

Fase 2: Planeando un software con GenAI

Vamos a iniciar con una investigación del tipo de sistemas que queremos trabajar, utilizando GenAI:

Prompt 1

*Eres un experto en producto, con experiencia en **AUTOSENSE una solución automotriz**.*

*¿Qué funcionalidades básicas tiene **AUTOSENSE una solución automotriz**?*

Describémelas en un listado, ordenado de mayor a menor prioridad

Respuesta:

1. Diagnóstico en tiempo real

Conexión directa (por ejemplo, mediante un dispositivo OBD-II) que permite leer al instante datos críticos del vehículo (motor, sensores, emisiones, etc.) para detectar cualquier anomalía.

2. Interpretación y análisis de códigos de error

Traduce los códigos diagnósticos en descripciones claras, facilitando la identificación de posibles causas y orientando hacia soluciones o reparaciones.

3. Alertas y notificaciones inmediatas

Envía avisos automáticos cuando se detectan fallas críticas o condiciones que requieran atención, permitiendo actuar de forma preventiva antes de que se agrave el problema.

4. Historial y registro de datos

Guarda un seguimiento cronológico de los diagnósticos y mediciones, lo que ayuda a identificar tendencias, detectar desgaste progresivo y planificar mantenimientos futuros.

5. Recomendaciones y asesoramiento de mantenimiento

Basándose en el análisis de datos, ofrece sugerencias personalizadas sobre rutinas de mantenimiento, reparaciones y ajustes necesarios para optimizar el rendimiento del vehículo.

6. Integración con servicios y talleres

Facilita la localización de centros de servicio autorizados y, en algunos casos, permite agendar citas o solicitar asistencia remota directamente desde la app.

7. Generación de reportes y análisis detallados

Permite la exportación de informes completos del estado del vehículo, útiles para compartir con técnicos o para llevar un control exhaustivo del historial del automóvil.

Prompt 2

*¿Qué beneficios obtiene el cliente de un **sistema de diagnóstico automotriz como Autosense** para considerar su uso?*

Respuesta:

1. Detección temprana de fallas:

Permite identificar problemas o anomalías en tiempo real, lo que facilita la intervención antes de que se conviertan en reparaciones mayores.

2. Ahorro en costos de mantenimiento y reparación:

Al anticipar problemas, se pueden programar mantenimientos preventivos y evitar daños que podrían resultar en gastos significativos.

3. Mejora en la seguridad:

Un diagnóstico continuo asegura que el vehículo se mantenga en condiciones óptimas, reduciendo el riesgo de accidentes por fallas mecánicas.

4. Optimización del rendimiento:

Monitorear constantemente parámetros críticos ayuda a mantener el motor y otros sistemas funcionando de forma eficiente, mejorando el rendimiento general y el consumo de combustible.

5. Historial y seguimiento del estado del vehículo:

La app registra y almacena datos históricos, lo que permite detectar tendencias, planificar mantenimientos futuros y llevar un control detallado del estado del automóvil.

6. Facilidad en la comunicación con talleres y técnicos:

Al contar con reportes detallados y códigos de error interpretados, se facilita la comunicación con profesionales, acelerando el diagnóstico y la reparación.

7. Accesibilidad y comodidad:

El uso de una app móvil permite al usuario acceder a la información y gestionar el diagnóstico desde cualquier lugar, sin la necesidad de acudir a un taller para obtener un análisis básico.

Prompt 3

*¿Cómo es el customer journey normal de un cliente que usa un **sistema de diagnóstico automotriz como Autosense**? Describe paso a paso todas las interacciones*

Respuesta:

1. Descubrimiento y Conciencia

Exposición inicial: El cliente se entera de AUTONSENSE a través de publicidad digital, recomendaciones de conocidos o contenido en redes sociales y blogs especializados en automoción.

Interés generado: La propuesta de valor—como el diagnóstico en tiempo real y alertas preventivas—captura su atención.

2. Investigación y Evaluación

Búsqueda de información: El usuario investiga las funcionalidades y beneficios en el sitio web, foros o reseñas de otros clientes.

Comparativa de soluciones: Evalúa cómo AUTONSENSE se posiciona frente a otras herramientas diagnósticas, valorando tanto el dispositivo OBD-II como la aplicación móvil.

Contacto inicial: Puede comunicarse con el equipo de ventas o soporte para aclarar dudas técnicas o funcionales.

3. Adquisición y Configuración

Decisión de compra: Con base en la información recabada, el cliente decide adquirir el producto.

Descarga e instalación: Descarga la app en su smartphone y, si es necesario, adquiere e instala el dispositivo OBD-II en su vehículo.

Configuración inicial: Sigue un proceso guiado (tutorial o setup interactivo) para vincular el dispositivo con la app, establecer preferencias y personalizar alertas.

4. Onboarding y Primeros Usos

Tutorial de bienvenida: La app ofrece una introducción paso a paso sobre cómo interpretar los datos, configurar notificaciones y utilizar cada funcionalidad.

Prueba de diagnóstico: El usuario realiza su primer análisis en tiempo real, comprobando que la lectura de datos y la interpretación de códigos funcionan correctamente.

5. Uso Continuo y Monitoreo Diario

Monitoreo en tiempo real: El cliente consulta regularmente la app para verificar el estado del vehículo y recibe alertas inmediatas en caso de detectar anomalías.

Registro de datos: La solución almacena automáticamente el historial de diagnósticos, facilitando la revisión de tendencias y el seguimiento del mantenimiento.

6. Interacción con Soporte y Talleres

Soporte técnico: Si surge alguna duda o se detecta un error, el usuario puede contactar con el servicio de asistencia para recibir ayuda personalizada o asesoramiento técnico.

Conexión con profesionales: La app, en algunos casos, permite localizar talleres o programar citas, integrando el diagnóstico con la solución de problemas reales.

7. Feedback y Fidelización

Retroalimentación del servicio: El cliente puede dejar reseñas, calificar la experiencia y sugerir mejoras, lo que permite afinar el producto.

Actualizaciones y mejoras: Con base en el feedback, se lanzan actualizaciones que optimizan la funcionalidad del sistema, consolidando la confianza del usuario y fomentando la fidelidad.

Ahora iniciemos con la planeación del sistema desde los casos de uso

Prompt 1

*Eres un analista de software experto. Estoy construyendo un MVP de un **sistema de diagnostico automotriz llamado Autosense** que sólomente haga **Diagnóstico en tiempo real**. Enumera y describe brevemente los casos de uso más importantes a implementar para lograr una funcionalidad básica*

Respuesta:

1. Emparejamiento y conexión del dispositivo OBD-II

Permite al usuario emparejar y conectar el dispositivo OBD-II con la app (usualmente vía Bluetooth o WiFi), estableciendo la comunicación necesaria para el flujo de datos.

2. Lectura en tiempo real de datos del vehículo

La aplicación inicia la lectura continua de parámetros críticos (RPM, temperatura, presión, velocidad, entre otros) directamente del vehículo, garantizando datos actualizados en cada instante.

3. Procesamiento y análisis inmediato de datos

Los datos capturados se procesan en tiempo real, aplicando algoritmos o reglas predefinidas para detectar desviaciones o anomalías en los parámetros del vehículo.

4. Visualización dinámica de diagnósticos

La app presenta de manera intuitiva la información procesada a través de gráficos, indicadores o dashboards, permitiendo al usuario monitorear el estado del vehículo de forma clara y rápida.

5. Generación de alertas en caso de anomalías

Si el sistema identifica parámetros fuera de rango o condiciones críticas, se activan alertas (visuales o sonoras) que informan al usuario sobre la necesidad de atención inmediata.

6. Cierre y desconexión segura de la sesión

Permite al usuario finalizar la sesión de diagnóstico de forma segura, desconectando el dispositivo OBD-II y reseteando el estado de la app para futuros diagnósticos.

Prompt 2

Representa estos casos de uso en el tipo de diagrama más adecuado usando el formato plantUML. Diferencia entre usuarios visitantes y usuarios logueados. Acorde a la sintaxis y buenas prácticas UML, define y describe lo que sea necesario.

Respuesta:

```
@startuml
left to right direction

actor "Visitante" as Visitor
actor "Usuario Logueado" as LoggedUser

rectangle "AUTOSENSE MVP – Diagnóstico en Tiempo Real" {
    usecase "Iniciar Sesión / Registro" as UC0
    usecase "Emparejamiento y conexión del dispositivo OBD-II" as UC1
    usecase "Lectura en tiempo real de datos del vehículo" as UC2
    usecase "Procesamiento y análisis inmediato de datos" as UC3
    usecase "Visualización dinámica de diagnósticos" as UC4
    usecase "Generación de alertas en caso de anomalías" as UC5
    usecase "Cierre y desconexión segura de la sesión" as UC6
}

Visitor --> UC0 : "Se autentica"
LoggedUser --> UC1
LoggedUser --> UC2
LoggedUser --> UC3
LoggedUser --> UC4
LoggedUser --> UC5
LoggedUser --> UC6

@enduml
```

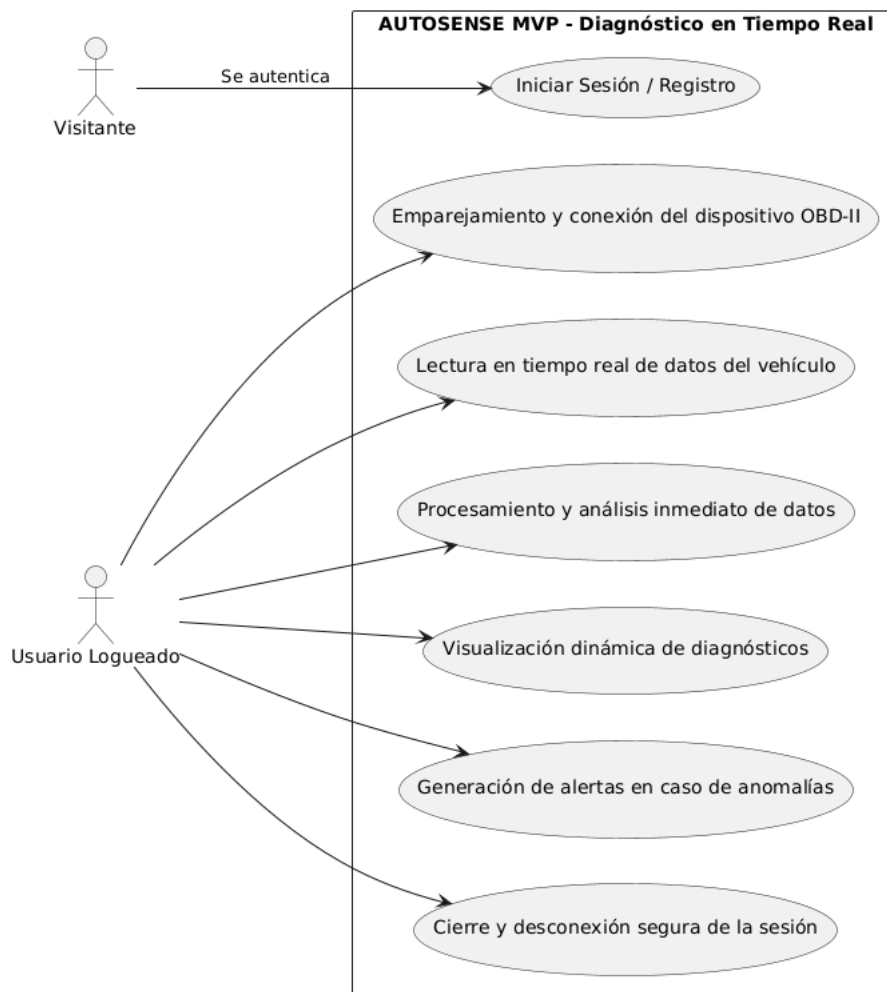


Imagen 3 – Diagrama de casos de uso

Descripción de los elementos:

Visitante: Usuario que aún no se ha autenticado. Su principal interacción es iniciar sesión o registrarse.

Usuario Logueado: Usuario autenticado, quien puede acceder a la funcionalidad completa del sistema.

Casos de uso:

Iniciar Sesión / Registro (UC0): Permite al visitante autenticarse para convertirse en usuario logueado.

Emparejamiento y conexión del dispositivo OBD-II (UC1): Conecta el dispositivo del vehículo con la app.

Lectura en tiempo real de datos del vehículo (UC2): Recupera continuamente los parámetros críticos del vehículo.

Procesamiento y análisis inmediato de datos (UC3): Interpreta la información en tiempo real para detectar anomalías.

Visualización dinámica de diagnósticos (UC4): Muestra de forma clara y gráfica los datos y resultados del diagnóstico.

Generación de alertas en caso de anomalías (UC5): Notifica al usuario si se detectan condiciones críticas.

Cierre y desconexión segura de la sesión (UC6): Permite terminar la sesión de diagnóstico y desconectar el dispositivo de forma segura.

Diseñemos ahora las entidades principales para la base de datos:

Prompt 1

*Eres un arquitecto de software experto. Cuales son las 3 entidades de modelo de datos esenciales en un **sistema de diagnostico automotriz llamado Autosense**? Dame algunos campos esenciales de cada una y cómo se relacionan*

Respuesta:

1. Usuario

Campos Esenciales:

id_usuario: Identificador único del usuario.

nombre: Nombre completo del usuario.

correo: Correo electrónico para notificaciones e inicio de sesión.

contraseña: Clave cifrada para la autenticación.

fecha_registro: Fecha en que se registró el usuario.

Relación: Un usuario puede poseer uno o varios vehículos, estableciendo una relación de uno a muchos con la entidad Vehículo.

2. Vehículo

Campos Esenciales:

id_vehículo: Identificador único del vehículo.

id_usuario: Clave foránea que asocia el vehículo a un usuario.

VIN: Número de Identificación Vehicular único para cada automóvil.

marca: Marca del vehículo (ej. Toyota, Ford).

modelo: Modelo del vehículo.

año: Año de fabricación.

Relación: Cada vehículo está asociado a un único usuario.

Un vehículo puede tener múltiples registros de diagnósticos, lo que lo vincula en una relación de uno a muchos con la entidad *Diagnóstico*.

3. Diagnóstico (Sesión de Diagnóstico en tiempo real)

Campos Esenciales:

id_diagnóstico: Identificador único para la sesión de diagnóstico.

id_vehículo: Clave foránea que relaciona el diagnóstico con el vehículo.

fecha_hora_inicio: Marca de tiempo que indica el inicio de la sesión.

fecha_hora_fin: Marca de tiempo que indica la finalización (puede ser nula si es en curso).

datos_sensor: Registro (posiblemente en formato JSON) que contiene datos en tiempo real (RPM, temperatura, presión, velocidad, etc.).

alertas_generadas: Indicadores o registros de alertas si se detectan anomalías en los parámetros.

Relación: Cada sesión de diagnóstico se vincula a un vehículo, permitiendo mantener un historial que puede ser consultado para análisis de tendencias y seguimiento del estado del automóvil.

Relaciones generales:

• **Usuario → Vehículo:**

Un usuario puede tener varios vehículos, lo que permite administrar múltiples autos dentro de la misma cuenta.

- **Vehículo → Diagnóstico:**

Cada vehículo puede generar y almacenar múltiples sesiones de diagnóstico a lo largo del tiempo, facilitando un monitoreo continuo de su estado.

Prompt 2

Eres un arquitecto de software. Eres capaz de diseñar, explicar y diagramar los diferentes aspectos de un sistema de software.

*Estoy construyendo un sistema de **diagnostico automotriz llamado Autosense**. He definido las entidades **usuario, vehiculo y diagnostico** con sus campos y relaciones.*

Respuesta:

1. Diagrama de Modelo de Datos (Clases)

```
@startuml
' Definición de la clase Usuario
class Usuario {
  +id_usuario: int
  +nombre: String
  +correo: String
  +contraseña: String
  +fecha_registro: DateTime
}

' Definición de la clase Vehículo
class Vehiculo {
  +id_vehiculo: int
  +id_usuario: int
  +VIN: String
  +marca: String
  +modelo: String
  +anio: int
}

' Definición de la clase Diagnóstico
class Diagnostico {
  +id_diagnostico: int
  +id_vehiculo: int
  +fecha_hora_inicio: DateTime
  +fecha_hora_fin: DateTime
  +datos_sensor: JSON
  +alertas_generadas: String
}
```

```
' Relaciones
Usuario "1" -- "0..*" Vehiculo : posee
Vehiculo "1" -- "0..*" Diagnostico : genera
@enduml
```

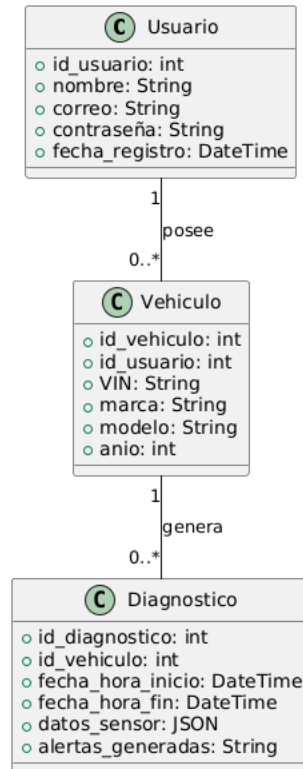


Imagen 4 – Diagrama de modelo de datos

2. Arquitectura del Sistema de Alto Nivel

```
@startuml
!define RECTANGLE class

rectangle "Aplicación Móvil" as MobileApp
rectangle "API Gateway" as APIGateway
rectangle "Lógica de Negocio" as BusinessLogic
rectangle "Integración OBD-II" as DeviceIntegration
rectangle "Base de Datos" as Database

MobileApp --> APIGateway : Solicitudes REST/HTTPS
APIGateway --> BusinessLogic : Enrutamiento de solicitudes
BusinessLogic --> DeviceIntegration : Solicita datos en tiempo real
DeviceIntegration --> BusinessLogic : Envía datos del vehículo
BusinessLogic --> Database : CRUD de entidades
APIGateway --> Database : Acceso a datos (lecturas históricas)
@enduml
```

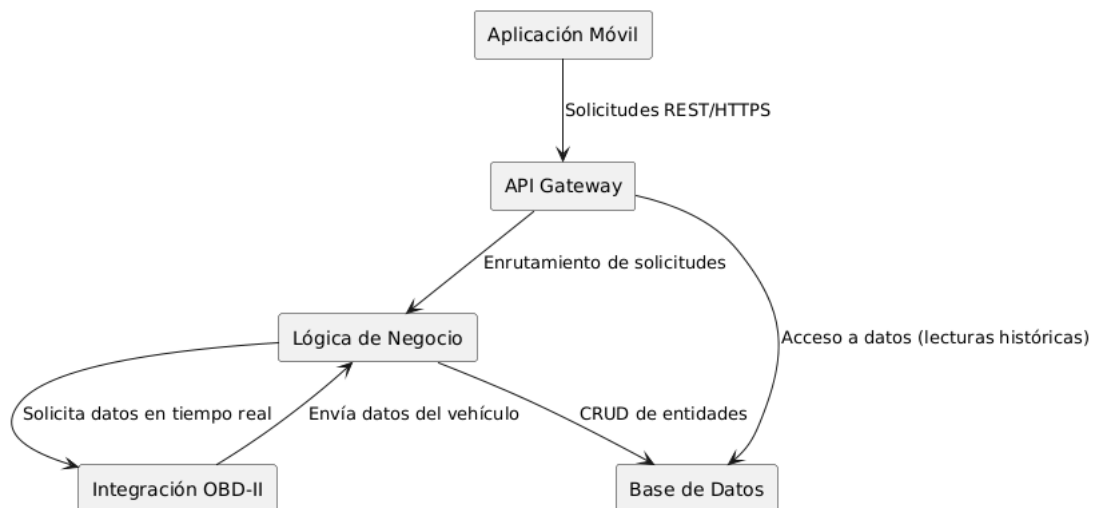


Imagen 5 – Diagrama de modelo de datos

Para el MVP de AUTONSENSE, se recomienda una arquitectura modular y escalable que se divida en capas o componentes. A continuación, se describe un esquema de la arquitectura:

Componentes Principales

- *Aplicación Móvil/Cliente:*

Permite la interacción del usuario, mostrando datos en tiempo real, alertas y gráficos de diagnóstico.

- *API Gateway:*

Actúa como punto de entrada para las solicitudes desde el cliente, gestionando la autenticación y enrutamiento a los microservicios o componentes correspondientes.

- *Servicio de Lógica de Negocio:*

Se encarga de procesar los datos del dispositivo OBD-II, aplicar algoritmos de análisis y gestionar la generación de alertas.

- *Servicio de Integración con Dispositivos:*

Gestiona la conexión y comunicación con el dispositivo OBD-II, recibiendo datos en tiempo real del vehículo.

- *Base de Datos:*

Almacena las entidades definidas (Usuario, Vehículo y Diagnóstico) y permite consultar historiales y tendencias.

Explicación:

Aplicación Móvil: Permite al usuario autenticarse, visualizar datos en tiempo real y recibir alertas.

API Gateway: Centraliza las peticiones, implementando seguridad, autenticación y balanceo de carga.

Lógica de Negocio: Procesa los datos recibidos del dispositivo, detecta anomalías y gestiona el flujo de información.

Integración OBD-II: Establece y mantiene la conexión con el dispositivo que extrae los datos del vehículo.

Base de Datos: Almacena la información de usuarios, vehículos y diagnósticos para consultas y análisis históricos.

Conclusión

Este diseño proporciona una base sólida para un MVP de AUTOSENSE enfocado en el diagnóstico en tiempo real. Se cuenta con:

Un modelo de datos bien definido que facilita la gestión de la información del usuario, sus vehículos y los registros de diagnóstico.

Una arquitectura modular que separa la interfaz de usuario, la lógica de negocio, la integración con hardware (OBD-II) y la persistencia de datos, permitiendo una evolución escalable y mantenible del sistema.

Esta estructura asegura que el sistema pueda crecer y adaptarse a futuras funcionalidades, manteniendo al mismo tiempo la simplicidad y eficiencia necesarias para un MVP.

Prompt 3

Generame un diagrama Mermaid de este sistema

```
graph TD
    subgraph Cliente
        A[Aplicación Móvil]
    end
    subgraph Backend
        B[API Gateway]
        C[Lógica de Negocio]
        D[Integración OBD-II]
        E[Base de Datos]
    end
    A -->|Solicitudes REST/HTTPS| B
    B -->|Enrutamiento de solicitudes| C
    C -->|Solicita datos en tiempo real| D
    D -->|Envía datos del vehículo| C
    C -->|CRUD de entidades| E
    B -->|Acceso a datos: lecturas históricas| E
```

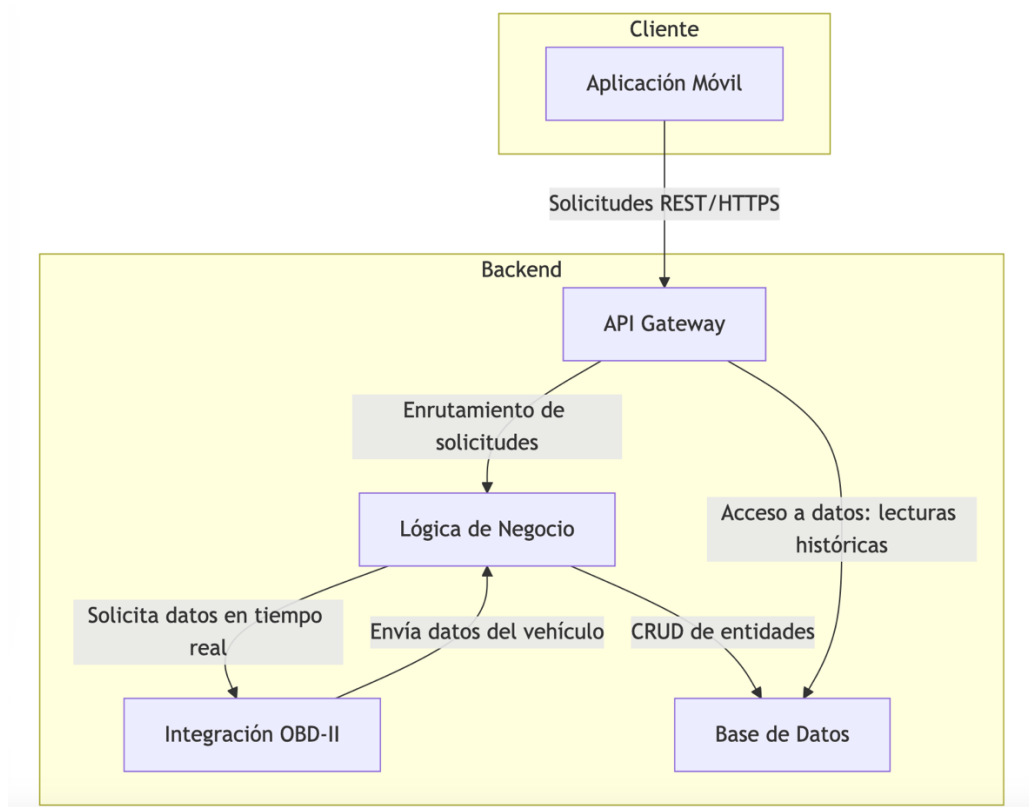


Imagen 6 – Diagrama mermaid del sistema

Ahora pruebe a generar el diagrama de arquitectura con **DiagramGPT**

Prompt 1

A microservices architecture for an automotive diagnostic app called Autosense system like Expedia

Each MS has its own database

Also has a frontend that communicates through API

Cloud provider is AWS, use proper services

Include load balancing and CDN

Autosense System Microservices Architecture

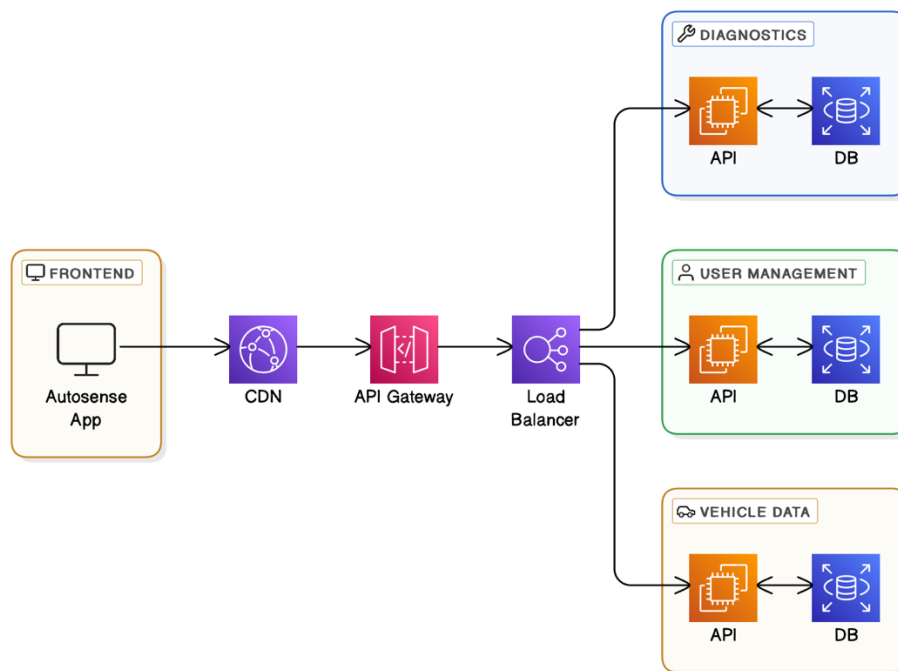


Imagen 7 – DiagramGPT