ESTRUCTURA Y DISEÑO DE ARQUITECTURA DE SOFTWARE PARA UN SISTEMA DE FLOTAS INTELIGENTE

Nombre: Daniel Hernandez Palacio

**1. Funcionalidad Principales**

Seguimiento en Tiempo Real:

- Esto nos ayudara a poder monitorear en tiempo real, la ubicación, velocidad y el estado de los vehículos

- El funcionamiento sería este: Los vehículos envían datos de sensores como el GPS, Velocidad, estado. Con esto el microservicio de Seguimiento procesa los datos y los almacena en una base de datos real, para después tener los datos disponibles en una interfaz web para los administradores.

- Para usar esto, se debe tener unas cosas puntuales, que sin esto, este microservicio no serviría, las cuales son: El Vehículo ( Que debe tener el sensor de GPS y telemetría ) , Servicio de Seguimiento, Base de datos de Telemetría

Rutas Inteligentes:

- Esto nos ayudara a poder escoger las mejores rutas en tiempo real, nos ahorrara tiempo y tendremos más eficiencia en las rutas

- El funcionamiento sería este: El sistema recibe los datos en tiempo real de los sensores GPS de los vehículos, consulta servicios externos para obtener información de tráfico y clima, calculamos la ruta optima y enviamos las instrucciones al conductor a través de una página web o una app móvil

- Para poder usar este microservicio, se necesita de APIS que nos den los datos en tiempo real del tráfico, clima y restricciones en las carreteras, se necesita la app móvil que el conductor usara para poder localizar las mejores rutas y se necesita el algoritmo que nos hará la optimización de las rutas y nos seleccionara la mejor opción

Mantenimiento Preventivo:

- Esto nos ayudara a intentar siempre prevenir posibles fallos mecánicos

- El funcionamiento sería este: Los sensores de los vehículos recopilan datos sobre el estado del motor, neumáticos, frenos y demás cosas importantes. Necesitamos un algoritmo que nos haga un análisis de los datos y nos detecte posibles fallos o si necesitamos un mantenimiento. El sistema genera alertas y las envía a los administradores y al personal técnico mediante la app móvil, así ellos sabrán que vehículos tienen posibles fallas, y cuales son las posibles fallas de estos

- Para poder usar este microservicio, se necesitan de los sensores de los vehículos, la interfaz que nos muestra las alertas

Gestión de conductores:

- Esto nos ayuda a asignar conductores a vehículos de una forma mas eficiente, poder monitorear su desempeño

- El funcionamiento sería este: Los administradores ingresan o actualizan el perfil de los conductores, el sistema asignara el conductor a un vehículo. Durante los viajes, los sensores recopilan datos relacionados con el cómo conduce el conductor, cosas como la velocidad, frenadas, cosas así, para al final de cada jornada, el sistema generar un reporte de desempeño, para así poder ver el desempeño de este conductor

- Debemos de tener una Base de Datos de los conductores en donde tenemos sus datos personales, se deben necesitar los sensores y la aplicación web o móvil que nos enseñe el reporte final

Integración con Terceros:

- Esto nos ayuda facilitar la conexión con servicios externos como mapas

- El funcionamiento sería este: Necesitamos llamar a varias APIs externas para poder tener informaciones, Los servicios retornan información que el sistema almacena o procesa según la funcionalidad requerida. Los datos son mostrados en la interfaz de usuario o utilizados automáticamente por otros microservicios

- Debemos tener las APIs Externas

**2. Requisitos Técnicos**

- Escalabilidad: Se usarán los Microservicios, cada funcionalidad (seguimiento, rutas, mantenimiento) se implementa como un servicio independiente, lo que permite escalar cada uno según la demanda. Usaremos Docker para poder empaquetar los servicios y Kubernetes para escalar automáticamente según el tráfico.

- Disponibilidad: Los microservicios más usados, se replicarán en múltiples servidores para así asegurar que sea muy difícil que se caigan, ya que, si uno falla, tenemos el otro. También implementaríamos balanceadores para distribuir el tráfico entre instancias activas

Modularidad: Gracias a que cada módulo funciona de forma independiente, esto nos permitirá actualizaciones o cambios sin afectar al resto del sistema.

Usaremos API Gateway ya que centraliza la comunicación y asegura que todos los módulos sean autónomos y puedan ser modificados sin romper la interacción entre cada uno de ellos.

Los módulos se prueban y se despliegan de forma aislada, para así poder garantizar una buena calidad antes de ser integrados

Almacenamiento Eficiente:

Usaremos bases de datos Hibridas, como PostgreSQL para datos estructurados como perfiles de conductores o configuraciones y también usaremos MongoDB para datos no estructurados como registros de sensores y datos históricos. Usaremos un sistema llamado Comprensión y Archivado, ya que datos antiguos serán comprimidos y almacenados en un sistema de bajo costo

Seguridad:

Cumplimiento con GDPR, implementar control de acceso basado en roles, para proteger la información sensible, Anonimizacion de datos personales donde sea posible. Uso de protocolos seguros como HTTPS y cifrado AES para proteger datos en tránsito y en reposo

**3. Restricciones**

- Implementación Gradual:

Enfocarse en las funcionalidades criticas para la gestión de flotas logísticas, como el Seguimiento en tiempo real, rutas inteligentes y mantenimiento predictivo. Limitar la cantidad inicial de vehículos soportados para validar el diseño. Usar microservicios desde el inicio, para que las futuras expansiones sean más fáciles.

- Múltiples zonas horarias y monedas:

Guardaremos todas las fechas y horas en formato UTC en las bases de datos y convertirlas a la zona horaria local en la interfaz de usuario

Implementar un servicio de conversación de monedas conectado a APIs externas, almacenaremos las tarifas y costos en la moneda base del cliente y convertirlas en tiempo real según el contexto

- Presupuesto Limitado en la Fase Inicial:

Usaremos una infraestructura mínima viable. Usaremos servicios en la nube con esquemas de pago por uso, para así poder minimizar costos. (AWS, entre otras)

Evitaremos configuraciones sobredimensionadas durante esta fase, solo implementamos los microservicios necesarios y justos

Usaremos bases de datos locales o de menor costos los primeros meses, Monitorizar y optimizar el uso de recursos para identificar posibles reducciones de gasto.

En lugar de integrar múltiples APIs desde el inicio, vamos a priorizar las mas importantes, como servicios de mapas

**4. Desafíos Adicionales:**

Manejo de picos de trafico

Es posible que, durante eventos especiales o emergencias, el sistema podría experimentar un aumento significativo en el numero de solicitudes. Lo que podremos hacer es usar un Balanceador de carga, que esto nos ayudaría a distribuir el tráfico equitativamente entre instancias

Garantizar sincronización de datos en áreas con conectividad limitada

Los vehículos podrían operar en zonas con baja conectividad, afectando obviamente la sincronización de datos en tiempo real. Lo que podríamos hacer es usar un Almacenamiento Local Temporal, que en si seria, configurar dispositivos en los vehículos para almacenar datos localmente hasta que vuelvan a tener una conexión

Soporte para Múltiples Interfaces de Usuario:

El sistema necesita ser accesible desde aplicaciones web, móviles y APIs para integraciones. Para las aplicaciones WEB, utilizaríamos REACT, ya que es un lenguaje que nos sirve para construir interfaces muy amigables para administradores, Para las aplicaciones Móviles usaremos FLUTTER, ya que sirve mucho para crear una experiencia multiplataforma para conductores.