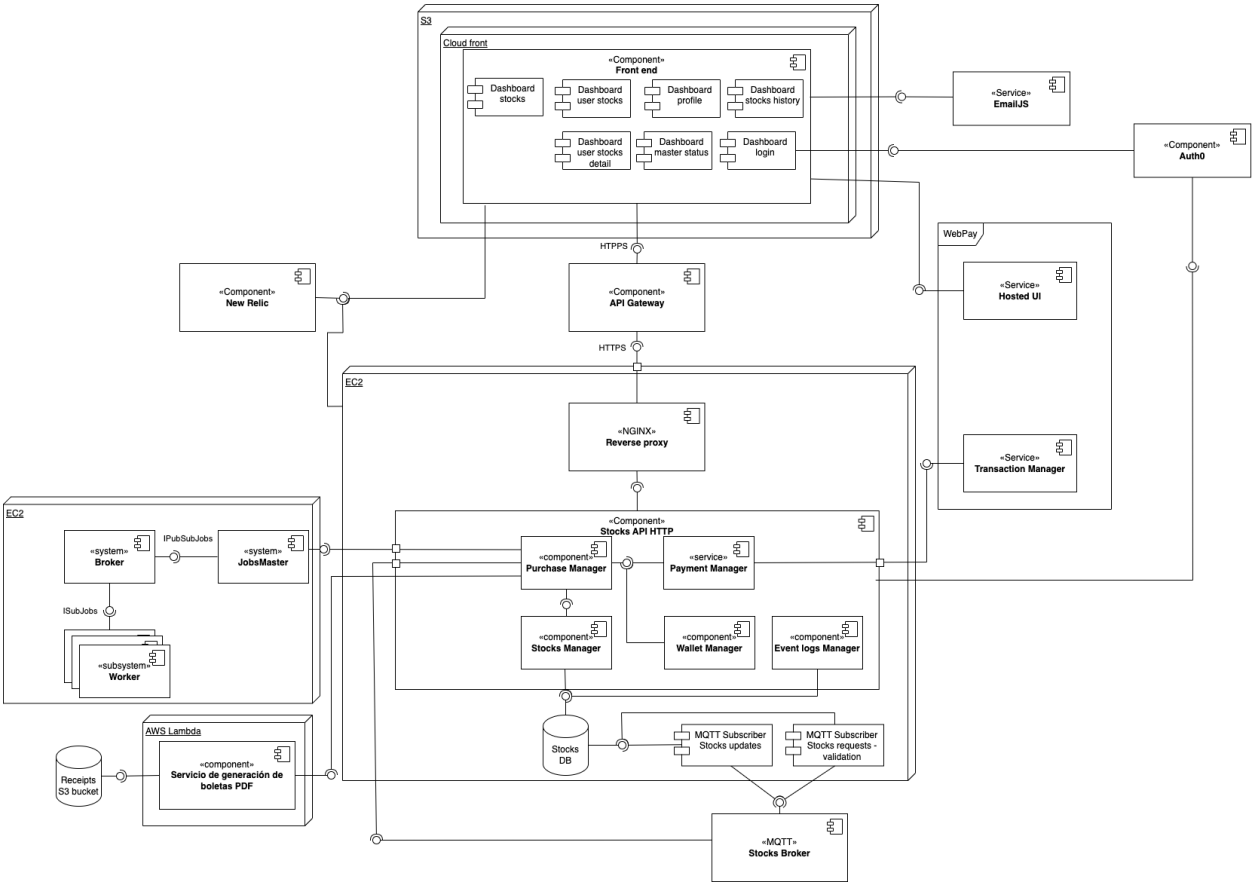
Pontificia Universidad Católica de Chile
Departamento de Ciencias de la Computación
IIC2173- Arquitectura de Sistemas de Software

Proyecto Semestral Entrega 2: Diagrama UML

Grupo 27

---

1. Diagrama UML



2. Explicación

Frontend

- Almacenado en AWS S3 y distribuido a través de CloudFront.
- Contiene el componente `Front End`, el cual incluye los siguientes *dashboards*:
  - `Dashboard stocks`: Página de inicio de la aplicación.

- `Dashboard user stocks:` Muestra todos los *stocks* recibidos desde el broker.
- `Dashboard profile:` Presenta el correo electrónico, nombre y billetera del usuario.
- `Dashboard stocks history:` Visualiza el historial de eventos de los *stocks*.
- `Dashboard user stocks detail:` Muestra los *stocks* adquiridos por el usuario.
- `Dashboard master status:` Indica si el servidor de `MasterJob` está activo.
- `Dashboard login:` Permite iniciar sesión en el sistema.

Autenticación

- El *frontend* se conecta con el componente `Auth0`, el cual es el encargado de la autenticación de usuarios en el sistema. Cuando un usuario inicia sesión desde el *frontend*, es redirigido al *Hosted UI* de Auth0, donde se realiza el proceso de autenticación. Una vez verificado, Auth0 entrega un token JWT que el *frontend* utiliza para realizar solicitudes autenticadas. Estas solicitudes llegan al *backend* alojado en la instancia EC2, donde el token es validado mediante una consulta a los servidores de`Auth0`.

API Gateway

- El *API Gateway* actúa como el punto de entrada para todas las solicitudes HTTP que llegan al sistema.En términos de interconexión, recibe las solicitudes provenientes del *frontend* y, tras pasar por los procesos de validación, las redirige hacia el *Reverse Proxy* **(NGINX)** si son consideradas válidas.

EmailJS

- El servicio *EmailJS* se encarga de enviar correos electrónicos desde el *frontend* sin tener que conectarse con el *backend*. En el contexto del proyecto esto se utiliza para confirmar la creación de un usuario, recibir confirmación de compras y enviar las boletas.

***Backend* (EC2)** El *backend* está alojado en una instancia EC2 de AWS, donde se ejecutan los siguientes componentes y servicios:

- **NGINX Reverse Proxy:** Controla el tráfico entre los clientes y los servicios internos. Recibe las peticiones provenientes del *API Gateway* y, entre sus funciones, realiza balanceo de carga. Redirige las solicitudes hacia el servicio `Stocks API HTTP`, también alojado en la instancia EC2.
- **Stocks API HTTP:** Es el núcleo del *backend* del sistema, compuesto por múltiples microservicios y componentes:
  - `Purchase Manager:` Componente encargado de gestionar la compra de acciones. Valida que el usuario disponga de saldo suficiente, actualiza las bases de datos y registra los logs correspondientes. Solicita pagos al `Payment Manager`, consulta la disponibilidad de acciones al `Stocks Manager`, solicita descuentos de fondos al `Wallet Manager` y envía eventos al `Event Logs Manager`. Además, se comunica con el `JobMaster` para enviar la información de la compra y permitir la estimación correspondiente.
  - `Payment Manager:` Servicio responsable del procesamiento de pagos. Trabaja directamente con el `Transaction Manager` de Webpay y comunica los resultados de las transacciones al `Purchase Manager`.
  - `Stocks Manager:` Componente encargado de gestionar las acciones, incluyendo stock disponible y valores actuales. Atiende las consultas del `Purchase Manager` y se comunica con la base de datos `Stocks DB`.
  - `Wallet Manager:` Administra el saldo del usuario, consultando y modificando su balance. Interactúa con el `Purchase Manager` para validar y ejecutar compras realizadas mediante el medio de pago *wallet*.

    - `Event Logs Manager`: Responsable de gestionar y registrar todos los eventos y transacciones relacionados con los *stocks* y el sistema en general.

- **Stocks DB**: Base de datos donde se guardan las acciones, wallets, etc. Es accedida por los dos MQTT Subscribers y el componente de `Stocks Manager`

- **MQTT Subscriber - Stocks Updates:** Este componente recibe mensajes en tiempo real desde el broker sobre actualizaciones de acciones. Al recibir un mensaje, se conecta con la base de datos para actualizar la información correspondiente y registra un evento en el historial del sistema. Está tambien conectada al `Stocks Broker`.

- **MQTT Subscriber - Stocks request - validation:** Este componente se conecta al broker para gestionar solicitudes relacionadas con la compra de acciones. También se comunica con la base de datos, donde registra información sobre las solicitudes, transacciones, usuarios y eventos. Está tambien conectada al `Stocks Broker`.

MQTT Stocks Broker Es el *broker* de mensajes MQTT, encargado de permitir la comunicación en tiempo real y de forma asincrónica.

Broker, JobsMaster y Worker (EC2): Estos tres componentes se encuentran desplegados en una misma instancia EC2 de AWS. Conforman el sistema de procesamiento asincrónico del backend.

- **Broker:** Es el componente responsable de publicar trabajos que deben ser procesados de forma asincrónica. Recibe tareas desde el `JobsMaster` y las envia al `Worker`.

- **JobsMaster:** Cumple la función de gestionador de trabajos. Administra una cola de tareas y se encarga de planificar y distribuir dichas tareas hacia los `Worker`. Envía las tareas que recibe al componente `Broker`, el cual se encarga de comunicarlas a los `Worker`.

- **Worker:** Es el componente encargado de ejecutar las tareas como estimacion de ganancias.

Generación de Boletas (AWS Lambda) Este componente utiliza funciones `Lambda` para la generación automática de boletas en formato PDF una vez completada una compra. Se activa mediante eventos provenientes del `Purchase Manager` o del `Worker`, y guarda las boletas generadas en el bucket `Receipts S3` para su posterior descarga por parte del usuario.

Sistema de Pagos (WebPay) Este subsistema permite a los usuarios completar el proceso de pago con tarjeta de crédito o débito. Está compuesto por una interfaz de usuario y un componente que gestiona el ciclo completo de las transacciones.

- **Hosted UI:** Interfaz visual que permite al usuario realizar el pago. Se comunica tanto con el `Frontend` como con el `Payment Manager` para iniciar y gestionar la operación de pago.

- **Transaction Manager:** Componente encargado de supervisar las distintas etapas de las transacciones (inicio, éxito o fallo). Se comunica directamente con la `Hosted UI` y el `Payment Manager`, y notifica a otros servicios para completar o abortar procesos según el resultado de la transacción.

Monitoreo (New Relic) Servicio encargado de recolectar métricas, monitorear el rendimiento, y detectar errores en tiempo real. Puede integrarse con diversos componentes del sistema como `NGINX` y `Stocks API HTTP`, facilitando la observabilidad del sistema tanto para desarrolladores como administradores.