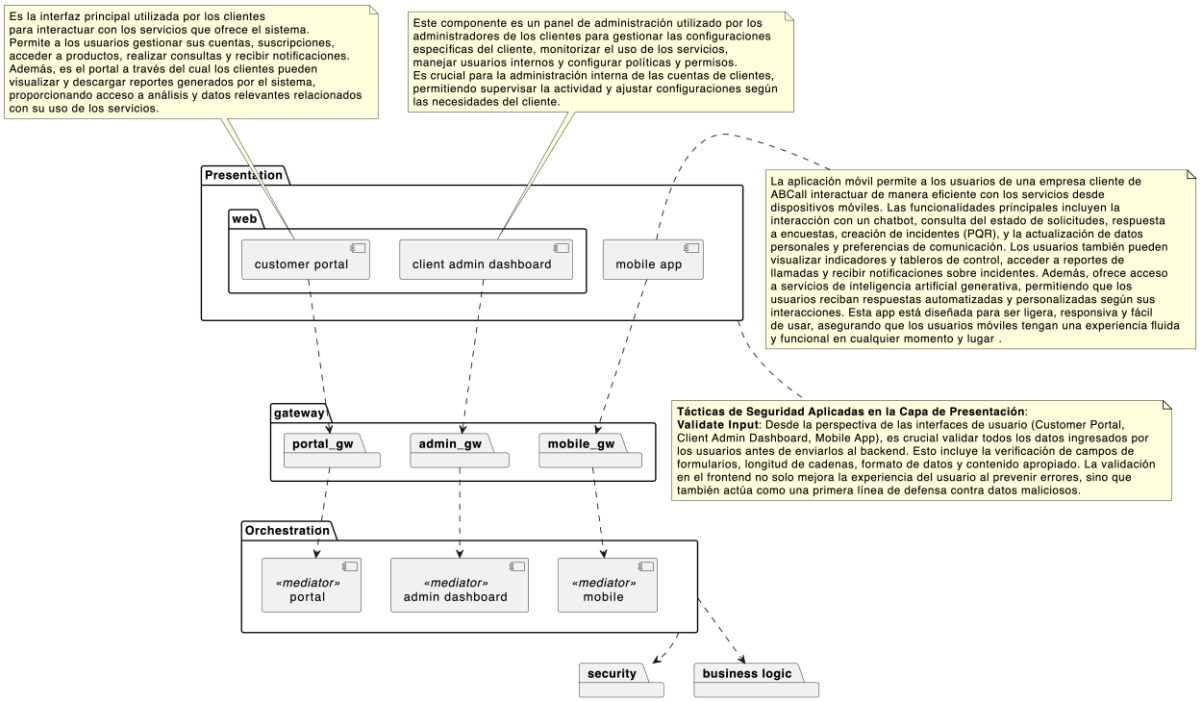
A continuación, se detallan los modelos aplicables al punto de vista de desarrollo del diseño detallado de la solución:

Modelo de componente para la vista de Niveles:

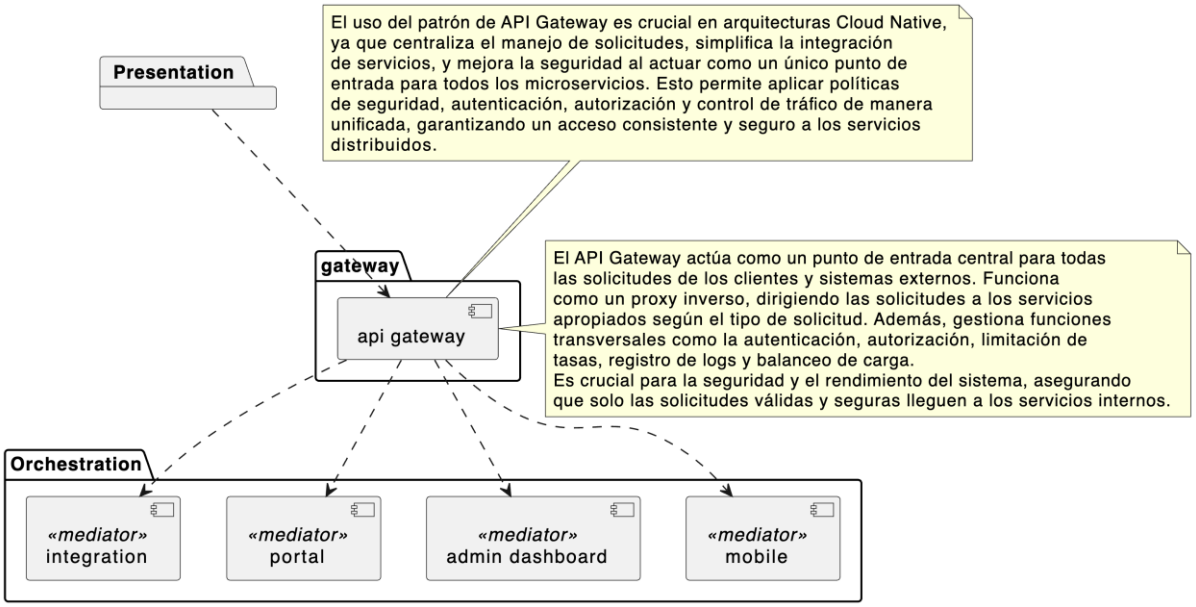
Contiene una vista global de los n-niveles planteados para la distribución de responsabilidades de la solución.



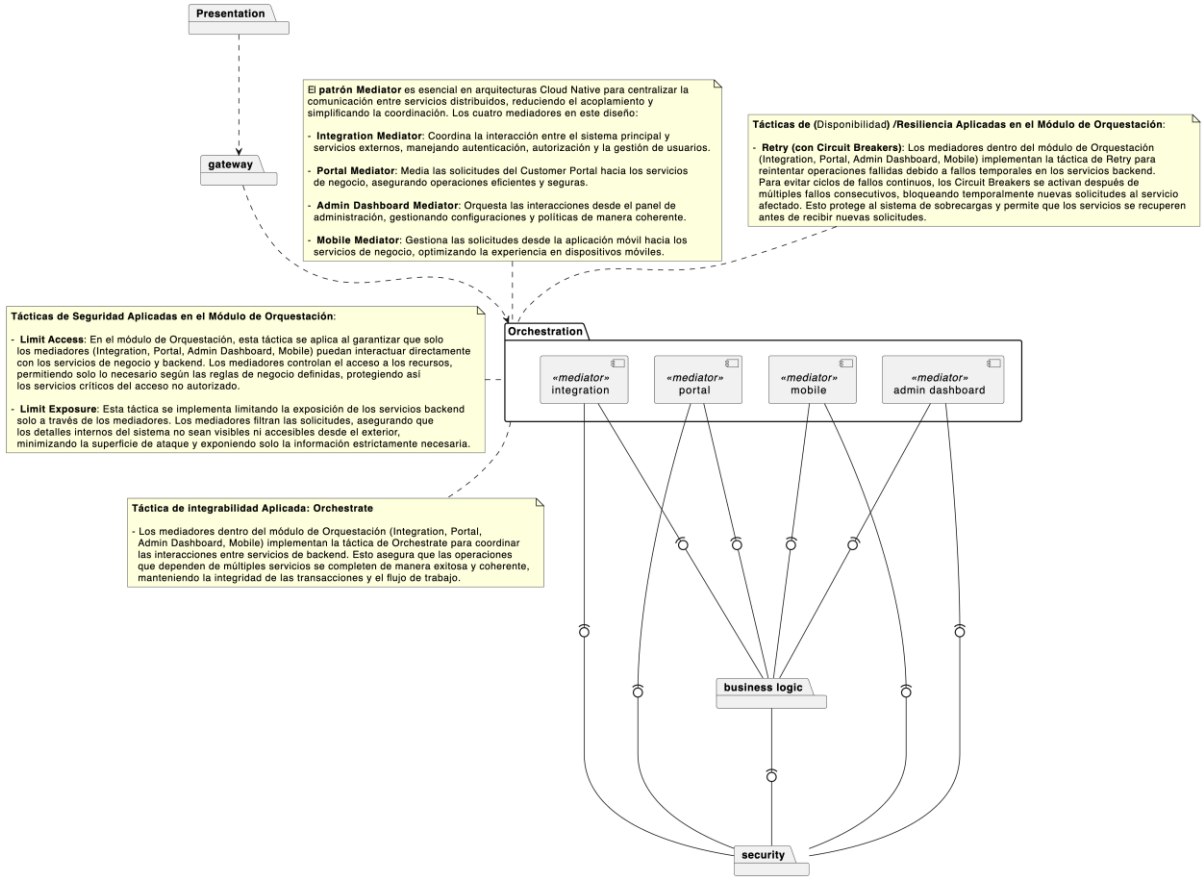
Modelo de componentes para la vista de presentación:



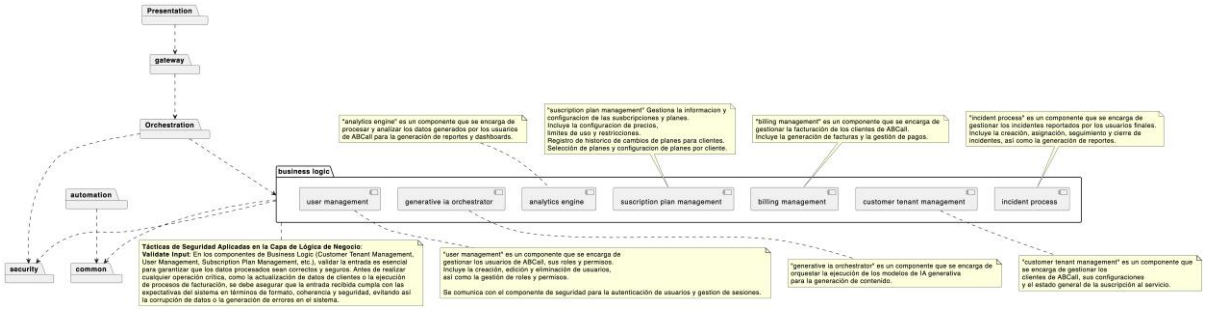
Modelo de componentes para la vista del Gateway:



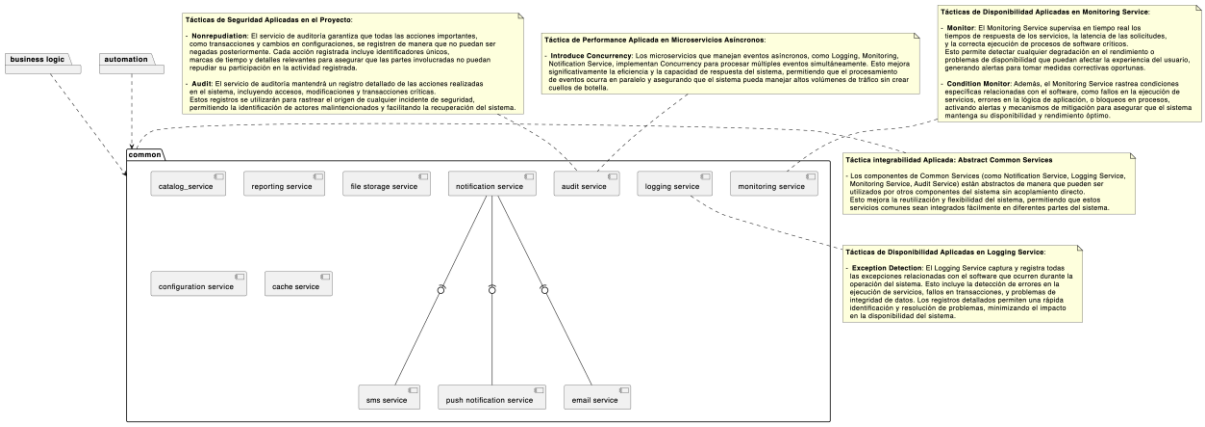
Modelo de componentes para la vista del Orchestrator:



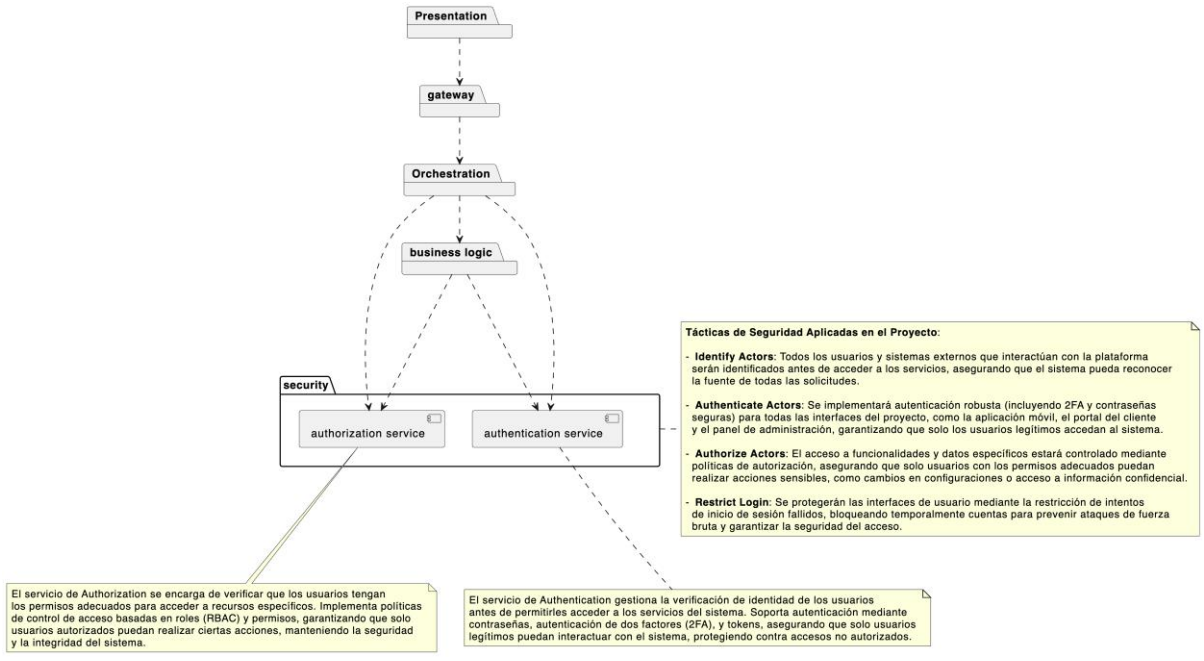
Modelo de componentes para la vista del Business Logic:



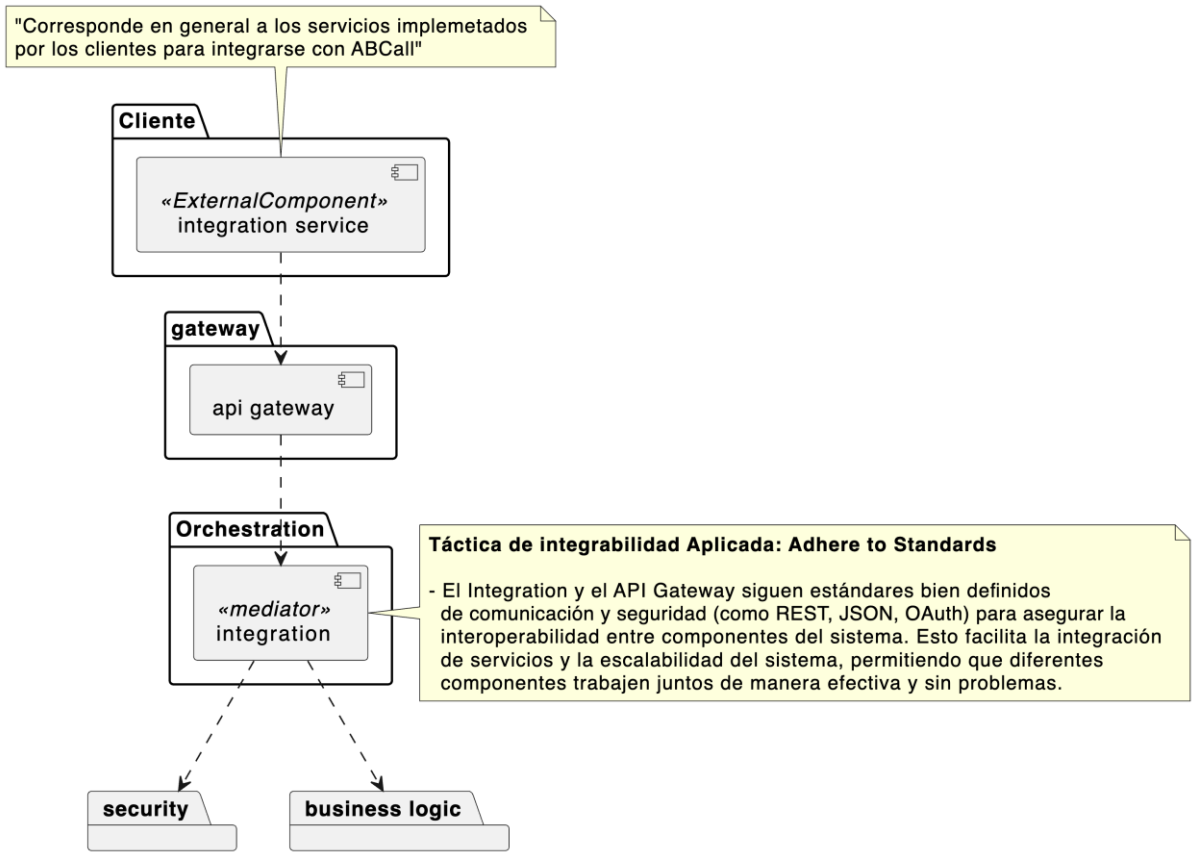
Modelo de componentes para la vista de Commons:



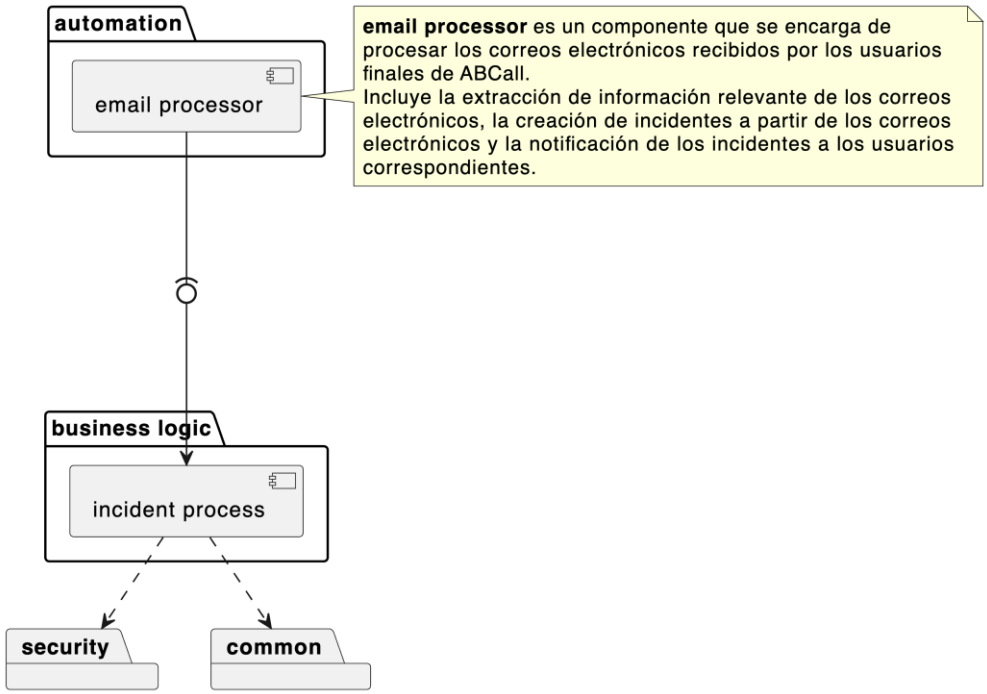
Modelo de componentes para la vista de Security:



Modelo de componentes para la vista del Integration:



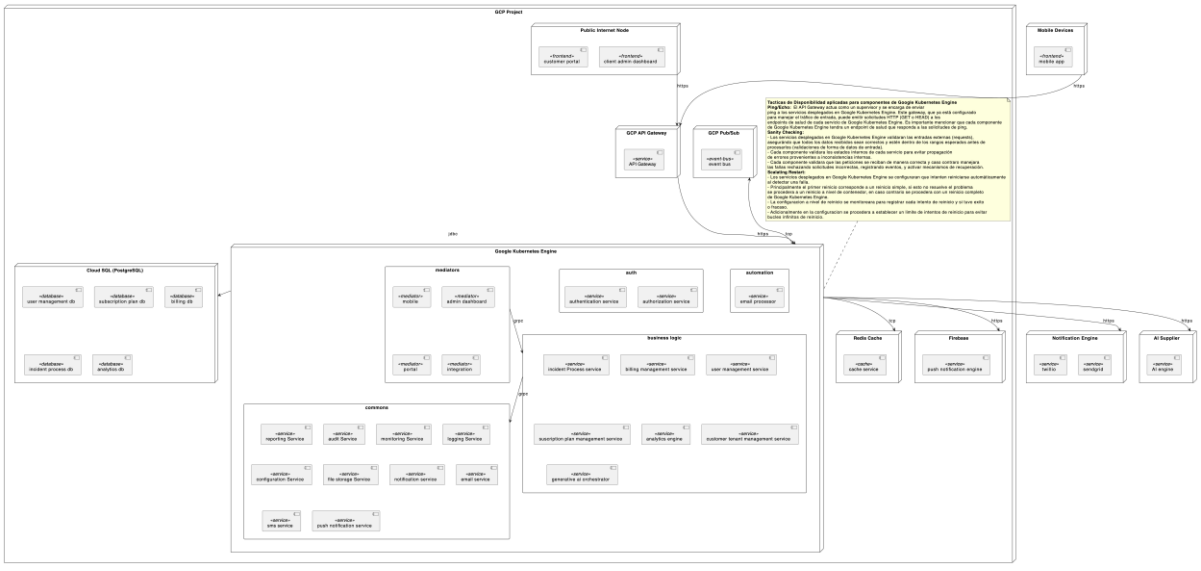
Modelo de componentes para la vista de Automation:



Modelo de Despliegue:

Para una mejor visualización del modelo de componentes se anexa la url:

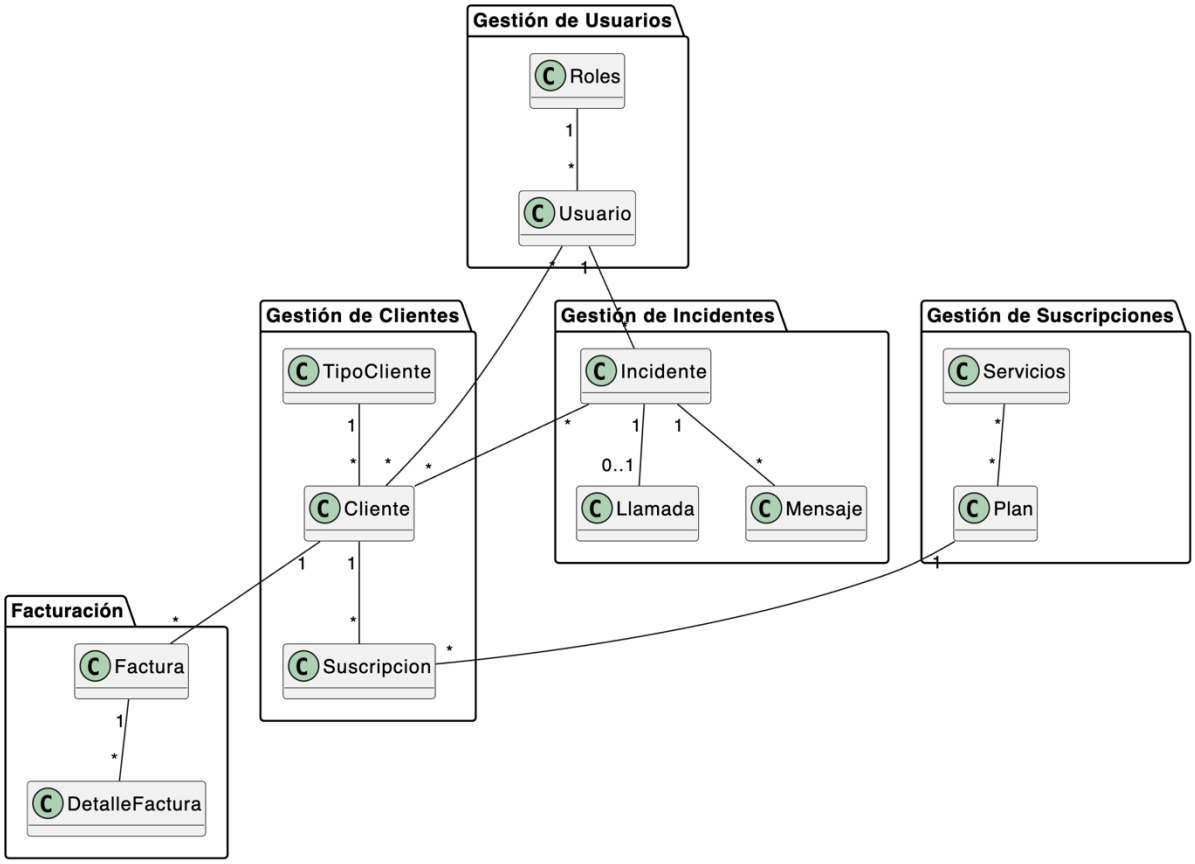
https://raw.githubusercontent.com/MISO-Proyecto-Final/Entregas-Proyecto-Final-G2/main/diagramas-arquitectura/ModeloDespliegue.png



Modelo de Dominio:

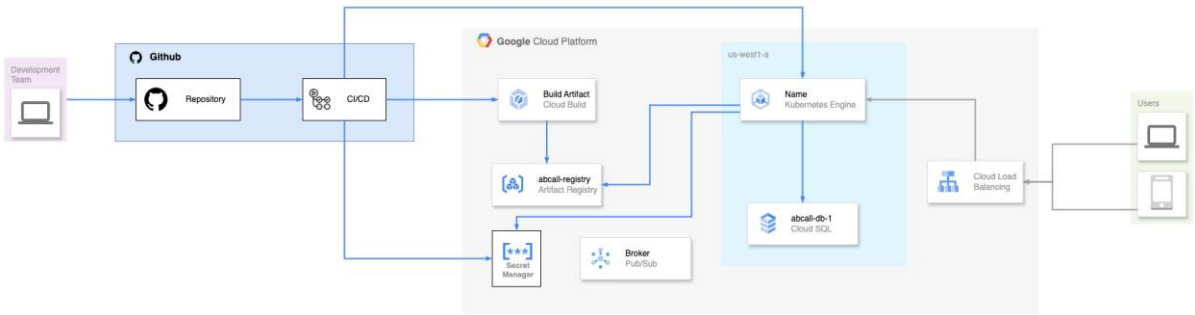
Para una mejor visualización del modelo de componentes se anexa la url:

https://github.com/MISO-Proyecto-Final/Entregas-Proyecto-Final-G2/blob/main/diagramas-arquitectura/Modelo_Dominio_ABCall.png



Vista Cloud

Para una mejor visualización del modelo de servicios cloud a utilizar se anexa la url: https://github.com/MISO-Proyecto-Final/Entregas-Proyecto-Final-G2/blob/main/Semana%205/Vista%20Cloud%20ABCall%20GCP.png



Decripcion detallada de componentes del sistema, tácticas y decisiones de arquitectura:

Este apartado proporciona una descripción de los componentes clave que conforman el sistema de ABCall, junto con las tácticas de diseño y las decisiones de arquitectura que sustentan su construcción. La arquitectura de software es fundamental para garantizar que el sistema no solo cumpla con sus objetivos funcionales, sino que también responda eficazmente a los desafíos de seguridad, disponibilidad, integrabilidad y rendimiento.

Cada componente ha sido diseñado con un enfoque en la modularidad y la escalabilidad, permitiendo que el sistema se adapte a cambios y expansiones futuras sin comprometer su estabilidad. Además, se han implementado tácticas de arquitectura específicas para abordar los requisitos no funcionales, garantizando que el sistema sea resiliente frente a fallos, seguro contra amenazas externas, y capaz de manejar eficientemente las demandas operativas y de usuario.

A lo largo de esta descripción, se detallan los distintos paquetes, componentes y sus relaciones, explicando cómo cada elemento del sistema contribuye al funcionamiento general de ABCall. Se hace énfasis en cómo las tácticas de arquitectura, como la validación de entradas, el control de acceso, la auditoría y la orquestación de servicios, se aplican para cumplir con los objetivos del sistema y asegurar su operatividad en un entorno complejo y dinámico. Esta exploración detallada proporciona una visión clara y comprensible de las decisiones arquitectónicas que aseguran que ABCall sea un sistema robusto, eficiente y preparado para enfrentar los desafíos del entorno digital actual.

Presentation Layer

La **Presentation Layer** es la capa responsable de proporcionar interfaces directas para los usuarios finales del sistema, permitiendo la interacción y el acceso a los diferentes servicios y funcionalidades que ABCall ofrece. Esta capa incluye varios componentes clave que desempeñan un papel fundamental en la experiencia del usuario, asegurando que las interacciones sean eficientes, seguras y fáciles de usar.

Customer Portal

El **Customer Portal** es la interfaz principal a través de la cual los clientes de ABCall acceden a los servicios ofrecidos. Este portal está diseñado para permitir a los usuarios gestionar todos los aspectos de sus cuentas y suscripciones. Los usuarios pueden acceder a productos, realizar consultas, recibir notificaciones, y lo más importante, visualizar y descargar reportes generados por el sistema. Estos reportes proporcionan acceso a análisis y datos relevantes relacionados con el uso de los servicios, lo que permite a los clientes tomar decisiones informadas sobre su suscripción y el uso de los servicios.

El **Customer Portal** no solo es un punto de acceso crítico para los servicios, sino que también juega un papel vital en la validación de entradas. La táctica de **Validate Input** se implementa aquí para garantizar que todos los datos introducidos por los usuarios sean verificados antes de ser enviados al backend. Esto incluye la verificación de la

longitud de las cadenas, el formato de los datos y el contenido adecuado. Esta validación inicial no solo mejora la experiencia del usuario al prevenir errores, sino que también actúa como una primera línea de defensa contra posibles datos maliciosos, protegiendo la integridad del sistema.

Client Admin Dashboard

El **Client Admin Dashboard** es un panel de control diseñado específicamente para los administradores de los clientes, quienes son responsables de gestionar la configuración y supervisión de las cuentas dentro del sistema. Este componente permite a los administradores monitorizar el uso de los servicios, manejar usuarios internos, y configurar políticas y permisos.

El **Client Admin Dashboard** es crucial para la administración interna de las cuentas de clientes, permitiendo a los administradores supervisar la actividad y ajustar configuraciones según las necesidades del cliente. La táctica de **Validate Input** también es fundamental en este componente, asegurando que todas las configuraciones y políticas establecidas sean correctas y seguras antes de ser implementadas en el sistema. Esto no solo garantiza la coherencia operativa, sino que también previene posibles problemas de seguridad derivados de configuraciones incorrectas.

Mobile App

La **Mobile App** es una aplicación diseñada para dispositivos móviles que permite a los usuarios de una empresa cliente interactuar con los servicios de ABCall desde cualquier lugar y en cualquier momento. La aplicación móvil ofrece una serie de funcionalidades esenciales, incluyendo la interacción con un chatbot, la consulta del estado de solicitudes, la respuesta a encuestas, la creación de incidentes (PQR), y la actualización de datos personales y preferencias de comunicación. Además, los usuarios pueden visualizar indicadores y tableros de control, acceder a reportes de llamadas, y recibir notificaciones sobre incidentes.

Una característica destacada de la **Mobile App** es el acceso a servicios de inteligencia artificial generativa, que permite a los usuarios recibir respuestas automatizadas y personalizadas según sus interacciones. Esta aplicación está diseñada para ser ligera, responsiva y fácil de usar, asegurando que los usuarios móviles tengan una experiencia fluida y funcional en todo momento. Como en los otros componentes de la capa de presentación, la táctica de **Validate Input** es esencial para garantizar que todas las entradas de los usuarios sean validadas antes de ser procesadas, lo que protege la seguridad y la funcionalidad de la aplicación móvil.

Integration Service

El **Integration Service** es un componente externo que proporciona servicios diseñados para permitir a los clientes integrar sus propios sistemas con los de ABCall. Este servicio es fundamental para asegurar que los sistemas de los clientes puedan comunicarse eficazmente con los servicios de ABCall, permitiendo una integración fluida y eficiente.

En términos de arquitectura, el **Integration Service** sigue la táctica de **Adhere to Standards**, lo que significa que utiliza estándares de comunicación y seguridad bien definidos, como REST, JSON, y OAuth, para asegurar la interoperabilidad entre los sistemas del cliente y los servicios de ABCall. Esta adhesión a los estándares es crucial para facilitar la integración y escalabilidad del sistema, permitiendo que diferentes componentes y sistemas trabajen juntos de manera efectiva y sin problemas.

Gateway Layer

La **Gateway Layer** actúa como el punto de entrada central para todas las solicitudes del sistema. Esta capa es fundamental para gestionar y dirigir las solicitudes entrantes a los servicios backend apropiados, asegurando que cada solicitud sea manejada de manera eficiente y segura. El componente principal de esta capa es el **API Gateway**, que desempeña un papel crucial en la arquitectura del sistema.

API Gateway

El **API Gateway** es un componente crítico que actúa como un proxy inverso, manejando las solicitudes que provienen tanto de clientes externos como de sistemas internos. Este gateway no solo dirige las solicitudes a los servicios backend adecuados, sino que también gestiona varias funciones transversales esenciales para la seguridad y el rendimiento del sistema. Estas funciones incluyen la autenticación y autorización, la limitación de tasas (rate limiting), y el registro de logs.

Una de las principales razones para utilizar un **API Gateway** en arquitecturas **Cloud Native** es su capacidad para centralizar el manejo de solicitudes, simplificando la integración de servicios y mejorando la seguridad. El **API Gateway** actúa como un único punto de entrada para todos los microservicios, lo que permite aplicar políticas de seguridad, autenticación y autorización de manera unificada. Además, el **API Gateway** implementa **Rate Limiting** para controlar la cantidad de solicitudes que se procesan en un período de tiempo determinado. Esto es crucial para evitar la

sobrecarga del sistema, asegurando un rendimiento constante y protegiendo los servicios backend de picos de tráfico inesperados.

El uso de la táctica **Use an Intermediary** es evidente en la función del **API Gateway**, ya que este actúa como un intermediario entre la capa de presentación y los servicios backend. Al reducir el acoplamiento entre estas capas, el **API Gateway** facilita la escalabilidad y la flexibilidad del sistema, permitiendo que los componentes de frontend y backend evolucionen de manera independiente sin afectar la integridad general del sistema.

Orchestration Layer

La **Orchestration Layer** es responsable de coordinar las interacciones entre los diferentes servicios backend, asegurando que las operaciones distribuidas se realicen de manera coherente y eficiente. Esta capa es esencial para manejar transacciones complejas en un entorno distribuido y para garantizar que todas las partes del sistema trabajen juntas de manera armoniosa.

Integration Mediator

El **Integration Mediator** es un componente fundamental dentro de la **Orchestration Layer** que coordina las interacciones entre el sistema principal y los servicios externos. Este mediador se encarga de manejar la autenticación, la autorización y la gestión de usuarios en un entorno distribuido.

La táctica **Orchestrate** se implementa en el **Integration Mediator** para asegurar que todas las operaciones distribuidas se ejecuten de manera coherente y eficiente. Además, el **Integration Mediator** utiliza **Sagas** para gestionar transacciones distribuidas. Cada paso de la Saga es una transacción independiente, con su correspondiente operación de compensación (Rollback) en caso de fallo. Esto asegura que el sistema mantenga su consistencia incluso en situaciones donde una parte de la transacción falla.

La táctica **Retry**, combinada con el uso de **Circuit Breakers**, también es implementada por el **Integration Mediator**. Cuando una operación falla debido a un fallo temporal en los servicios backend, el mediador reintenta la operación varias veces antes de activar un **Circuit Breaker**. Este mecanismo protege al sistema de sobrecargas y permite que los servicios se recuperen antes de recibir nuevas solicitudes.

Portal Mediator

El **Portal Mediator** es responsable de mediar las solicitudes provenientes del **Customer Portal** hacia los servicios de negocio, asegurando que las operaciones iniciadas en el portal se ejecuten correctamente en los sistemas backend.

En este componente, la táctica **Limit Access** se aplica para controlar qué recursos backend están disponibles para los usuarios del **Customer Portal**. El **Portal Mediator** filtra las solicitudes y aplica políticas de acceso que garantizan que solo las solicitudes autorizadas lleguen a los servicios backend. Además, la táctica **Limit Exposure** se implementa limitando la exposición de los detalles internos del sistema, asegurando que solo la información estrictamente necesaria sea visible para los usuarios.

Admin Dashboard Mediator

El **Admin Dashboard Mediator** orquesta las interacciones iniciadas desde el panel de administración de clientes, manejando configuraciones, políticas y la supervisión de actividades. Este mediador es crucial para asegurar que la administración de las cuentas de clientes se realice de manera coherente y segura.

Al igual que otros mediadores en esta capa, el **Admin Dashboard Mediator** implementa la táctica **Orchestrate** para coordinar las operaciones distribuidas de manera eficiente. Este componente también asegura que las configuraciones y políticas administrativas se apliquen correctamente en todo el sistema, garantizando la integridad y la coherencia de las operaciones administrativas.

Mobile Mediator

Mobile Mediator (continuación)

El **Mobile Mediator** también implementa **Rate Limiting** para controlar la cantidad de solicitudes que pueden ser procesadas en un período de tiempo determinado, lo que es particularmente importante para proteger los servicios backend de sobrecargas causadas por un aumento repentino en el tráfico de usuarios móviles. Esta táctica asegura que el sistema mantenga un rendimiento constante, incluso durante picos de demanda, protegiendo la disponibilidad y la capacidad de respuesta del sistema.

Además, el **Mobile Mediator** aplica la táctica **Concurrency** para manejar eventos asincrónicos de manera eficiente. Esto es especialmente relevante en entornos móviles, donde la capacidad de procesar múltiples eventos simultáneamente es crucial para ofrecer una experiencia de usuario fluida y responsiva. Implementar

concurrencia permite que el sistema maneje varios procesos en paralelo, mejorando significativamente la eficiencia y reduciendo los tiempos de respuesta.

Security Layer

La **Security Layer** es esencial para proteger los recursos del sistema, asegurando que solo los usuarios y sistemas autorizados puedan acceder a la información y realizar operaciones dentro del entorno de ABCall. Esta capa implementa varios servicios que trabajan en conjunto para proporcionar un entorno seguro, donde cada acceso y operación se controla y verifica rigurosamente.

Authentication Service

El **Authentication Service** es el componente encargado de verificar la identidad de los usuarios antes de permitirles el acceso al sistema. Este servicio utiliza mecanismos como contraseñas y tokens de seguridad para garantizar que solo los usuarios legítimos puedan acceder a los servicios ofrecidos por ABCall.

La táctica **Authenticate Actors** se implementa en este componente para asegurarse de que cada solicitud que llega al sistema provenga de un usuario autenticado y verificado. Esta verificación es crítica para prevenir accesos no autorizados y proteger la integridad de los datos y servicios del sistema. La integración de este servicio con todos los mediadores de la **Orchestration Layer** asegura que ninguna operación pueda ser realizada sin una autenticación previa, lo que fortalece significativamente la seguridad general del sistema.

Authorization Service

El **Authorization Service** controla el acceso a los recursos del sistema mediante la implementación de políticas de control de acceso basadas en roles (RBAC) y permisos. Este servicio verifica que los usuarios no solo estén autenticados, sino que también tengan los permisos adecuados para acceder a recursos específicos o realizar operaciones sensibles.

La táctica **Authorize Actors** se aplica a través de este servicio para garantizar que solo los usuarios con los permisos correctos puedan realizar acciones que pueden afectar críticamente al sistema, como la modificación de configuraciones o el acceso a datos confidenciales. Este servicio también se integra con los mediadores en la **Orchestration Layer** para asegurar que todas las solicitudes se revisen y autoricen correctamente antes de que se permita cualquier operación.

Además, se implementa la táctica **Restrict Login** para proteger las interfaces de usuario contra ataques de fuerza bruta y otros intentos de acceso no autorizados. Esta táctica incluye la restricción de intentos fallidos de inicio de sesión y el bloqueo temporal de cuentas después de un número determinado de intentos incorrectos, lo que agrega una capa adicional de seguridad.

Business Logic Layer

La **Business Logic Layer** es donde reside la lógica central del negocio, encargándose de gestionar las operaciones críticas que permiten a ABCall ofrecer sus servicios a los clientes. Esta capa es responsable de la gestión de clientes, planes de suscripción, facturación, y otros procesos fundamentales, asegurando que el sistema funcione de manera coherente y eficiente.

Customer Tenant Management

El componente **Customer Tenant Management** es responsable de gestionar la configuración y el estado de las suscripciones de los clientes. Este componente maneja todos los aspectos relacionados con las cuentas de los clientes, incluyendo la creación, actualización, y eliminación de cuentas, así como la gestión de sus datos y configuraciones.

En este componente, la táctica **Validate Input** es crucial para garantizar que todos los datos que se procesan sean correctos y seguros. Antes de realizar cualquier operación, como la actualización de los datos de un cliente, el sistema valida que la entrada cumpla con los requisitos de formato, coherencia, y seguridad. Esto ayuda a prevenir errores y garantiza que los datos almacenados sean siempre confiables.

Además, el **Customer Tenant Management** implementa la táctica **Audit** para mantener un registro detallado de todas las acciones realizadas en las cuentas de los clientes. Este registro es esencial para garantizar la transparencia y para proporcionar una base para la auditoría y el cumplimiento normativo. Cualquier cambio en las configuraciones de los clientes, o cualquier operación crítica realizada, queda registrado para su revisión posterior.

De igual manera, el Customer Tenant Managemente implementa la Táctica de Separación de Entidades, esta se refiere a la división lógica o física de los recursos para evitar el acceso no autorizado entre diferentes partes del sistema. En un contexto de multitenancy, esta táctica es crítica para aislar los datos y procesos de cada tenant, de

modo que los clientes (o tenants) no puedan acceder a la información de otros. Esto puede lograrse mediante diferentes enfoques, como:

- Separar las bases de datos de cada tenant, manteniendo una base de datos por cliente.
- Aislar la lógica de procesamiento a través de contenedores o máquinas virtuales.

En un patrón de multitenancy, la separación de entidades asegura que, aunque todos los tenants compartan recursos de infraestructura, sus datos y procesos estén completamente aislados.

User Management

El componente **User Management** se encarga de la gestión de los usuarios dentro del sistema, incluyendo la creación, actualización y eliminación de cuentas de usuario. Este componente es fundamental para asegurar que cada usuario tenga acceso a las funcionalidades y recursos adecuados según su rol y permisos.

Similar al **Customer Tenant Management**, el **User Management** también aplica la táctica **Validate Input** para garantizar que todos los datos de los usuarios sean validados antes de ser procesados. Esto incluye la validación de las credenciales de los usuarios y la coherencia de los datos asociados a sus cuentas. La implementación de **Audit** en este componente asegura que todas las operaciones relacionadas con la gestión de usuarios sean registradas, lo que permite un control detallado y la capacidad de auditar cualquier cambio realizado en las cuentas de usuario.

Subscription Plan Management

El componente **Subscription Plan Management** es responsable de gestionar todos los aspectos relacionados con las suscripciones y planes que ABCall ofrece a sus clientes. Este componente maneja la configuración de precios, límites de uso, restricciones, y el historial de cambios de planes para los clientes. También permite la selección y configuración de planes personalizados para cada cliente, asegurando que las ofertas de ABCall se adapten a las necesidades específicas de cada usuario.

En este contexto, la táctica **Validate Input** se aplica para asegurar que todas las configuraciones de planes de suscripción sean correctas y seguras antes de ser implementadas. Este proceso de validación ayuda a prevenir errores que podrían afectar la facturación o el acceso de los clientes a los servicios. Además, el **Subscription Plan Management** implementa **Audit** para registrar todas las modificaciones realizadas a los planes de suscripción. Estos registros son

fundamentales para garantizar la transparencia en la gestión de suscripciones y para proporcionar evidencia en caso de disputas o auditorías.

Billing Management

El componente **Billing Management** se encarga de todos los procesos de facturación dentro del sistema. Esto incluye la generación de facturas, la gestión de pagos, y la aplicación de cargos basados en el uso de los servicios por parte de los clientes. Este componente es crucial para asegurar que ABCall maneje sus ingresos de manera precisa y eficiente.

Para garantizar la exactitud en la facturación, el **Billing Management** aplica la táctica **Validate Input**, verificando que todos los datos financieros sean correctos antes de procesar cualquier operación. Esto incluye la validación de montos, fechas, y la coherencia de los registros de uso. Además, la táctica **Audit** se implementa para mantener un registro detallado de todas las transacciones financieras. Estos registros son esenciales para el cumplimiento normativo y para asegurar que el proceso de facturación sea transparente y rastreable.

Incident Process

El **Incident Process** es el componente encargado de gestionar y resolver los incidentes reportados por los usuarios, como problemas técnicos o consultas sobre el servicio. Este componente se comunica con otros servicios del sistema, como **Notification Service** y **Logging Service**, para manejar la notificación y el seguimiento de incidentes.

La táctica **Audit** es fundamental en el **Incident Process**, ya que asegura que todos los incidentes y las acciones tomadas para resolverlos sean registrados de manera detallada. Esto permite un seguimiento completo de cada incidente, facilitando la revisión posterior y asegurando que se tomen las medidas correctivas adecuadas. La capacidad de auditar los incidentes también es crucial para identificar patrones recurrentes y mejorar la calidad del servicio.

Generative IA Orchestrator

El **Generative IA Orchestrator** es un componente avanzado que coordina el uso de la inteligencia artificial generativa para ofrecer respuestas automatizadas y personalizadas a los usuarios. Este componente utiliza algoritmos de IA para analizar las interacciones de los usuarios y generar contenido que responda de manera efectiva a sus necesidades.

La táctica **Validate Input** es particularmente importante en el **Generative IA Orchestrator**, ya que asegura que todas las solicitudes procesadas por la IA sean apropiadas y válidas antes de su procesamiento. Esto evita respuestas incorrectas o inadecuadas que podrían afectar la experiencia del usuario. Además, el **Generative IA Orchestrator** implementa **Audit** para registrar el uso de la IA, asegurando que todas las interacciones y decisiones tomadas por el sistema sean transparentes y rastreables.

Analytics Engine

El **Analytics Engine** es el componente encargado de procesar y analizar grandes volúmenes de datos para ofrecer insights y reportes que ayuden en la toma de decisiones estratégicas. Este componente es fundamental para extraer valor de los datos recopilados por el sistema y para proporcionar análisis que guíen el desarrollo futuro de ABCall.

Antes de analizar los datos, el **Analytics Engine** aplica la táctica **Validate Input** para garantizar que los datos sean precisos y coherentes. Esta validación es esencial para evitar errores en los análisis que podrían conducir a conclusiones incorrectas. Además, el **Analytics Engine** implementa **Audit** para mantener un registro detallado de los procesos analíticos y de las conclusiones derivadas. Esto permite rastrear cómo se han llegado a ciertas conclusiones y proporciona la capacidad de auditar los análisis para asegurar su exactitud y confiabilidad. La capacidad de auditar los resultados analíticos es especialmente importante en un entorno empresarial, donde las decisiones estratégicas a menudo se basan en los insights derivados de los datos analizados.

Common Services

La **Common Services Layer** proporciona una serie de servicios reutilizables y compartidos que son esenciales para el funcionamiento de diversas partes del sistema. Estos servicios están diseñados para ser abstractos y desacoplados, lo que permite su integración en múltiples componentes del sistema sin generar dependencia directa. Esto mejora la flexibilidad y escalabilidad del sistema al permitir que los servicios comunes sean utilizados de manera eficiente en diferentes contextos.

Catalog Service

El **Catalog Service** gestiona el catálogo de productos y servicios ofrecidos por ABCall, asegurando que la información esté siempre actualizada y accesible para otros componentes del sistema. Este servicio es crucial para la operación de varios

componentes de la capa de negocio, ya que proporciona datos consistentes y centralizados sobre los productos y servicios disponibles.

La táctica **Abstract Common Services** se aplica en el **Catalog Service** para garantizar que este servicio sea reutilizable por múltiples componentes del sistema sin acoplamiento directo. Esto permite que el **Catalog Service** sea integrado fácilmente en diferentes partes del sistema, proporcionando una fuente única de verdad para la información de catálogo y mejorando la coherencia en todo el sistema.

Notification Service

El **Notification Service** gestiona la entrega de notificaciones a los usuarios a través de diferentes canales, como correo electrónico, SMS, y notificaciones push. Este servicio es fundamental para mantener a los usuarios informados sobre eventos y actualizaciones importantes, así como para manejar la comunicación automatizada del sistema.

Para manejar eficientemente las notificaciones, el **Notification Service** implementa la táctica **Concurrency**, que permite procesar múltiples eventos de notificación de manera simultánea. Esto es crucial para mantener la capacidad de respuesta del sistema, especialmente en momentos de alta demanda, donde múltiples notificaciones pueden necesitar ser enviadas en un corto período de tiempo. La concurrencia asegura que el sistema pueda manejar altos volúmenes de tráfico sin crear cuellos de botella, lo que mejora significativamente la eficiencia y la capacidad de respuesta del servicio.

Logging Service

El **Logging Service** captura y registra eventos y excepciones que ocurren dentro del sistema, proporcionando un historial detallado de las operaciones realizadas. Este servicio es esencial para la supervisión, el diagnóstico y la auditoría del sistema, ya que proporciona información crucial sobre el comportamiento y la salud del sistema.

La táctica **Exception Detection** se implementa en el **Logging Service** para asegurar que todas las excepciones relacionadas con el software sean capturadas y registradas de manera adecuada. Esto incluye la detección de errores en la ejecución de servicios, fallos en transacciones, y problemas de integridad de datos. Los registros detallados generados por este servicio permiten una rápida identificación y resolución de problemas, minimizando el impacto en la disponibilidad y el rendimiento del sistema.

Monitoring Service

El **Monitoring Service** es responsable de supervisar el estado y rendimiento del sistema en tiempo real. Este servicio detecta problemas de disponibilidad y rendimiento, permitiendo que se tomen medidas correctivas antes de que afecten negativamente a los usuarios. La supervisión en tiempo real es crucial para mantener la alta disponibilidad del sistema y garantizar una experiencia de usuario satisfactoria.

La táctica **Monitor** se aplica en el **Monitoring Service** para supervisar continuamente los tiempos de respuesta de los servicios, la latencia de las solicitudes, y la correcta ejecución de procesos de software críticos. Esto permite detectar cualquier degradación en el rendimiento o problemas de disponibilidad que puedan afectar la experiencia del usuario. Cuando se detecta un problema, el sistema genera alertas para que los operadores puedan tomar medidas correctivas oportunas.

Además, el **Monitoring Service** implementa la táctica **Condition Monitor**, que rastrea condiciones específicas relacionadas con el software, como fallos en la ejecución de servicios, errores en la lógica de aplicación, o bloqueos en procesos. Este nivel de supervisión detallada es fundamental para mantener la disponibilidad y el rendimiento óptimo del sistema. Las alertas generadas por el **Condition Monitor** permiten que los operadores respondan rápidamente a problemas potenciales antes de que se conviertan en incidentes mayores.

Audit Service

El **Audit Service** es responsable de mantener un registro detallado de todas las acciones y transacciones importantes que ocurren dentro del sistema. Este servicio asegura que cada operación crítica, como accesos, modificaciones, y transacciones financieras, quede registrada de manera que no pueda ser negada posteriormente. Esto es esencial para garantizar la transparencia, la seguridad, y el cumplimiento normativo dentro del sistema.

La táctica **Nonrepudiation** se implementa en el **Audit Service** para garantizar que todas las acciones registradas incluyan identificadores únicos, marcas de tiempo, y detalles relevantes. Esto asegura que las partes involucradas en cualquier operación registrada no puedan repudiar su participación en la actividad registrada, lo que es crucial para la integridad y la seguridad del sistema.

Además, la táctica **Audit** se aplica en el **Audit Service** para mantener un registro completo de todas las acciones y transacciones realizadas en el sistema. Estos registros son esenciales para rastrear el origen de cualquier incidente de seguridad, permitiendo la identificación de actores malintencionados y facilitando la

recuperación del sistema en caso de incidentes. La capacidad de auditar las acciones y transacciones dentro del sistema es crucial para mantener la confianza de los usuarios y cumplir con las regulaciones de seguridad.

Automation Layer

La **Automation Layer** maneja procesos automatizados que son esenciales para el funcionamiento continuo del sistema. Estos procesos se ejecutan de manera autónoma, integrándose con otras capas del sistema para completar tareas específicas que optimizan la operación general del sistema.

Email Processor

El **Email Processor** es un componente de la **Automation Layer** que procesa correos electrónicos de manera automatizada, gestionando notificaciones y manejando incidentes reportados por los usuarios a través de correo electrónico. Este componente es crucial para asegurar que los incidentes reportados sean registrados y manejados eficientemente, sin necesidad de intervención manual.

El **Email Processor** aplica la táctica **Automate Repetitive Tasks** para optimizar la gestión de incidentes reportados por correo. Al automatizar este proceso, se mejora la eficiencia del sistema y se reduce la posibilidad de errores humanos, lo que asegura una gestión más rápida y precisa de los incidentes. Además, la integración del **Email Processor** con el **Incident Process** en la capa de lógica de negocio asegura que todos los incidentes sean gestionados de manera coherente, desde su reporte inicial hasta su resolución final.

Conclusión General diseño de componentes

Este diseño arquitectónico combina una serie de tácticas avanzadas de seguridad, integrabilidad, disponibilidad y rendimiento, asegurando que el sistema de ABCall sea robusto, seguro y escalable. Cada capa y componente está cuidadosamente diseñado para cumplir con los requisitos funcionales y no funcionales, asegurando que el sistema no solo cumpla con las expectativas actuales, sino que también esté preparado para adaptarse a futuras necesidades y desafíos. La implementación de estas tácticas, respaldada por una arquitectura bien definida y modular, garantiza que ABCall pueda ofrecer sus servicios de manera eficiente, segura y confiable, proporcionando una experiencia superior tanto para los usuarios como para los administradores del sistema.

Tácticas utilizadas en infraestructura Cloud Run

Dentro de la infraestructura propuesta para el proyecto, se plantea utilizar Cloud Run en GCP, donde se está implementando las siguientes tácticas:

Ping/Echo

El API Gateway actúa como un supervisor y se encarga de enviar ping a los servicios desplegados en Cloud Run. Este gateway, que ya está configurado para manejar el tráfico de entrada, puede emitir solicitudes HTTP (GET o HEAD) a los endpoints de salud de cada servicio de Cloud Run. Es importante mencionar que cada componente de Cloud Run tendrá un endpoint de salud que responda a las solicitudes de ping.

Sanity Checking

- Los servicios desplegados en Cloud Run validarán las entradas externas (requests), asegurándose de que todos los datos recibidos sean correctos y estén dentro de los rangos esperados antes de procesarlos (validaciones de forma de datos de entrada).
- Cada componente validará los estados internos de cada servicio para evitar propagación de errores provenientes a inconsistencias internas.
- Cada componente validará que las peticiones se reciban de manera correcta y en caso contrario manejará las fallas rechazando solicitudes incorrectas, registrando eventos, y activando mecanismos de recuperación.

Scalating Restart

- Los servicios desplegados en Cloud Run se configurarán que intenten reiniciarse automáticamente al detectar una falla.
- Principalmente el primer reinicio corresponde a un reinicio simple, si esto no resuelve el problema se procederá a un reinicio a nivel de contenedor, en caso contrario se procederá con un reinicio completo de Cloud Run.
- La configuración a nivel de reinicio se monitoreará para registrar cada intento de reinicio y si tuvo éxito o fracaso.
- Adicionalmente en la configuración se procederá a establecer un límite de intentos de reinicio para evitar bucles infinitos de reinicio.

Vista General de la Arquitectura, Decisiones cloud

Descripción General

Google Cloud fue elegido como la plataforma principal para este proyecto debido a su capacidad para ofrecer una infraestructura robusta y altamente escalable, que se adapta automáticamente a las necesidades cambiantes de la aplicación. La decisión se basó en la necesidad de garantizar un rendimiento óptimo y una alta disponibilidad, al tiempo que se minimiza la complejidad operativa. Además, el equipo cuenta con experiencia previa en el manejo de los servicios de Google Cloud, lo que facilita su implementación y optimización, reduciendo así los tiempos de desarrollo y permitiendo un enfoque más ágil en la entrega del proyecto.

En cuanto a la arquitectura del sistema, se optó por utilizar Google App Engine en su entorno flexible para el backend. Esta elección permite desplegar y escalar aplicaciones Node.js desarrolladas con NestJS sin la necesidad de gestionar servidores de forma manual. De esta manera, la infraestructura puede ajustarse automáticamente en función de la demanda, lo que asegura que la aplicación mantenga un alto rendimiento bajo diferentes cargas de trabajo. La combinación de estas características de Google Cloud con la experiencia del equipo crea un entorno ideal para el desarrollo y la operación eficiente de la solución propuesta.

Angular para Desarrollo Web: Angular es un framework de desarrollo web robusto y escalable que permite la creación de aplicaciones dinámicas y de alto rendimiento. Para el proyecto web, Angular proporcionará una arquitectura modular que facilita la integración de servicios, la reutilización de componentes y la gestión eficiente del estado de la aplicación. La elección de Angular garantiza la implementación de mejores prácticas en términos de mantenimiento, pruebas y optimización de la aplicación, todo esto desplegado de manera continua a través de Firebase Hosting, lo que asegura una alta disponibilidad y escalabilidad.

Ionic Angular para Desarrollo Móvil: Ionic Angular es una poderosa solución para el desarrollo de aplicaciones móviles multiplataforma, aprovechando las capacidades de Angular junto con el poder de la API de Ionic. Esto permitirá la creación de una aplicación móvil que no solo comparte gran parte de la lógica y componentes con la versión web, sino que también ofrece una experiencia de usuario nativa optimizada para dispositivos móviles. La distribución y actualización de la aplicación móvil se realizará a través de Google Play Store, asegurando que los usuarios puedan acceder a las versiones más recientes de la aplicación con un proceso de despliegue eficiente y seguro en la plataforma de Android.

La arquitectura del sistema también incorpora Firebase Authentication para gestionar la autenticación de usuarios. Esta elección se alinea con la decisión de utilizar Google Cloud, aprovechando la integración nativa y fluida entre Firebase y los otros servicios

en la nube de Google. Firebase Authentication proporciona un sistema seguro y escalable para la gestión de identidades, permitiendo a los usuarios autenticarse mediante diversos métodos, como correo electrónico, redes sociales y autenticación anónima, sin la necesidad de implementar y mantener un sistema de autenticación personalizado.

Al integrar Firebase Authentication, se facilita el manejo de la seguridad y la autenticación dentro de la aplicación, permitiendo al equipo enfocarse en otras áreas críticas del desarrollo. Además, esta solución es altamente compatible con el resto de la infraestructura en Google Cloud, lo que asegura una implementación cohesiva y eficiente.

El sistema también tiene la capacidad de implementar notificaciones a través de Firebase Cloud Messaging (FCM) si se considera necesario en etapas posteriores del desarrollo. FCM ofrece una solución robusta para el envío de mensajes y notificaciones push a los usuarios, permitiendo una interacción más dinámica y personalizada con la aplicación, tanto en entornos móviles como web.

Inicialmente, se evaluará la necesidad de estas notificaciones push en función de los requerimientos del proyecto y del comportamiento de los usuarios. Si se determina que las notificaciones pueden mejorar la experiencia del usuario o cumplir un objetivo estratégico, FCM se integrará en la arquitectura existente de manera eficiente, aprovechando la infraestructura ya establecida en Google Cloud. Esto permitirá mantener la flexibilidad del sistema y garantizar una implementación ágil en caso de que se decida avanzar con esta funcionalidad.

En cuanto al almacenamiento de datos, la arquitectura del sistema se ha diseñado para aprovechar diferentes servicios en función del tipo de datos que se maneje. Para los datos estructurados, se utilizará Cloud SQL con PostgreSQL, lo que proporciona una base de datos relacional completamente gestionada y escalable. Esto garantiza la integridad y consistencia de los datos, así como un rendimiento óptimo para consultas complejas y transacciones.

Por otro lado, los datos no estructurados se almacenarán en Cloud Firestore, una base de datos NoSQL que ofrece una escalabilidad automática y una gestión simplificada. Cloud Firestore es ideal para manejar datos flexibles que no encajan en un esquema relacional tradicional, permitiendo una rápida recuperación y sincronización en tiempo real, especialmente útil para aplicaciones móviles y web que requieren una alta reactividad.

La combinación de Cloud SQL y Cloud Firestore en la arquitectura asegura que los datos se manejen de manera eficiente, aprovechando las fortalezas de cada servicio en

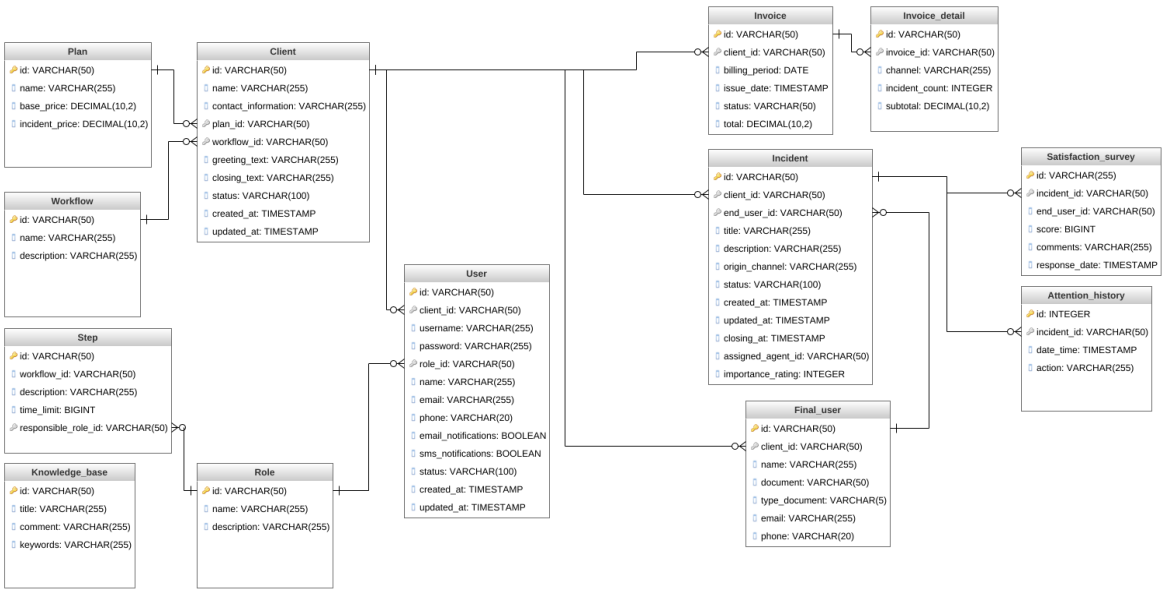
función de las necesidades específicas del sistema. Esta estrategia proporciona una solución de almacenamiento robusta y flexible, adaptada a los diferentes tipos de datos y patrones de uso que se esperan en la aplicación.

Para el manejo de archivos estáticos y contenido multimedia, el sistema utilizará Google Cloud Storage. Este servicio ofrece un almacenamiento escalable y altamente duradero para archivos como imágenes, videos, documentos y otros tipos de contenido estático que son fundamentales para la operación de la aplicación.

Cloud Storage es ideal para manejar grandes volúmenes de datos, ofreciendo características como el versionado de archivos, la gestión de permisos granulares y la integración con otros servicios de Google Cloud, lo que facilita la gestión y distribución de contenido. Además, permite un acceso rápido y seguro a los archivos desde cualquier parte del mundo, lo que es esencial para aplicaciones que sirven a usuarios distribuidos geográficamente.

Al integrar Cloud Storage en la arquitectura del sistema, se garantiza que los archivos multimedia y estáticos se gestionen de manera eficiente, con un alto rendimiento y confiabilidad, complementando el uso de Cloud SQL y Cloud Firestore para datos estructurados y no estructurados, respectivamente. Esto completa una solución de almacenamiento integral que cubre todas las necesidades del proyecto.

Vista de Información



Descripción General:

El diagrama de datos de ABCall revela la estructura de información esencial para construir un sistema robusto de atención al cliente. Modelando entidades como

clientes, planes, flujos de trabajo, incidentes y encuestas de satisfacción, el diagrama proporciona la base para automatizar la gestión de solicitudes, integrar diversos canales de comunicación, personalizar la experiencia del usuario y aplicar analítica avanzada para mejorar la calidad del servicio.

Entidades Principales:

- Client: Representa a un cliente de la empresa. Almacena información como nombre, información de contacto, plan suscrito y workflow asociado.
- Plan: Describe un plan de servicio con un precio base y un precio por incidente.
- Workflow: Define un flujo de trabajo para la atención al cliente, con un nombre y una descripción.
- Step: Representa un paso dentro de un workflow, con una descripción, tiempo límite y rol responsable.
- User: Representa a un usuario del sistema, ya sea un cliente, administrador o agente. Contiene información como nombre de usuario, contraseña, rol, información de contacto y preferencias de notificación.
- Role: Define los roles dentro del sistema, con un nombre y una descripción.
- Invoice: Representa una factura emitida a un cliente, con información sobre el período de facturación, fecha de emisión, estado y total.
- Invoice_detail: Detalla cada elemento de una factura, como el canal, número de incidentes y subtotal.
- Incident: Representa un incidente o solicitud de servicio reportado por un cliente. Incluye información como el usuario final, título, descripción, canal de origen, estado, agente asignado y nivel de importancia.
- Satisfaction_survey: Representa una encuesta de satisfacción enviada a un usuario final después de la resolución de un incidente.
- Attention_history: Registra el historial de atención de un incidente, con la fecha, hora y acción realizada.
- Knowledge_base: Representa una base de conocimiento con artículos y recursos para la atención al cliente.
- Final_user: Representa al usuario final que interactúa con el servicio de atención al cliente.

3. Componentes Principales

En este apartado se detallan algunos puntos a nivel de decisiones del Stack tecnológico, seleccionados por el equipo para abordar los requerimientos de la construcción del producto:

- **Frontend**:

    - **Web**: Desarrollado en **Angular** y desplegado en **Firebase Hosting**. La aplicación interactúa con el backend a través de APIs REST y GraphQL.
    - **Móvil**: Desarrollado en **IONIC Angular** y también desplegado en **Firebase Hosting** para aplicaciones móviles (iOS y Android). Comparte gran parte del código con la aplicación web.
    - **Autenticación**: Gestionada por **Firebase Authentication** que ofrece autenticación basada en OAuth2, con soporte para múltiples proveedores de identidad.
- **Backend**:
    - **Google Cloud Run**: Proporciona el entorno de ejecución para el backend construido con **NestJS**. Maneja automáticamente la escalabilidad y el balanceo de carga.
    - **API Gateway (Google)**: Punto de entrada para todas las solicitudes al backend. Gestiona la autenticación, autorización y la enrutación a los servicios internos.
    - **Tablero de Control**: Servicio responsable de consolidar y mostrar datos en tiempo real. Utiliza **WebSockets** para actualizaciones instantáneas.
    - **Servicio de Notificaciones**: Envía notificaciones a través de correo electrónico, SMS y push notifications usando **Firebase Cloud Messaging (FCM)** y **SendGrid** (para correos).
- **Bases de Datos y Almacenamiento**:
    - **Cloud SQL (PostgreSQL)**: Base de datos relacional para datos estructurados como incidentes y usuarios.
    - **Cloud Firestore**: Base de datos NoSQL para datos no estructurados, como logs de eventos y datos en tiempo real.
    - **Cloud Storage**: Almacenamiento de archivos estáticos y multimedia.
- **Integración y APIs**:
    - **APIs REST y GraphQL**: Implementadas en **NestJS** para la comunicación entre frontend y backend. REST se utiliza para operaciones estándar, mientras que GraphQL se usa para consultas más específicas y optimizadas.
- **Despliegue y Escalabilidad**:
    - **Google Cloud Run** Gestiona automáticamente la escalabilidad del backend.
    - **Cloud Load Balancing**: Distribuye el tráfico entrante entre las instancias de CloudRun.
    - **Cloud CDN**: Optimiza la entrega de contenido estático, mejorando la latencia para los usuarios finales.

4. Decisiones Arquitectónicas Clave A nivel del Stack Tecnologico

Adicional al detalle reportado a nivel de las decisiones arquitectónicas aplicadas en las tácticas de los diferentes componentes a continuación se detallan algunos puntos a nivel de tecnología relevantes para la construcción del producto:

- **Google App Engine**: Se seleccionó para evitar la complejidad de la gestión de infraestructura mientras se garantiza la escalabilidad automática y la integración con otros servicios de GCP.
- **Firebase para el Frontend**: Firebase Hosting y Authentication ofrecen una solución gestionada y segura para desplegar y autenticar usuarios en aplicaciones web y móviles.
- **Cloud SQL y Firestore**: La combinación de bases de datos SQL y NoSQL permite una gestión eficiente de datos estructurados y no estructurados, con alta disponibilidad y soporte transaccional.
- **SendGrid, Twillio y Firebase para Notificaciones**: Proporcionan una solución fiable y escalable para las notificaciones de usuario.