EJERCICIOS PRÁCTICOS

TWITTER

Daniel Blanco Calviño

REQUISITOS

- **Requisitos funcionales.**
  - Publicar posts. Pueden ir acompañados de contenido multimedia.
  - Seguir usuarios.
  - Feed con todas las publicaciones de los usuarios que seguimos.
  - Búsquedas sobre los posts.

- **Requisitos no funcionales.**
  - Alta disponibilidad.
  - Baja latencia al publicar posts.
  - Baja latencia al consultar el feed.

HIPÓTESIS Y ESTIMACIONES

- 200 millones de usuarios al día.

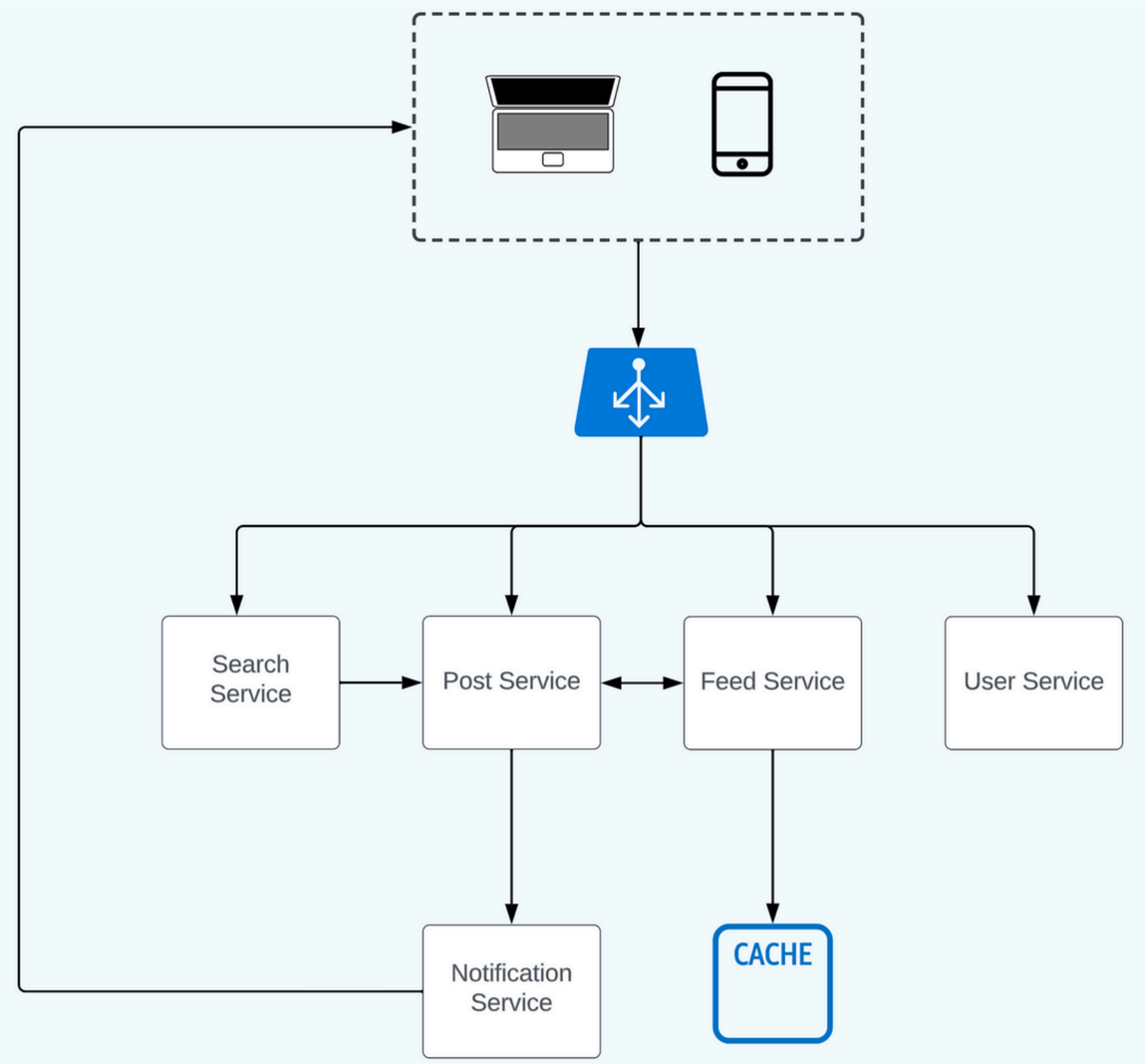
- Cada usuario publica 5 tweets de 200 caracteres de media al día.
  - 1000 millones de posts * 200 bytes = **0.2 terabytes al día.**

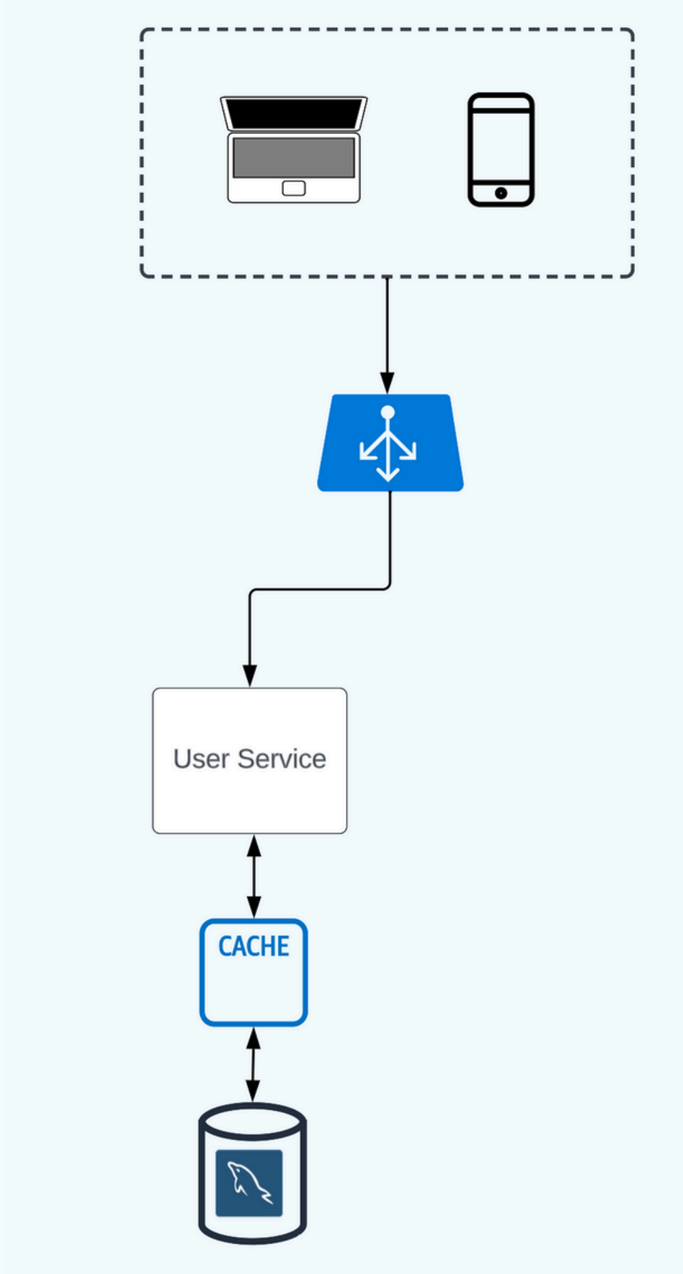
- Uno de cada diez posts tienen contenido multimedia. 1MB de media.
  - **100 terabytes adicionales al día.**

- Sistema **read-heavy. Ratio de 100:1** entre lecturas y escrituras.

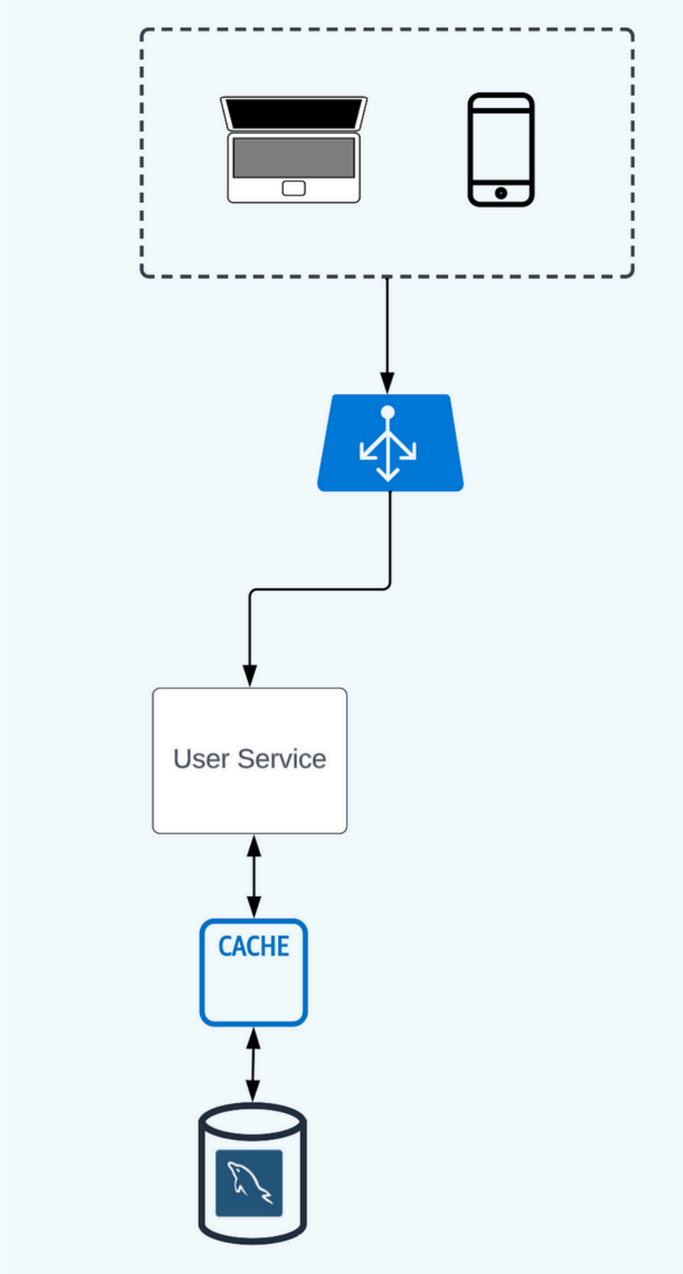
- Consultas por segundo (QPS)
  - 1000M posts * 100 = 100.000M consultas al día = **1.15M al segundo.**

DISEÑO ALTO NIVEL



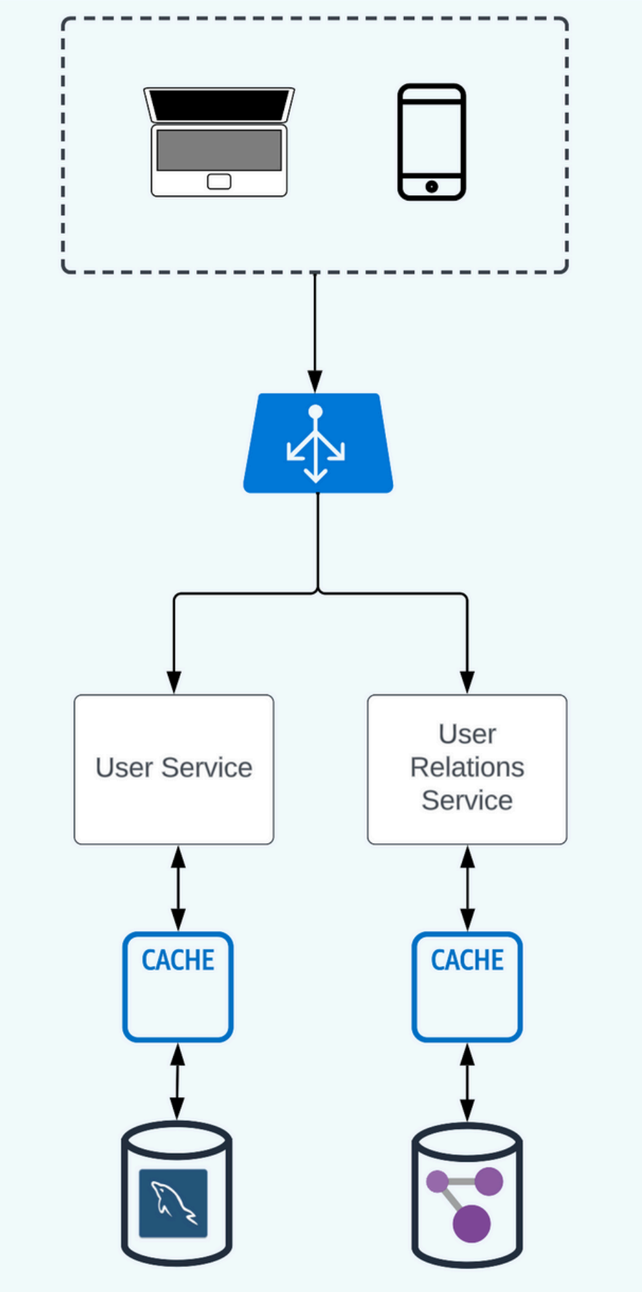


- **Información estructurada + pocas escrituras.**
  - BBDD relacional.
  - Replicación de datos + sharding en caso de ser necesario.



- **Información estructurada + pocas escrituras.**
  - BBDD relacional.
  - Replicación de datos + sharding en caso de ser necesario.

- **Problemas en la gestión de seguidores.**
  - Muchas escrituras. Difícil de escalar.
  - Consultas lentas al tener tantos registros.



- **Sistema orientado a grafos.**
  - Eficiente para almacenar y consultar relaciones entre usuarios.

PUBLICACIÓN DE POSTS Y FEED

- Necesitamos **baja latencia** en la **publicación** y en la obtención del **feed.**

- Método **pull.**
  - Al publicar un post sólo se almacena. **Al consultar el feed se construye de cero.**
  - Eficiente para publicar posts. Muy poco eficiente al consultar el feed.

PUBLICACIÓN DE POSTS Y FEED

- Necesitamos **baja latencia** en la **publicación** y en la obtención del **feed.**

- Método **pull.**
  - Al publicar un post sólo se almacena. **Al consultar el feed se construye de cero.**
  - Eficiente para publicar posts. Muy poco eficiente al consultar el feed.

- Método **push.**
  - Al publicar un post se actualiza el feed de los seguidores en la caché.
  - Muy rápido para obtener el feed. Muy lento en la publicación.
  - Recursos desperdiciados con usuarios inactivos.
  - Inasumible para usuarios con un número alto de seguidores.

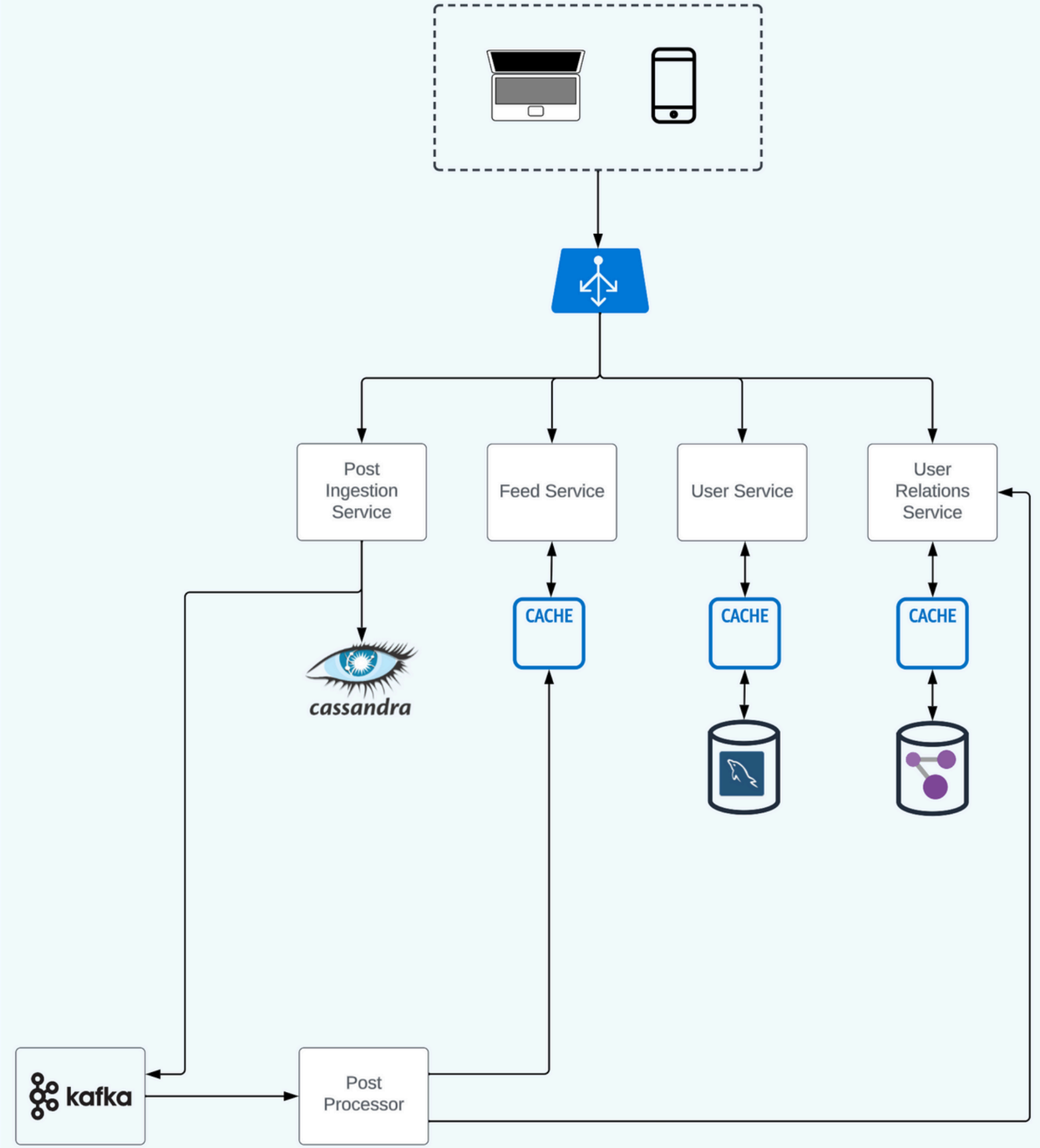
MÉTODO HÍBRIDO

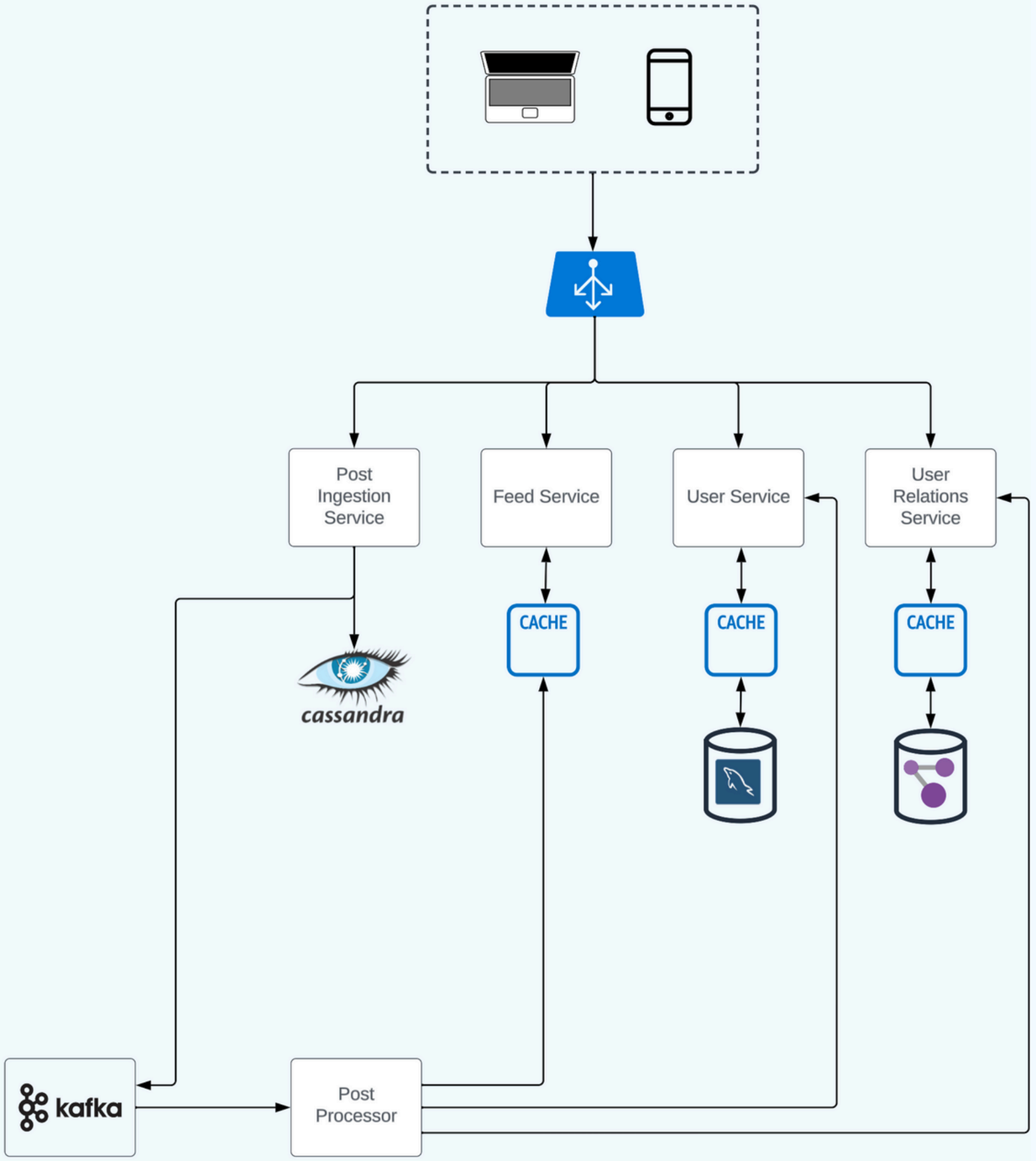
- El método **push** es el más apropiado para la mayoría de usuarios.
  - No se puede construir el feed de los usuarios a baja latencia en un sistema así.

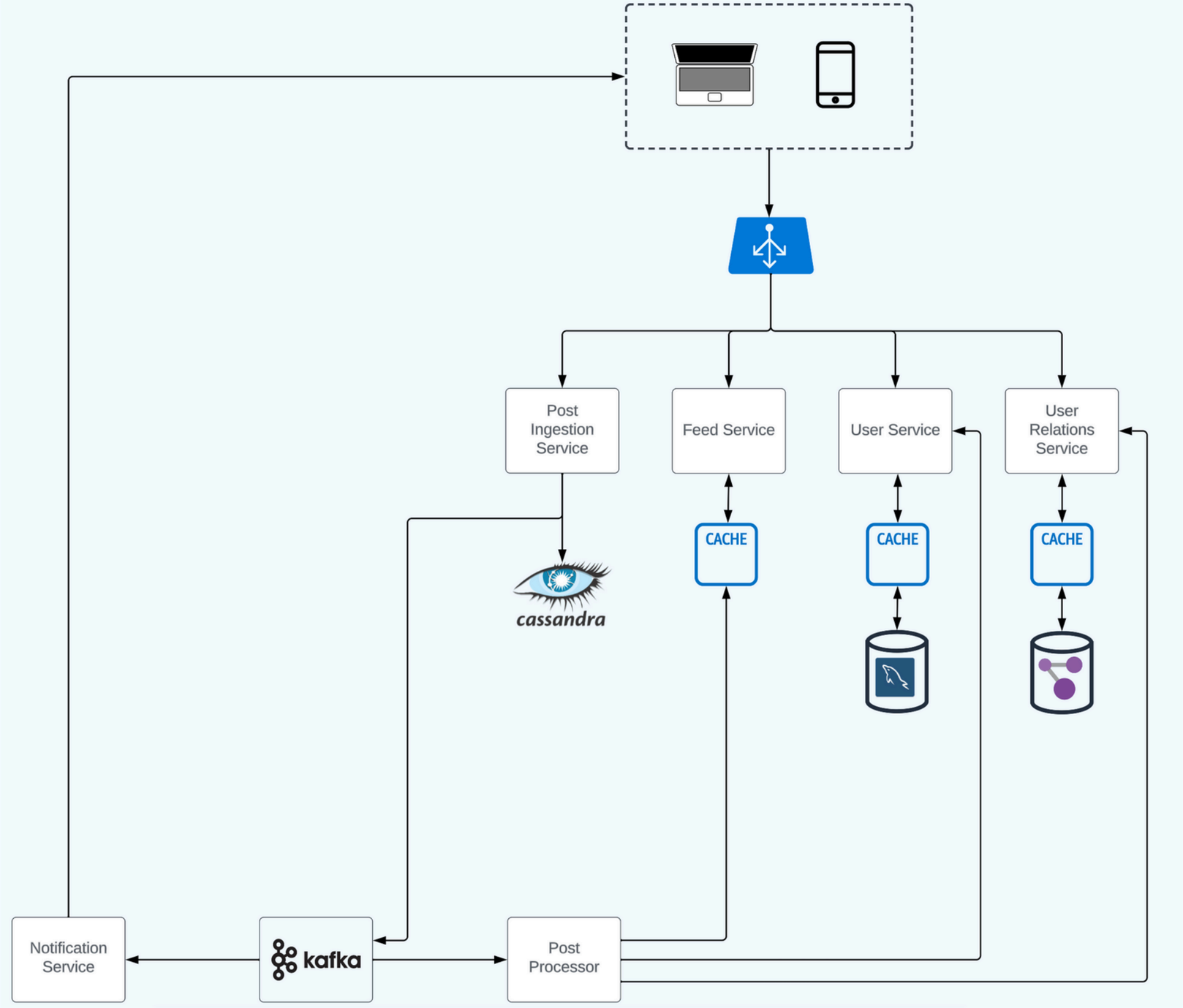
MÉTODO HÍBRIDO

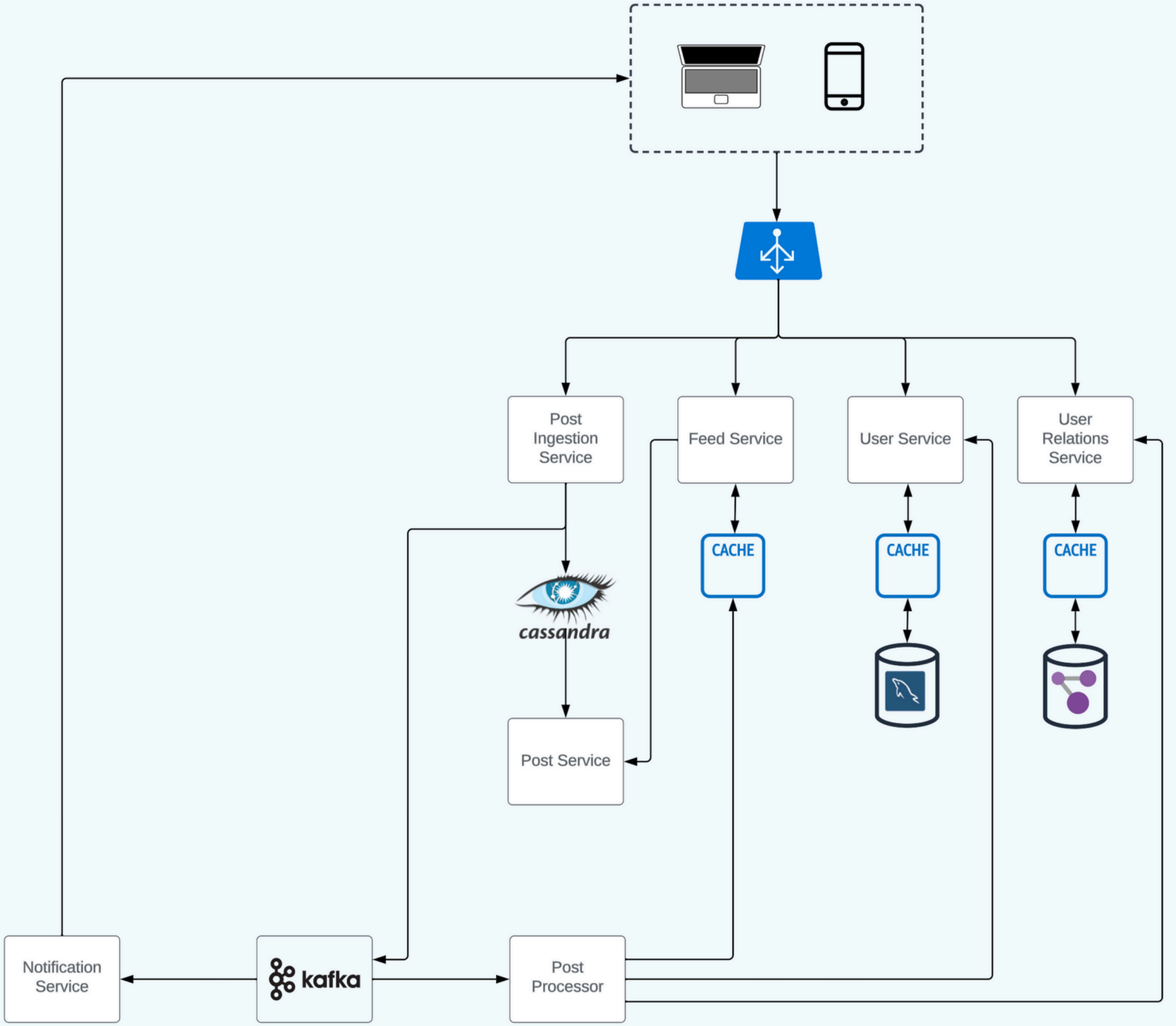
- El método **push** es el más apropiado para la mayoría de usuarios.
  - No se puede construir el feed de los usuarios a baja latencia en un sistema así.

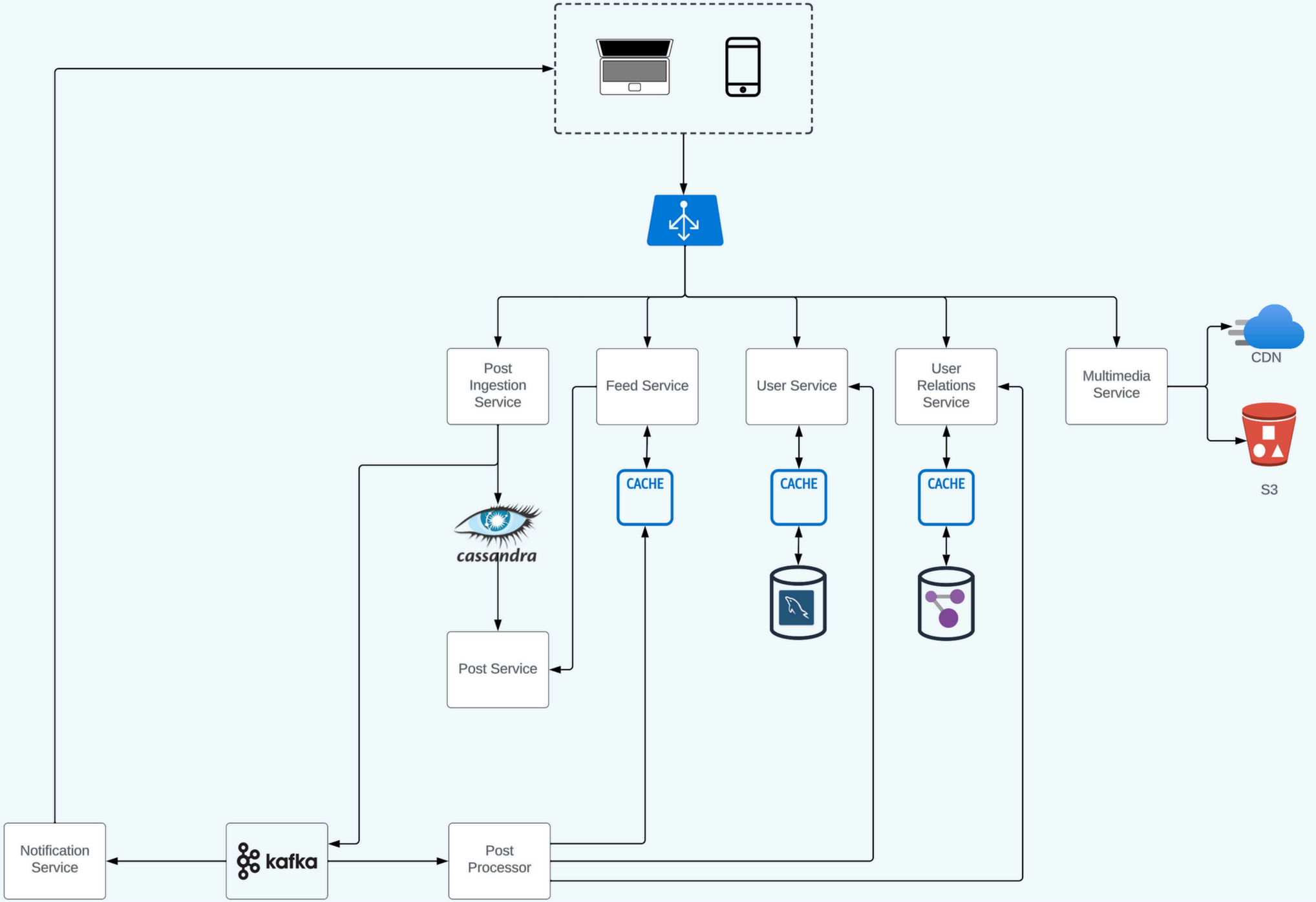
- Debemos tratar los casos especiales.
  - **Ignoraremos los usuarios inactivos.**
    - Cuando se conecten, se hace pull esa primera vez.
  - Utilizaremos el método **pull para los usuarios famosos.**

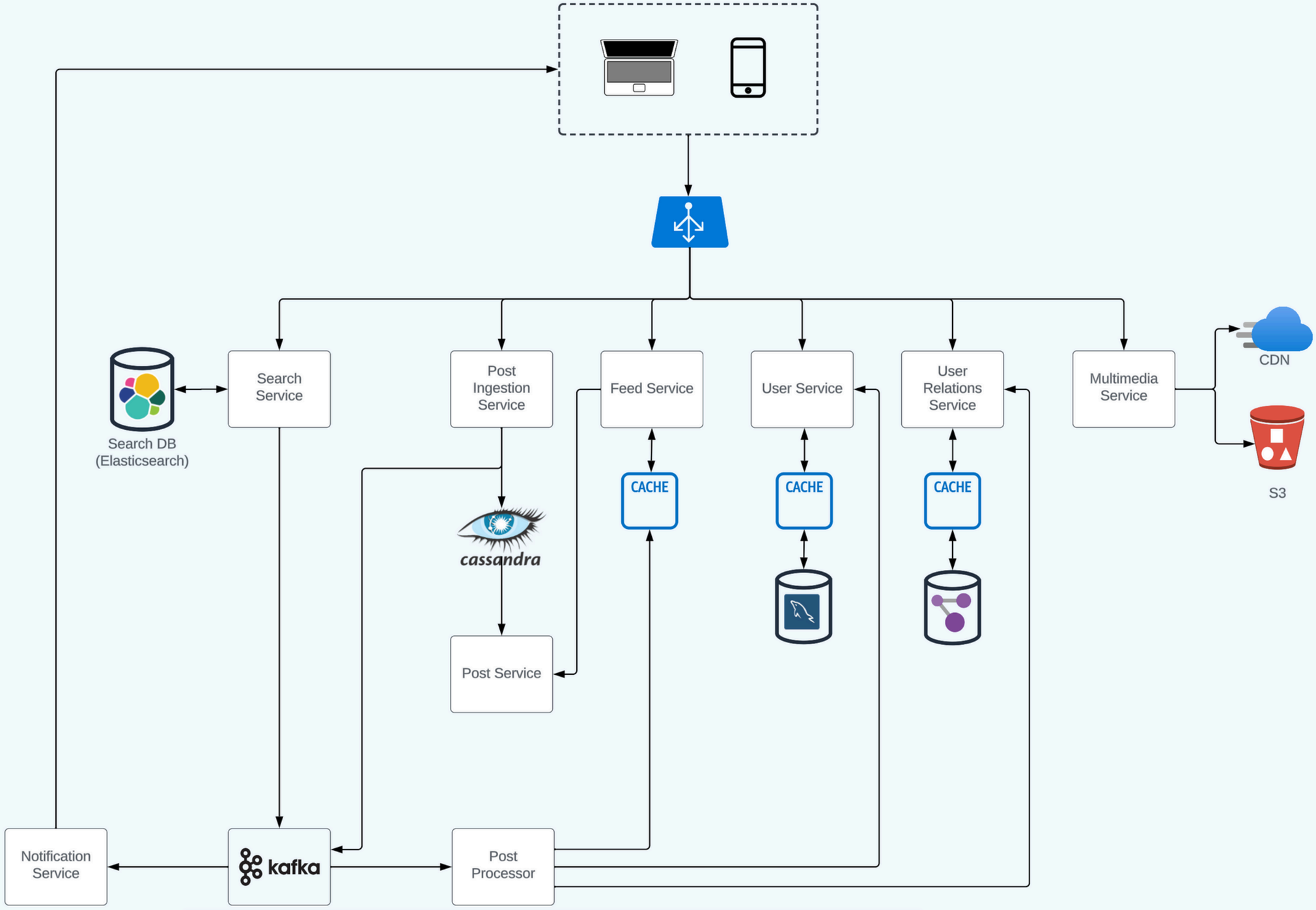

Post Ingestion Service
Feed Service
User Service
User Relations Service
CACHE
CACHE
CACHE
cassandra
kafka
Post Processor

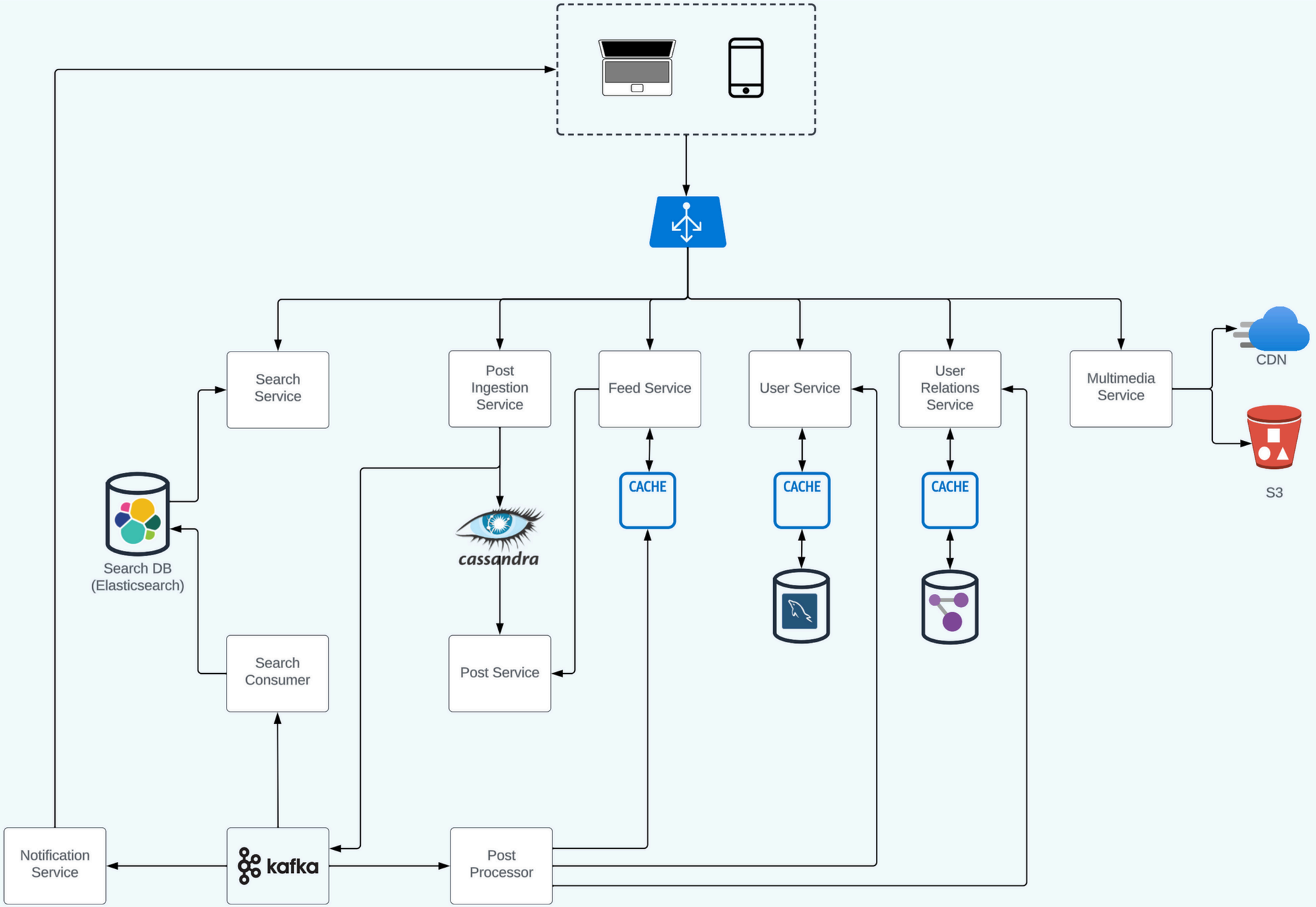

Post Ingestion Service
Feed Service
User Service
User Relations Service
CACHE
CACHE
CACHE
cassandra
kafka
Post Processor


Post Ingestion Service
Feed Service
User Service
User Relations Service
CACHE
CACHE
CACHE
cassandra
Notification Service
kafka
Post Processor


Post Ingestion Service
Feed Service
User Service
User Relations Service
CACHE
CACHE
CACHE
cassandra
Post Service
Notification Service
kafka
Post Processor


Post Ingestion Service
Feed Service
User Service
User Relations Service
Multimedia Service
CDN
S3
CACHE
CACHE
CACHE
cassandra
Post Service
Notification Service
kafka
Post Processor


Search Service
Post Ingestion Service
Feed Service
User Service
User Relations Service
Multimedia Service
Search DB (Elasticsearch)
CDN
S3
CACHE
CACHE
CACHE
cassandra
Post Service
Notification Service
kafka
Post Processor


Search Service
Post Ingestion Service
Feed Service
User Service
User Relations Service
Multimedia Service
CDN
S3
CACHE
CACHE
CACHE
Search DB (Elasticsearch)
cassandra
Search Consumer
Post Service
Notification Service
kafka
Post Processor