Brainstorm

* Temas: Fish farm, videojuegos, libros de Sci-Fi
* La página de Kaggle tiene datasets de fish farming que podría usar:

<https://www.kaggle.com/datasets/ogbuokiriblessing/sensor-based-aquaponics-fish-pond-datasets> o con imágenes e información recopilada por IoTs en una granja en producción.

* Predictor del peso del pez.
* Esta página tiene datos de videojuegos: <https://www.3djuegos.com/juegos/>.
  + Predictor de sold-out de juegos o un sistema de recomendaciones
  + Puedo utilizar técnicas de scraping/crawling para extraer datos no estructurados.

Diseño del DAaaS (Data Analytics as a Service)

### Definición la estrategia del DAaaS

* Crear un servicio de dashboard que actualice semanalmente los datos de la granja y que entregue predicciones del peso promedio de los peces para los próximos meses.
* El servicio Dashboard proveerá visualizaciones y reportes que permitan al usuario realizar análisis de los datos semanales, establecer correlaciones entre variables, tendencias y obtener pronósticos de crecimiento a futuro.

### Arquitectura DAaaS

* Dashboard con ElasticSearch/Kibana en Google Compute Engine para las visualizaciones y reportes semanales.
* VM con MongoDB con colección/documentos que contienen datos de:
  + mediciones de sensores ambientales.
  + metadata (uri.address) de imágenes capturadas por las cámaras.
  + datos de tamaño y peso promedio.
* Cloud Function que genera CSV en Google Storage con los resultados de analitica de datos.
* Cloud Function que genera colección con los resultados del procesamiento de imágenes y cálculos de tamaño y peso.
* Hadoop/Cluster/Job con Dataproc que lleva CSV con datos de parámetros a MongoDB.
* Bucket/Google Storage para almacenamiento del dataset.
* Cloud Function/Scheduler para carga automática del dataset (imágenes y parámetros de sensores) al Google Storage.

### DAaaS Operating Model Design and Rollout

1. Al final de cada día (Cloud Scheduler) se ejecuta el Cloud Function: ***Load\_Info*** para la carga del Dataset a la nube. Para ello, se establece la conexión IP a las cámaras y sensores instalados en la jaula de peces y se importan las imágenes y parámetros ambientales del día a un Bucket creado en Google Storage.
2. Al terminar la función *Load\_Info* se ejecuta desde Hadoop/Cluster el **Job** para meter los datos de los parámetros.csv en la base de datos de Mongo. El Job es un script escrito en Python que se conecta a MongoDB Atlas e importa los datos.
3. Una vez terminado el Job, se ejecuta la segunda Cloud Function: ***Process\_Image*** para procesar cada imagen, calcular el tamaño y peso del pez utilizando código Python y la librería OpenCV (real-time computer vision).

La metadata de las imágenes (uri, etc) y los resultados de tamaño y peso promedio se guardan como una colección/documento en MongoDB.

1. Se ejecuta la siguiente función Cloud Function: ***Predictions*** para generar analitica de datos y predicciones del peso promedio para los próximos meses.

Mediante lenguaje Python, la librería Prophet y el acceso a los datos talla-peso guardados en MongoDB, se realizan cálculos y predicciones basado en un modelo de pronósticos con series de tiempo. Con la opción de Trigger del Cloud Function se guardan automáticamente los resultados en Google Storage (analitica\_predicciones.csv).

1. Con código Python se ejecuta el proceso de transformación e importación de información a ElasticSearch, mediante el envío de solicitudes HTTP al API REST de la herramienta para poder indexar los datos del CSV: analitica\_predicciones.
2. Utilizando Kibana se accede al índice/documentos en ElasticSearch y se genera el servicio del Dashboard con las visualizaciones y proyecciones requeridas por el cliente (proceso manual).
3. Mediante acceso-web vía https el cliente podrá analizar los gráficos y reportes de producción, así como ver las predicciones del peso promedio del pez para las próximas semanas y meses.

### 

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

*.*

El desarrollo del DAaaS se realizará completando los siguientes pasos de implementación:

1. Construcción y configuración de los elementos definidos en la arquitectura en la plataforma de Google Cloud:
   1. Creación de **Cloud Functions** para la ejecución automática de los procesos internos de carga y transformación de datos, procesamiento de imágenes, analitica y cálculo de predicciones.
   2. Creación de **Bucke**t del proyecto en Google Storage, como repositorio para el almacenamiento de imágenes y archivos CSV.
   3. Creación de **VM** (instancia) para base de datos en MongoDB.
   4. Creación de **Cluste**r con Hadoop en Google Dataproc para ejecutar job que importa CSV a MongoDB.
   5. Creación de segundo **VM** para correr herramientas como ElasticSearch/Kibana con funciones de indexado, búsqueda y visualización de datos.
2. Generar códigos (incompleto !!)

Como referencia algunos ejemplos con Python para ejecutar los procesos internos descritos en la arquitectura.

* 1. **Importar CSV a ElasticSearch usando REST API**:

*import csv*

*import json*

*import requests*

*# Elasticsearch REST API endpoint*

*url = "http://****34.27.33.14:9200****/fish\_index/\_doc/"*

*# Read CSV file and index data into Elasticsearch*

*with open('****analitica\_predicciones.csv****', mode='r') as file:*

*reader = csv.DictReader(file)*

*for row in reader:*

*# Transform row data as needed*

*document = {*

***"talla": row["25"****],*

***"peso": row["300"],***

*# Add other fields as needed*

*}*

*# Convert document to JSON*

*json\_document = json.dumps(document)*

*# Index document into Elasticsearch*

*response = requests.post(url, json=document)*

*if response.status\_code != 201:*

*print(f"Failed to index document: {response.text}")*

*else:*

*print("Document indexed successfully.")*

* 1. **Ejecutar job para meter datos en MongoDB:**

*import csv*

*from pymongo import MongoClient*

*# Connect to MongoDB*

*client = MongoClient('mongodb://****34.20.37.17:27017****')*

*db = client['****mongo\_database****']*

*collection = db[‘****fish\_collection****']*

*# Read CSV file and insert data into MongoDB*

*with open('****parametros.csv****', mode='r') as file:*

*reader = csv.DictReader(file)*

*for row in reader:*

*# Insert each row as a document into MongoDB*

*collection.insert\_one(row)*

1. Pruebas de conectividad a cámaras y sensores instalados en sitio y acceso a ficheros de imágenes y mediciones.
2. Pruebas de carga, así como de transformación de datos. Se recomienda ejecutar los scripts de Python en un Colab para probar que funcionan antes de llevarlos como código final a los Cloud\_Fuction y el Job configurado en Hadoop.
3. Construir el dashboard utilizando ElasticSearch y Kibana,
4. Configurar acceso seguro (https) al Dashboard desde Internet.

Link a Diagrama:

* Github: <https://github.com/mdosreis69/Practicas_BootCamp/blob/main/Big-Data-Arquitectura/diagrama_arquitectura.drawio>

