**[TripGO]**

**(SAD) Software Architecture Document**

**Versión 1.0**

**Identificación de Documento**

| **Identificación** | Proyecto de título “TripGO” |
| --- | --- |
| **Proyecto** | Itinerario Inteligente |
| **Versión** | 1.0 |

| **Documento mantenido por** | Felipe Romero, Alejandro Martinez y Natanael Huenullan |
| --- | --- |
| **Fecha de última revisión** | 23/9/2025 |
| **Fecha de próxima revisión** | 03/10/2025 |

| **Documento aprobado por** | Juan Gana |
| --- | --- |
| **Fecha de última aprobación** |  |

**Historia de Revisiones**

| **Fecha** | **Versión** | **Descripción** | **Autor** |
| --- | --- | --- | --- |
| 14/9/25 | 1.0 | Muestra de la arquitectura del sistema móvil | Alejandro Martinez |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Tabla de Contenidos**

[**Introducción 3**](#_heading=h.30j0zll)

[Contexto del Problema 3](#_heading=h.1fob9te)

[Propósito 3](#_heading=h.3znysh7)

[Ámbito 3](#_heading=h.2et92p0)

[Definiciones, acrónimos y abreviaciones 3](#_heading=h.tyjcwt)

[Referencias 3](#_heading=h.3dy6vkm)

[Resumen ejecutivo 4](#_heading=h.1t3h5sf)

[Representación 5](#_heading=h.4d34og8)

[**Metas y Restricciones de la Arquitectura 6**](#_heading=h.2s8eyo1)

[Metas de la arquitectura 6](#_heading=h.3rdcrjn)

[Restricciones de la Arquitectura 6](#_heading=h.26in1rg)

[Otros antecedentes y consideraciones 6](#_heading=h.lnxbz9)

[**Vista de Casos de Uso / Escenarios de Calidad 6**](#_heading=h.35nkun2)

[Modelo de Casos de Uso 7](#_heading=h.1ksv4uv)

[Especificación de Casos de Uso Relevantes 7](#_heading=h.44sinio)

[Especificación de los Escenarios de Calidad Relevantes 8](#_heading=h.2jxsxqh)

[**Vista Lógica 9**](#_heading=h.z337ya)

[Parte Estructural 9](#_heading=h.3j2qqm3)

[Parte Dinámica 11](#_heading=h.1y810tw)

[**Vista de Procesos 13**](#_heading=h.4i7ojhp)

[**Vista de Implementación 14**](#_heading=h.2xcytpi)

[**Vista de Despliegue 15**](#_heading=h.1ci93xb)

[**Decisiones de Diseño y Selección de Alternativas 16**](#_heading=h.u9b0xyd4mlu9)

[**Análisis de Reutilización 17**](#_heading=h.mt86njyuk8re)

# Introducción

## Contexto del Problema

Imagina que es sábado por la mañana. Tienes que ir al banco, comprar pintura en la ferretería, cortarte el pelo y pasar al supermercado. Normalmente pasarías un buen rato pensando en qué hacer primero, revisando horarios y rutas… y aun así probablemente terminarás llegando a un local cerrado o dando vueltas de más.

## Propósito

Con nuestra app, solo escribes tus pendientes y listo: te arma un itinerario inteligente. Primero el banco porque cierra temprano, después la peluquería que queda de camino, luego la ferretería más cercana y finalmente el supermercado de regreso a casa. Todo optimizado según tu ubicación, horarios y transporte.

## Ámbito

Nuestra app cubrirá todo el proceso de planificación de pendientes diarios. Desde que el usuario ingresa sus tareas, hasta que recibe un itinerario optimizado considerando horarios, ubicación y rutas disponibles. Se incluirá la integración con mapas y transporte, la validación de horarios de locales y la propuesta de trayectos eficientes.

El proyecto abarca el diseño de la aplicación móvil, el desarrollo de la lógica de itinerarios, las pruebas de calidad y la publicación en las principales tiendas de aplicaciones.

## Definiciones, acrónimos y abreviaciones

| **ACRÓNIMO** | **DESCRIPCIÓN** |
| --- | --- |
| *API* | Application Programming Interface. Conjunto de funciones que permiten la interacción con servicios externos. |
| *GPS* | Global Positioning System. Sistema de geolocalización por satélite. |
| *UX* | User Experience. Experiencia del usuario en la interacción con la aplicación. |
| *DB* | Database. Sistema de almacenamiento estructurado de datos. |

## Referencias

A continuación se listan las referencias a otros documentos:

[**Especificación de Casos de Uso**](https://docs.google.com/document/d/1Gblcbnvo3OkbOip0B6aKGaoze7Kr1x6H/edit?usp=sharing&ouid=116527649750314309237&rtpof=true&sd=true)

## Resumen ejecutivo

**Propósito:** El propósito de este informe es describir los casos de uso y escenarios de negocio relacionados con la implementación de **TripGO**, una aplicación móvil que genera itinerarios inteligentes para optimizar las actividades diarias de los usuarios. La app combina datos de geolocalización, horarios de locales y transporte en tiempo real para entregar rutas personalizadas y eficientes.

**Procesos Principales:**

* **Gestión de Usuarios:** Permite el registro, autenticación y administración de perfiles de usuario.
* **Ingreso de Pendientes:** Los usuarios pueden registrar sus tareas diarias indicando tipo de actividad, horarios y preferencias.
* **Optimización de itinerarios:** El sistema consulta APIs externas (Google Maps, Directions and Places) y genera un itinerario ordenado según cercanía, horarios y tiempos de transporte.
* **Notificaciones y Recordatorios:** La app envía alertas para recordar al usuario próximas actividades y cambios en el itinerario.
* **Integración con Negocios Locales:** Los comercios pueden suscribirse para aparecer en las recomendaciones de la app, generando un modelo de negocio sostenible.

**Áreas Organizacionales Involucradas:**

* **Usuarios Finales:** Personas que buscan optimizar su tiempo en tareas diarias como compras, trámites y visitas a servicios.
* **Negocios Locales (Pymes):** Comercios como peluquerías, restaurantes, ferreterías y supermercados que pueden destacarse en la aplicación mediante suscripción.
* **Instituciones Académicas (Patrocinador):** DUOC UC como entidad patrocinadora del desarrollo y validación del sistema.

**Módulos del Nuevo Sistema Descritos:**

* **Registro y Autenticación:** Módulo que gestiona cuentas de usuario y seguridad de acceso.
* **Planificación de itinerarios:** Motor central de optimización de rutas que integra ubicación, transporte y horarios.
* **Integración de APIs Externas:** Encargado de la comunicación con Google Maps y otros servicios de transporte.
* **Notificaciones y Alertas:** Módulo que informa al usuario sobre eventos próximos, cambios de ruta y recordatorios.
* **Gestión de Suscripciones:** A pesar de que este proceso sea de forma externa, se permitirá a los negocios contratar servicios premium para aparecer en las recomendaciones.

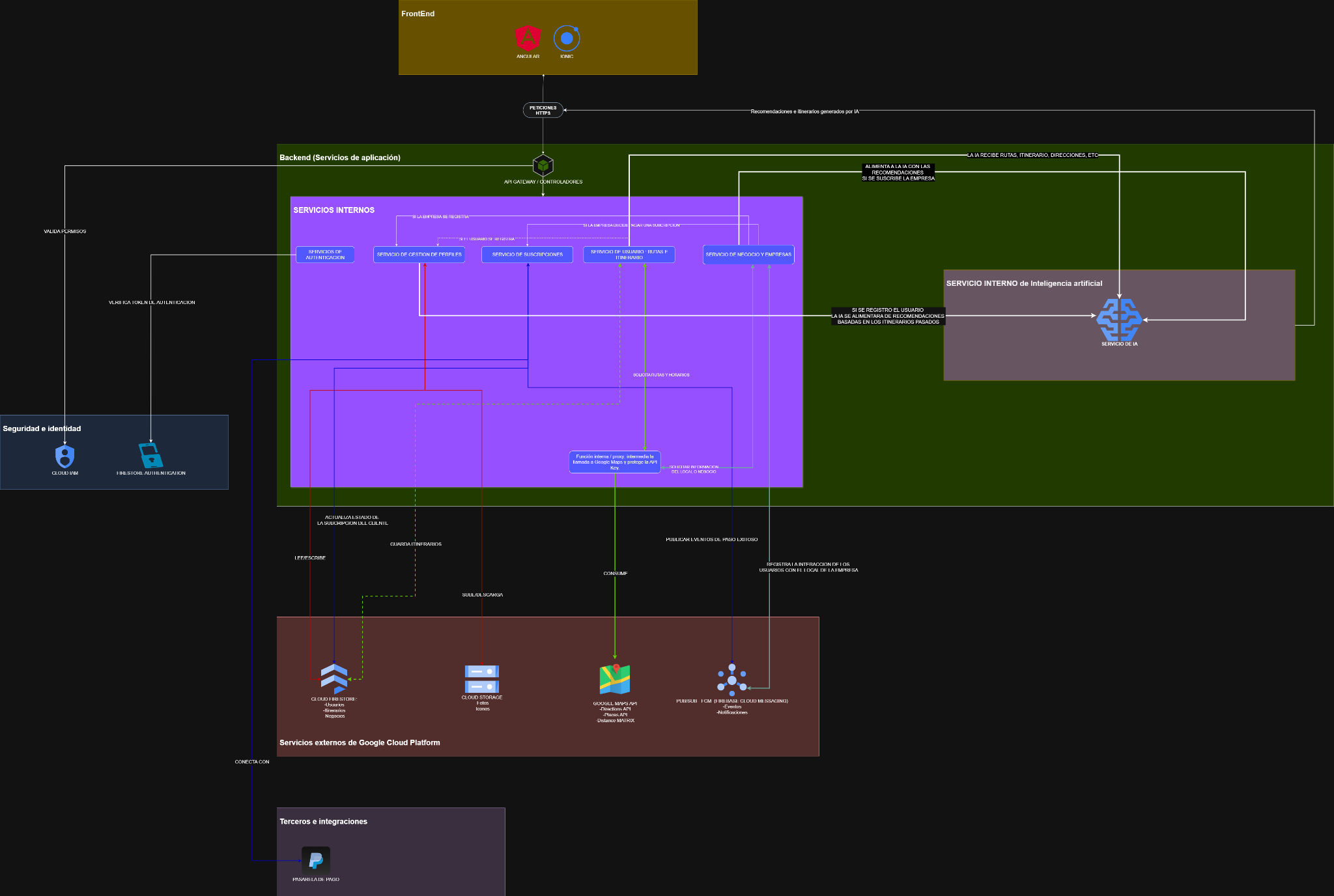
## Representación

La arquitectura del sistema TripGo está representada siguiendo el enfoque del framework 4+1 y las recomendaciones del proceso unificado. Las vistas incluidas en esta versión del documento son:

* **Vista de Casos de Uso y Escenarios de Calidad**: Describe los casos de uso más significativos, presenta los actores y una descripción de sus casos de uso asociados. De igual forma describe los escenarios de calidad más relevantes para la arquitectura.
* **Vista de Metas y Restricciones**: Describe restricciones tecnológicas, normativas, estándares, etc., los cuales influyen sobre las decisiones arquitectónicas, del producto y del proceso de desarrollo.
* **Vista Lógica**: Describe la arquitectura del sistema presentando varios niveles de refinamiento. Indica los módulos lógicos principales, sus responsabilidades y dependencias. Usa el view type Módulos para representar la estructura lógica y el view type Componentes y Conectores para representar el comportamiento.
* **Vista de Procesos**: Describe los procesos involucrados para darle sentido a la ejecución del sistema, así como sus relaciones de comunicación y sincronización.
* **Vista de Implementación**: Describe los componentes de deployment construidos y sus dependencias.

# 

# Propuesta Arquitectónica

**Link :** [**DIAGRAMA DE IMPLEMENTACION HD**](https://drive.google.com/file/d/1MY1JzrCIg7mtdehwBwO4hqreHuNxT9II/view?usp=sharing)

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Metas y Restricciones de la Arquitectura

## Metas de la arquitectura

De acuerdo a las reuniones y al análisis de los requerimientos, se listan los principales conductores iniciales de la arquitectura los cuales corresponden a las metas arquitectónicas iniciales:

* **Desempeño:** La aplicación debe responder en menos de 2 segundos al generar itinerarios cortos y en menos de 5 segundos para itinerarios complejos.
* **Tolerancia a fallos:** El sistema debe recuperarse de un fallo en menos de 30 minutos y mantener la consistencia de la información del usuario.
* **Seguridad:** Se deben implementar protocolos de cifrado (HTTPS, AES), autenticación de usuarios y protección de datos sensibles conforme a la Ley 19.628 de Chile.
* **Modificabilidad/Reuso:** La arquitectura debe permitir la adición de nuevos módulos (ej. integración con transporte local) sin modificar los existentes. Se prioriza el uso de frameworks de inyección de dependencias.
* **Operatividad:** El sistema debe ser fácil de instalar, mantener y monitorear, soportando métricas de salud (logging, auditoría de accesos, alertas).

## Restricciones de la Arquitectura

Existen restricciones que han sido levantadas con los stakeholders, las cuales se presentan a continuación:

* **Tiempo de construcción**: se cuenta con un plazo estrecho de tiempo para su construcción, -- semanas según la planificación.
* **Infraestructura**: Se cuenta con un ambiente 100% en la nube mediante la suite de google llamada GCP.
* **Otros componentes de software**: no se considera la adquisición y licenciamiento de otros componentes de software.

## Otros antecedentes y consideraciones

La empresa desarrolladora contará con un stack basado en Angular e Ionic, que considera los siguientes componentes y librerías para satisfacer los requerimientos arquitectónicos:

* **Framework de UI y componentes móviles (Ionic):** Permite la creación de interfaces responsivas y multiplataforma (iOS, Android y web). Facilita la reutilización de componentes y la consistencia visual, cumpliendo con los requerimientos de usabilidad y experiencia de usuario.
* **Framework de inyección de dependencias (Angular):** Soporta la modularización y encapsulación de los componentes, lo que facilita la mantenibilidad, escalabilidad y testeo del sistema. Angular ofrece un performance eficiente y optimización en tiempo de ejecución.

# **Vista de Casos de Uso / Escenarios de Calidad**

Esta sección describe en detalle el conjunto de escenarios funcionales y no funcionales que obtuvieron la mayor prioridad en el análisis. Para esto se presenta y describe el diagrama de casos de uso y los casos de uso prioritarios, así como los escenarios en que uno o más atributos de calidad se ven involucrados de manera significativa.

## Modelo de Casos de Uso

El modelo de casos de uso puede ser encontrado en el documento “Casos de Uso”.

## Especificación de Casos de Uso Relevantes

Los casos de uso considerados los más relevantes para el desarrollo de la arquitectura fueron determinados. Los criterios usados para dicha determinación fueron:

* Su implementación implica varios nodos de la vista de despliegue.
* Su implementación es de alto riesgo.
* Incluye muchos conceptos y relaciones del dominio.
* Incluye posibles escenarios críticos de calidad.

A continuación se listan los casos de uso relevantes, los cuales pueden ser encontrados con su especificación detallada en el documento “Casos de Uso”.

| **Código** | **Nombre** | **Actores** | **Prioridad** |
| --- | --- | --- | --- |
| CU-001-001 | Exportar saldos y puntos a vencer |  | Baja |
| CU-002-001 | Exportar actividades |  | Alta |
| CU-002-002 | Importar deuda vencida por PDV |  | Muy Alta |
| CU-004-001 | Generación Archivo PDA Importación |  | Media |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Especificación de los Escenarios de Calidad Relevantes

Después de un análisis en conjunto con los stakeholders, los escenarios de calidad se expresan a continuación:

**ID: QS1**

**Nombre:** Desempeño: Tiempo de respuesta en la generación de los archivos PDA.

**Sinopsis**: Tiempos de respuesta en la generación de archivos para la PDA que no afecten el actual proceso.

**Entorno**: Proceso normal de operación del sistema.

**Cambio en el entorno**: ejecución de proceso de carga PDA.

**Comportamiento esperado**: El proceso genera los archivos de carga en el file system.

**Medida**: debe demorar menos de 2 minutos por cada PDA cargada.

**Prioridad Arquitectónica**: Alta

**Aplicación**: Global

**ID: QS2**

**Nombre:** Tolerancia a fallos: Recuperación del sistema ante un fallo.

**Sinopsis**: El Sistema debe restablecerse en un tiempo razonable.

**Entorno**: El sistema está trabajando en su carga normal.

**Cambio en el entorno**: El sistema sufre un crash.

**Comportamiento esperado**: Se debe restablecer el sistema para seguir con la operación normal.

**Medida**: debe demorar menos de 30 minutos en restablecer el sistema.

**Prioridad Arquitectónica**: Alta

**Aplicación**: Local

# Vista Lógica

A continuación se presenta una vista lógica de la aplicación expresado en dos diagramas, uno de ellos que muestra la parte estructural o estática de la aplicación (módulos), y otra vista que representa la parte dinámica (componentes y conectores).

## Parte Estructural

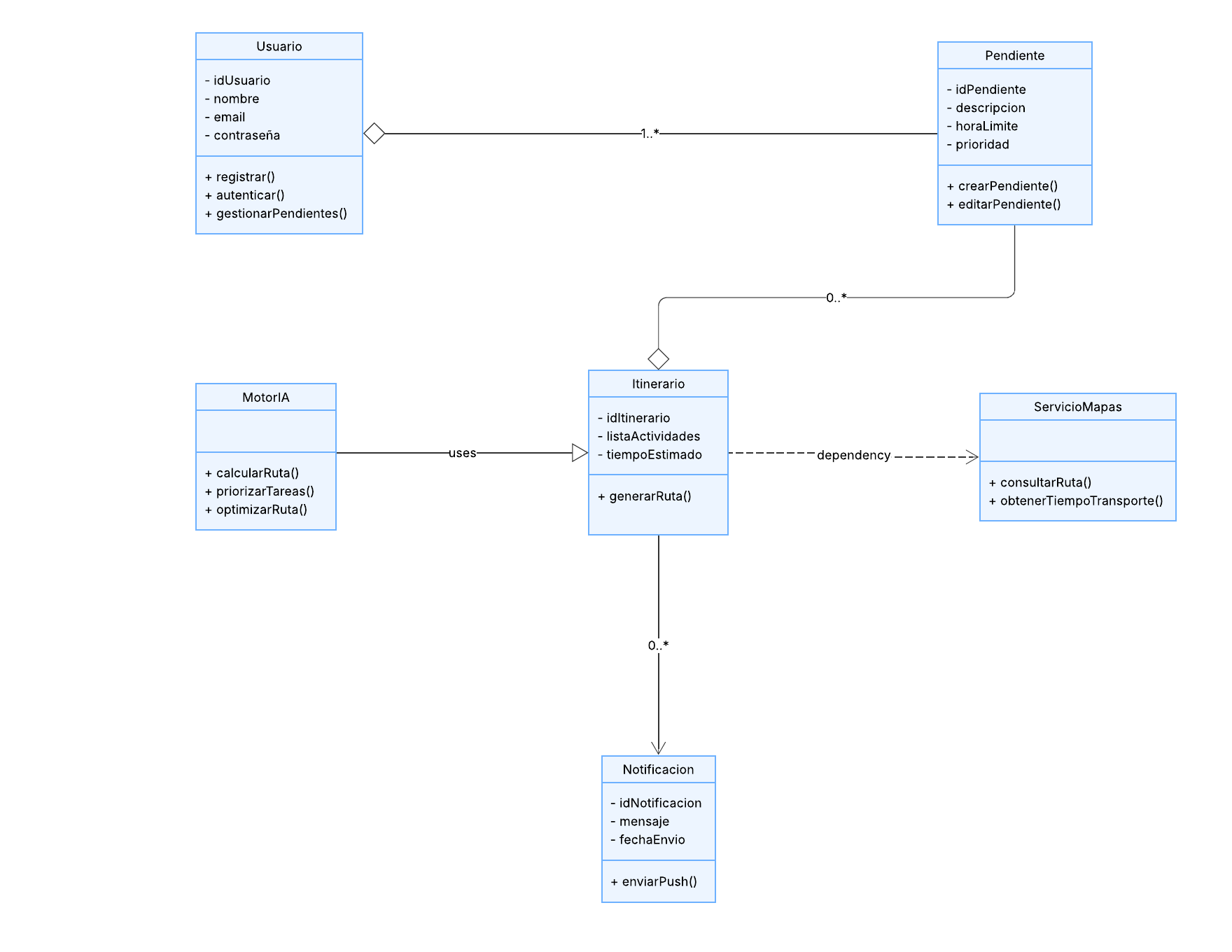
En el siguiente diagrama de clases UML se representa la estructura lógica principal del sistema **TripGO**, mostrando las entidades más relevantes, sus atributos, operaciones y relaciones.

* **Usuario**: clase que modela al cliente final de la aplicación. Permite registrar, autenticar y gestionar los pendientes asociados a su cuenta.
* **Pendiente**: corresponde a las tareas ingresadas por el usuario (ejemplo: ir al banco, comprar en la ferretería). Incluye atributos como descripción, hora límite y prioridad. Cada usuario puede tener múltiples pendientes.
* **Itinerario**: clase central del sistema que organiza y optimiza los pendientes en una ruta eficiente. Contiene atributos como lista de actividades y tiempo estimado, e implementa operaciones para generar y optimizar rutas.
* **MotorIA**: módulo de inteligencia artificial dentro del backend. Procesa los pendientes y define un orden óptimo en base a ubicación, horarios y transporte.
* **Notificación**: clase encargada de los recordatorios enviados al usuario. Contiene mensaje, fecha de envío y permite generar alertas derivadas de los itinerarios.
* **Servicio Mapas**: representa la integración con APIs externas (Google Maps, Directions, Distance Matrix). Permite consultar rutas y obtener tiempos de transporte.

Las multiplicidades expresadas en el modelo indican que:

* Un **Usuario** puede tener cero o muchos **pendientes**.
* Un **itinerario** debe estar compuesto al menos por una pendiente, pero puede contener varios.
* Un **itinerario** puede generar cero o muchas **notificaciones**.
* El **itinerario** depende tanto del **MotorIA** como del **Servicio Mapas** para producir rutas optimizadas.

**Diagrama de Clases UML**



## **Parte Dinámica**

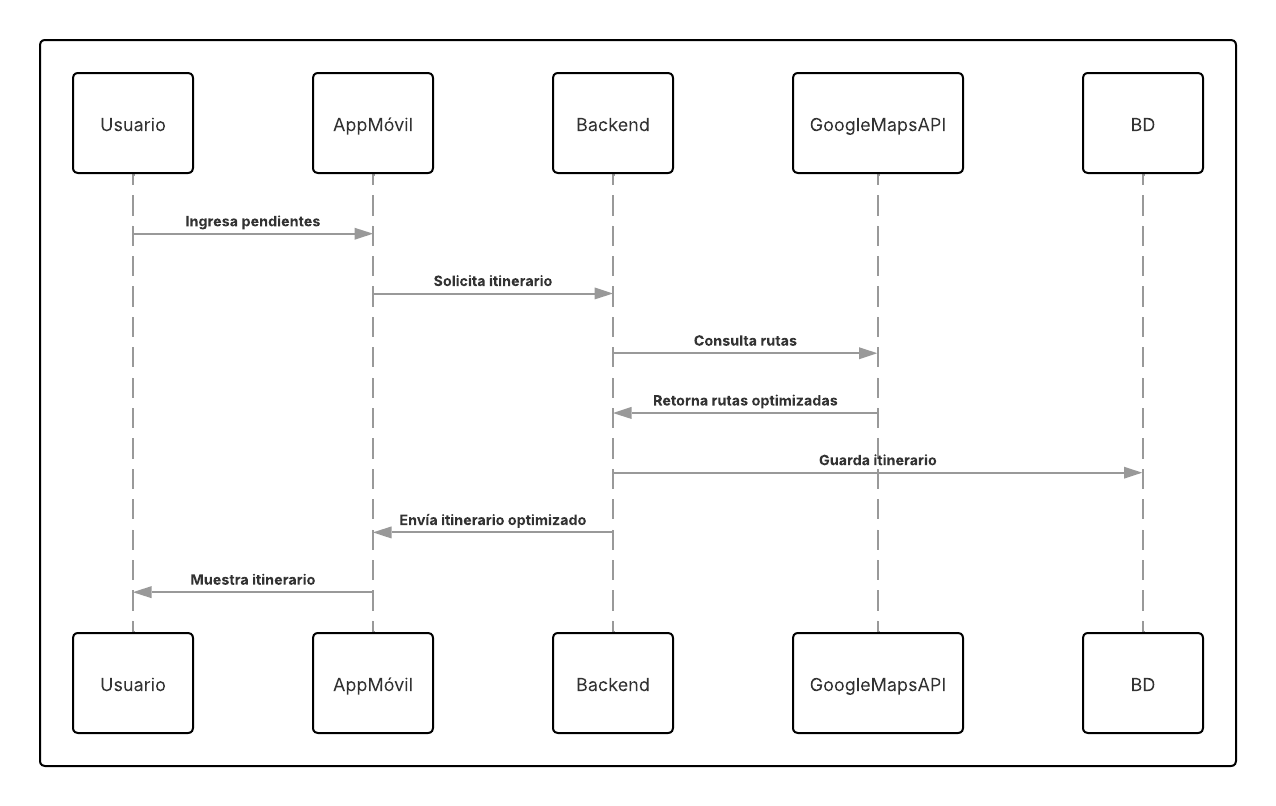
La parte dinámica de la vista lógica se representa mediante un diagrama de secuencia, el cual modela la interacción entre los distintos componentes del sistema durante la ejecución de un caso de uso clave: Generar Itinerario.

En este diagrama se observa el flujo de mensajes:

* El Usuario ingresa sus pendientes a través de la Aplicación Móvil.
* La aplicación envía la solicitud al Backend, encargado de procesar la información.
* El Backend posee funcionalidades claves para obtener rutas y tiempos de transporte en base a la ubicación y horarios tales como el servicio de la IA, el cual debe saber que hacer con cada ruta que ingresa el usuario por la aplicación.
* El resultado se materializa en un itinerario, que es almacenado en la Base de Datos y posteriormente enviado a la aplicación del usuario.
* Finalmente, se generan notificaciones asociadas para recordar al usuario sus próximas actividades.

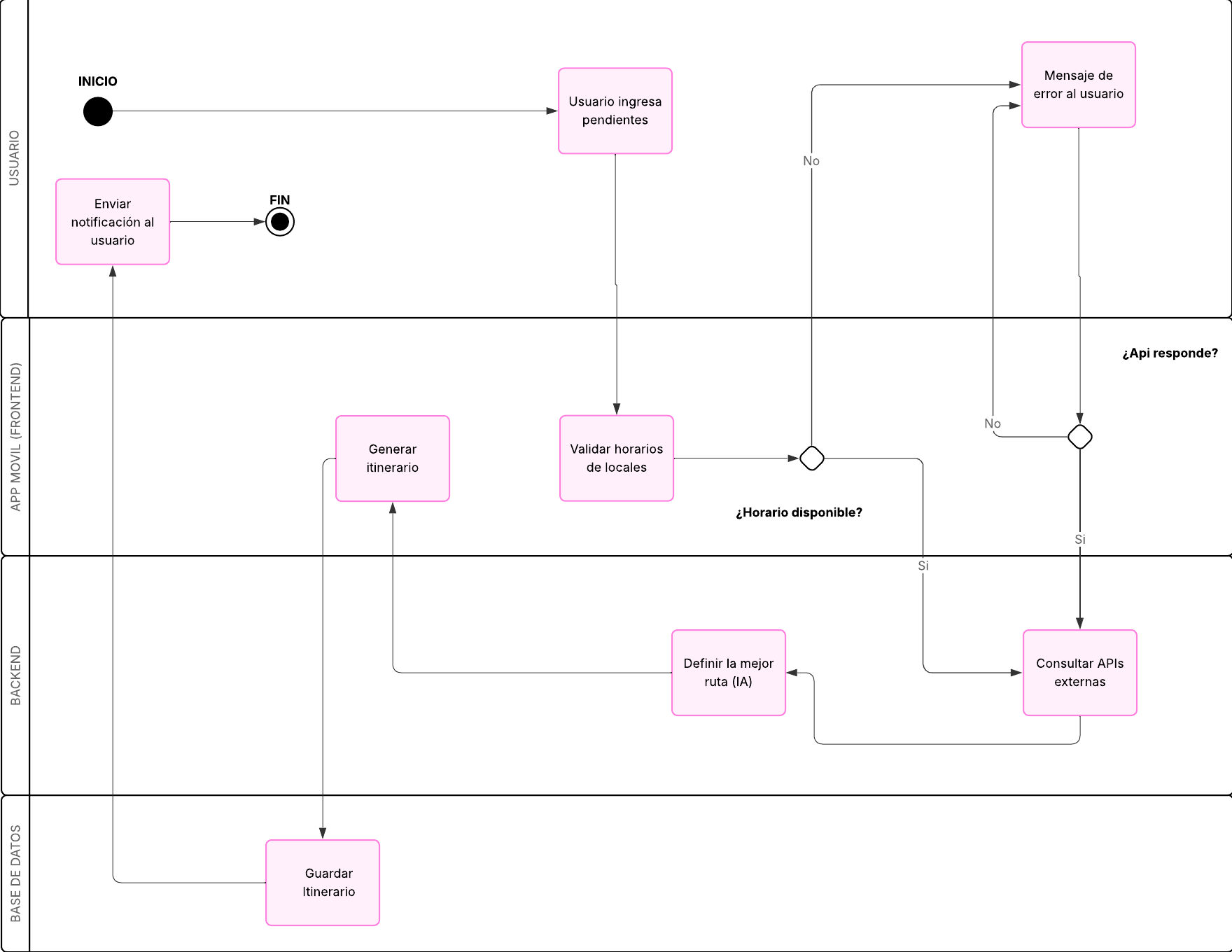
Este modelo dinámico evidencia cómo los distintos módulos cooperan en tiempo de ejecución para transformar los datos de entrada en un itinerario optimizado, integrando tanto lógica interna como servicios externos.

**Diagrama de secuencia**



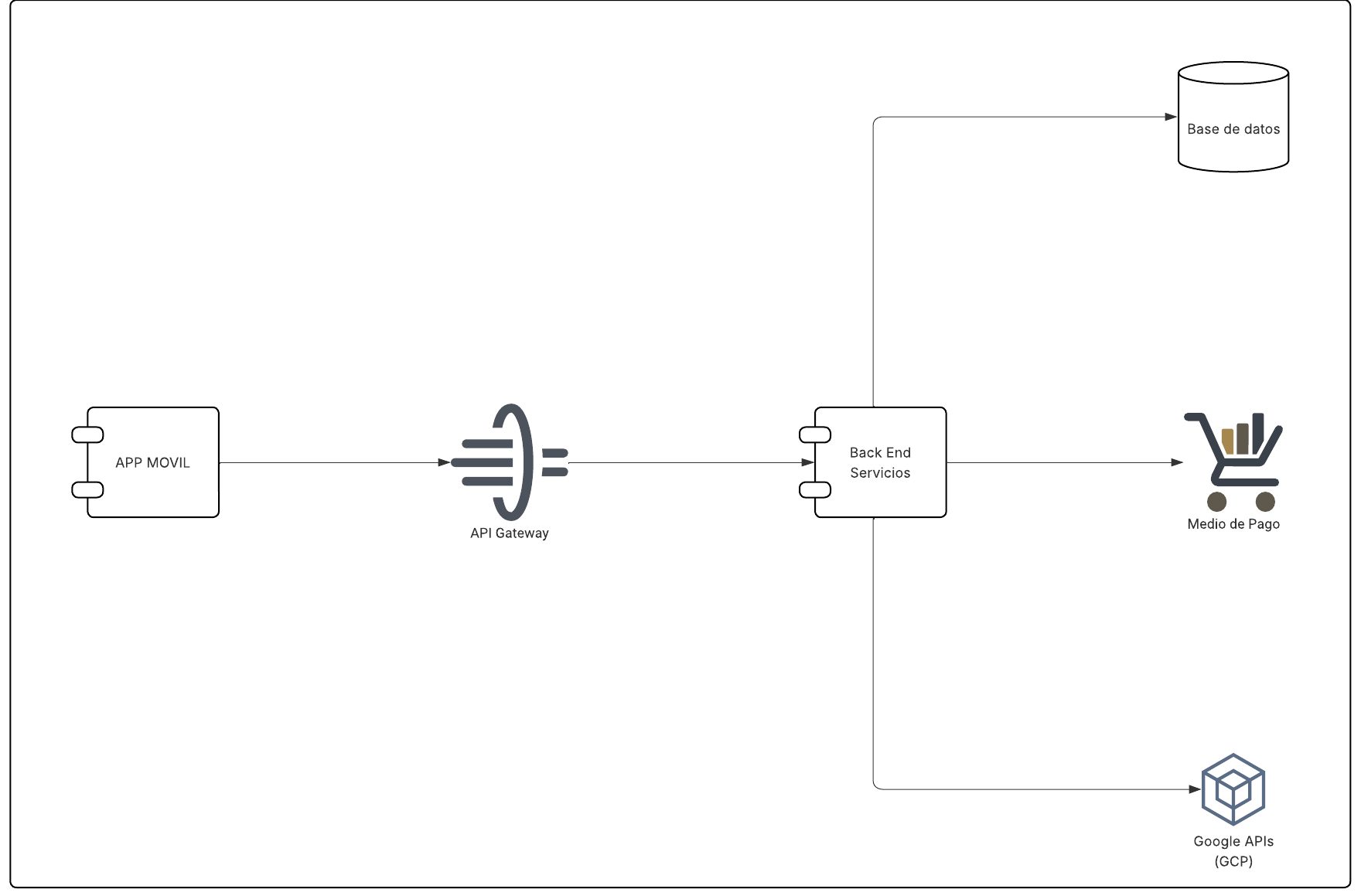
# Vista de Procesos

En esta sección, se presenta el flujo detallado de los procesos clave del Usuario. Para ofrecer una representación clara y comprensible de las interacciones y acciones que ocurren en tiempo real, se ha optado por utilizar un diagrama de actividades. Este tipo de diagrama permite ilustrar las actividades secuenciales y las decisiones dentro del flujo, mostrando las interdependencias entre las diferentes acciones del sistema. Así, se consigue una visión precisa de cómo los procesos se desarrollan, permitiendo identificar posibles cuellos de botella, optimizaciones y puntos críticos en la ejecución de inicio a fin.

**Diagrama de Actividades**

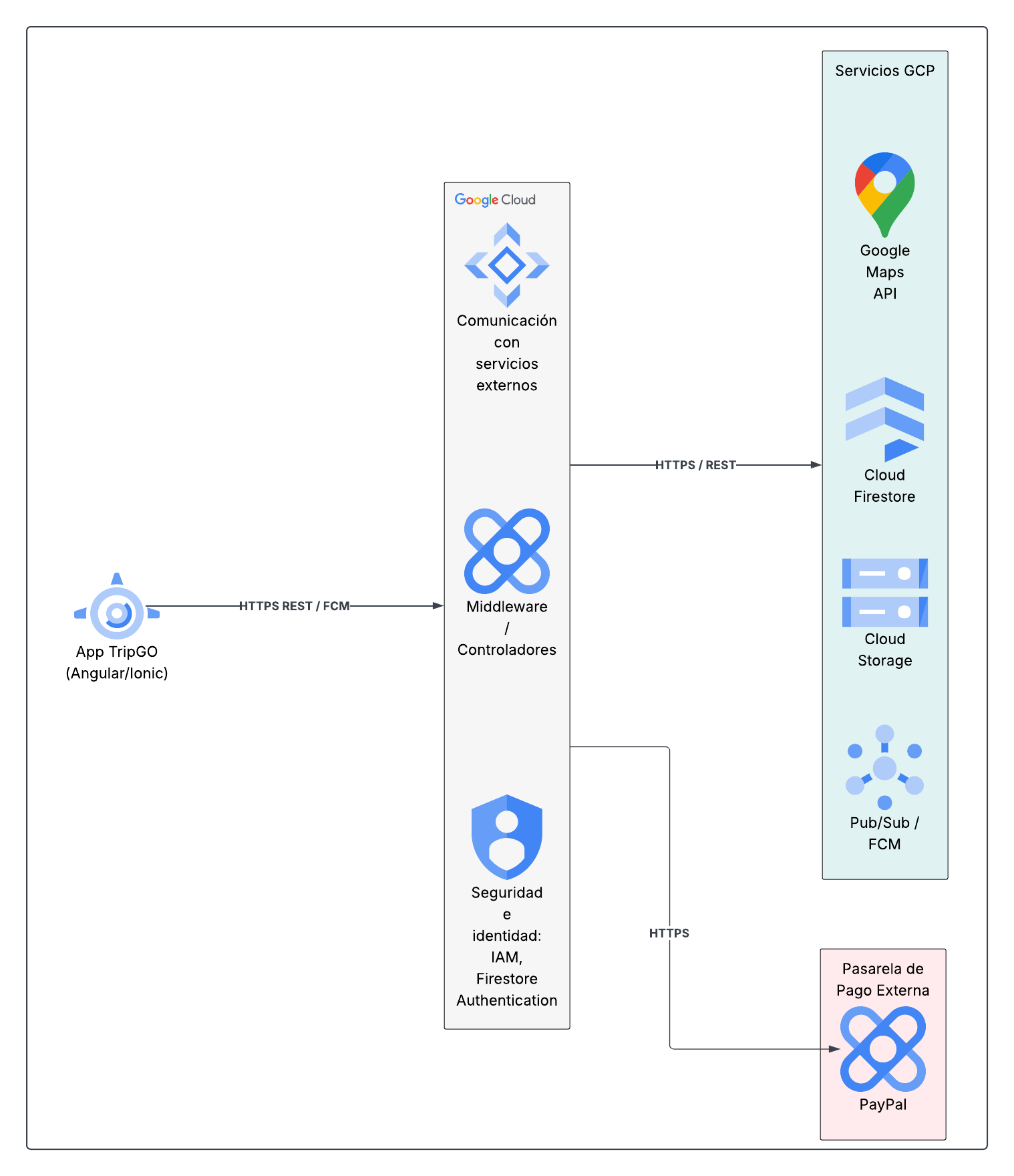
# Vista de Implementación

Muestra los componentes de software del sistema TripGO, sus dependencias y relaciones, destacando cómo se organiza la lógica del backend, frontend y módulos internos.



# 

# Vista de Despliegue

Representa la distribución física de los nodos y artefactos del sistema TripGO en la infraestructura de GCP, incluyendo la app móvil, backend, servicios externos y cómo se interconectan entre sí.

# 

# Decisiones de Diseño y Selección de Alternativas

Las principales decisiones arquitectónicas se tomaron en consideración de la restricción **Tiempo de Construcción**. Dado que el proyecto debe implementarse en un tiempo ajustado y sin holguras, se privilegió la adopción de una arquitectura conocida y que presente un bajo riesgo en su implementación.

Asimismo, la arquitectura se modularizó con el primer objetivo de separar con cernimientos de forma que permita paralelización en construcción de dichos componentes, y que a su vez sea módulos testeables unitariamente de forma de asegurar (mediante la suite Junit) que cada pieza tenga una baja tasa de fallas.

Un segundo elemento fue considerado en la arquitectura, que corresponde a la restricción de **Infraestructura** con que debe cumplir la aplicación, combinado con el escenario de calidad de **Tolerancia a Fallos**, nos condiciona la modularización de la aplicación en una **aplicación web activa-activa** y una **aplicación de servicios activa-pasiva**.

El escenario de calidad relacionado con la **mantenibilidad** nos conduce al modelamiento pensando en la separación de concernimiento de los componentes y a la utilización del patrón **provider** de forma que el sistema pueda delegar sus requerimientos de información hacia sistemas externos a piezas de software no acopladas que nos permitan su extensibilidad a futuro.

Por último, se eligió que la estrategia para implementar los providers externos en aquellos servicios asíncronos de entrada se implementarían mediante un temporizador (quartz) que levantará los procesos que verifican la llegada de información a las colas de entrada (mensajería asíncrona, archivos de texto en directorios). Esta estrategia fue seleccionada para disminuir el riesgo pues es una solución simple y efectiva.

# Análisis de Reutilización

[Poner razonamiento de componentes que se utilizarán o que se desarrollarán]