Brainstorm

“<https://mm.tt/app/map/3164794583?t=AZMBY5lZt1>”

El mundo de las finanzas está en constante cambio, con mercados volátiles y oportunidades emergentes. En este ambiente dinámico, es crucial mantenernos ágiles y abiertos a nuevas ideas y enfoques de inversión.

Es por esto por lo que, la problemática que se va a tratar, es el diseño de una arquitectura de software vinculada a la inversión.

Las ideas que surgen a priori en relación con el tema, son las expuestas a continuación, posteriormente explicadas.

* *Inversión:* La inversión es el acto de asignar recursos para la compra o creación de activos o de capital, es toda materialización de medios financieros en bienes que van a ser utilizados en un proceso productivo de una empresa o unidad económica.Esto es; la inversión implica colocar capital en un proyecto o activo con propósito de obtener ingresos o ganancias futura.
* *Cortoplacismo:* Estrategia de inversión basada en responder a los cambios en los precios más que en el valor fundamental.
* *Trading*: comercialización de instrumentos financieros con el objetivo de obtener beneficios.
* *Bolsa de valores:* es un mercado abierto donde se compran y venden productos financieros, facilitando el encuentro entre demandantes y oferentes.
* *Mercado de valores:* Espacio físico o virtual de compra y venta de instrumentos financieros.
* *Acciones:* Partes en las que se decide el capital de una empresa.
* *Índice:* Grupo o canasta de valores, derivados u otros instrumentos financieros, que representa y mide el desempeño de un mercado específico.
* *Divisas:* Unidad monetaria que se compra o vende con el propósito de que su valor aumente en comparación a la moneda de origen.
* *Forex:* Mercado de divisas.
* *Derivados:*Es un producto financiero cuyo valor depende de la evolución del precio de otro activo (activo subyacente).
* *Criptomonedas:* activo digital que emplea un cifrado criptográfico para garantizar su titularidad y asegurar la integridad de las transacciones, y controlar la creación de unidades adicionales.
* *Futuros:* Productos financieros derivados, esto es, un contrato por el que se acuerda un intercambio de determinado activo subyacente en una fecha futura predeterminada, a un precio convenido de antemano
* *Apalancamiento:* Relación entre capital propio e inversión total en una operación financiera.
* *Broker:* Agente intermediario en operaciones financieras o comerciales que percibe una comisión por su intervención*.*
* *Velas:* representación gráfica del precio de mercado financiero en forma de velas

Diseño del DAaaS

### Definición la estrategia del DAaaS

El objetivo general es dar un servicio en forma de mailchimp o chatbot que envíen al usuario señales de inversión y que además pueda predecir los valores futuros según los análisis y estrategias determinados para anticipar la opción adecuada, según el tipo de inversión, el capital disponible, el mercado, etc.

Se atenderán estas peticiones de usuarios mediante un Google Formulary, donde se recogerán los diferentes datos y preferencias, de cada usuario.

Con estos datos, sabiendo el capital disponible, además de otro tipo de información recogida en el formulario, se enviará vía email o plataforma de comunicación, bajo petición del usuario, las distintas señales de inversión, dónde invertir según su capital, flexibilidad, objetivos y con qué frecuencia.

Para diseñar la arquitectura de modelo de microservicios de una solución que incluya un chatbot o Mailchimp para el envío de señales de inversión y predicciones de valores, es importante considerar varios aspectos, incluyendo la escalabilidad, la modularidad, la seguridad y la integración con sistemas externos.

### Los componentes principales son:

* Interfaz de Usuario (Chatbot o Mailchimp): Interfaz a través de la cual los usuarios interactúan con un servicio. Permite a los usuarios solicitar señales de inversión, recibir predicciones, etc.
* La capa de negocio contiene la lógica de negocio del servicio, incluyendo la generación de señales de inversión y predicciones de valores. Se encarga de procesar las solicitudes de los usuarios y generar respuestas relevantes.
* Capa de Acceso a Datos: Gestiona el acceso a los datos necesarios para generar las señales de inversión y predicciones de valores. Puede incluir integraciones con fuentes de datos externas, como feeds de mercado, bases de datos financieras, APIs de servicios de datos, etc.
* Motor de Predicción: Componente encargado de ejecutar algoritmos de predicción para generar predicciones de valores futuros. Puede utilizar técnicas de aprendizaje automático u otros métodos de análisis predictivo.
* Servicios de Comunicación: Gestiona la comunicación con los usuarios a través del chatbot y Mailchimp y envía señales de inversión y predicciones de valores a los usuarios de manera oportuna y personalizada.
* Gestión de Usuarios y Autenticación: Gestiona la autenticación de usuarios y la gestión de perfiles de usuario además de permitir a los usuarios registrarse, iniciar sesión y gestionar sus preferencias de comunicación.

A continuación se muestra la arquitectura propuesta

+---------------------------+

| Interfaz de Usuario |

+---------------------------+

|

v

+---------------------------+

| Capa de Negocio |

+---------------------------+

|

+----------------+----------------+

| |

+-----------v-----------+ +------------v-----------+

| Capa de Acceso a Datos | | Motor de Predicción |

+-----------------------+ +------------------------+

| |

+------------+--------+------------+

| |

+-----------------v--+ +--v------------------+

| Servicios de | | Gestión de Usuarios |

| Comunicación | | y Autenticación |

+---------------------+ +---------------------+

De otro modo, se realizaría un modelo DaaS integrado con SOA, que será la propuesta de uso para este servicio. Sus componentes principales serían:

* + Servicios de Negocio (SOA):Estos servicios representan las funcionalidades principales del sistema, como la generación de señales de inversión, predicción de valores, gestión de usuarios, etc. Este servicio es responsable de la lógica de negocio relacionada con la generación de señales de inversión y predicciones de valores. Puede incluir algoritmos de análisis de datos, modelos de predicción, y lógica para determinar cuándo enviar señales de inversión.Cada servicio está diseñado para realizar una tarea específica y puede ser invocado por otros servicios o aplicaciones a través de interfaces bien definidas. Se necesitará un entorno de desarrollo que admita la implementación de algoritmos de análisis de datos y aprendizaje automático, como Python con bibliotecas como Pandas, Scikit-learn y TensorFlow.
  + Servicios de Datos (DaaS): Estos servicios proporcionan acceso a conjuntos de datos específicos, como datos históricos de mercado, información financiera, datos de usuarios, etc. Los datos se exponen a través de APIs estandarizadas, lo que permite a los servicios y aplicaciones acceder y utilizar los datos de manera consistente y segura. Este servicio almacena y proporciona acceso a los datos necesarios para generar señales de inversión y predicciones de valores. Se necesitará un sistema de gestión de bases de datos para almacenar datos históricos y en tiempo real, como PostgreSQL, MongoDB o Elasticsearch.
  + Servicios de Integración (SOA):Estos servicios facilitan la integración entre los servicios de negocio y los servicios de datos.Pueden ser responsables de realizar operaciones de lectura y escritura en los servicios de datos, transformar datos según sea necesario y proporcionar datos procesados a los servicios de negocio. Este servicio se encarga de integrar el servicio de negocio y el servicio de datos, coordinando la comunicación entre ellos y asegurando que los datos necesarios estén disponibles para la generación de señales de inversión y predicciones de valores.Requerimos de un sistema de mensajería o middleware para facilitar la comunicación entre los diferentes servicios, como RabbitMQ, Apache Kafka, en este caso, o AWS SQS.
  + Servicios de Presentación (SOA):Este servicio proporciona interfaces de usuario para que los usuarios interactúen con el sistema, como un chatbot para comunicarse con los usuarios o una plataforma de correo electrónico (Mailchimp) para enviarles señales de inversión y predicciones de valores. Se usarán herramientas de desarrollo web o móvil para crear la interfaz de usuario, como React.js para aplicaciones web o Flutter para aplicaciones móviles.

### 

### 

### 

### 

### La arquitectura propuesta:

+-------------------------------------+

| Servicios de Presentación |

+-----------------+-------------------+

|

v

+-------------------------------------+

| Servicios de Negocio |

+-----------------+-------------------+

|

+---------------------------------+----------------------------------+

| | |

v v v

+------------------+ +------------------+ +------------------+

| Servicios de | | Servicios de | | Servicios de |

| Datos (DaaS) | | Integración | | Integración |

+------------------+ +------------------+ +------------------+

-Características Principales:

* Desacoplamiento: Los servicios de negocio están desacoplados de los servicios de datos, lo que permite una mayor flexibilidad y reutilización de los servicios.
* Reutilización: Los servicios de negocio y los servicios de datos pueden ser reutilizados por múltiples aplicaciones y sistemas, lo que promueve la interoperabilidad y la eficiencia en el desarrollo de software.
* Estándares de Interfaz: Se definen estándares de interfaz para los servicios de negocio y los servicios de datos, lo que facilita la integración y la comunicación entre los diferentes componentes del sistema.
* Gestión de Datos Centralizada: Los datos se gestionan de forma centralizada a través de los servicios de datos, lo que facilita el acceso y la gestión de datos en todo el sistema.
* Escalabilidad: La arquitectura permite escalar los servicios de negocio y los servicios de datos de forma independiente según sea necesario para satisfacer los requisitos de rendimiento y escalabilidad del sistema.

Flujo de datos:

El servicio de negocio consulta datos históricos y en tiempo real del servicio de datos.

Utilizando algoritmos de análisis de datos y modelos de predicción, el servicio de negocio genera señales de inversión y predicciones de valores.

Estas señales y predicciones se envían al servicio de presentación para su distribución a los usuarios.

El servicio de presentación utiliza un chatbot o Mailchimp para enviar las señales y predicciones a los usuarios a través de mensajes de chat o correos electrónicos.

**Recolección de datos:**

* ***Fuentes de datos****:* Para la elaboración de este proyecto se seleccionarán distintas fuentes de datos para la generación de estas señales.

Se tendrán en cuenta como principales fuentes de datoslos análisis de los diferentes ***datasets:***

*“*[*https://www.kaggle.com/datasets/taipanda9686/real-time-social-sentiment-for-stocks-crypto*](https://www.kaggle.com/datasets/taipanda9686/real-time-social-sentiment-for-stocks-crypto)*”*

*“*[*https://www.kaggle.com/datasets/zsinghrahulk/crypto-currency-bitcoin-and-ethereum-data*](https://www.kaggle.com/datasets/zsinghrahulk/crypto-currency-bitcoin-and-ethereum-data)*”*

*“*[*https://www.kaggle.com/datasets/saeedaghasoleimani/forex-fundumental-news-for-usd*](https://www.kaggle.com/datasets/saeedaghasoleimani/forex-fundumental-news-for-usd)*”*

*“*[*https://www.kaggle.com/datasets/tanay001/nseindia-futures-options-daily*](https://www.kaggle.com/datasets/tanay001/nseindia-futures-options-daily)*”*

*“*[*https://www.kaggle.com/datasets/soumendraprasad/stock*](https://www.kaggle.com/datasets/soumendraprasad/stock)*”*

*“*[*https://www.kaggle.com/datasets/saketk511/2019-2024-us-stock-market-data*](https://www.kaggle.com/datasets/saketk511/2019-2024-us-stock-market-data)*”*

*“*[*https://www.kaggle.com/datasets/paultimothymooney/stock-market-data*](https://www.kaggle.com/datasets/paultimothymooney/stock-market-data)*”*

*“*[*https://www.kaggle.com/datasets/iolalla/ibex3519942020*](https://www.kaggle.com/datasets/iolalla/ibex3519942020)*”*

*“*[*https://www.kaggle.com/datasets/andrewmvd/sp-500-stocks?resource=download*](https://www.kaggle.com/datasets/andrewmvd/sp-500-stocks?resource=download)*”*

**Procesamiento de datos:**

Para replicar un escenario real, se utilizarán todos los dataset anteriormente citados, además de los derivados mediante Scrapy los mencionados a continuación. Con todos estos datos, se generarán tablas anonimizadas en un Data Lake en MongoDB, además de los datos autogenerados por el propio sistema de mensajería de Kafka para simular eventos en tiempo real, que son controlados de manera completamente aleatoria.

*“*[*https://www.marketwatch.com/markets/us*](https://www.marketwatch.com/markets/us)*”*

*“*[*https://www.metatrader4.com/en*](https://www.metatrader4.com/en)*”, “*[*https://www.binance.com/*](https://www.binance.com/)*”*

Las fases de la arquitectura, las clasificaremos en fases:

* Data: Se incluyen los datos en bruto descargados desde Kaggle, ficheros que sirven para crear el modelo de datos de entrenamiento. Los datos adicionales se conectarán vía API de Google BigQuery, data lake o bucket de S3, u otras base de datos creadas para simular un escenario real, tanto para los datos de testing, como los datos simulados.
* Collect: En ella se usará la plataforma Kafka,para simular datos entrantes nuevos. Estos eventos se pueden envíar a otros sistemas, base de datos PostgreSQL y Spark en Streaming, mediante un conector consumer.
* Process: Fase de procesamiento donde se diferencian dos flujos, el de streaming donde los datos serán procesados en tiempo real, para luego generar y agregar nuevos datos y almacenarlos.

El segundo flujo, batch o por lotes, realizará el propio entrenamiento con datos históricos y entrantes para otros análisis.

* Store: Esta fase consiste en el almacenado, los datos agregados y mejorados se harán disponibles en diferentes formatos, según la disponibilidad y el valor de los propios. Se utilizarán sistemas distribuidos, HDFS para los históricos, y para uso interno, se guardarán en la base de datos en formato NoSQL (MongoDB).
* Visualize: Fase de exploración de datos con el fin de monitorizar los recursos internos de cada pieza clave, desde la ingesta, al procesado, así como la explotación de cada recurso.

Los datos recogidos en el Data Lake, serán depurados mediante herramientas automatizadas para el mismo fin, además del uso de pandas de python para eliminar valores nulos o duplicados.

Es importante tener en cuenta la escalabilidad y la eficiencia del proceso, dado el gran volumen y la variedad de datos que se manejan.

Una vez depurados los datos serán integrados en una base de datos NoSQL, que relacionamos con la que tiene los datos de los usuarios y así, poder enviarles las señales correctas según las peticiones solicitadas.

El procesamiento de datos y su transformación se realizarán con el framework de computación en clúster construido en Scala, Apache Spark.Es un motor rápido y general para el procesamiento de datos en paralelo a gran escala, amplía el popular modelo MapReduce. Podemos decir que una de las características más importantes de Spark es que funciona en la memoria. Este hecho implica que es más eficiente realizar cálculos como algoritmos iterativos, consultas interactivas y procesamiento de secuencias debido a que se evita el cuello de botella de lectura / escritura del disco.

Se utilizará la plataforma Kafka para poder recoger y almacenar datos de eventos en tiempo real. La transmisión de eventos asegura así un flujo continuo y una interpretación de los datos para que la información correcta esté en el lugar correcto, en el momento correcto.

Kafka combina tres capacidades para que permita implementar el caso de uso

de transmisión de eventos:

* Publicar y suscribir flujos de eventos a otros sistemas
* Almacenar transmisiones de eventos de manera duradera y confiable
* Procesar flujos de eventos a medida que ocurren.

La base de datos utilizada, MongoDB, proporciona un modelo de almacenamiento de datos elástico que permite almacenar y consultar tipos de datos multivariados con facilidad. Esto no solo simplifica la administración de la base de datos, sino que también crea un entorno altamente escalable para aplicaciones y servicios multiplataforma. Esta escalabilidad es necesaria según aumente de forma significativa el número de usuarios.

***Arquitectura orientada a servicios:***

Complementando al DAas para que los sistemas de información sean más flexibles, escalables y eficientes en el acceso y gestión de datos, también será utilizado un diseño SOA de procesos distribuidos, se considera esta arquitectura ya que será importante tener la información de cada mercado, valores, etc, actualizada en todo momento.

Para ocuparse de la disponibilidad de datos, cada uno de los sistemas internos se podría hacer altamente disponible. Cada sistema necesitaría estar altamente disponible para igualar la disponibilidad del servicio enfocado al cliente. Obviamente para una arquitectura de proceso distribuida sería considerablemente más costoso proporcionar una respuesta altamente disponible y completa que una arquitectura centrada en datos.

***Definición de las interfaces de servicios en un enfoque de Service-Oriented Architecture (SOA) para un Data as a Service (DaaS) para nuestro servicio:***

Señales de inversión:

#### Envío de Señales:

* **Descripción**: Permite solicitar señales de inversión para un mercado específico.
* **Método:** POST
* **URI**: /api/signal
* **Parámetros de entrada:** Mercado (acciones, divisas), preferencias de inversión ( corto plazo, largo plazo).
* **Respuesta:** Mensaje de confirmación o señal de inversión generada.

Predicciones de valores:

#### Obtención de Predicciones:

* **Descripción:** Permite obtener predicciones de valores para un mercado específico.
* **Método**: GET
* **URI**: /api/prediction/{market}
* **Parámetros de entrada:** Mercado (acciones, divisas).
* **Respuesta:** Datos de predicción, como el valor estimado y la confianza del modelo.

#### Actualización de Modelos:

* **Descripción:** Permite al sistema actualizar los modelos de predicción con nuevos datos.
* **Método:** POST
* **URI**: /api/prediction/update
* **Parámetros de entrada:** Nuevos datos de mercado.
* **Respuesta:** Confirmación de actualización exitosa.

Integración con plataformas externas:

#### Envío de Señales por Correo Electrónico:

* **Descripción:** Permite recibir señales de inversión por correo electrónico.
* **Método:** POST
* **URI:** /api/integration/email
* **Parámetros de entrada:** Dirección de correo electrónico, preferencias de señales.
* **Respuesta:** Confirmación de suscripción al servicio de señales por correo electrónico.

#### Interfaz de Chatbot:

* **Descripción:** Permite interactuar con el servicio de señales a través de un chatbot.
* **Método:** Interfaz de usuario conversacional.
* **Funcionalidades**: Solicitar señales de inversión, consultar predicciones de valores, suscribirse a notificaciones, etc.
* **Respuesta:** Respuestas del chatbot con señales de inversión, predicciones, información adicional, etc.

### Administración y Configuración:

#### Gestión de Usuarios:

* **Descripción:** Permite administrar los usuarios del servicio, incluyendo registro, inicio de sesión y gestión de suscripciones.
* **Método**: POST / GET / PUT / DELETE
* **URI:** /api/user
* **Parámetros de entrada:** Información de usuario, credenciales, preferencias, etc.
* **Respuesta:** Confirmación de operación exitosa, datos de usuario, etc.

#### Configuración del Servicio:

* **Descripción:** Permite configurar opciones y preferencias del servicio, como intervalos de actualización, fuentes de datos, modelos de predicción, etc.
* **Método**: POST / GET / PUT
* **URI:** /api/config
* **Parámetros de entrada:** Opciones de configuración, valores de ajuste, etc.
* **Respuesta:** Confirmación de configuración actualizada, datos de configuración, etc.

Seguridad y Autenticación:

Se implementarán mecanismos de autenticación y autorización para garantizar que solo los usuarios autorizados puedan acceder a los servicios y datos del sistema. Esto podría incluir autenticación basada en tokens, OAuth, etc. se implementarán medidas de seguridad en cada componente de la arquitectura, incluyendo autenticación, autorización, cifrado de datos y protección contra ataques como inyecciones SQL y XSSy se cumplirán las regulaciones como GDPR y PCI-DSS.

Para un servicio de Mailchimp, que se utiliza comúnmente para enviar correos electrónicos masivos y administrar campañas de marketing por correo electrónico, las interfaces de servicio podrían incluir operaciones para gestionar listas de correo, crear y enviar campañas, gestionar suscriptores, etc.

### Gestión de listas de correo:

#### Crear Lista:

* **Descripción:** Permite crear una nueva lista de correo en Mailchimp.
* **Método:** POST
* **URI:** /api/lists
* **Parámetros de entrada**: Nombre de la lista, opciones de configuración (
* **Respuesta:** Confirmación de lista creada con éxito, ID de la lista.

#### Obtener Lista:

* **Descripción:** Permite obtener detalles de una lista de correo específica.
* **Método:** GET
* **URI:** /api/lists/list\_id
* **Parámetros de entrada:** ID de la lista.
* **Respuesta:** Datos de la lista, incluyendo nombre, configuración y estadísticas.

#### Actualizar Lista:

* **Descripción:** Permite actualizar los detalles de una lista de correo existente.
* **Método:** PUT
* **URI**: /api/lists/{list\_id}
* **Parámetros de entrada:** ID de la lista, nuevos datos de la lista.
* **Respuesta:** Confirmación de lista actualizada con éxito.

#### Eliminar Lista:

* **Descripción:** Permite eliminar una lista de correo existente.
* **Método:** DELETE
* **URI:** /api/lists/{list\_id}
* **Parámetros de entrada:** ID de la lista.
* **Respuesta:** Confirmación de lista eliminada con éxito.

Gestión de Suscriptores:

#### Agregar Suscriptor:

* **Descripción**: Permite agregar un nuevo suscriptor a una lista de correo específica.
* **Método:** POST
* **URI**: /api/subscribers
* **Parámetros de entrada**: Datos del suscriptor, lista de destino.
* **Respuesta:** Confirmación de suscriptor agregado con éxito.

#### Actualizar Suscriptor:

* **Descripción:** Permite actualizar los detalles de un suscriptor existente.
* **Método**: PUT
* **URI**: /api/subscribers/{subscriber\_id}
* **Parámetros de entrada:** ID del suscriptor, nuevos datos del suscriptor.
* **Respuesta:** Confirmación de suscriptor actualizado con éxito.

#### Eliminar Suscriptor:

* **Descripción**: Permite eliminar un suscriptor de una lista de correo específica.
* **Método**: DELETE
* **URI:** /api/subscribers/{subscriber\_id}
* **Parámetros de entrada:** ID del suscriptor.
* Confirmación de suscriptor eliminado con éxito.

Para integrar el servicio de chatbot o Mailchimp en otras aplicaciones y servicios se utilizará un API REST, esto permitirá que otras aplicaciones y servicios interactúen con el servicio utilizando solicitudes HTTP estándar. Se definirán endpoints en la API para las operaciones específicas que se ofrecen.

Otra opción para integrar el servicio sería una integración directa,utilizando bibliotecas de cliente específicas o llamadas a la API directamente desde el código de la aplicación. Esta integración directa proporciona un mayor control y flexibilidad sobre cómo se comunican las aplicaciones y servicios entre sí.

Se realizará en una máquina virtual de una nube pública para aprovechar el mejor rendimiento en memoria y CPU.

Modelización:

La solución se estructurará teniendo, en primer lugar, un archivo en el directorio central que actuará como bucle principal para llamar a funciones de otros paquetes, importandoles. También se realizarán procesos internos, de transformación y tratamiento de los valores (limpieza, atribución y creación de nuevas variables), para que pasen al modelado empleando los métodos de clasificación disponibles en las librerías de Python o Spark, para que, sucesivamente el modelo sea guardado en formato y disponibilizado para los datos entrantes.

La segunda parte de la modelización, con el fin de conocer las variables y realizar un estudio más profundo de estos datos, se generan en notebooks en dos entornos diferentes (Python y SparkML de Databricks), para obtener una exploración de datos exhaustiva, conocer su distribución, detectar posibles anomalías, estudiar las variables y correlaciones.

Flujo de datos en Kafka:

Esta solución realiza la simulación de datos entrantes desde el flujo completo, empleando el sistema de producer/consumer de Kafka.

Flujo de datos en Spark Streaming:

En este flujo se leerán lo datos desde Kafka, como conector de consumidor,

tendrá su parte de procesado, agregados y nuevamente guardado en diferentes

soluciones de almacenado de datos, distribuidos por ficheros en HDFS, o en MongoDB como base de datos.

*Definir el catálogo de servicios que proporcionará la plataforma DAaaS, que incluye incorporación de datos, limpieza de datos, transformación de datos, datapedias, bibliotecas de herramientas analíticas y otros.*

### Arquitectura DAaaS

*Definir la selección de componentes, la definición de procesos de ingeniería y el diseño de interfaces de usuario. Diseño y ejecución de Proofs-of-Concept (PoC) para demostrar la viabilidad del enfoque DAaaS.*

### DAaaS Operating Model Design and Rollout

*Personalizar los modelos operativos DAaaS para cumplir con los procesos, la estructura organizacional, las reglas y el gobierno de los clientes individuales. Realizar seguimiento de consumo y mecanismos de informe.*

### Desarrollo de la plataforma DAaaS. (ligera descripción del desarrollo)

*Construcción iterativa de todas las capacidades de la plataforma, incluido el diseño, desarrollo e integración,* ***pruebas****, carga de datos, metadatos y población de catálogos, y despliegue.*

Link a Diagrama:

<https://drive.google.com/file/d/1b2z8WeGCZ8i3hL62Qsh3wGqKIYcC4Zzd/view?usp=drive_link>