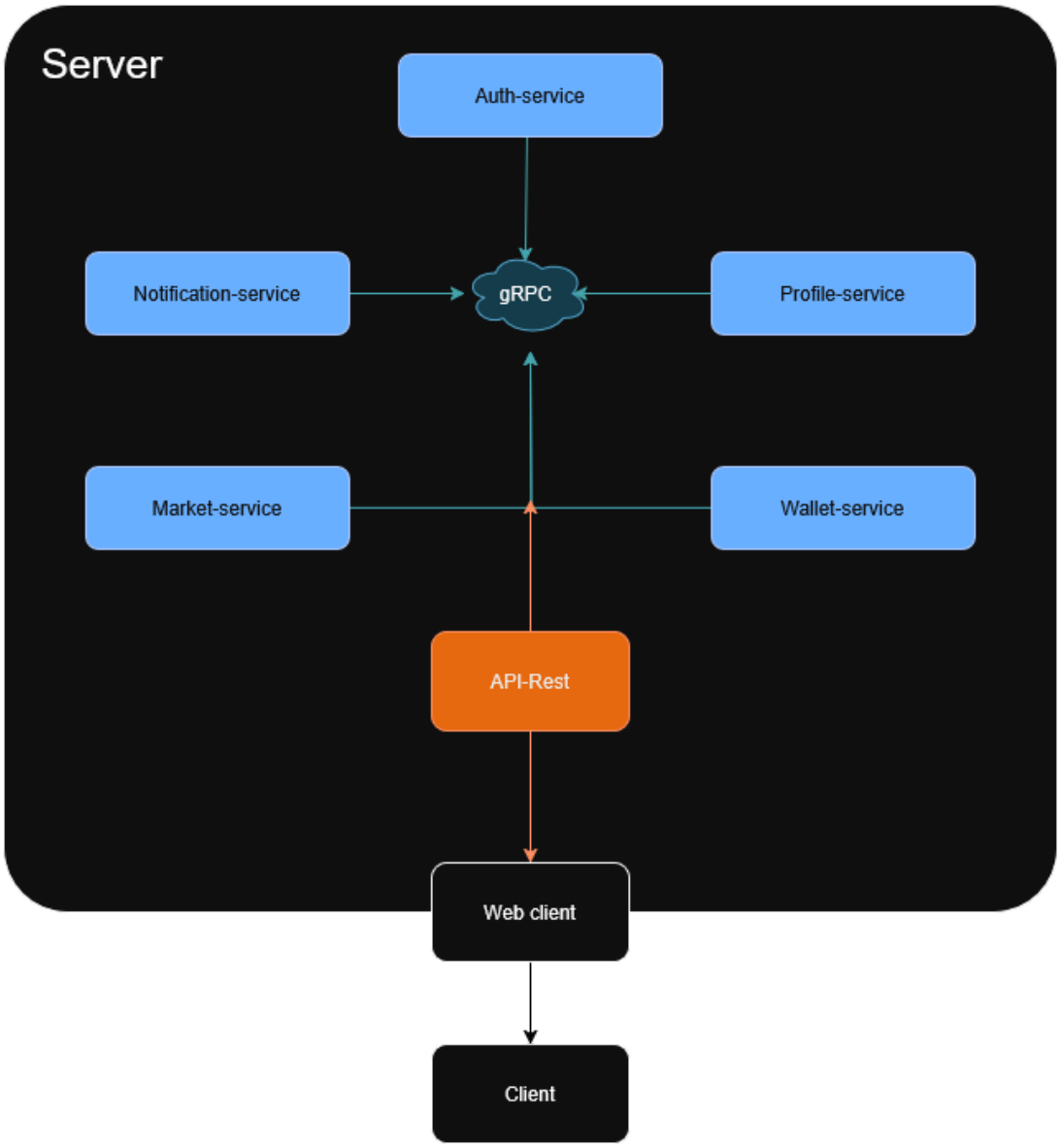
Arquitectura



Servicios

1. Auth-service:

a. Brindará las sesiones
b. Controlará las sesiones activas
c. Verificará las cookies de sesión de las requests
d. Genera códigos para los cambios de contraseña y verificación de email.

2. Profile-service:

a. Almacenará los datos del usuario: Nombre, Apellido, email, contraseña (hash), foto de perfil, dinero y wallet-id.

3. Notification-service

a. Mandará mail para confirmación de email
b. Mandará mail para cambiar la contraseña

4. Market-service

a. Devolverá el precio de una compañía
b. Devolverá el precio de muchas compañías
c. Devolverá los precios históricos de una compañía

5. Wallet-service

a. Creará una wallet por cada cliente
b. Las wallets almacenarán historial de compra / venta y las acciones compradas
c. Devolverá el valor de las acciones

6. Api-rest

a. Manejara todas las comunicaciones con el exterior
b. Brindará los servicios al cliente

7. Web client

a. Página de inicio
b. Página principal con todas las acciones

c. Página de compra / venta de acciones
d. Página de wallet
e. Página de perfil
f. Página de login / registro
g. Página de cambio contraseña
h. Página confirmación de email
i. Página de error

Comunicación

Para la comunicación entre servicios se utilizará **gRPC** para que haya una comunicación muy veloz entre servicios

Almacenamiento de data

Para almacenar los datos sin que se reinicien al actualizar el archivo de kubernetes usare PVCs con kubernetes

Frameworks de cada servicio y funcionamiento

Auth-service

1. **DB**: Para manejar las sesiones hay que almacenarlas en una base de datos, se usará **Valkey**
2. **Back-end**: Se usará spring Boot con **JWT** para generar las claves de sesión y manejarlas. Además para los login con google se usará **OAuth2 client**

Profile-service

1. **DB**: Para guardar toda la información de los usuarios se usará **PostgreSQL**
2. **Fotos**: Las fotos de perfil se almacenarán en el disco, en /var/lib/profile-service/images/. Para que se puedan identificar el nombre será así {userId.jpg}

Notification-service

1. Para hacer un servicio de mensajería con mails se requiere mucho tiempo y experiencia, por lo cual se dejará para el final, mientras tanto se usará **SendGrid**

Market-service

1. Para el servicio de market service se creará una api con Spring boot la cual usará la api de **IEX Cloud** para obtener la información de la bolsa con un retraso de 15 min.

Wallet-service

1. Se usará una base de datos **postgreSQL** ya que no será una app de uso real. Muy importante que en la api principal después maneje los datos de manera encapsulada para un posible paso a transacciones reales.

Api-rest

1. Se usará **Spring Boot** e intentaré hacer un código muy escalable y robusto.

Web client

1. Se usará **Next.js** para hacer una interfaz amigable y responsiva. Y en un futuro se podría hacer una app móvil con **React Native** por la facilidad de escalado.

Kubernetes

1. Cluster con K3s
2. Los servicios usarán namespaces para controlar los despliegues (dev, prod)
3. Cada servicio tendrá que tener su despliegue propio para que de este modo sea más escalable
4. Se configurara un ingress controller para la mediación entre el exterior y la api principal (Traefix)
5. Los servicios que requieran de almacenar datos utilizaran PVCs para persistencia de datos
6. Para credenciales en servicios externos como IEX Xloud se usarán ConfigMaps y para claves privadas se usará Secrets
7. Para los despliegues se usará CI/CD con GitHub Actions usando kubectl.

   Extra
   Implementación de Horizontal Pod Autoscaler y copias de seguridad.
   Seguridad con NetworkPolicies y PodSecurityContext.
   Servicio de mail privado