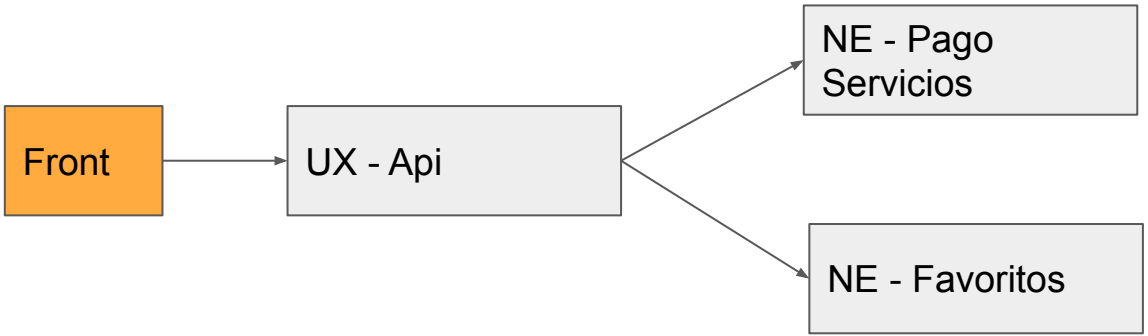
# Java 3- Reto integrador

*Contexto*:

Una empresa del sector bancario, que ya cuenta con soluciones digitales disponibles para sus clientes, requiere habilitar **el pago de servicios** en sus diferentes canales digitales, por lo que requiere realizar la implementación de las siguientes API's con las siguientes características.



# Detalle Técnico:

- Construir una **API Negocio de Pago de servicios** que exponga dos servicios:
  - Obtiene lista de servicios de la base de datos que se pueden pagar por canal (Ejemplo Código de canal: BM: Banca Móvil, BI: Banca por internet)
  - Realiza la transacción.
    - Recibirá como input los siguientes campos: código de servicio, número de suministro y monto a pagar.
    - Registrar la fecha y hora de la transacción.
    - Implementar validaciones de formato.
    - Escalar a 02 instancias de este servicio.
- Construir un **API Negocio de Favoritos** que exponga un servicio.
  - Registra el favorito.
    - Recibirá como input los siguientes campos: nombre y tipo de favorito, código de servicio y los almacenará en un repositorio.
- Construir una **API de Experiencia** que exponga dos servicios:
  - Consume endpoint del API Negocio de Pago de servicios, enviando el código de canal, para obtener los servicios que se pueden pagar por ese canal.
    - Antes de devolver la respuesta a front, almacenar la información en caché por un periodo de tiempo de 5 minutos.
  - Consume endpoint del API Negocio de Pago de servicios para realizar la transacción.
    - Antes de realizar la transacción, validar que el código de servicio se encuentre en la lista almacenada en caché.
    - En caso haya algún error, realizar 02 reintentos en intervalo de 02 segundos.
  - Como flujo alternativo, el cliente al momento de realizar la transacción puede guardar la operación como **Favorito.**
    - Si es el caso y la transacción es exitosa, debe consumir el endpoint de registrar del **API Negocio de Favoritos.**

## Conocimientos a aplicar:

- Arquitectura de microservicios
- Programación en reactiva con WebFlux.
- Spring Security con tokens JWT y oAuth2 con Keycloak.
- Uso de BD Sql y NoSQL .
- Contenerización de microservicios.
- Uso de Config Server y Eureka.
- Spring Cloud Gateway (opcional)
- Despliegue y orquestación en Kubernetes.

## Entrega:

- Subir el código a un repositorio github público.
- Subir al repositorio un archivo de evidencias de pruebas de reto. (formato excel,word u otro) .
- Agregar las indicaciones necesarias para realizar el despliegue y prueba del reto.
- Subir los scripts de creación de BD y generación de data.