ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Desarrollo de módulo de analítica de datos para aplicación web enfocada en el monitoreo de piscinas camaroneras

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingeniero en Ciencias de la Computación

Presentado por:
Eduardo Andrés Gonzalez Beltrán
George Alberto Henríquez Ronquillo

GUAYAQUIL - ECUADOR
Año: 2022

DEDICATORIA

El presente proyecto se lo dedico a todos los seres queridos que me apoyaron incondicionalmente en mi vida, desde mi infancia hasta mi adultez.

Personas que les debo todo.

Eduardo Gonzalez Beltrán

El presente proyecto lo dedico a mi familia por haberme apoyado a lo largo de toda mi carrera y mi vida, aportando tanto a mi formación como profesional y como ser humano.

George Henríquez Ronquillo

AGRADECIMIENTOS

Le agradezco a mi madre, padre, abuela y tías, que me ayudaron a formarme con persona. Agradezco a mi hermano mi compañero de vida. Agradezco a Doris Pita y Samira Suárez compañeras durante toda mi formación académica. Y agradezco a la empresa BioDynamics que me ha permitido integrarme como profesional.

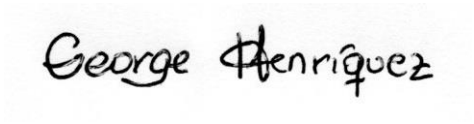
Eduardo Gonzalez Beltrán

En primer lugar, agradezco a Dios por permitirme lograr terminar uno de mis objetivos en mi vida. A mis padres que constantemente me estuvieron guiando y aconsejando. A mis hermanos que me motivaban a seguir adelante. A mis tíos y mi abuela que estuvieron apoyándome día a día. Finalmente, a los docentes que supieron guiarme durante mi formación académica.

George Henríquez Ronquillo

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, nos corresponde conforme al reglamento de propiedad intelectual de la institución; Eduardo Andrés Gonzalez Beltrán y George Alberto Henríquez Ronquillo y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"



George Alberto
Henríquez Ronquillo

EDUARDO ANDRES GONZALEZ BELTRAN

Firmado digitalmente por EDUARDO ANDRES GONZALEZ BELTRAN
Fecha: 2022.11.15 10:28:11 -05'00'

Eduardo Andrés
Gonzalez Beltrán

EVALUADORES

Firmado electrónicamente por:
RONALD RAUL CRIOLLO BONILLA

Criollo Ronald

PROFESOR DE LA MATERIA

Firmado electrónicamente por:
DANIEL ERICK OCHOA DONOSO

PhD. Daniel Ochoa Donoso

PROFESOR TUTOR

RESUMEN

La empresa ECRobotics (ECR) desarrolló modelos de analítica de datos que utilizan registros de datos de piscinas y tienen como fin mejorar la producción de camarones. Sin embargo, no se ha logrado desarrollar por completo un módulo de analítica que los permita usar desde su aplicación web. La finalidad del proyecto es implementar una arquitectura de software orientada al web que permita habilitar el uso de modelos de analítica de datos acuícolas para los clientes de ECR.

La arquitectura propuesta define una nueva lógica de ejecución de los modelos que permitió: ejecutar y monitorear los procesos asociados, manejar errores en la ejecución de procesos, intercambiar datos con servicios de terceros y mostrar los resultados en la aplicación web. Además, para tener flexibilidad en la implementación se utilizaron imágenes de Docker que contienen el ambiente informático necesario para cada modelo.

Al finalizar el desarrollo, se logró modularizar el código y se proveyó de un único objeto que se encarga de la gestión de los modelos. Además, los programadores de ECR fueron entrenados en el despliegue del módulo de analítica con la nueva arquitectura. Lo que permitió que el tiempo de despliegue de un producto se reduzca considerablemente, ya que hay una clara diferenciación entre el desarrollo de la aplicación web y los modelos de analítica.

Palabras Clave: Camaroneras, Docker, Modelos de inteligencia artificial, Arquitectura web, módulo de analítica.

ABSTRACT

ECRobotics (ECR) developed a software solution to improve shrimp farms' production by creating multiple data analytics models that use records of grow-up ponds. However, they have not been able to fully develop a module that allows the models to be used from their website. The goal of this project is to implement a web-oriented software architecture that facilitates the use of aquaculture data analytics models for the company's clients.

The proposed architecture defines a new model execution logic that allows them to: execute and supervise processes associated with the models, handle errors in process execution, exchange data with third-party services, and display results on their website. In addition, to allow flexibility in the implementation Docker images containing the computing environment required for each model were used.

At the end of development, it was possible to modularize the code and provide a single object responsible for managing the models. In addition, company programmers were trained in the deployment of models with the new architecture. This allowed the deployment time of a product to be reduced.

Keywords: Shrimp farms, Docker, artificial intelligence models, web architecture, analytics module.

ÍNDICE GENERAL

ABREVIATURAS

ESPOL Escuela Superior Politécnica del Litoral

ECR ECRobotics

AWS Amazon Web Services

SIMBOLOGÍA

s	Segundo
ha	Hectáreas
cam	Camarones

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. INTRODUCCIÓN

En la última década, la industria acuícola se ha posicionado como una de las más importantes del Ecuador. En 2021, hubo un aumento del 23% de exportación de camarones que generó un incremento del 39% en ingresos, en comparación al 2020 [1].

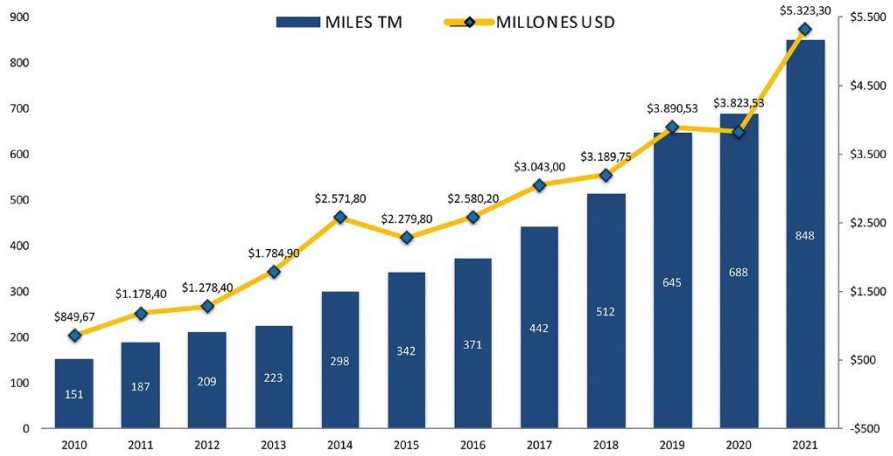


Figura 1.1 Exportación ecuatoriana anual de camarones [1].

Las granjas camaroneras en Ecuador aun evalúan el comportamiento de los animales de forma rudimentaria, para evitar posibles enfermedades, determinar su crecimiento, etc. Actividades como estimar la población de camarones, donde se capturan animales en sectores aleatorias de la piscina, se cuentan los camarones y estableciendo un promedio de los camarones por hectárea.

ECRobotics (ECR) ha desarrollado tecnología de software y hardware para: el monitoreo de las piscinas camaroneras combinando de sensores e inteligencia artificial que permiten tomar mediciones de calidad de agua, temperatura, alimento suministrado, actividad de animales, etc.

Adicionalmente, ECR ha construido varios modelos de aprendizaje automático y profundo para: predecir el crecimiento de camarones con datos de temperatura, detectar eventos acústicos anómalos, cambios de biomasa con ultrasonido, entre otros. Para que sus clientes interactúen con dichos modelos, ECR desarrolló una implementación parcial del módulo de analítica de datos en una aplicación web.

1.1 Descripción del problema

Los camaroneros recolectan datos de sus operaciones tales como: distribución de alimento, calidad del agua, densidad de camarones, porcentaje de supervivencia, entre otros, utilizando mecanismos manuales. Los camaroneros utilizan los datos recolectados, su experiencia y conocimiento biológico sobre los animales para dar solución a los problemas específicos o determinar el estado de las piscinas camaroneras. Sin embargo, las conclusiones pueden estar sesgadas y no siempre son soportadas por datos.

ECR desarrolló una plataforma web para el monitoreo de la piscina camaronera, alimentada por equipos en campo que recolectan datos y controlan automáticamente la alimentación de camarones. Además, existen varios modelos de analítica de datos que tienen como fin mejorar la producción de la cosecha. Estos modelos tienen como entrada los registros de datos almacenados por piscina: bitácora de crecimiento, archivos multimedia, valores de parámetros ambientales, etc.

El proceso que ECR sigue para realizar análisis descriptivos o predictivos comprenden tres pasos: 1) Ejecutar scripts que extraen registros de base de datos, archivos CSV y/o parámetros proveídos por el usuario, y los transforma en una estructura de datos; los parámetros de análisis dependen del modelo que sea seleccionado. 2) Ejecución de algoritmos de inteligencia artificial (modelo) que toman como entrada la estructura de datos y 3) Almacenar en un gestor de bases de datos o como archivos estáticos los resultados obtenidos.

El fin de los modelos es que sean utilizados por los clientes como un servicio desde la aplicación web como soporte para la toma de decisiones. ECR desarrolló

parcialmente un Backend (ml_backend) que integra la ejecución de modelos con la aplicación web por medio de un módulo web de analítica de datos, el API de datos de la empresa provee los registros de las piscinas. El ml_backend se encarga de: inicializar un análisis predictivo, preprocesar los datos, ejecutar los algoritmos de inteligencia artificial y almacenar los resultados, a estos pasos descritos, ECR los denomina su arquitectura de procesamiento de datos [6].

ECR implementó los elementos de su arquitectura de forma independiente y no están complemente integrada a la aplicación web todavía. Consecuentemente, no están disponibles para el uso de los clientes. Para habilitar un análisis se requiere integrar y modularizar de forma eficiente las tareas de preprocesamiento de datos.

En la práctica, hay dos tipos de desarrolladores: web y de modelos. Los desarrolladores web de la empresa adaptan los programas que implementan los desarrolladores de modelos para su uso directo desde la aplicación. Dado que este tipo de desarrolladores no necesariamente participan en el desarrollo de los modelos de inteligencia artificial, adaptar el código puede ser complejo porque necesitan de cierto nivel de conocimiento en matemáticas y ciencias computacionales que ellos no disponen. Un cambio o un error de digitación puede ocasionar que se altere el comportamiento de un modelo.

De igual modo, los programadores encargados de desarrollar los modelos no conocen el funcionamiento de las herramientas web o la infraestructura disponible para el procesamiento de datos. Lo que obliga a que haya una supervisión y comunicación permanente entre ambos equipos para habilitar un nuevo modelo o modificar uno existente. Si bien esta situación no se presenta con regularidad actualmente, se espera que, en el futuro, conforme aumente la demanda de servicios de datos, se vuelva un potencial cuello de botella para los procesos operativos de ECR.

1.2 Justificación del problema

ECR considera que hay varias ventajas en el proyecto propuesto:

1) Reducir el tiempo para la generación y puesta en operación de nuevos modelos de análisis de datos.
2) Reducir la probabilidad de errores de programación en el despliegue de un modelo.
3) Sentar las bases tecnológicas para crear servicios de procesamiento de datos y un mercado electrónico para algoritmos hechos por terceros.
4) Usar los registros recolectados de las piscinas para obtener información relevante que permita mejorar las condiciones y competitividad de sector acuícola.

1.3 Objetivos

1.3.1 Objetivo General

Implementar una arquitectura web para habilitar a los usuarios el acceso a modelos de analítica de datos acuícolas.

1.3.2 Objetivos Específicos

1. Establecer estándares en el formato de entrada para el procesamiento de datos y salida de los algoritmos de analítica de datos.
2. Codificar la implementación de modelos de analítica para la acuicultura utilizando la arquitectura de procesamiento de datos de ECR.
3. Documentar procesos y técnicas que ayuden a mejorar la compatibilidad de los modelos de analítica con el ml_backend.

1.4 Marco teórico

Los análisis de datos se manejan con una cola de trabajo asíncronos. Consiste en: 1) la definición de los recursos computacionales, 2) ejecución de los scripts de preprocesamiento de los registros que se encarga de transformarlos, limpiarlos y agruparlos en otra estructura de datos, es manejado por una herramienta Extraer-Transformar-Cargar (Sección 1.4.2), 3) la ejecución de los algoritmos de inteligencia artificial y 4) el almacenamiento de los resultados.

1.4.1 Colas de trabajos asíncronos

Las colas de trabajos asíncronos son herramientas que permiten distribuir la carga computacional en un sistema informático por medio de la planificación de procesos [2]. Las tareas en Celery [2] están a la espera de ser ejecutados. Celery es una de las herramientas más usadas para la gestión de trabajos asíncronos, debido a su simplicidad y flexibilidad para manejar grandes cantidades trabajo. En las arquitecturas web se utiliza colas para delegar trabajo a un servicio externo con el objetivo de responder a las solicitudes de los clientes.

Celery está implementada en Python, se compone de 4 componentes:

- Tareas de Celery (task): Se definen como operaciones en un script, las cuales van a ser ejecutadas por los trabajadores [2].
- Trabajador (worker): Es un proceso que se ejecuta en segundo plano y se mantiene a la espera de nuevas tareas; por lo general, se ejecutan varios trabajadores con el fin de procesar varias tareas simultáneamente [2].
- Corredor (Broker): Es un mecanismo de paso de mensajes por pipes a Redis [16], el cual se encarga de obtener las solicitudes almacenadas del servidor principal y enviarlas a Celery para su ejecución [2].
- Backend: es un mecanismo basado en Redis [16] usado para almacenar y retornar los resultados de la ejecución de una tarea [2].

1.4.2 Herramientas Extraer-Transformar-Cargar

Las herramientas Extraer-Transformar-Cargar (también ETL por sus siglas en inglés) son utilizadas para el preprocesamiento de datos, con el objetivo de obtener y almacenar datos coherentes procesados desde múltiples fuentes o tipos de datos [3]. El proceso se divide en 3 etapas:

- Extraer: Consiste en copiar o importar datos desde diferentes fuentes (bases de datos, archivos de texto, etc.), pueden ser estructurados o no estructurados [3].
- Transformar: Los datos extraídos se someten a operaciones de limpieza, eliminación de duplicados, validaciones, cálculos, cambio formatos que define el programador. Finalmente, los datos se transforman y se agrupan [3].

- Cargar: Los datos transformados se almacenarán en algún gestor (bases de datos o texto plano) para su futura consulta [3].

Luigi es una librería de Python que permite la ejecución automatizada de trabajo de procesamiento por lotes de ejecución prolongada. Por lo general están relacionadas con la ejecución de algoritmos de aprendizaje automático, cargar de datos con base de datos, entre otras. Luigi también se encarga de gestionar el flujo de trabajo, de manera que el trabajo de Luigi sea ejecutado eficientemente [4].

Esta herramienta ETL se basa su de ejecución basada en objetivos, solo se requiere la declaración de las instrucciones necesarias en código de Python. La ejecución de las instrucciones divide en 5 componentes [5]:

1. Tareas de Luigi: Son las representaciones de unidades atómicas de carga de trabajo que constituyen un flujo de trabajo, cada tarea depende de otras tareas y de los objetivos de salida [5] como se observa en la imagen 1.2.
2. Objetivos: Las tareas definen sus objetivos como salida [5]. Los objetivos pueden ser desde un archivo de texto, una entrada a una base de datos, etc. [4]. Luigi tiene un soporte de formatos robusto.
3. Parámetros: Los parámetros de entrada definen el comportamiento de la tarea y su secuencia de ejecución dependiendo de las condiciones que se definan. Los parámetros pueden cambiar durante el tiempo de ejecución [5].
4. Trabajador: Planea las tareas y las ejecuta dentro del proceso de Luigi, además de almacenar el estado de ejecución, tiempo de ejecución y datos de error [5].
5. Planificador: Determina el orden de ejecución de los trabajadores [5].

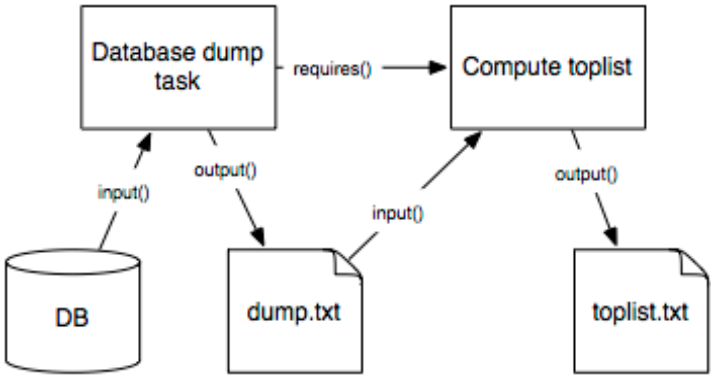


Figura 1.2 Lógica de Luigi [4].

En la figura 1.2 se muestra un pequeño esquema de la ejecución de tareas de Luigi, donde Compute toplist requiere de un archivo de texto que genera Data base dump task, que a su vez necesita de los registros de la base de datos para su ejecución. La salida de las tareas en otro archivo de texto.

1.5 Modelos de inteligencia artificial

Un modelo en inteligencia artificial se puede describir como el aprendizaje de una función (f) que asigna una o más variables de entrada (X) a una variable de salida (Y), con el objetivo de realizar predicciones, identificar patrones, agrupar datos, etc. [7] En este trabajo, no nos interesa tanto el fin de los modelos sino la representación de datos de entrada y salida que requieren los algoritmos para su ejecución

Los modelos pueden tener como variables de salida de valor único y discreto, o continuo [8]. Un algoritmo puede retornar una estructura de datos, vector o matriz, en particular, si el resultado es una agrupación de datos similares [9].

Ciertos modelos de inteligencia artificial implican predecir objetos estructurados tales como una secuencia, un grafo o árboles [10]. Las salidas estructuradas han sido aplicadas en la extracción de información [11].

CAPÍTULO 2

2. METODOLOGÍA

2.1 Análisis

De acuerdo con lo discutido, ECR cuenta con una aplicación web en Angular donde los clientes camaroneros ven la información y gestionan sus piscinas. Con el fin de darle funcionalidades adicionales para definir análisis que usan los modelos desarrollados por la empresa, se construyó parcialmente un módulo de analítica de datos y una arquitectura web que integra la arquitectura de procesamiento de datos de ECR, sin embargo, aún no se encuentra en producción y solo permite ejecutar un análisis (predicción de peso por temperatura) [6], la lógica de ejecución de los análisis es gestionada por el ml_backend.

Los registros, plantillas y logs de los análisis, se almacenan en MongoDB. Esta base de datos es indiferente de otros servicios de ECR.

Además, en abril del 2022 se realizó una migración masiva del gestor de base de datos, que provocó que varias APIs que consultaban el ml_backend para extraer registros quedaran inutilizables, consecuentemente el único análisis que estaba disponible ya no podía ser usado. Adicionalmente, se desarrollaron dos nuevos modelos de inteligencia artificial que se mencionan en la sección 2.3.3, y uno que se desarrolla en conjunto a este proyecto.

2.1.1 Requerimientos

Se definieron requerimientos de los clientes (camaroneros), los desarrolladores web y de modelos de ERC, y las características que se requieren mejorar o corregir en la arquitectura de preprocesamiento de datos existente.

2.1.2 Requerimientos Funcionales

2.1.2.1 Cliente

1) El módulo de analítica de datos debe permitir ejecutar un análisis, cuyos parámetros dependen del modelo seleccionado.

2) El módulo debe permitir ejecutar un análisis para una sola piscina o para un grupo de piscinas.

3) El módulo debe permitir conocer el estado de los análisis, si está pendiente, en ejecución o finalizado, y así mismo su respuesta.

4) El módulo debe permitir visualizar los resultados de los análisis ejecutados correctamente en gráficos estadísticos desde la interfaz web.

2.1.2.2 Programadores

5) Establecer un formato genérico de salida del modelo, que se utilizará como entrada de los gráficos estadísticos.

6) Documentar los pasos para integrar y/o adaptar los scripts de preprocesamiento de datos en tareas de la librería Luigi, de modo que facilite la compatibilidad con el ml_backend.

7) El módulo debe permitir habilitar un análisis con el mínimo de cambios posibles en el código del ml_backend.

2.1.2.3 Características para mejorar o refactorizar

8) Codificar una técnica para extraer los registros desde múltiples instancias del API principal de ECR, con el objetivo de indicarle al ml_backend cuál es la fuente de datos que requiere consultar por análisis.

9) Refactorizar los métodos de estación de registros del ml_backend para rehabilitarlos, además que permita filtrar por granja y/o usuario.

10) Codificar una técnica que permita subir, acceder y descargar archivos multimedia para los tipos de análisis que lo requieran.

11) Enviar una notificación al panel de control cuando se finalice el análisis, para cuando un análisis actualice su estado.

2.1.3 Requerimientos No-Funcionales

12) El Backend requiere ser desplegado en contenedores de Docker en los servicios de ECR en Amazon Web Services (AWS).

2.1.4 Alcance y limitaciones de la solución

- **Limitaciones**

ERC por cada cliente despliega una instancia de API principal en un ordenador de placa reducida, que se encarga de gestionar y almacenar los registros capturados de granjas, esta instancia no necesariamente tiene comunicación con otras instancias del API, ni son manejas por los servicios de AWS, además el equipo no cuenta con conexión a Internet estable.

El ml_backend es dependiente de AWS y de la conexión a internet, además de no ser recomendado ejecutarlo en los equipos de bajos recursos que no soporte o les represente una carga alta de trabajo, debido a que puede comprometer el funcionamiento de otras actividades que realizan.

- **Alcance**

Los modelos solo pueden ser ejecutados en el lenguaje de programación Python, no se integrará para otros soportes en otros lenguajes de programación.

Integrar y generar un prototipo de la visualización de datos de todos los análisis descritos en la sección 2.2, y al menos un análisis debe ser estar completamente terminado con las recomendaciones del cliente.

2.1.5 Riesgos y beneficios de la solución

- **Riesgo**

La ejecución de los análisis depende completamente de los registros obtenidos del API, por lo que cualquier cambio futuro en la extracción de los registros de la base de datos puede comprometer el funcionamiento del ml_backend.

La ejecución de múltiples tareas concurrente en Celery de múltiples usuarios puede crear un futuro cuello de botella que comprometa el rendimiento del ml_backend.

- **Beneficios**

La implementación del módulo de analítica permitirá la integración de nuevos modelos de forma rápida y costo-efectiva.

La documentación se espera que sirva como guía para los programadores que se encargan de integrar nuevos modelos.

2.1.6 Usuarios de la solución

Los camaroneros podrán usar los análisis que estarán disponibles en los servicios de ECR.

Los programadores encargados de las tecnologías web de ECR usarán las herramientas de software y documentación para desplegar nuevos análisis rápidamente.

2.2 Prototipado

Los análisis dependen de una plantilla escrita en formato JSON, en esta se definen la fuente de los registros que serán procesados, los formularios para los parámetros que ingresa el usuario, el modelo que se ejecuta, descripción e información adicional referente al análisis o parámetros.

Cada análisis puede tener variaciones del mismo modelo de inteligencia artificial, que puede requerir de diferentes parámetros (parámetros del modelo), para entregar una respuesta diferente considerando su propia definición de ejecución. Se creo una técnica para actualizar el formulario dependiendo del modelo seleccionado.

El módulo de analítica de datos se creó con la intención de ser simple, con pocas pantallas para el formulario que se renderiza dependiendo del análisis seleccionado.

Se usaron componentes previamente desarrollados. A continuación, se presentan las pantallas más importantes:

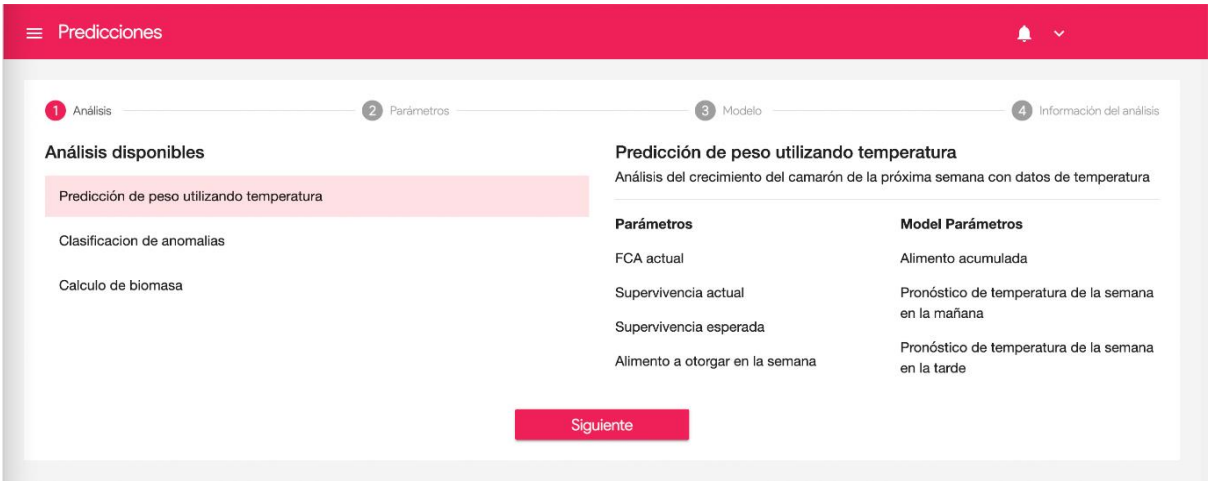


Figura 2.1 Selección de un análisis [autoría propia].

En la figura 2.1 se observa la sección para escoger uno de los análisis habilitados. Se muestra el titulo y una descripción corta del comportamiento del análisis, también se muestran los parámetros que debe ingresar el usuario en el formulario. Existen los parámetros y parámetros del modelo, los parámetros del modelo varían según el modelo que se escoja, esto se ve reflejado en los formularios posteriormente.

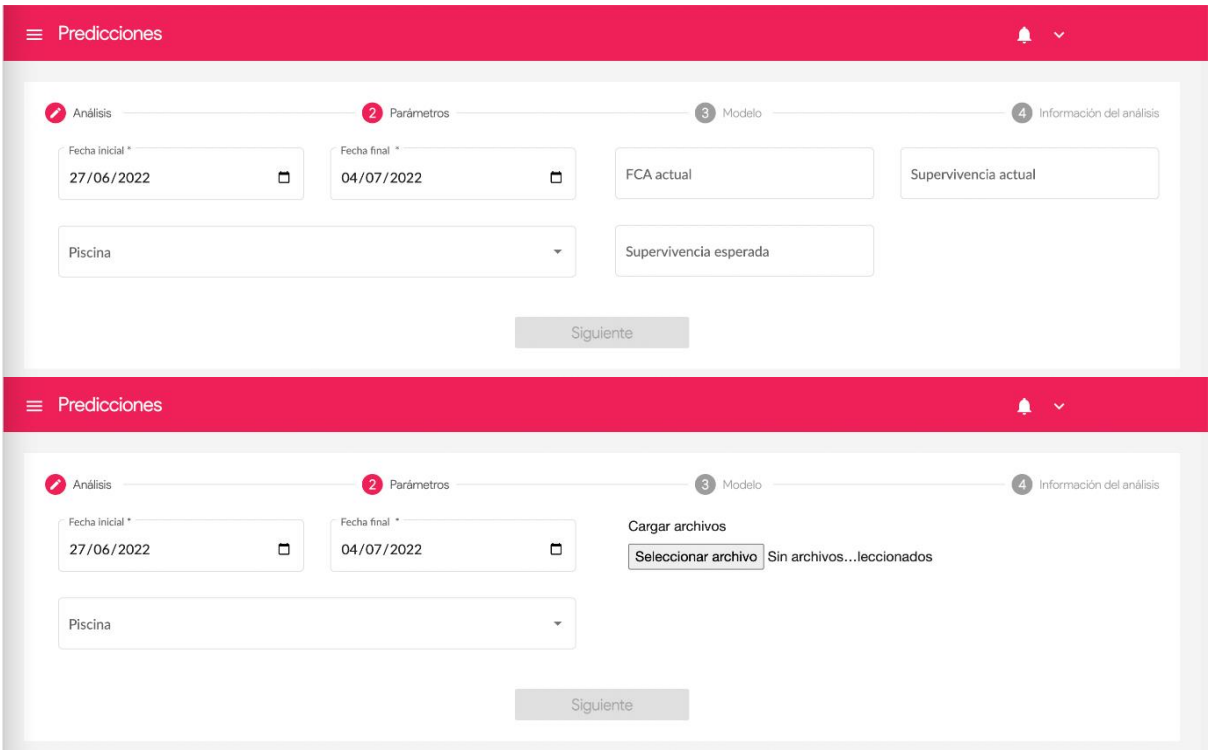


Figura 2.2 Formulario de parámetros [autoría propia].

En la figura 2.2 se muestra la sección de parámetros del formulario que permite ingresar los parámetros. Se refactorizó para que las etiquetas del formulario cambien según la plantilla del análisis seccionado y se habilitó una opción para subir archivos multimedia; como se muestra en la figura 2.2 que existen dos formularios de análisis diferentes.

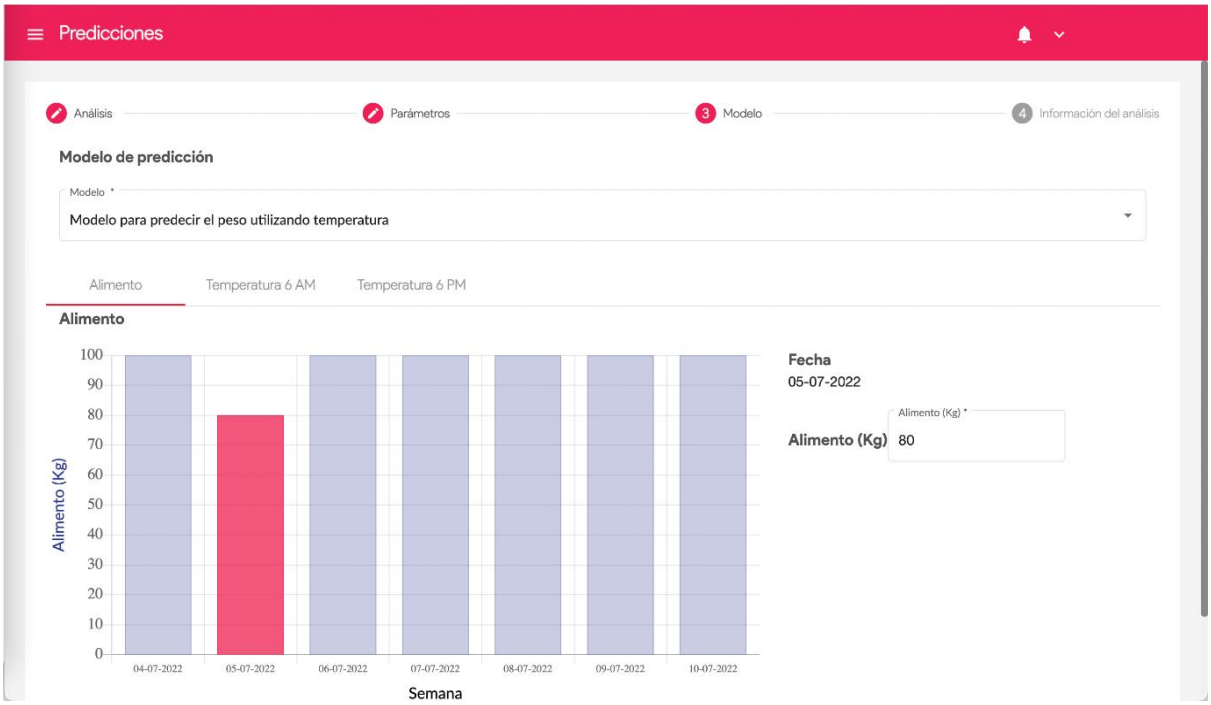


Figura 2.3 Formulario para escoger un modelo [autoría propia].

En la figura 2.3 se muestra la sección de modelo del formulario, se actualizó la disposición de los elementos del formulario en esta sección. Para seleccionar un modelo de los disponible y variar sus parámetros del modelo.

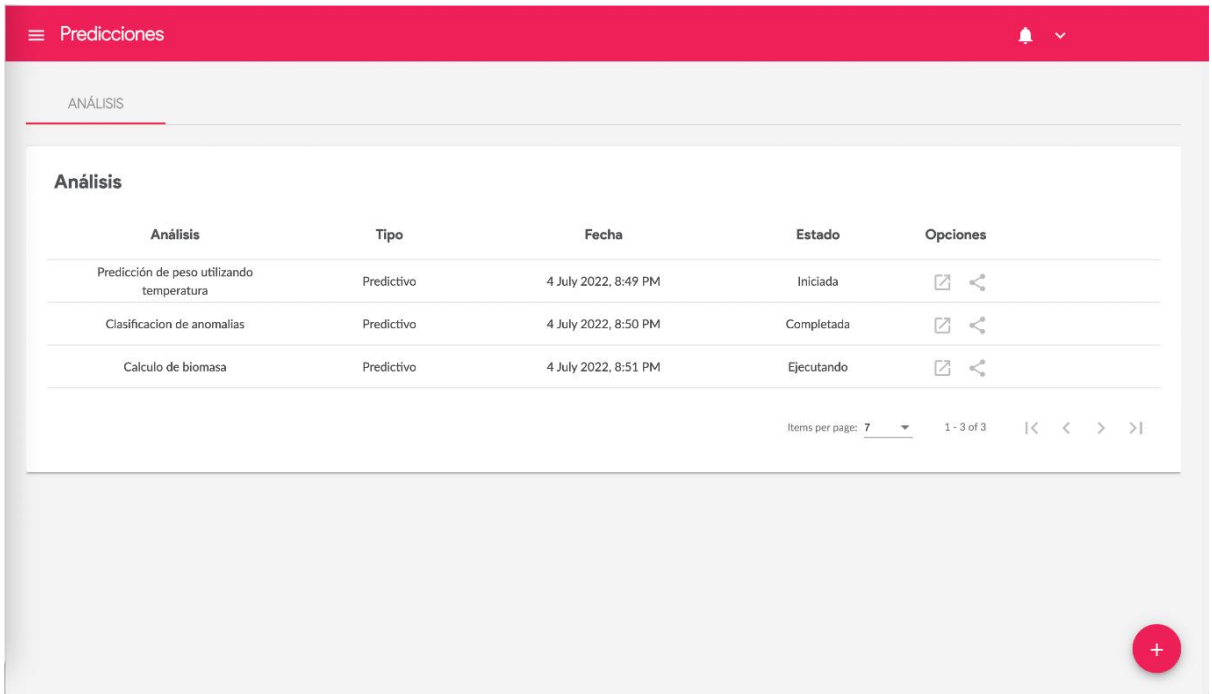


Figura 2.4 Tabla de tareas de análisis [autoría propia].

Posterior a encolar un análisis para ser ejecutado, se cambia a esta pantalla (figura 2.4) que muestra una tabla que presenta la lista de análisis ejecutados, se puede visualizar su estado, el tipo de análisis, descripción y acciones. La tabla ya estaba desarrollada, sin embargo, se la adapto para mostrar los nuevos análisis.

2.3 Evaluación

Tras evaluar el prototipo con ECR, se sugirió añadir las unidades de todos los valores que se muestran por pantallas y usar la terminología adecuada para la acuicultura. Además, se sugirió que el análisis predictivo para crecimiento de camarones tenga su respuesta en gráficos estadísticos.

2.4 Diseño de la solución

A continuación, se explican las mejoras y cambios que se realizaron a la arquitectura de preprocesamiento de datos de ECR.

2.4.1 Diagrama de Despliegue

2.4.1.1 Hardware

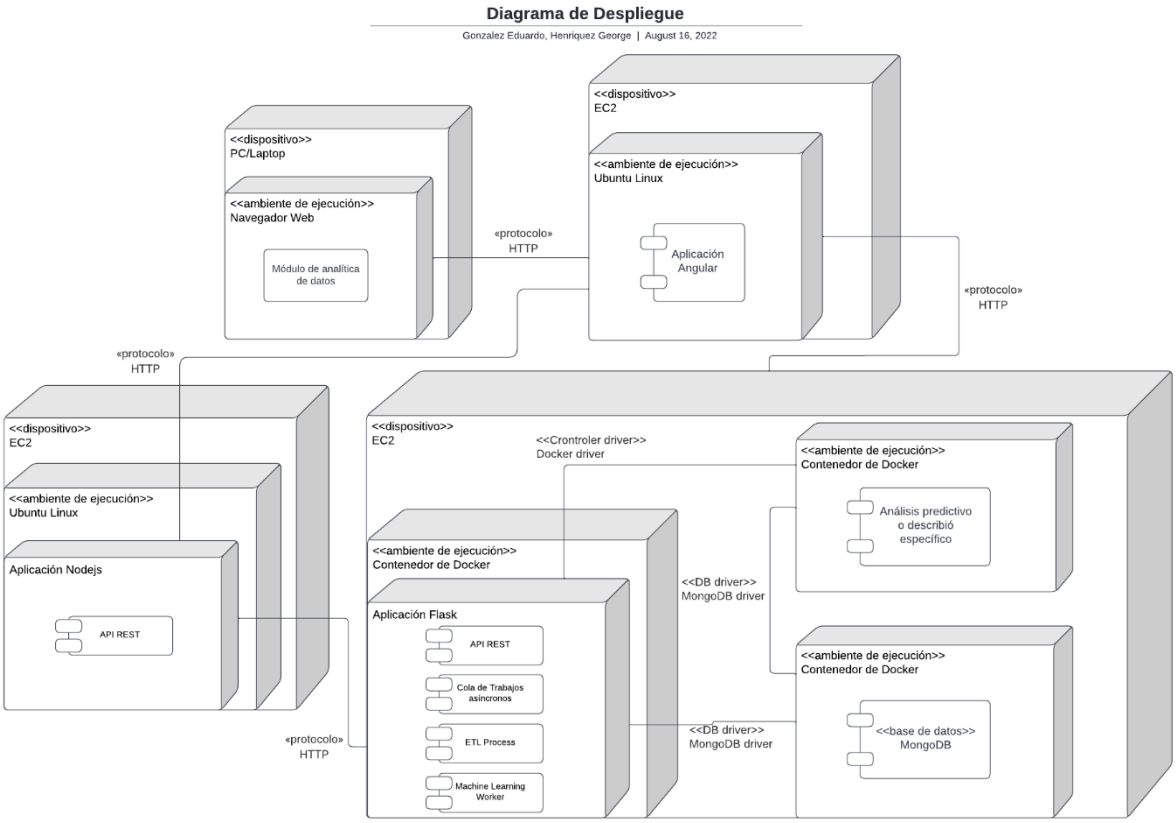


Figura 2.5 Diagrama de despliegue [autoria propia].

La arquitectura web se desplegará en una máquina de AWS (Amazon Web Services) con sistema operativo Ubuntu. La solución consiste en múltiples contenedores de Docker que disponen de servicios y/o ejecutan análisis específicos que define el programador. Se comunica por requerimientos HTTP con la aplicación web y el API principal de ECR. Todos los contenedores se comunican con el contenedor de MongoDB a través de los drivers del gestor de base de datos, y los drives de Docker para levantar nuevos contendores de Docker que se requieran por análisis.

2.4.2 Uso de contenedores Docker

Como las dependencias de los modelos pueden ser incompatibles, se decidió por crear imágenes de Docker [14] que contengan el ambiente de ejecución necesario

para correr cada análisis como un proceso independiente. Estas imágenes se usarán para crear contendores, que son añadidas a la misma network [15] del gestor de base de datos, así poder establecer una conexión con este. Se le provee de variables de entorno que podrá usar en su ejecución, tales como el identificador del registro del análisis en la base de datos, el nombre del volume de Docker y el enlace para la conexión con MongoDB. Los contenedores se eliminan al finalizar su ejecución.

Los modelos que puedan ser ejecutados en el propio ambiente del ml_backend no requiere de un contenedor.

Los registros de la base de datos de MongoDB y los archivos multimedia se almacenan a través de api en un Volume de Docker [13], que es un enlace a directorio en el host (máquina de AWS). Docker gestiona los directorios compartirlos con más contenedores, donde se define una ruta absoluta en el contenedor que estará enlazada con el volumen, todo cambio en la ruta asignada repercutirá en cambios del Volume. Se usan los Volumes [13] para conservar los datos, aunque se elimine los contendedores ya que son ajenos a estos, si se requiere de eliminar los datos se debe hacer manualmente.

2.4.3 Machine Learning Backend

El servicio denominado ml_backend, se encarga de extraer y transformar los datos, que serán usados por algoritmos de inteligencia artificial (modelos) y retorna una respuesta [6].

La aplicación web se comunica por requerimientos HTTP con ml_backend, para añadir una nueva tarea a ser ejecutada. El ml_backend se comunica por requerimiento HTTP al API principal de ECR, donde se extraen los registros a transformar.

Por cada solicitud de inicio de análisis se genera una tarea de Celery que se añade a una cola de trabajos asíncronos manejada en segundo plano por Celery, se

mantienen suspendidas hasta que puedan ser ejecutadas, de tal forma que las tareas no representen una gran carga computacional al servidor principal [6].

El proceso ETL se ejecuta con la librería Luigi, que define el programa en tareas, de modo que recoge y transforma los datos de entrada para los modelos de inteligencia artificial [6]. El proceso ETL (Apéndice 1) de ECR descrito solo soporta el preprocesamiento del análisis predictivo mencionado en la sección 2.4.6.

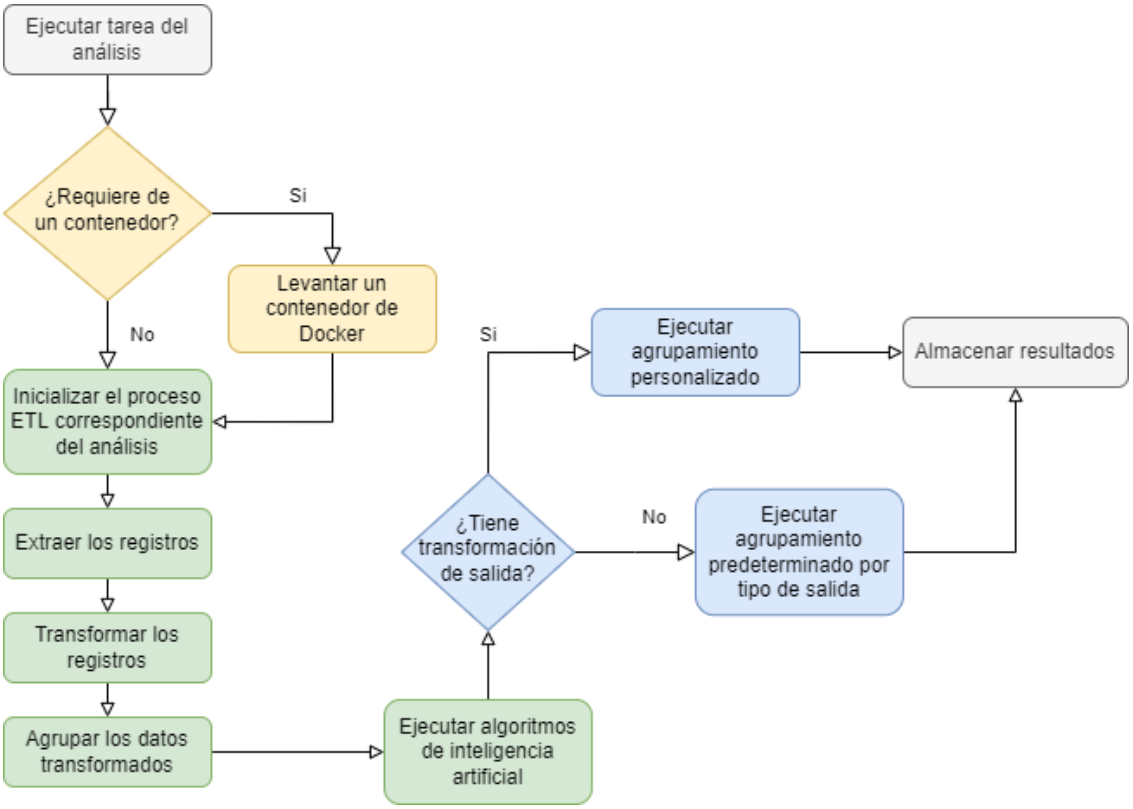


Figura 2.6 Nueva lógica del Proceso ETL y ejecución del análisis [autoria propia].

Como existen distintos análisis que definen su propia entrada, que van desde consultas HTTP a la API, archivos locales y/o entradas de formularios en estructura de datos, se adaptó y desacopló el método que agrupa los datos extraídos en una estructura de datos (diccionario), para que soporte la consulta y descarga de archivos multimedia, y la descompresión de archivos ZIP.

Se requiere desacoplar las tareas de Luigi, de tal forma que se pueda definir un conjunto de tareas propias que se ejecuten secuencialmente por cada análisis, y que puedan ser interpretado por el proceso ETL de ECR.

Para cada análisis, las tareas se ejecutan siguiendo una secuencia de etapas ya establecidas. Estas etapas son las siguientes:

- Manejador**:** Evalúa e inicializa las tareas específicas por cada análisis, también puede levantar un contenedor de Docker con las especificaciones que se mencionaron en la sección 2.4.2.2.
- Extractor**:** Importa registros desde múltiples fuentes, y las agrupa en un diccionario.
- Trasformador**:** evalúa los datos obtenidos y ejecuta el proceso de trasformación correspondiente.
- Agrupador**:** los datos transformados en una estructura para el algoritmo de inteligencia artificial.

Los análisis dentro de los contendores de Docker pueden seguir una implementación diferente a la planteada. Se aplicó la misma definición de tareas de Luigi para el cálculo de biomasa, en su propio ambiente de Python 2 sección 2.4.6. El proceso descrito se muestra en un diagrama de flujo en la figura 2.2.

La respuesta de los algoritmos de inteligencia artificial es almacenada o transformada nuevamente en estructuras, tal como arreglos, diccionarios, o matrices. La estructura puede ser fácilmente interpretadas en gráficos estadístico a través de un archivo de configuración, donde se define el tipo de grafico como: histograma, diagrama de cajas, etc. y la estructura de datos necesaria. Los programadores pueden evaluar la estructura que necesitan y crear método de agrupamiento específicos de acuerdo las necesidades.

2.4.4 Diseño de la Base de Datos

Para darle soporte a nuevas funcionalidades al ml_backend se requirió modificar esquemas de la base de datos de MongoDB. Los campos marcados con verde en la figura 2.8 son adicionales, mientras que las rojas son sustracciones.

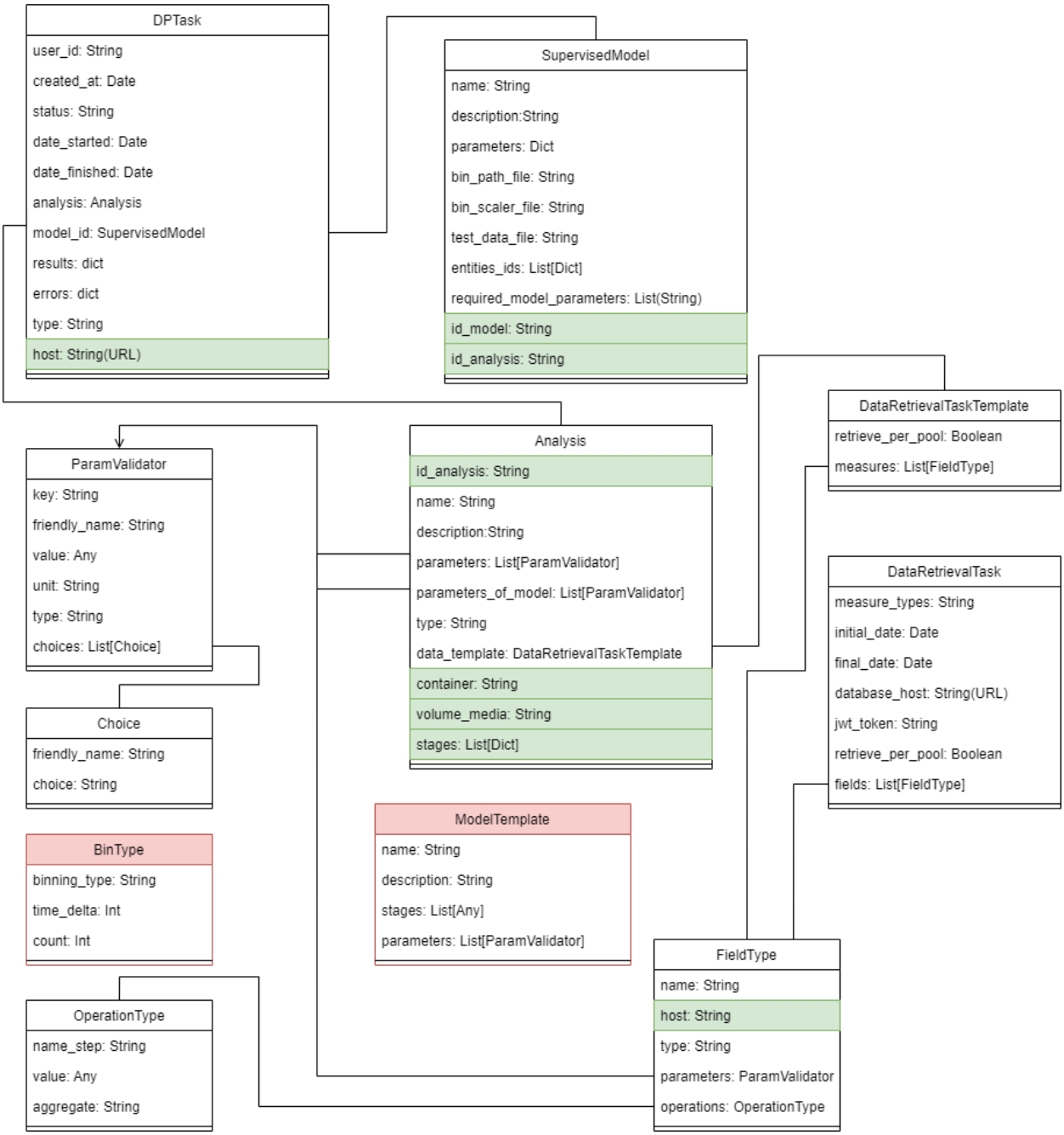


Figura 2.7 Diagrama de colecciones actualizado [Autoria propia].

Para dar soporte a la consulta de múltiples instancias del APIs de ECR, se agregó un nuevo campo al esquema de la tarea (DPTask), que es URL (host) de donde provino el requerimiento HTTP, posteriormente se usa este campo para generar las APIs para extraer los registros.

En la definición de los análisis se añadió un nuevo campo que determina si el análisis necesita de un contendor de Docker para su ejecución, el valor del campo es el nombre del contenedor, además de volumen_media que indica la ruta absoluta de los archivos multimedia, los campos son opcionales.

Se añadieron también identificadores secundarios (id_analysis) para establecer una relación entre los esquemas Analysis y SupervisedModel. Las colecciones ModelTemplatate y BinType se eliminaron debido a que ya no se estaban utilizando en ningún contexto.

2.4.5 Modelos disponibles

2.4.5.1 Análisis predictivo para crecimiento de camarones

Tiene como objetivo el predecir el crecimiento de los camarones usando redes neuronales en base a datos de alimento suministrado o calidad de agua [6]. El cliente debe proveer los parámetros (formulario) necesarios para ejecutar un análisis [6], donde se especifican:

- Cantidad de alimento suministrado (Kg) hasta la fecha de inicio.
- Factor de conversión.
- Porcentaje de supervivencia de los camarones en la fecha final.

Los registros son extraídos de la bitácora por ciclo de engorde de los camarones por medio del API principal, tales como: biomasa, peso, porcentaje de supervivencia, alimento diario y alimento acumulado.

El análisis tiene una implementación parcial en el módulo de analítica, que se requirió refactorizar. Se desacopló su primera implementación del proceso ETL para ejecutarlo a través de tareas individuales de Luigi, además de habilitar la consulta de

las demás variantes del modelo principal que pueden o no usar datos que depende de las necesidades del usuario.

Las redes neuronales y redes neuronales convolucionales retornan matrices multidimensional de tamaño m x n, donde m representa el número de muestras y n el número de salidas o etiquetas [8]. Debido a que la salida de estos algoritmos tiene una estructura de tipo matriz, es posible transformar los datos de forma que tengan una estructura similar. Para el caso de la matriz se establecieron métodos genéricos que retornan diccionarios con los arreglos por etiqueta, de tal forma que pueda ser almacenada como un documento.

La salida de este análisis es un arreglo de enteros, que se representa en un gráfico de líneas de la predicción obtenida vs días de la siguiente semana.

2.4.5.2 Estimación de biomasa usando Imágenes acústicas

Análisis del crecimiento del camarón con datos de sonar (ultrasonidos). Requiere como único parámetro, los registros de los sensores (datos crudos) obtenidos por una API o desde una ruta del equipo. Los datos se usan para generar imágenes y algunos archivos CSV.

El análisis requiere de un entorno específico en Python 2, para ejecutar los algoritmos de generación de imágenes. Además, usa los algoritmos de visión por computador escritos en lenguaje C.

Ya que este análisis es un trabajo en progreso no se estableció una salida genérica para este caso solo se proveyó del URL de descarga para los archivos generadas.

2.4.5.3 Análisis de eventos anómalos.

Analiza los audios capturados por los hidrófonos de una piscina, con el objetivo de determinar si el audio capturado grabó un evento anómalo. Tiene como único parámetro un archivo de audio de n duración. El audio se carga y envía al preprocesamiento de datos, que lo transforma en uno o más espectrogramas

(representación visual), para ser analizado por una red neuronal convolucional que determina la clasificación.

Se añadieron nuevas funcionalidades subir y leer un archivo de audio almacenado en los volumes de Docker cuando el análisis sea ejecutado.

CAPÍTULO 3

3. RESULTADOS Y ANÁLISIS

El módulo de analítica de datos creado en este proyecto se logró integrar de manera eficiente con la aplicación web de ERC, y se encuentra disponible para los usuarios, de igual forma los desarrolladores web de ECR cuentan con herramientas para el despliegue de futuros análisis.

3.1 Pruebas

Las pruebas de ejecución de análisis con los datos reales de las piscinas fueron realizadas por los usuarios de ECR y por los programadores que desarrollaron los modelos. Los usuarios accedieron al módulo de analítica desde su navegador web. Las tareas de los análisis mostrados están siendo ejecutadas en la máquina de AWS de ECR ya en producción.

Se probaron todos los modelos descritos en la sección 2.4.6, además de un nuevo análisis para la clasificación de audios no definido con anterioridad (apéndice C), este se desarrolló en conjunto por otro equipo de desarrolladores de ECR. El nuevo análisis fue desplegado y puesto en producción en 3 horas.

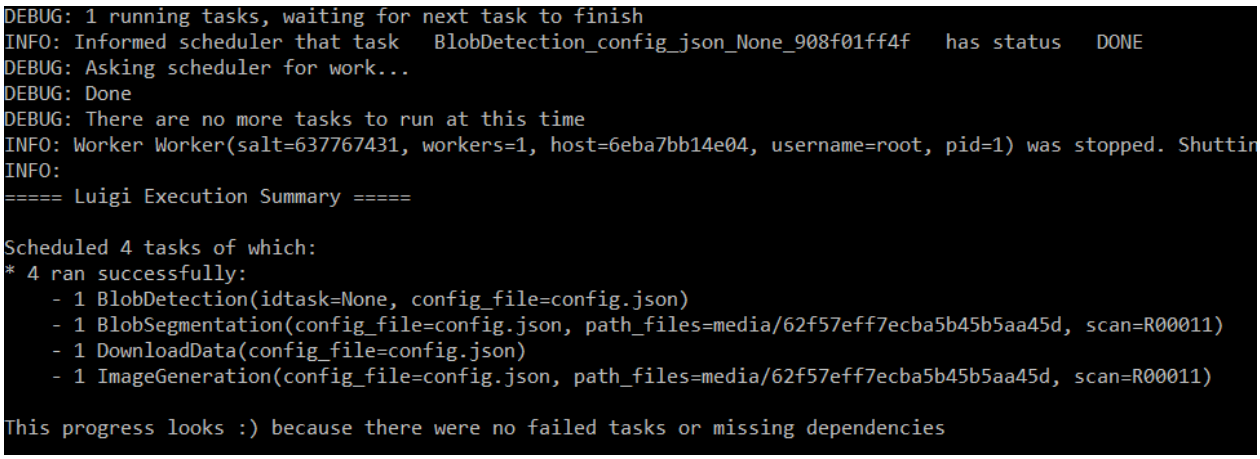


Figura 3.1 Resumen de ejecución de tareas de Luigi del calculo de biomasa [autoria propia].

Las tareas que se muestran en la figura 3.1 heredan de una clase TaskLuiguiTemplate (apéndice H) que tiene los recursos que provee la librería Luigi. Se encarga del manejo y consulta del análisis a la base de datos, actualizar los estados y resultados del análisis y de la comunicación con la aplicación web a través de un websocket. De igual forma maneja los errores que son notificados por la aplicación web a los usuarios y el administrador por correo electrónico. En el apéndice C se reportan las pruebas realizadas.

3.2 Plan de Implementación

La implementación se hizo en varias etapas. Se adjunta un diagrama de Gantt en la sección de apéndice D, donde se detallan las actividades y las fechas de inicio y fin, durante las 16 semanas en la que se desarrolló este proyecto.

Se estableció un objetivo por semana referente al proyecto, además de detallar las semanas en que se realizaron pruebas con los usuarios, la documentación y manuales respectivos. Los manuales de usuario e instalación se encuentran en la sección apéndice E y F donde se muestra el proceso para usar la respectiva aplicación.

3.3 Resultados de la implementación

En base al prototipado desarrollado en el capítulo 2, se desarrolló el módulo de analítica de datos capaz de planificar y ejecutar el ambiente que los análisis necesitan para ejecutar los modelos de forma óptima. A continuación, se presentan las principales funcionalidades desarrolladas:

- **Inicializar un análisis**

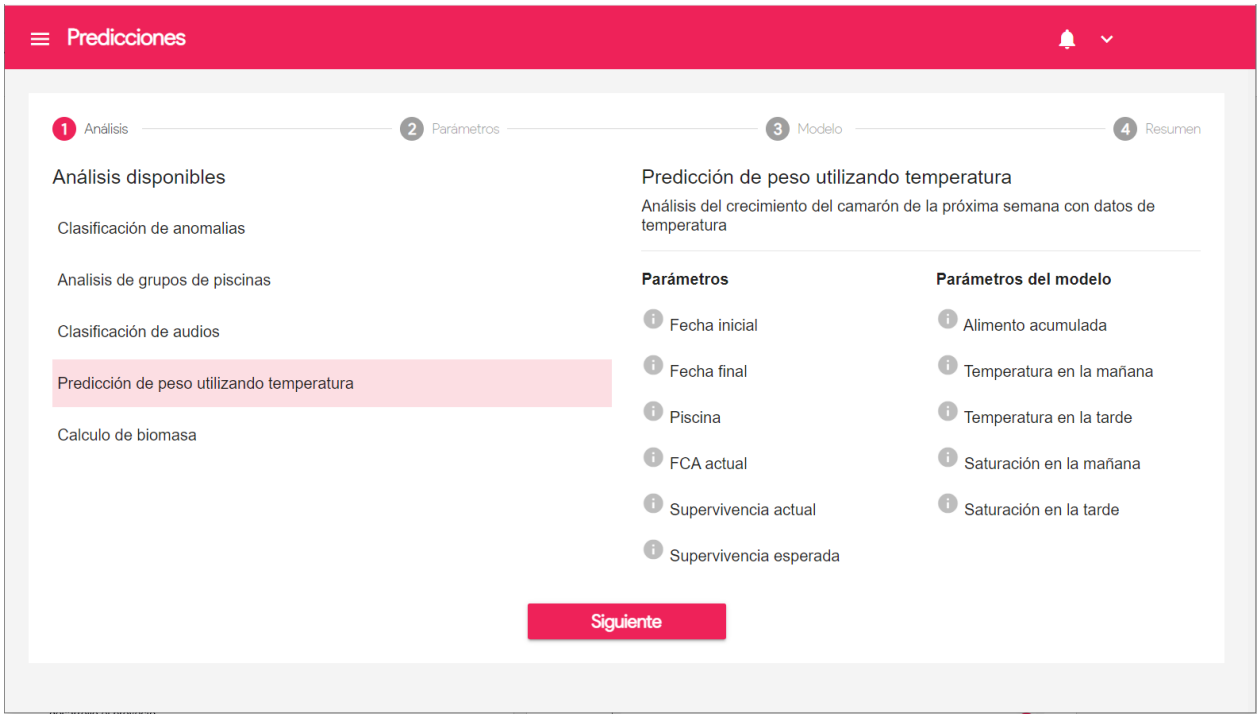


Figura 3.2 Selección de análisis [autoria propia].

Sección que permite escoger el análisis que será ejecutado y encolado en Celery, se muestra un resumen y descripción detallada de los parámetros del modelo. También indica los formularios a renderizar a la aplicación web. El módulo verifica que los análisis puedan ser ejecutados en el servidor, caso contario no permite iniciarlo.

- **Tabla de resultados**

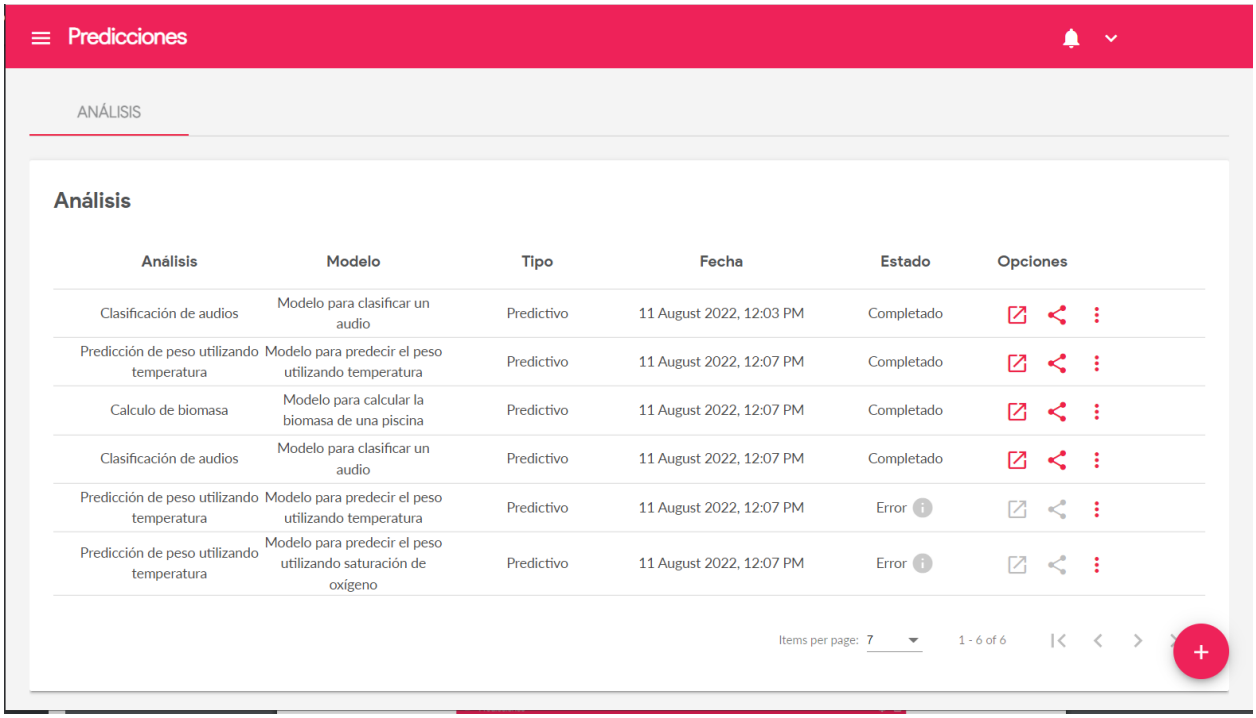


Figura 3.3 Tabla de resultados [autoria propia].

Las tareas en ejecución y/o terminadas se listan en una tabla, donde se visualiza su estado, actualiza las etapas, fecha de ejecución, tipo de análisis y modelo de análisis. En caso de que el análisis falle en su ejecución se muestra el error generado. En el costado derecho se da una opción para ver los resultados del análisis. Se presenta más información sobre las funcionales desarrolladas en el apéndice E.

3.4 Análisis de costos

Se consideraron el costo de desarrollar el módulo basándose en el esfuerzo y tiempo de los programadores para desarrollar el modelo de analítica. Se ha cálculo un aproximado del sueldo por las 16 semanas que duró el desarrollo de este proyecto.

El horario para desarrollar el proyecto fue de medio tiempo, por los 5 días laborables de la semana, 4 horas diarias, 20 horas a la semana por 16 semanas, siendo un total 320 horas por programador para desarrollar el módulo. El costo por hora es de 4.37

dólares americanos, al ser dos programadores necesarios entonces el costo total es 2800 $.

Tabla 3.1 Análisis de costo del proyecto por programador. [autoría propia]

Actividades	Cantidad	Total ($)
Horas laborables por programador	320	1400
Costo por horas ($)	4.37	

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

Se logró desarrollar el módulo de analítica de datos en su totalidad además de llevarlo a producción para que pueda ser usado por ECR. El sistema desde ahora podrá ser utilizado para obtener valor de los registros extraídos de las piscinas. Se diseñó:

- Una arquitectura web sobre la arquitectura de procesamiento de datos de ECR capaz de ejecutar múltiples análisis de datos, en diferentes ambientes de dependencias de Python que no suelen ser compatibles entre sí, dando una escalabilidad, flexibilidad y robustez al sistema.
- Se actualizó el módulo de analítica de datos para que soporte las nuevas funcionalidades.
- Se modularizó el código y se provee de una clase padre en Python encargada de toda la gestión de las tareas de Luigi, así como de obtener las métricas y los parámetros de entrada, requeridos por los análisis, en un diccionario. Además, maneja los errores que se puedan ejecutar durante el análisis.
- Los desarrolladores podrán usar esta arquitectura web para desplegar los futuros modelos de inteligencia artificial desarrollados por ECR de una forma más rápida y escalable, ahorrando mucho tiempo que podrá ser usado para otras actividades dentro de la empresa.
- Se proveyó de documentación detallada y organizada, dependiendo de las necesidades de los programadores, la cual está disponible en línea. En esta se explica con conceptos y ejemplos el cómo usar múltiples piezas de código que podrán ser reutilizadas para generar: interfases web, contenedores de Docker, tareas de Luigi, gestión de archivos multimedia, manejo de errores, notificaciones y definición de plantillas de nuevos análisis en archivos de texto en formato JSON.

4.2 Recomendaciones

1. Se recomienda desplegar nuevos análisis en otros lenguajes de programación, para que los nuevos análisis no necesariamente deban estar

escritos en Python. Lenguajes tales como R, Rust u otros lenguajes que son usados para procesar datos.

2. Implementar un manejo aún más detallado de la recolección de errores de las subtareas de Luigi en la ejecución de los análisis.
3. Desarrollar un modo administrador que pueda permitir habilitar o deshabilitar análisis desde un panel de control y hacer cambios en la descripción de los parámetros o en la plantilla de los análisis.
4. Implementar un subsistema que permita, desde la plataforma web, hacer y probar variaciones en los modelos de inteligencia artificial ya implementados, algoritmos y/o en el procesamiento de datos. Así mismo, que permita realizar test con datos ingresados por el administrador, para comprobar que el funcionamiento de los análisis sea óptimo a lo largo del tiempo.

BIBLIOGRAFÍA

[1] CNA Ecuador (2021). Camarón – Reporte de Exportaciones Ecuatorianas Totales [Online]. Disponible en: https://www.cna-ecuador.com/estadisticas/

[2] Introduction to Celery (2021). What's a Task Queue? [Online]. Disponible en: https://docs.celeryq.dev/en/stable/getting-started/introduction.html

[3] IBM Cloud Learn Hub (2020). ETL (Extract, Transform, Load) [Online]. Disponible en: https://www.ibm.com/cloud/learn/etl

[4] Luigi (2020). Luigi [Online]. Disponible en: https://luigi.readthedocs.io/en/stable/

[5] M Erdmann, B Fischer, R Fischer, & M Rieger (2017). Design and Execution of make-like, distributed Analyses based on Spotify's Pipelining Package Luigi. *Journal of Physics: Conference Series, 898, 072047*, doi :10.1088/1742-6596/898/7/072047

[6] C Burgos, L Moya (2021), *Herramienta Web de analítica de datos que permita explotar los modelos de Machine Learning*, Tesis de Estudios Superiores, FIEC, ESPOL, Guayaquil, Ecuador, 2021.

[7] Stuart J. Russell, Peter Norvig, Artificial Intelligence: A Modern Approach (2010), Prentice Hall.

[8] Nasteski, V. (2017). An overview of the supervised machine learning methods.

[9] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[10] Martins, A. F., Smith, N. A., Figueiredo, M., & Aguiar, P. (2011, July). Structured sparsity in structured prediction. In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing.

[11] Brefeld, U., & Scheffer, T. (2006, June). Semi-supervised learning for structured output variables. In Proceedings of the 23rd international conference on Machine learning.

[12] Open-Source Computer Vision. Disponible en: https://docs.opencv.org/4.x/index.html

[13] Docker (2022). Docker volumes [Online]. Disponible en: https://docs.docker.com/storage/volumes/

[14] Docker (2022). Docker volumes [Online]. Disponible en: https://docs.docker.com/develop/develop-images/image_management/

[15] Docker (2022). Manage images [Online]. Disponible en: https://docs.docker.com/network/

[16] Redis (2022). Redis [Online]. Disponible en: https://redis.io/

APÉNDICES

APÉNDICE A

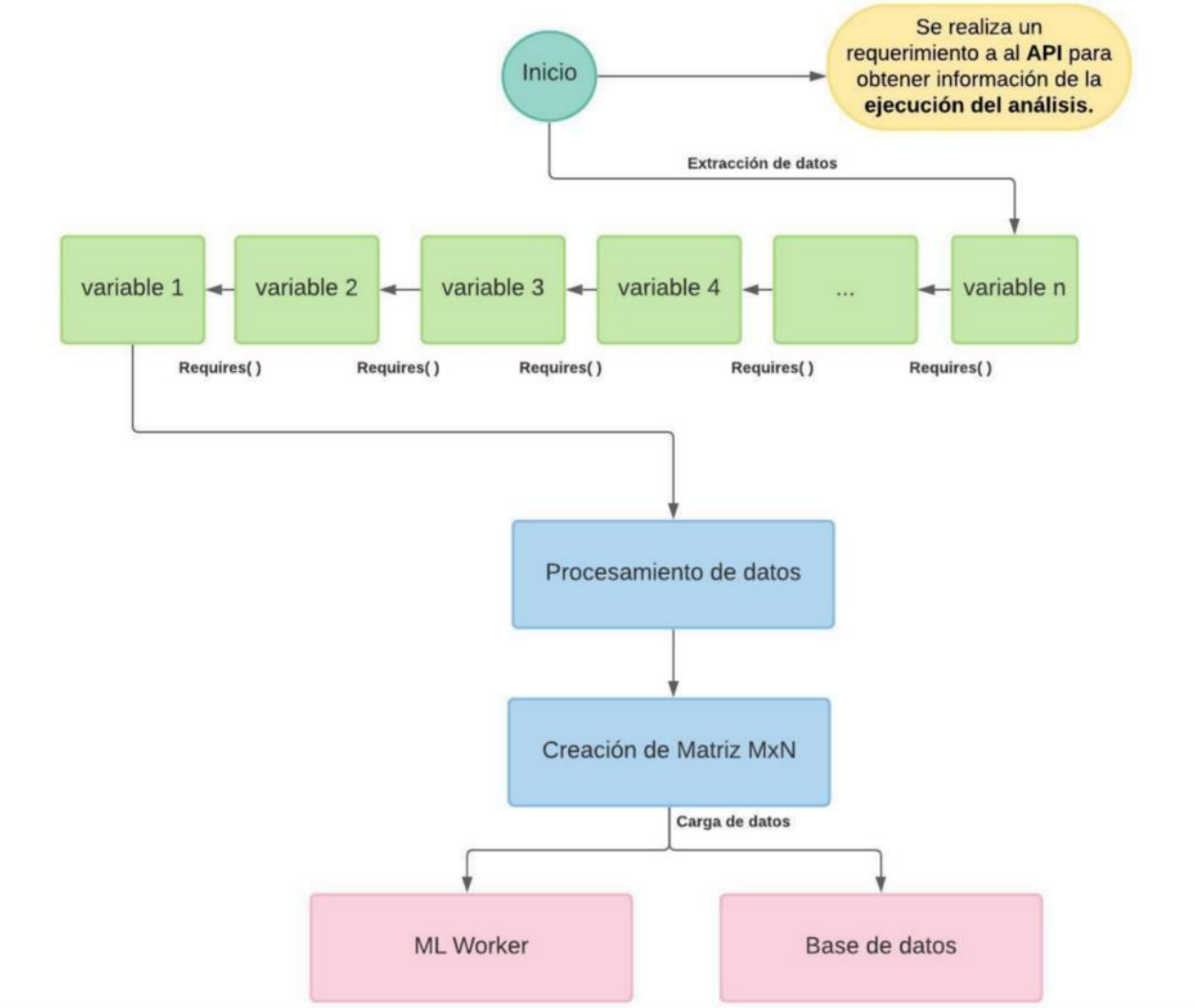


Figura A. 1 Anterior lógica del proceso ETL de la arquitectura de procesamiento de datos. [6]

APÉNDICE B

Reporte de ejecución de los modelos.

El planificador desarrollado verifica si los análisis requieren de un contendor de Docker o se pueden ejecutar en el mismo ambiente del ml_backend. El planificador se encarga de crear el contendor con las variables de entorno necesarias para ejecutar el análisis, siendo el identificado de la tarea en la base de datos, la conexión en el contendor de MongoDB, además, en caso de requerirlo el volumen de Docker donde se almacenan los archivos multimedia.

Esto permite ejecutar múltiples tareas de Luigi en el mismo proceso de Celery, donde se recogen las salidas de los contendores de Docker y se puede tener un monitoreo de las actividades que se realizaron. Tales como se muestran a continuación:

```
[2022-08-11 17:11:59,559: WARNING/ForkPoolWorker-1] INFO:
===== Luigi Execution Summary =====

Scheduled 1 tasks of which:
* 1 ran successfully:
    - 1 TaskPredictive(idtask=DPTask object)

This progress looks :) because there were no failed tasks or missing dependencies

===== Luigi Execution Summary =====
[2022-08-11 17:11:59,559: INFO/ForkPoolWorker-1]
===== Luigi Execution Summary =====

Scheduled 1 tasks of which:
* 1 ran successfully:
    - 1 TaskPredictive(idtask=DPTask object)

This progress looks :) because there were no failed tasks or missing dependencies

===== Luigi Execution Summary =====

[2022-08-11 17:11:59,560: WARNING/ForkPoolWorker-1] Tiempo de Ejecución:
[2022-08-11 17:11:59,560: WARNING/ForkPoolWorker-1] 4.902209997177124
```

Figura B.1 Resumen de ejecución de tareas de Luigi del modelo de predicción de peso [autoria propia]

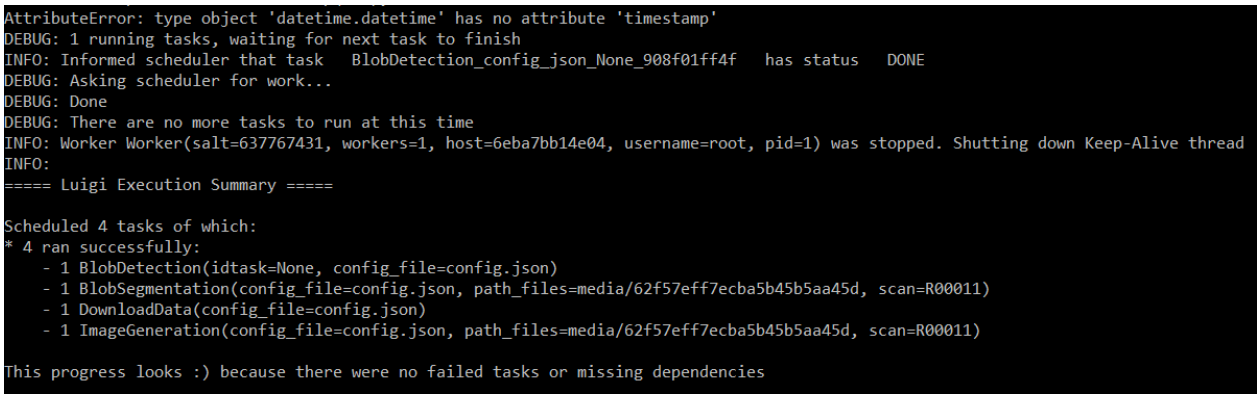


Figura B.2 Resumen de ejecución de tareas de Luigi del modelo de cálculo de biomasa [autoria propia]

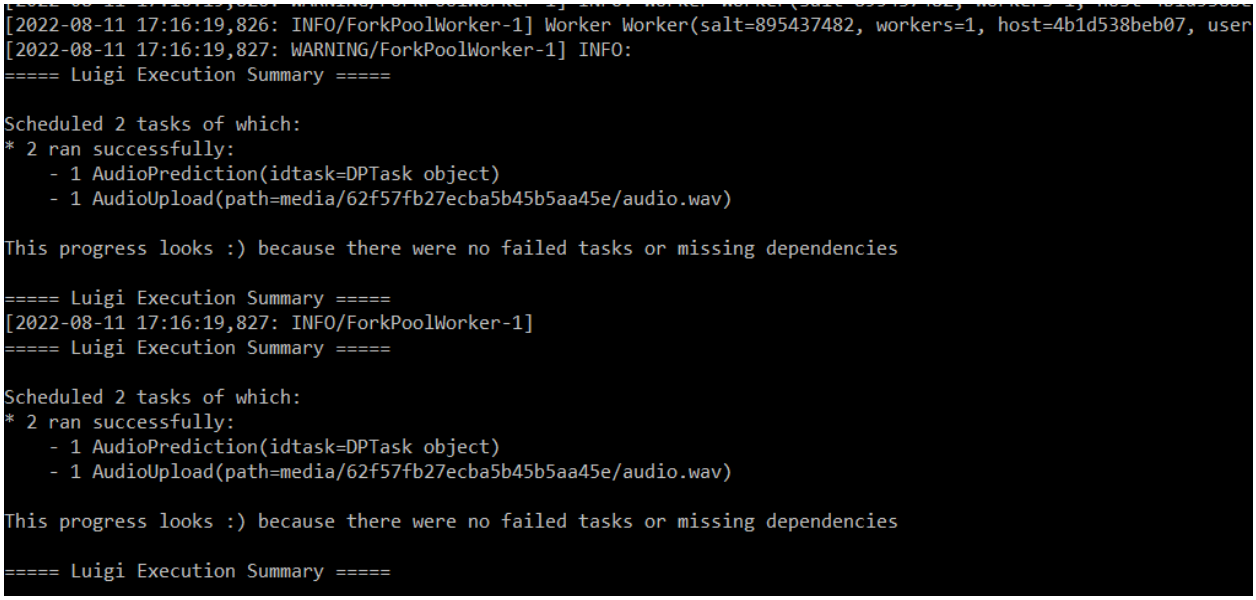


Figura B.3 Resumen de ejecución de tareas del modelo de clasificación de audios [autoria propia]

APÉNDICE C

Análisis de clasificación de audios.

Analiza los audios capturados por los hidrófonos de una piscina, con el objetivo de determinar una clasificación. Tiene como único parámetro un archivo de audio de n duración. El audio se carga y envía al preprocesamiento de datos, que lo analizar y determina a que clase pertenece.

El análisis se ejecutó en el mismo ambiente del ml_backend por lo que no requiere de un contenedor de Docker. Se proveyó de una forma para poder descargar los archivos que genera este análisis en la vista de resultados.

APÉNDICE D

N Actividades	2-Jun	9-Jun	16-Jun	23-Jun	30-Jun	7-Jul	14-Jul	21-Jul	28-Jul	4-Ago	11-Ago	18-Ago	25-Ago	1-Set	8-Set	15-Set
Reunión con el tutor y cliente																
Plantear problema, objetivos y alcance del proyecto																
Estudiar las librerías usadas en la arquitectura de procesamientos de datos																
Estudiar la integración parcial del módulo de analítica con la aplicación WEB de BioDynamics																
Desarrollar nuevas APIs a utilizar																
Modularizar ejecución de tareas de Luigi para Python																
Definir los parámetros y resultados de los modelos																
Desarrollar la interfaz de la aplicación																
Desarrollar los métodos para estandarizar la entrada de datos a los algoritmos de inteligencia artificial																
Establecer métodos para el manejo de errores																
Establecer métodos para el almacenamiento de contenido multimedia, y volumes de Docker																
Crear estándar para crear contenedores de Docker																
Desarrollar el planificador de contenedores de Docker																
Integrar los modelos de predicción de pesos, calculo de biomasa y eventos anómalos																
Pruebas																
Documentación de usuario e implementación																

APÉNDICE E

Manual de usuario

- **Ejecutar un análisis**

Por medio de un formulario que se renderiza y actualiza dependiendo del análisis que se quiera realizar. Se puede mandar a ejecutar un análisis que será encolado a la lista de tareas.

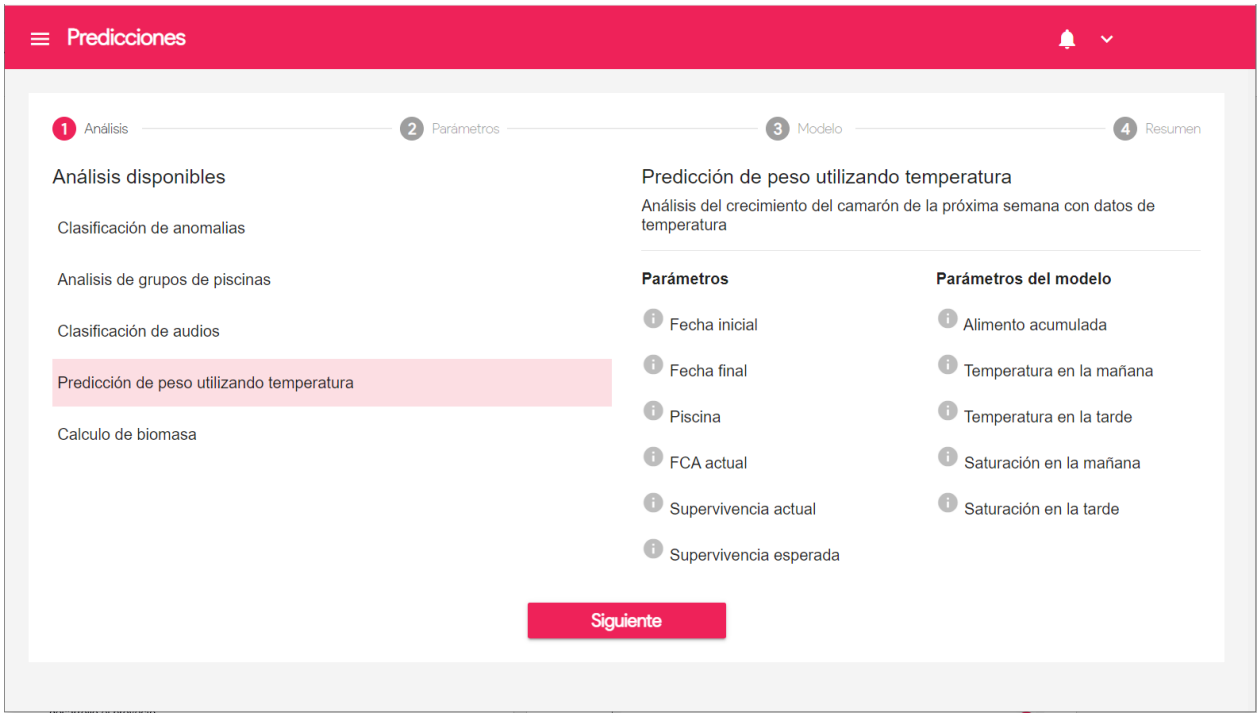


Figura D.1 Selección de análisis [autoria propia]

Dependiendo del análisis y los parámetros que el programador haya definido se renderizarán inputs específicos que el usuario debe de rellenar, los formularios tienen soporte para todos los tipos de inputs de HTML y también para archivos multimedia.

Las tareas que se manden a ejecutar serán listadas en una tabla filtradas por el usuario.

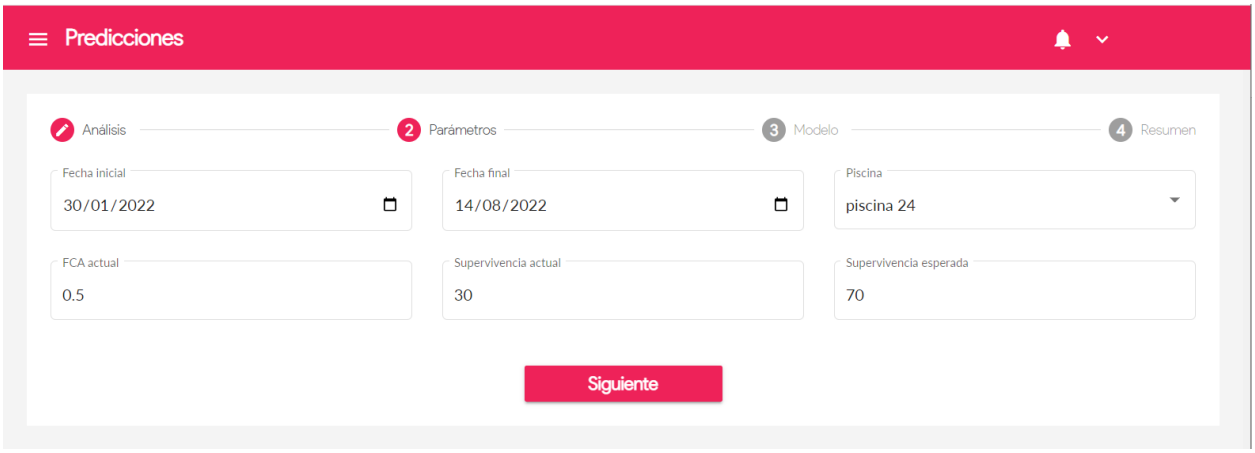


Figura D.2 Formulario para predicción de peso [autoria propia]

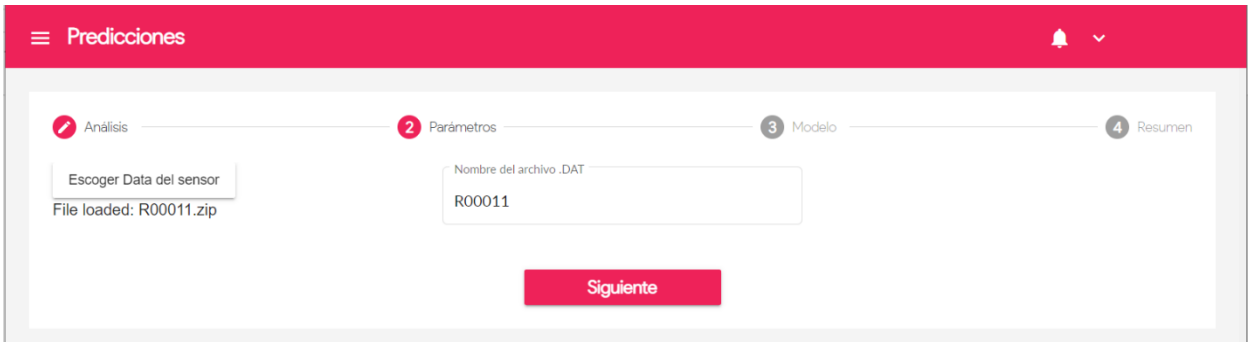


Figura D.3 Formulario para cálculo de biomasa [autoria propia]

También se le dio soporte a los análisis que pueden usar más de un modelo o variaciones de este para ejecutarse; si el modelo seleccionado tiene sus propios parámetros se pueden renderizar y variar dependiendo de la selección.

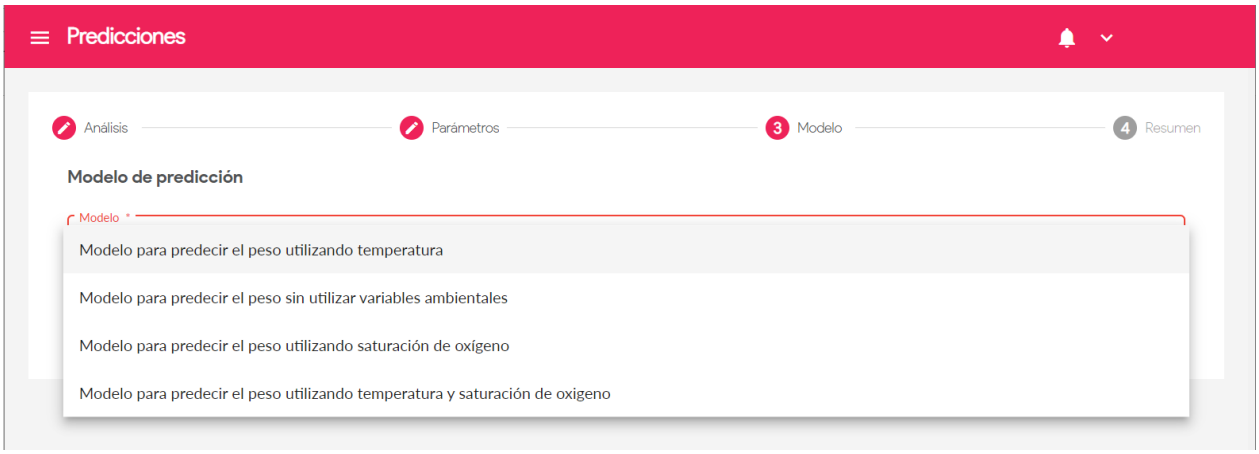


Figura D.4 Formulario para seleccionar un modelo [autoria propia]

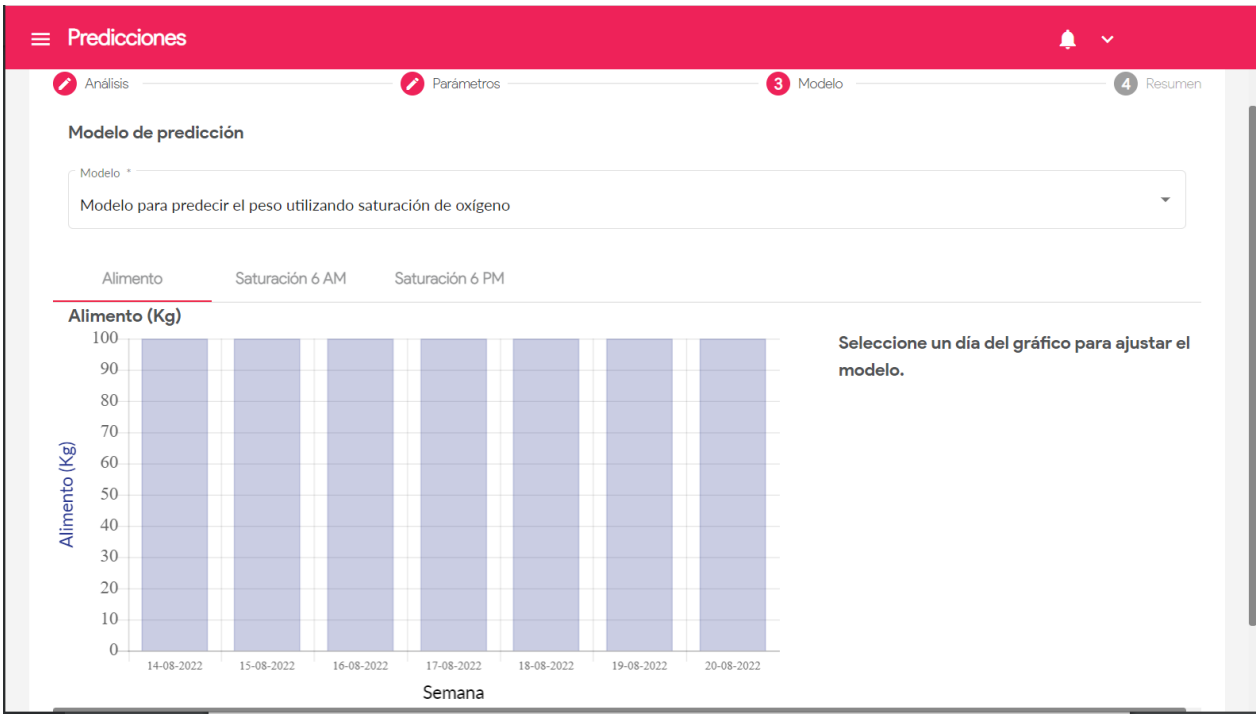


Figura D.5 Parámetros del modelo [autoria propia]

Figura 3.9 Finaliza con un resumen de los parámetros ingresados e información del análisis [autoria propia]

Las tareas en ejecución se listan en una tabla, donde se visualiza su estado, fecha de ejecución, tipo y modelo de análisis. En caso de error se actualiza el estado.

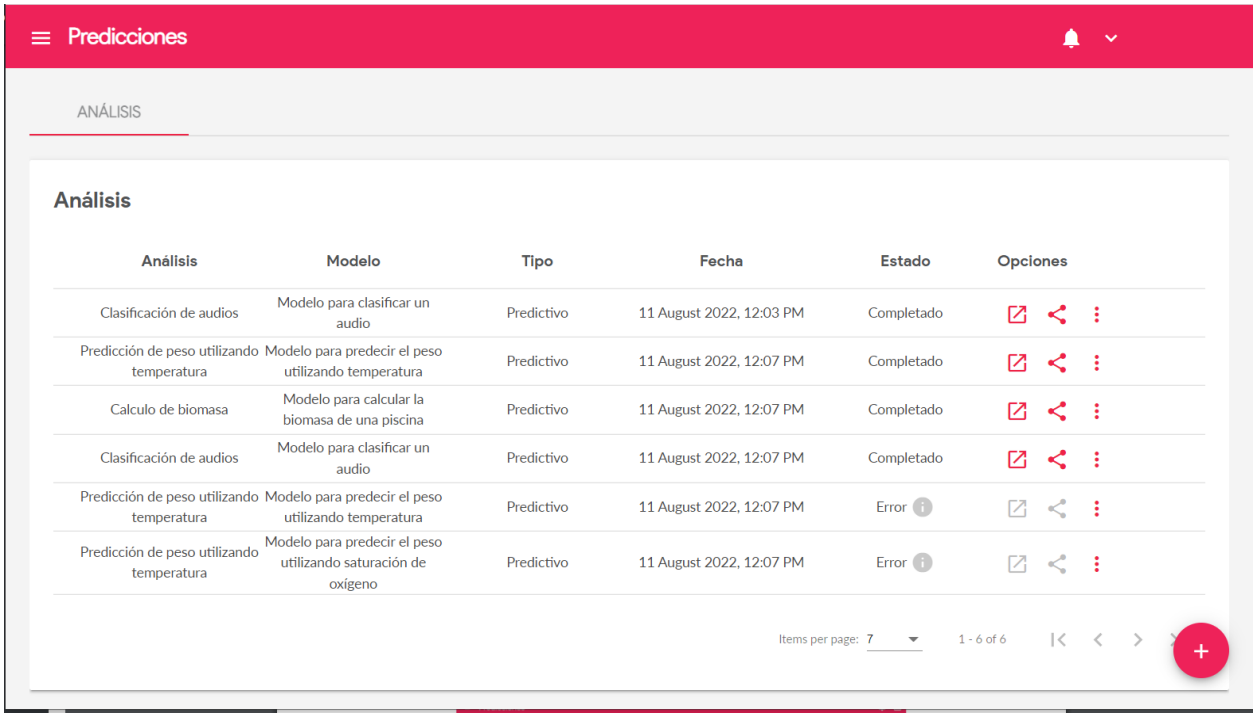


Figura D.6 Tabla de análisis encolados [autoria propia]

- **Visualización de resultados**

Cada análisis define su propia forma de representar los resultados, se proveyó de métodos que permiten hacer gráficas simples, descarga archivos en un comprimido o archivos individuales, reproducir contenido multimedia, audios o imágenes, y previsualización de archivos. También, se documentó como usar estos métodos para definir personalización en resultados de los análisis.

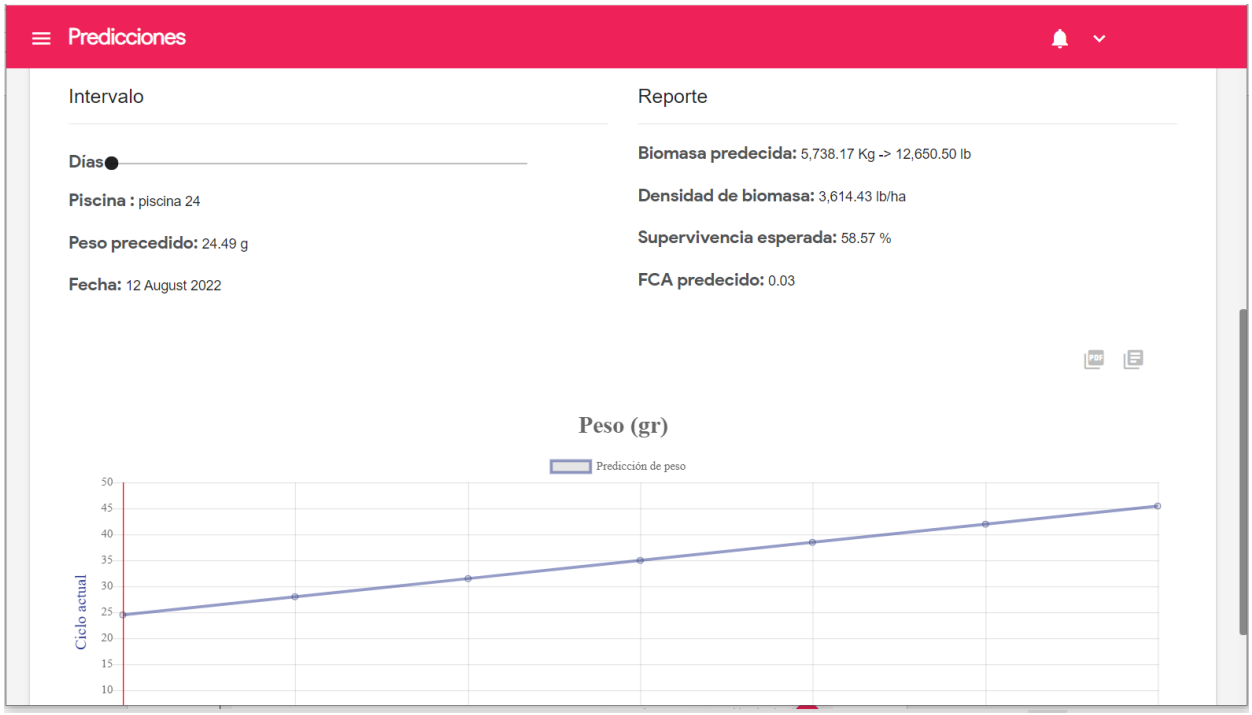


Figura D.7 Resultados de la predicción de peso [autoria propia]

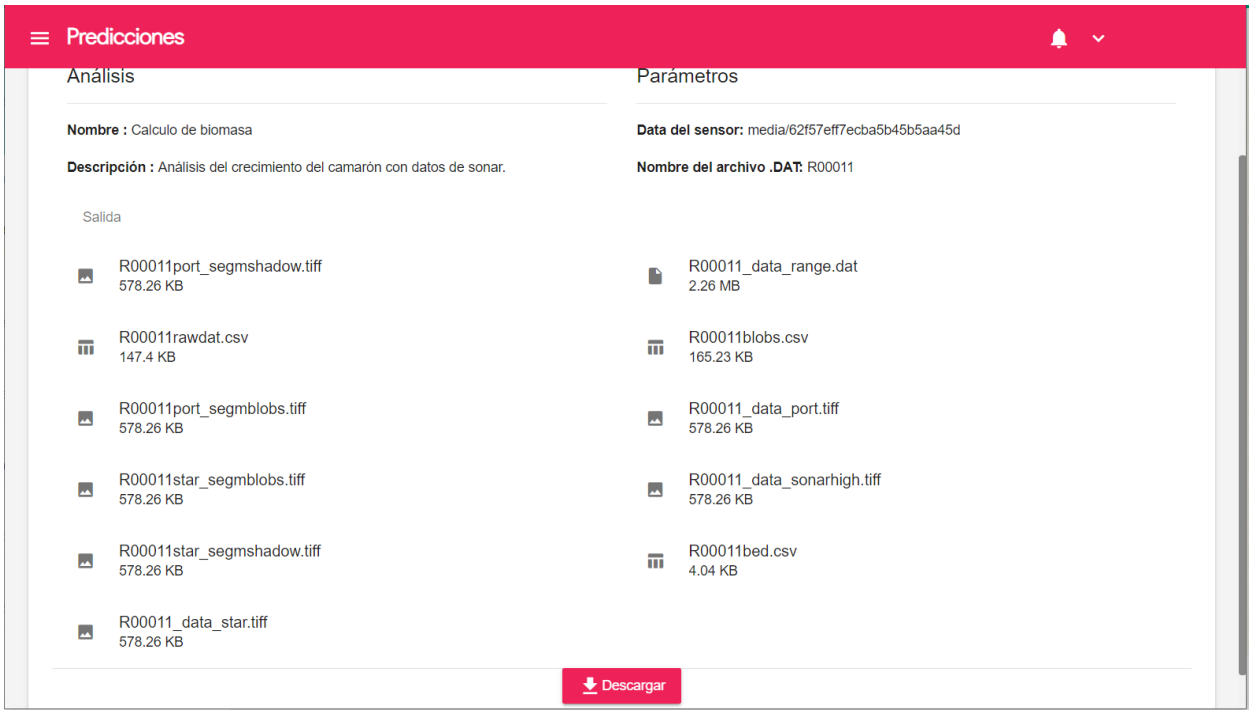


Figura D.8 Resultados del cálculo de biomasa [autoria propia]

APÉNDICE F

Manual de instalación

- **Levantar un contenedor**

Si el análisis requiere de un ambiente que no puede ser ejecutado en el propio ambiente del ml_backend, entonces es necesario crear un contenedor de Docker capaz de ejecutar el análisis con las dependencias necesarias.

Está disponible en la documentación en línea sección Docker, un instructivo detallado para levantar un contenedor de Docker que pueda ser usado por el Scheduler descrito en el capítulo 2.

- **Despliegue de contenedores de Docker**

Para automatizar el despliegue de los contenedores de Docker se creó un script capaz de hacerlo de forma automática, solo ejecutando un comando. Esto puede servir si en algún momento se necesita de implementar integración continua.

- Bajar los cambios de GitHub

Es suficiente con clonar o bajar los cambios del repositorio para que todo el código esté actualizado.

- Actualizar contenedores

Ejecutar el archivo deployment.sh para eliminar y volver a levantar los contenedores de Docker. El ml_backend está configurado de tal forma que los archivos multimedia y la base de datos se conservan sin importar que se eliminen los contenedores. Contenido del archivo:

```
docker-compose down
docker-compose up --build -d
```

El servicio se levantará en el puerto 5000, puede usar su enrutador de su preferencia para exponer el ml_backend, tal como **nginx**.

- Contenedores

Se crearán 4 contenedores en una misma red de Docker, encargados del: ml_backend, el worker que es cola de Celery, la base de datos en Mongo y la base de datos en Redis.

Los contenedores para los análisis que lo requieran, deben ser creados manualmente, antes o después de desplegar el ml_backend. En caso de no generarlos, el sistema comprueba la existencia de la imagen, y en caso de que no exista, no se inicializa el análisis.

- Estáticos de la aplicación

Es necesario que la aplicación desarrollada en Angular sea actualizada cuando haya cambios en el comportamiento de los formularios, o se muestren los resultados de forma diferente.

```
npx ng build --base-href=/app/ --configuration=production
```

Los archivos se generan en la carpeta dist.

Para exponer los estáticos generados puede usar su enrutador de su preferencia para exponer el ml_backend, tal como nginx.

APÉNDICE G

Diagrama de arquitectura

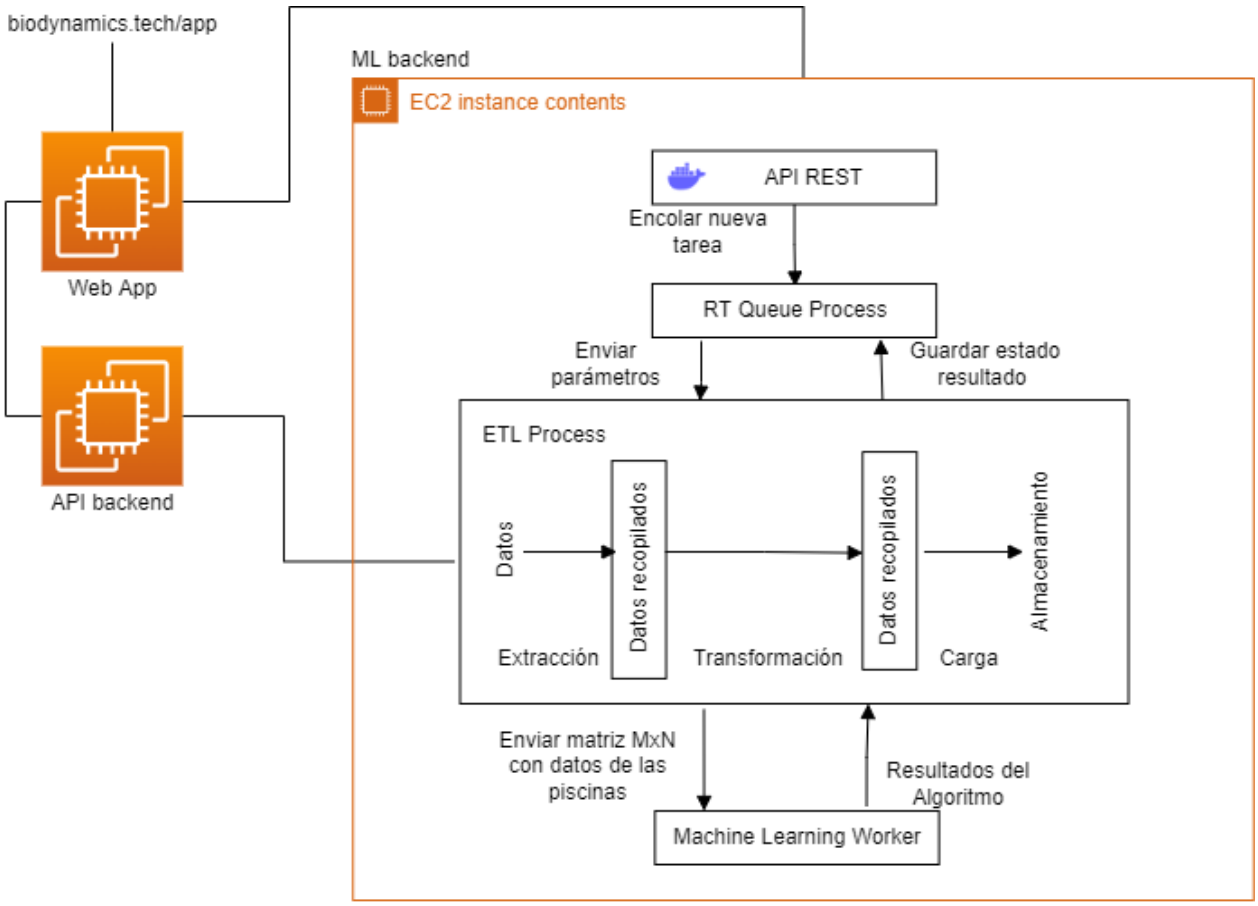


Figura 4.1 Arquitectura de las aplicaciones web de ECR [Autoria propia].

APÉNDICE H

Clase TaskLuiguiTemplate

Con el objetivo de desacoplar y reutilizar el código se creó una clase de Python llamada TaskLuiguiTemplate, que tiene como objetivo ser la clase padre de las tareas por análisis.

La clase contiene el funcionamiento de varios procesos de forma predeterminada, como el manejo de errores, extracción de registros, lectura de archivos multimedia, notificaciones, manejo de contenedores de Docker y envió de emails. Las funcionalidades descritas, también son clases y método en bloques de código. Si el desarrollador lo necesita, puede definir una nueva clase padre que solo integre funcionalidades que vea necesarias.