Arquitectura de Datos Mínima para una App de Delivery

Contexto

Aplicar conceptos teóricos a un escenario realista permite fijar conocimientos sobre componentes y buenas prácticas de arquitectura de datos.

Consigna

Diseña una arquitectura de datos mínima para una app de delivery que incluya fuentes, almacenamiento, procesamiento, acceso y seguridad.

Paso a paso

1. Identifica las principales fuentes de datos (usuarios, pedidos, restaurantes)
2. Elige tecnologías de almacenamiento (bases SQL/NoSQL, data lakes, etc) y justificarlas
3. Define cómo se procesarán los datos (ETL, en tiempo real, batch, etc)
4. Indica qué herramientas de acceso o visualización usarías (API, Dashboards, etc)
5. Esquematiza la solución en un diagrama y justifica cómo se cubren las buenas prácticas (gobernanza, escalabilidad, flexibilidad)

Desarrollo de la arquitectura

1. Principales fuentes de datos

- **Usuarios:** datos personales, dirección, preferencias.
- **Pedidos:** detalles del pedido, estado, método de pago.

- **Restaurantes:** menú, ubicación, horarios, disponibilidad.

- **Repartidores:** ubicación en tiempo real, estado.

- **Eventos del sistema:** logs de actividad, errores, métricas.

2. Tecnologías de almacenamiento y justificación

- **Base de datos relacional (SQL):** para datos estructurados y transaccionales como usuarios, pedidos, restaurantes y repartidores. Ejemplo: PostgreSQL o MySQL.
  Justificación: Consistencia, integridad referencial y soporte ACID, necesario para transacciones.

- **Base NoSQL (documental o key-value):** para datos no estructurados o semi-estructurados como logs, historial de ubicaciones o eventos. Ejemplo: MongoDB o Cassandra.
  Justificación: Escalabilidad horizontal y flexibilidad en el esquema.

- **Data lake (opcional para versión escalable):** para almacenar grandes volúmenes de datos históricos, analíticos y de logs en bruto. Ejemplo: almacenamiento en la nube como AWS S3 o Azure Blob Storage.

3. Procesamiento de datos

- **ETL batch:** para mover datos de las bases operacionales a un data warehouse o data lake para análisis históricos y reportes (por ejemplo, procesos nocturnos).

- **Procesamiento en tiempo real (streaming):** para seguimiento en vivo de repartidores, actualizaciones de estado de pedidos y análisis de eventos. Ejemplo: Apache Kafka o AWS Kinesis.

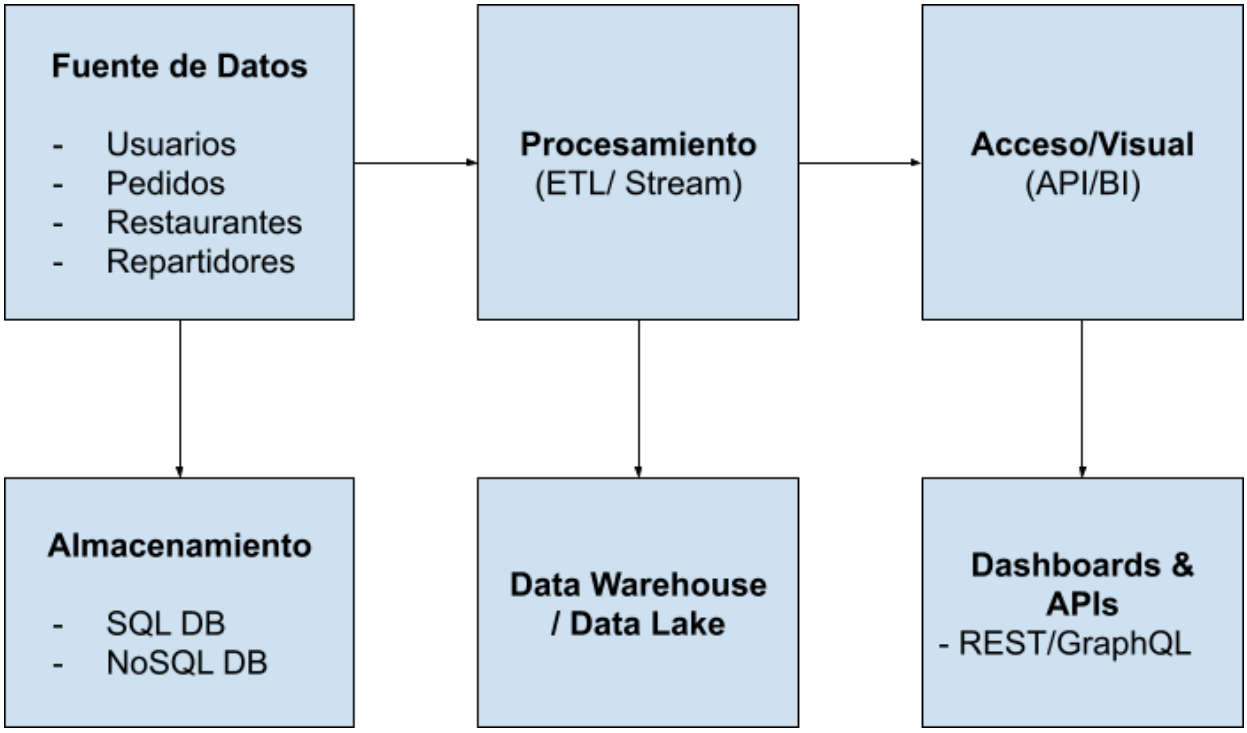
4. Herramientas de acceso y visualización

- **APIs REST/GraphQL:** para que el frontend de la app móvil y web consuman datos (pedidos, usuarios, restaurantes).

- **Dashboards de BI:** para monitoreo y análisis por parte del equipo (operaciones, marketing). Ejemplo: Power BI, Tableau, Metabase.

- **Herramientas de monitoreo y alertas:** para seguridad y salud del sistema (logs, métricas).

5. Esquema simplificado de la solución



Buenas prácticas cubiertas

- **Gobernanza:** roles definidos para acceso a datos, autenticación y autorización en APIs, cifrado de datos sensibles en reposo y tránsito.

- **Escalabilidad:** uso de bases NoSQL y procesamiento en streaming para escalar horizontalmente con el crecimiento.

- **Flexibilidad:** separación clara entre almacenamiento estructurado y no estructurado para facilitar adaptaciones futuras.

- **Seguridad:** protección de datos sensibles, monitoreo de accesos y backups regulares.