



Universidad de
Oviedo



Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

ÁREA DE ORGANIZACIÓN DE EMPRESAS

TRABAJO FIN DE MÁSTER

**CONCEPTUALIZACIÓN Y DESARROLLO DE UN PROTOTIPO DE
GEMELO DIGITAL PARA LA INDUSTRIA ACUÍCOLA**

**D. LÓPEZ TREITIÑO, Pedro
TUTOR: D. LUNA GARCÍA, Manuel**

FECHA: JULIO 2025



DECLARACIÓN DE ORIGINALIDAD

D./Dña. ...Pedro López Treitiño..... con DNI. ...10887271Z.....

como alumno/a del Máster Universitario en Ingeniería Informática de la Universidad de Oviedo, DECLARO que el Trabajo Fin de Máster titulado

CONCEPTUALIZACIÓN Y DESARROLLO DE UN PROTOTIPO DE GEMELO DIGITAL PARA LA INDUSTRIA ACUÍCOLA

.....
.....
.....
.....

es de mi autoría, es original y las fuentes bibliográficas utilizadas han sido debidamente citadas.

En Gijón a 11 de Julio de 2025

Firmado:.....

Pedro López Treitiño



UNIVERSIDAD DE OVIEDO

Escuela Politécnica de Ingeniería de Gijón

Hoja 3 de 142

Agradecimientos:

A mi tutor, gracias por su paciencia y valiosos consejos. A los profesores, gracias por compartir su conocimiento y por inspirarme a alcanzar nuevas metas.

Pedro López Treitiño

Resumen

La toma de decisiones en entornos empresariales es una tarea compleja, especialmente en sectores como el agroalimentario, donde confluyen múltiples factores ambientales, biológicos y económicos que dificultan la planificación y el control de las operaciones. En este contexto, resulta clave contar con nuevos sistemas de apoyo basados en datos, capaces de proporcionar una visión integrada y dinámica de los procesos productivos, facilitando una toma de decisiones más informada, precisa y flexible.

El presente Trabajo Fin de Máster consiste en la conceptualización y desarrollo de un prototipo de Gemelo Digital para la industria acuícola. Los Gemelos Digitales son equivalentes digitales de un objeto de la vida real del que reflejan, en un espacio virtual, su comportamiento y estados a lo largo de su vida útil. Desde el punto de vista de la gestión empresarial, estos ayudan a desvincular los flujos físicos de las tareas de planificación y control, permitiendo no solo gestionar las operaciones a distancia, sino también simular los efectos de decisiones concretas a partir de datos reales.

Los gemelos digitales ya han sido aplicados y validados en diferentes áreas de estudio, por lo que su aplicación en sectores de producción de productos agroalimentarios (agricultura, ganadería, acuicultura) podría ser una solución interesante de apoyo a la toma de decisiones, dadas las grandes ventajas que supondría para aumentar la eficiencia y la productividad. Sin embargo, su desarrollo en estos sectores es aún incipiente, lo que implica que no es posible abordar directamente la construcción de un gemelo digital sin una fase previa de conceptualización y análisis. En consecuencia, resulta necesario definir con claridad qué técnicas y métricas deben integrarse en estos sistemas para garantizar su utilidad predictiva y aplicabilidad real en el entorno operativo.

Palabras clave

Gemelo digital, IoT, Series temporales, Toma de decisiones, Software Libre.

Comentado [MLG1]: En general , hay que saber introducir la complejidad y el hecho de no irnos a un gemelo digital completo sino a su conceptualización y demás...

Esto es así para el resumen y también para el objetivo Te pongo un ejemplo, quizás muy extenso pero para que se vea la línea a transmitir

Summary

Decision-making in business environments is a complex task, particularly in sectors such as the agri-food industry, where multiple environmental, biological, and economic factors converge, making operational planning and control challenging. In this context, it is crucial to have new data-driven support systems capable of providing an integrated and dynamic view of production processes, facilitating more informed, precise, and flexible decision-making.

This master's Thesis focuses on the conceptualization and development of a Digital Twin prototype for the aquaculture industry. Digital Twins are virtual equivalents of real-life objects, mirroring their behavior and states throughout their lifecycle in a virtual space. From a business management perspective, they help decouple physical flows from planning and control tasks, not only enabling remote operation management but also allowing for the simulation of the effects of specific decisions based on real data.

Digital Twins have already been applied and validated in various fields of study, suggesting that their application in agri-food production sectors (agriculture, livestock, aquaculture) could be an interesting solution to support decision-making, given the significant advantages they offer for increasing efficiency and productivity. However, their development in these sectors is still nascent, which means that it is not possible to directly address the construction of a Digital Twin without a prior conceptualization and analysis phase. Consequently, it is necessary to clearly define which techniques and metrics should be integrated into these systems to ensure their predictive utility and real-world applicability in the operational environment.

Keywords

Digital Twin, IoT, Time Series, Decision Making, Free Software.

Con formato: Español (España)

Con formato: Normal

MEMORIA

Resumen	4
Palabras clave	4
Summary	5
Keywords	5
1 Introducción	11
1.1 Gemelo Digital	11
1.2 Clasificaciones y Construcción de Gemelos Digitales	14
1.2.1 Clasificación de Gemelos Digitales por Área de Aplicación	14
1.2.2 Clasificación de Gemelos Digitales por nivel de madurez	16
1.2.2.3 Conclusiones: Definiendo el Alcance del Gemelo Digital	17
1.3 Objetivo general de este TFM	18
2 Estado del arte de la aplicación de Gemelos digitales en las explotaciones del sector primario.	19
2.1 Granjas inteligentes	19
2.1.1 Ejemplo de granja inteligente: Proyecto SWAMP	20
2.2 Acuicultura	23
2.2.1 Ejemplo de piscifactoría terrestre operativa en Italia	25
2.2.1.1 La fase de observación.	26
2.2.1.2 La fase de interpretación.	27
2.2.1.3 La fase de decisión.	28
2.2.1.4 La fase de actuación	29
3 Análisis de requisitos	30
3.1 Requisitos funcionales	30
RF1 Gestión de Jaulas Marinas.	30
RF2 Modelos predictivos sobre series temporales.	31
RF3 Cuadro de mandos que muestre los datos al usuario.	32

Pedro López Treitiño

RF3.1 Representación de datos.	32
RF3.2 Detalle de datos de la balsa actual.	33
RF3.3 Representación de la balsa actual mostrando la biomasa.	33
RF3.4 Listado de balsas con detalle de valores óptimos de cosecha.	33
RF3.5 Control de tiempo actual y futuro.	33
RF3.6 Importación de datos.	34
RF3.7 Gestión de predicciones.	34
Requisitos funcionales fuera de alcance de este TFM (Oportunidades)	36
Alertas inteligentes: Notificaciones automáticas	36
3.2 Requisitos no funcionales	37
RNF 1. Ejecución multiplataforma.	37
RNF 2. Uso de librerías de ciencia de datos estándar.	37
RNF 3. Uso de software libre.	38
Requisitos No Funcionales fuera del alcance de este TFM (Oportunidades)	40
Conectividad (Futura)	40
Bases de datos	41
3.3 Roles de las Historias de Usuario	42
3.4 Casos de uso del TFM	43
4 Materiales y métodos	45
4.1 Series temporales	45
4.2 Análisis de series temporales.	50
4.3 Modelo Prophet	51
4.4 Modelo Gradient Boosting y LightGBM	54
4.5 Análisis de crecimiento de Biomasa	55
4.5.1 Crecimiento de los Peces y función logística	55
4.5.2 Modelo de Crecimiento de Thyholdt (2014) para el Salmón Noruego	58
4.5.3 Conclusión	60
4.6 Planificación	62
5 Diseño	64

5.1 Introducción	64
5.2 Arquitectura del sistema	66
5.3 Diseño de componentes	67
5.3.1 Modelo	67
5.3.1.1 Modelo GrowthModel	67
5.3.1.2 Modelo seaTemperature	74
5.3.1.3 Modelo priceModel	81
5.3.2 Vista	89
5.3.3 Controlador	89
5.4 Diseño de la interfaz de usuario (UI)	101
5.4.1 Diseño preliminar	102
5.4.2 Implementación Final	103
5.5 Diseño de sistema de persistencia de datos	106
5.5.1 Resumen del Diagrama: Sistema de persistencia actual de SalmonTwin	106
5.5.2 Flujo de Datos	108
5.5.3 Stack Tecnológico	109
6 Resultados y pruebas	110
6.1 Introducción	110
6.1.1 Tecnología utilizada	110
6.2 Plan de Pruebas	114
6.2.1 Resumen de pruebas realizadas	115
6.2.1.1 Modelo de crecimiento	115
6.2.1.2 Modelo de temperatura	117
6.2.1.3 Modelo de precios	121
6.2.2 Detalle de pruebas realizadas	123
6.3 Pruebas de usuario usando la aplicación directamente.	139
7 Referencias	140

Con formato: Justificado, Espacio Despues: 12 pto,
Interlineado: 1,5 líneas

Índice de Ilustraciones

Ilustración 1 CONCEPTUALIZACIÓN DEL GEMELO DIGITAL	13
Ilustración 2 Cinco niveles gemelo digital en las operaciones de un proceso (Udugama, 2021).....	16
Ilustración 3 Capas de arquitectura del proyecto SWAMP.....	20
Ilustración 4 Arquitectura y servicios usados	21
Ilustración 5 Tanques de canalización, el tanque de oxígeno líquido (tanque blanco a la izquierda) y el pórtico (que contiene la pasarela móvil que cruza las pistas de rodadura).	26
Ilustración 6 Las cuatro fases del funcionamiento del gemelo digital.	27
Ilustración 7 Esquema de entradas y salidas a los modelos dinámicos bioenergético y DO (oxígeno disuelto)	28
Ilustración 8 Diagrama de Casos de Uso.....	43
Ilustración 9 Serie de Viajes Aéreos Internacionales; los datos aumentan continuamente desde 1949 hasta 1961. La tendencia aumenta.....	46
Ilustración 10 Estacionalidad aditiva sin tendencia. La altura entre picos y valles permanece constante. ..	46
Ilustración 11 Estacionalidad aditiva con tendencia creciente. La altura entre picos y valles permanece constante siguiendo una tendencia lineal creciente.....	46
Ilustración 12 Estacionalidad Multiplicativa sin tendencia. La altura entre picos y valles aumenta siguiendo una constante multiplicativa. No se observa tendencia.....	47
Ilustración 13 Estacionalidad Multiplicativa con tendencia. La altura entre picos y valles aumenta siguiendo una constante multiplicativa. Se observa una tendencia creciente.	47
Ilustración 14 En el siguiente gráfico, puede ver los patrones que se repiten en cada 100.000 años.	48
Ilustración 15 Proceso de refinamiento de la previsión	51
Ilustración 16 Número de eventos creados en Facebook.	52
Ilustración 17 Ejemplo de Optimización Leaf-wise tree growth.....	54
Ilustración 18 función logística, curva logística o curva en forma de S	56
Ilustración 19 curvas logísticas para diversos valores de r con $N_0 = 10$ y $K = 100$	57
Ilustración 20 Thyholdt (2014) función de crecimiento.....	59
Ilustración 21 Gantt del Prototipo.....	62
Ilustración 22 Diagrama de componentes	66
Ilustración 23 Diagrama de clase del GrowthModel.....	67
Ilustración 24 Diagrama de clase de DataTemperature.....	74
Ilustración 25 Diagrama de Clase DataPrice	81
Ilustración 26 Diagrama de clase de dashBoardController	89

Con formato: Fuente: +Cuerpo (Aptos), 10 pto

Ilustración 27 QT Creator Community	101
Ilustración 28 Implementación final de la ventana del cuadro de mandos.	103
Ilustración 29 Pantalla de configuración y creación de balsas.....	103
Ilustración 30 Pantalla de selección de balsas.....	104
Ilustración 31 Pantalla de búsqueda de párametros del modelo de precios	104
Ilustración 32 Pantalla de búsqueda de parámetros finalizada.....	105
Ilustración 33 Ventanas de información y error	105
Ilustración 34 Sistema de persistencia actual	106
Ilustración 35 Flujo de datos de persistencia	108
Ilustración 36 Ejemplo de función de prueba	111
Ilustración 37 Ejemplo de Fixture de Pytest	112
Ilustración 38 Árbol de ejecución de pruebas.....	112
Ilustración 39 Ejemplo de mensaje de fallo en la prueba.....	113
Tabla 1 Constantes de crecimiento para varias especies.....	59

1 Introducción

La toma de decisiones en el ámbito empresarial implica enfrentarse a entornos complejos, donde la incertidumbre, la variabilidad de los datos y la necesidad de actuar con rapidez hacen que la planificación y el control de las operaciones resulten especialmente desafiantes. Esta complejidad se acentúa en sectores como el agroalimentario —y en particular en la acuicultura—, donde factores biológicos, climáticos, económicos y logísticos interactúan de forma dinámica y condicionan el rendimiento de los sistemas productivos.

En este contexto, una de las líneas de trabajo del grupo de investigación GIO (Grupo de ingeniería de organización de la universidad de Oviedo) se centra precisamente en el desarrollo de soluciones de apoyo a la toma de decisiones basadas en datos para la industria acuícola, aplicando técnicas de modelización, simulación y analítica avanzada. Este enfoque, junto al interés del alumno que está realizando este TFM, por profundizar en el potencial de los gemelos digitales y las técnicas predictivas de aprendizaje automático, motivó la elección de un tema aún incipiente como este, con el objetivo de explorar sus posibilidades de aplicación práctica y contribuir a sentar las bases de futuros desarrollos en este campo.

1.1 Gemelo Digital

Es un término auto explicativo que denota una versión digitalizada de un sistema físico. Una de las primeras referencias de este término se encuentra en un trabajo de ficción de 1991 escrito por David Gelernter [1], en el que se relata un mundo espejo dentro de un sistema informático. A su vez los ingenieros de la NASA en su “paper” titulado “Structures, Structural Dynamics, and Materials Conference” [2], en el año 2012, usan el término gemelo digital.

Sorprendentemente, ya en 1960 la NASA describe un “modelo viviente” de la misión Apollo [3] que empleaba múltiples simuladores para la evaluación de fallos. A principios de 1970 los grandes “mainframes” se utilizaban como sistemas digitales para

Comentado [MLG2]: Antes de esto, la introducción debería tener un apartado de par de párrafos justificando el tema elegido (más en este caso que es un poco raro lo de la acuicultura). Podría ser algo como:

Justificación del tema:

La toma de decisiones en el ámbito empresarial implica enfrentarse a entornos complejos, donde la incertidumbre, la variabilidad de los datos y la necesidad de actuar con rapidez hacen que la planificación y el control de las operaciones resulten especialmente desafiantes. Esta complejidad se acentúa en sectores como el agroalimentario —y en particular en la acuicultura—, donde factores biológicos, climáticos, económicos y logísticos interactúan de forma dinámica y condicionan el rendimiento de los sistemas productivos.

En este contexto, una de las líneas de trabajo del grupo de investigación GIO se centra precisamente en el desarrollo de soluciones de apoyo a la toma de decisiones basadas en datos para la industria acuícola, aplicando técnicas de modelización, simulación y analítica avanzada. Este enfoque, junto al interés del alumno por profundizar en el potencial de los gemelos digitales y las técnicas predictivas de aprendizaje automático, motivó la elección de un tema aún incipiente como este, con el objetivo de explorar sus posibilidades de aplicación práctica y contribuir a sentar las bases de futuros desarrollos en este campo.

Con formato: Título 2

monitorizar centrales eléctricas. Y a partir de 1980 los sistemas de CAD 2D permitían producir esquemas técnicos para visualizar complejas instalaciones. Entre los años 1990 y 2000 la llegada del CAD 3D, el aumento de la potencia computacional, las bases de datos interconectadas y las redes de sensores fue preparando el terreno para poder tener monitorizaciones en tiempo real y simulaciones más precisas y detalladas. En los sistemas actuales de monitorización en tiempo real se suelen emplear: Lo que se conoce actualmente, como el internet de las cosas, que engloba términos como: IoT, M2M, OT [16], etc.

Aunque los simuladores y los gemelos digitales utilizan modelos para replicar varios procesos de un sistema, un gemelo digital es un entorno virtual mucho más rico que un simulador. La principal diferencia es una cuestión de escala. Un simulador suele centrarse en un proceso, mientras que un gemelo digital, ejecuta varios procesos simultáneamente que pueden influir unos en otros como en un todo [4]. Las simulaciones, por lo general, no se benefician de tener datos en tiempo real. Sin embargo, los gemelos digitales están diseñados en torno a un flujo de información bidireccional que se produce cuando los sensores proporcionan datos al gemelo digital y luego vuelve a ocurrir cuando los datos procesados en los modelos vuelven al sistema físico y se comparan con la telemetría del sistema físico, para buscar discrepancias o actuar como mando del sistema físico directamente.

Los gemelos digitales existen dentro de los sistemas informáticos, y se programan para ejecutar escenarios precargados, que se comportan como el sistema real que virtualizan. Para ello, el gemelo digital ejecuta los modelos del sistema físico, y se le introducen grandes cantidades de datos provenientes de sensores del mundo real o guardados digitalmente en archivos o bases de datos, siendo capaces de responder a esos escenarios y predecir evoluciones futuras del sistema. Esto, permite "predecir" problemas futuros y probar posibles estrategias de actuación que las mitiguen. También son capaces de visualizar el estado actual con un grado de detalle definido. Así como permitir la monitorización y el control remoto del sistema virtualizado conectando los actuadores del sistema real al gemelo digital.

Comentado [MLG3]: No entiendo esta frase aquí suelta

Complementa la anterior ¿ debería ir seguida?

Comentado [PL4R3]: Es para explicar como se nombran las tecnologías que se utilizan más frecuentemente para estos temas.

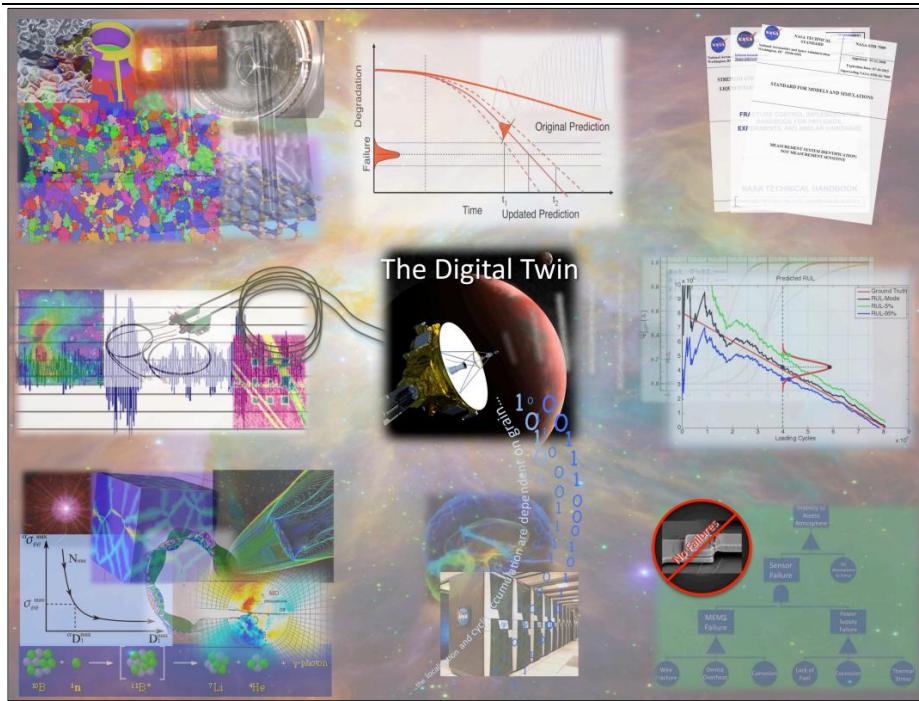


Ilustración 1 CONCEPTUALIZACIÓN DEL GEMELO DIGITAL

Con formato: Centrado

1.2 Clasificaciones y Construcción de Gemelos Digitales

Los **gemelos digitales** son réplicas virtuales avanzadas de objetos o procesos físicos. Para entenderlos mejor, podemos clasificarlos de dos formas principales: según su **área de aplicación** y su **nivel de madurez**. Estas dos clasificaciones están muy relacionadas, ya que el desarrollo de un gemelo digital a menudo sigue un camino desde elementos más pequeños y básicos hacia sistemas cada vez más complejos e integrados.

Sin embargo, no siempre es un proceso lineal. Se puede empezar con gemelos digitales de "proceso" (se define más adelante), que utilizan **sistemas "legacy" (antiguos)** y, al actualizar o mejorar estos sistemas, son convertidos en gemelos digitales de sistema. Esto significa que los gemelos digitales se pueden construir tanto de **abajo hacia arriba** (empezando por lo simple y añadiendo complejidad) como de **arriba hacia abajo** (empezando por un proceso completo e ir hacia abajo convirtiendo los sistemas que utilizan en gemelos digitales de sistema). En las siguientes secciones, explicaremos la clasificación desde un enfoque de **abajo hacia arriba**, yendo de lo más simple a lo más complejo.

1.2.1 Clasificación de Gemelos Digitales por Área de Aplicación

Esta clasificación categoriza los gemelos digitales según el alcance y la escala de lo que representan, avanzando desde componentes individuales hasta procesos operativos completos. Se puede imaginar que son bloques de construcción que se unen para formar un entorno virtual completo [4]. Se describen a continuación:



- **Gemelos Digitales de Componentes:** Son las unidades funcionales más básicas, que representan piezas individuales relevantes para el proceso que se está virtualizando. Por ejemplo, un sensor, una bomba o una válvula podrían tener cada uno su propio gemelo digital de componente. El nivel de detalle

de estos gemelos es crucial y se determina durante la fase de diseño.



- **Gemelos Digitales de Activos:** Cuando dos o más componentes trabajan juntos para realizar una función específica, forman un activo. Un gemelo digital de activo permite estudiar en profundidad cómo interactúan estos componentes. Esto genera datos de rendimiento valiosos que pueden analizarse para obtener información procesable, y también puede ayudar a identificar posibles incompatibilidades entre las diferentes partes.
- **Gemelos Digitales de Sistemas o Unidades:** Operando a un nivel superior, estos gemelos combinan múltiples activos individuales para representar un sistema o unidad más grande e integrado. Su propósito es comprender cómo estos activos funcionan colectivamente. Ejemplos incluyen un sistema eléctrico completo o el sistema de propulsión de un vehículo.
- **Gemelos Digitales de Procesos:** Son los gemelos digitales más completos en términos de alcance. Ilustran cómo varios sistemas y activos trabajan juntos para simular virtualmente una instalación de producción o un proceso operativo completo. Esto permite una observación detallada, como verificar si los sistemas están sincronizados para una máxima eficiencia o predecir cómo los retrasos en un sistema podrían propagarse y afectar a otros. Los gemelos digitales de procesos son invaluables para optimizar la eficacia operativa general.



1.2.2 Clasificación de Gemelos Digitales por nivel de madurez

En cuanto al nivel de madurez de uso se pueden clasificar en cinco niveles:

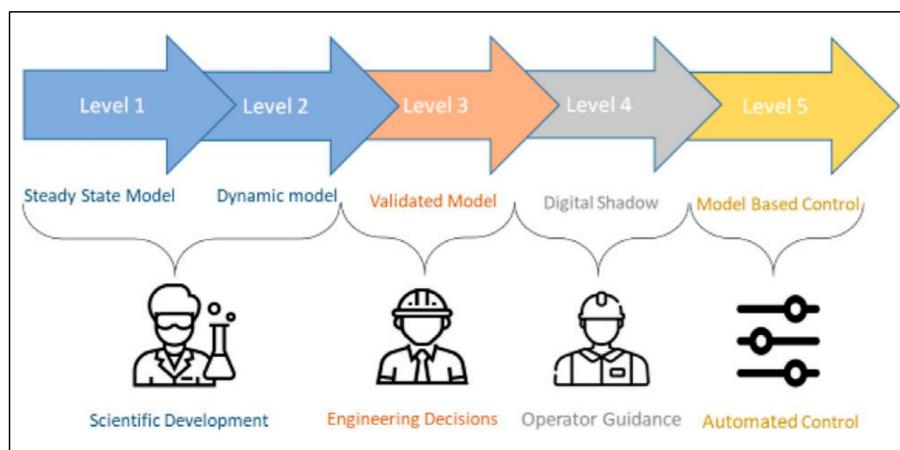


Ilustración 2 Cinco niveles gemelo digital en las operaciones de un proceso (Udugama, 2021).

Los niveles de madurez de un gemelo digital se clasifican según su complejidad y su grado de conexión con el proceso físico [5]:

- **Niveles Uno a Tres (Gemelos Basados en Modelos):** Estos gemelos digitales se fundamentan en modelos de procesos que van aumentando en complejidad y exhaustividad. En el **Nivel Tres**, el gemelo digital aún no está conectado en tiempo real con el proceso físico, pero se valida rigurosamente comparando sus resultados con los datos reales del proceso.
- **Nivel Cuatro (Sombra Digital):** Aquí, el gemelo digital se convierte en una **Sombra Digital**. Comienza a adquirir y utilizar datos en tiempo real directamente del proceso físico. Esta conexión en vivo permite al modelo predecir cómo evolucionarán las operaciones físicas en el futuro.

- **Nivel Cinco (Gemelo Digital Completo):** Este es el nivel más avanzado. Un gemelo digital completo para operaciones no solo utiliza datos del proceso en tiempo real para hacer predicciones futuras, sino que también tiene la capacidad de calcular y ejecutar acciones de control directamente sobre el proceso físico de producción. Es decir, el gemelo virtual puede intervenir y optimizar el funcionamiento de su contraparte real.

1.2.2.3 Conclusiones: Definiendo el Alcance del Gemelo Digital

Al abordar la creación de un **gemelo digital**, las fases iniciales de **análisis y diseño** son cruciales. Es en este punto donde se debe definir con precisión el **alcance** del gemelo. Esto implica tomar decisiones estratégicas basadas en las clasificaciones que hemos explorado: En primer lugar, hay que determinar el **nivel de madurez** deseado para el gemelo digital

Se debe decidir si será un modelo básico estático sólo representaciones CAD 2D/3D sin datos del proceso (Nivel 1), informativo con datos del proceso cargados de forma manual o automática y modelos predictivos (Nivel 2), informativo validando los datos de los modelos predictivos con los datos del proceso (Nivel 3), una Sombra Digital que predice el futuro a partir de datos obtenidos en tiempo real (Nivel 4), o un Gemelo Digital Completo capaz de ejecutar acciones de control (Nivel 5). Cada nivel de madurez implica una complejidad y una funcionalidad significativamente diferentes.

En segundo lugar, es vital especificar las **áreas de aplicación** que cubrirá el gemelo. Nos centraremos en un componente específico, un activo completo, un sistema integrado, o se buscará virtualizar un proceso productivo completo. La elección de estas "piezas" determinará la granularidad y la extensión de la réplica virtual.

La combinación de estas decisiones —el nivel de madurez y el área de aplicación— impactará directamente en la **magnitud del proyecto**, afectando significativamente el tiempo de desarrollo, los recursos necesarios para la construcción y, por supuesto, el presupuesto.

Finalmente, y no menos importante, el **tipo de industria o proceso** al que se destinará el gemelo digital es un factor determinante. Los requisitos y especificaciones del gemelo deben ajustarse meticulosamente a las características únicas de cada negocio. No es lo mismo diseñar un gemelo para la fabricación automotriz que para la acuicultura, ya que los datos, los modelos y los objetivos operacionales serán intrínsecamente diferentes. Adaptar el gemelo a estas particularidades es esencial para asegurar su utilidad y valor real en el entorno operativo.

1.3 Objetivo general de este TFM

El objetivo general de este Trabajo Fin de Máster es conceptualizar y desarrollar un prototipo de Gemelo Digital orientado a apoyar la toma de decisiones en una empresa de acuicultura marina con múltiples jaulas de producción. Este prototipo se enfocará en representar digitalmente el proceso productivo y anticipar escenarios futuros mediante el uso de técnicas predictivas, con el fin de facilitar la planificación y optimizar decisiones estratégicas como la fecha óptima de recogida y venta de la biomasa. Para ello, se estudiarán las necesidades de información, se seleccionarán modelos de predicción adecuados y se diseñará una interfaz de visualización útil para los responsables de gestión.

Comentado [MLG5]: Propuesta de nuevo más académica con objs específicos

Este objetivo general se desarrolla a través de los siguientes objetivos concretos:

- Conceptualizar el gemelo digital en base a las características y necesidades específicas del proceso de producción acuícola en jaulas marinas, identificando las variables clave para la toma de decisiones.
- Diseñar el prototipo del gemelo digital, considerando los niveles funcionales 1 a 3 (desconectado del sistema físico) y delimitando el alcance a la fase de planificación, excluyendo sensores, actuadores o sistemas de comunicación directa.

Con formato: Párrafo de lista, Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

Con formato: Párrafo de lista, Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

- Recolectar y estructurar los datos necesarios para alimentar los modelos del gemelo digital, incluyendo variables ambientales, técnicas, económicas y biológicas.
- Aplicar técnicas de predicción para estimar el crecimiento de biomasa, la evolución de la temperatura del agua y el precio esperado de venta, como base para la simulación de escenarios futuros.
- Elaborar un cuadro de mando visual, que permita a los gestores consultar de forma clara los resultados del gemelo digital y recibir recomendaciones sobre decisiones clave, como la fecha óptima de cosecha y comercialización del producto.

2 Estado del arte de la aplicación de Gemelos digitales en las explotaciones del sector primario.

2.1 Granjas inteligentes

El uso de agua dulce en la agricultura es responsable del 70% de la cantidad total de agua dulce utilizada en el mundo [6]. Con lo que el desarrollo de tecnologías como el Internet de las cosas (IoT) podría ser la clave para disminuir la cantidad de agua utilizada en la agricultura, así como para aumentar la cantidad y calidad de los alimentos producidos en las granjas. Este sistema se puede implementar en granjas para reconocer adecuadamente su medioambiente. Esto significa que una representación virtual de una granja no debería poder sólo recopilar información sobre los sistemas de riego de la finca y demás sistemas que tuviese, sino que debe atender también a las condiciones del medioambiente, humedad, temperatura, estado del terreno, estado de las plantas del cultivo, etc. Y además debe actuar en base al análisis y las decisiones tomadas por el agricultor y/o la inteligencia artificial del gemelo digital.

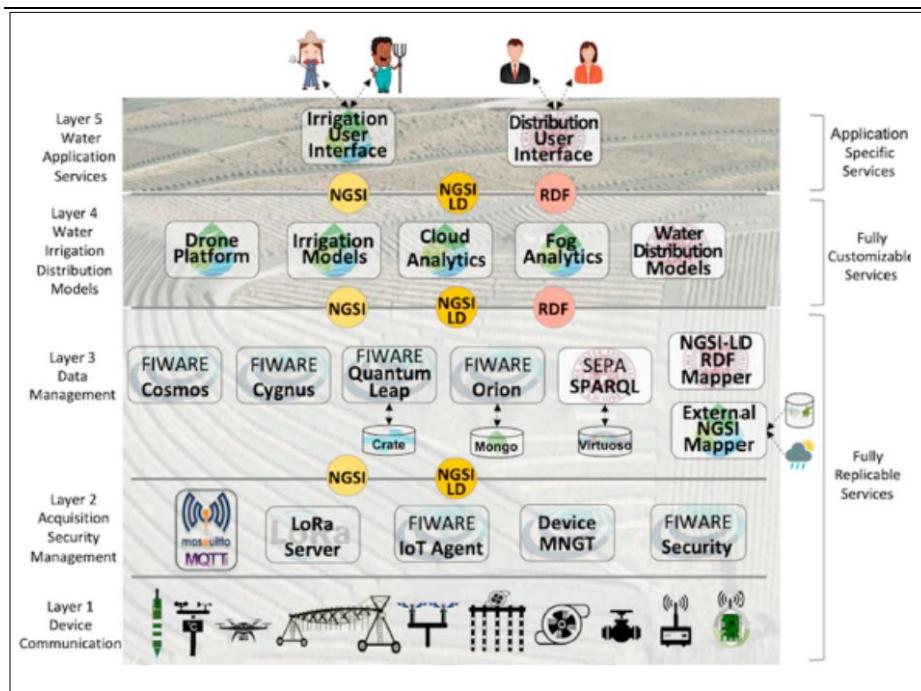


Ilustración 3 Capas de arquitectura del proyecto SWAMP

2.1.1 Ejemplo de granja inteligente: Proyecto SWAMP

El **proyecto SWAMP** implementa una **plataforma de gestión inteligente del agua basada en IoT** (Internet de las Cosas) para optimizar el **riego de precisión**. Esta iniciativa se está desplegando actualmente en Brasil, Italia y España [7], demostrando su adaptabilidad a diversos entornos, climas, tipos de suelo y cultivos gracias a su diseño configurable.

La plataforma SWAMP permite una **toma de decisiones flexible**: el agricultor puede gestionar completamente el proceso, optar por un **sistema de automatización basado en inteligencia artificial (IA)**, o emplear una combinación de ambos enfoques. En el campo, la infraestructura incluye una variedad de **dispositivos y sistemas conectados**, como sondas de suelo, estaciones meteorológicas, sistemas de riego,

sembradoras y cosechadoras. Todos estos equipos transmiten datos a la nube a través de un **"Gateway"** (**pasarela entre redes diferentes**), que actúa como intermediario enviando la información a un agente IoT para su procesamiento y análisis.

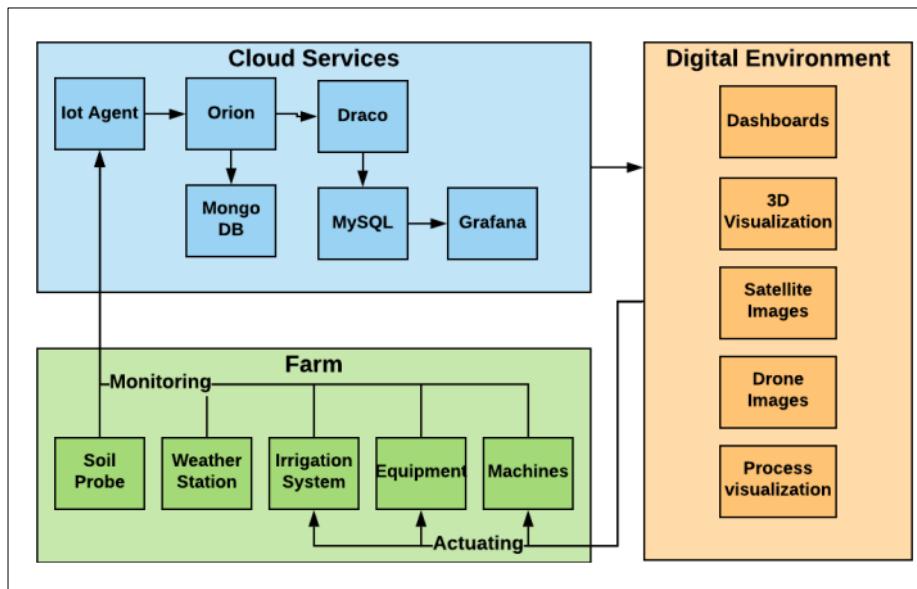


Ilustración 4 Arquitectura y servicios usados

La arquitectura del sistema propuesto por el proyecto SWAMP y los servicios utilizados en esta arquitectura se describen como:

Comentado [MLG6]: Propuesto por...

- Agente IoT [12]: Un servicio que traduce múltiples protocolos de comunicación IoT al utilizado dentro de la nube.
 - Orion [8]: Es un agente de contexto que le permite gestionar todo el ciclo de vida de la información de contexto, incluido actualizaciones, consultas, registros y suscripciones.
 - Mongo DB [9]: Es una base de datos para documentos que se utiliza para almacenar la última información actualizada en una estructura de datos

compuesta por pares de campos y valores. Los documentos de MongoDB son similares a los objetos JSON.

- Draco [10]: Es un habilitador genérico y es un mecanismo alternativo de persistencia de datos para gestionar el historial de citas contextuales. Draco es un conector encargado de conservar las fuentes de datos contextuales en otras bases de datos y sistemas de almacenamiento de terceros, creando una vista histórica del contexto. Internamente, Draco está basado en Apache NiFi. NiFi que es un marco popular para la gestión y el procesamiento de datos de múltiples fuentes. Draco desempeña el papel de conector entre Orion Context Broker (que es una fuente de datos NGSI [11]) y una amplia gama de sistemas externos como MySQL, MongoDB, etc. Se puede utilizar Draco si se necesita procesar y conservar datos de contexto para poder mantener un registro histórico. Draco también se puede utilizar para filtrar y volver a publicar datos de contexto en Orion.
- MySQL [13]: Es un sistema de gestión de bases de datos relacionales de código abierto que se puede utilizar para almacenar datos de series temporales. Aunque hay otras bases de datos específicas para series temporales como: TimescaleDB y InfluxDB.
- Grafana [14]: Es una solución de código abierto para el análisis y el monitoreo con el que se pueden crear paneles gráficos para visualizar datos.

El entorno digital (“Digital Environment”) está diseñado para informar, de forma visual, la información recopilada mediante IoT, así como para enviar información a los sistemas en base a la toma de decisiones del proceso realizado dentro de este entorno digital. En este momento sólo es posible visualizar la información en paneles de control de mando. Sin embargo, para desarrollar plenamente la granja digital inteligente, todos los demás entornos que se requieren deben desarrollarse de una manera integrada con los datos recopilados y analizados en él. Se debe conectar el entorno digital de la nube tal como se visualiza con en el sistema físico directamente (actuadores) si es posible o conectándose a través de Controladores Lógicos Programables (PLC) en el sistema de riego, sistemas, equipos y máquinas.

2.2 Acuicultura

Durante siglos, los recursos acuáticos han formado parte de la dieta humana. Esta tendencia ha ido aumentando con el paso del tiempo. De hecho, en la actualidad se consume una cantidad considerable de pescado, pues en muchos lugares es considerado un alimento básico. Se prevé que la mayor parte de la producción de pescado se consumirá como alimento (181 Mt en 2030) y solo 10% se destinará a usos no alimentarios (principalmente la harina y el aceite de pescado) [18]. Cerca de 72% del pescado para alimentación se consumirá en los países asiáticos. En 2030, se espera que la acuicultura proporcione 57% del pescado destinado al consumo humano, en comparación con 53% en el periodo base. Se prevé que el consumo mundial de pescado comestible se incrementará 1,3% anual, una bajada considerable en relación con la tasa de crecimiento de 2,3% anual registrada durante la década anterior. Esta disminución refleja la desaceleración de la demanda causada por ingresos más bajos a principios de la década, menor crecimiento demográfico y menores precios mundiales de la carne, en particular de la carne de aves de corral. Se prevé que el consumo mundial aparente de pescado para consumo humano sumará 21,2 kg per cápita en 2030, por encima de los 20,5kg per cápita registrados en el periodo base. El consumo de pescado per cápita aumentará en Asia, Europa y en América, en tanto que permanecerá estable en Oceanía y disminuirá en África, el continente con el crecimiento demográfico más rápido, el cual superará al crecimiento de su suministro de pescado para alimentación.

Aunque los términos **piscifactoría** y **granja acuática** suelen emplearse de forma intercambiable, es crucial diferenciarlos, ya que cada uno describe un tipo específico de instalación dentro del ámbito de la **acuicultura**. Además, la ubicación de estas instalaciones ya sea en tierra o en el mar, influye en la terminología y las técnicas de cultivo empleadas.

Una **piscifactoría** es una instalación industrial dedicada exclusivamente a la cría de **peces**. Su objetivo principal es la producción de diversas especies piscícolas para distintos fines. Entre las variedades más comunes cultivadas se encuentran salmones, bagres, tilapias, carpas, truchas y esturiones. Si bien la mayor parte de esta producción

se destina al **sector alimenticio**, una porción menor puede utilizarse para fines **ornamentales** o para la **repopulación de ecosistemas acuáticos** (lagos y ríos), fomentando así la pesca deportiva [17].

En contraste, el término **granja acuática** es mucho más amplio, refiriéndose a cualquier instalación o sistema donde se practica la **acuicultura**, es decir, la cría de **organismos acuáticos en general**. Esto incluye no solo peces (como en las piscifactorías), sino también una diversidad de **moluscos** (como ostras, mejillones o almejas), **crustáceos** (como gambas, langostinos o cangrejos), e incluso **plantas acuáticas** (como algas). En esencia, una piscifactoría es un tipo particular de granja acuática, enfocada únicamente en los peces, mientras que una granja acuática abarca una gama mucho más diversa de especies cultivadas en un entorno acuático controlado.

La ubicación de estas granjas determina también las tecnologías y los nombres específicos asociados a ellas:

- **Instalaciones en Tierra (Acuicultura Continental/Terrestre):** Cuando las granjas acuáticas se encuentran en tierra firme, suelen utilizar **agua dulce**, o en ocasiones, agua salobre o marina que es bombeada y tratada. Para estas instalaciones, los términos comunes incluyen **piscifactorías terrestres** (si son solo de peces) o **granjas de agua dulce** (para un espectro más amplio de organismos). Aquí se emplean frecuentemente **estanques de cultivo**, o sistemas más avanzados como los **Sistemas de Recirculación en Acuicultura (RAS)**, que recirculan y filtran el agua, minimizando el consumo y el impacto ambiental. También se utilizan **cultivos en tanques o canales** para producciones intensivas.
- **Instalaciones en el Mar (Acuicultura Marina/Maricultura):** Cuando las granjas se sitúan directamente en el mar o en áreas costeras influenciadas por el agua salada, se les denomina **piscifactorías marinas** (para peces) o **granjas marinas** (término más amplio para cualquier organismo marino). Los métodos más comunes en este entorno son las **jaulas marinas flotantes** para peces, y

los **cultivos en bateas** o “*long-lines*”, estructuras utilizadas principalmente para el cultivo de moluscos como mejillones u ostras. También existen los **viveros marinos** para la cría de moluscos en zonas costeras.

En resumen, la terminología específica (por ejemplo, “terrestre”, “marina”, “RAS” o “jaulas marinas”) se añade para indicar el entorno y el método de cultivo, aunque el concepto fundamental de “acuicultura” engloba todas estas modalidades.

2.2.1 Ejemplo de piscifactoría terrestre operativa en Italia

La contraparte virtual y digital de un objeto físico, denominada gemelo digital, deriva del Internet de las cosas (IoT) e involucra la adquisición en tiempo real y el procesamiento de grandes conjuntos de datos. Un gemelo digital completamente desarrollado permite la gestión remota y en tiempo real, así como la reproducción de escenarios reales y simulados como ya se ha comentado anteriormente. En el marco emergente de la acuicultura de precisión, que aporta principios de ingeniería de control a la producción pesquera, un prototipo de gemelo digital para granjas de peces en tierra firme; tiene como objetivo: Apoyar en la toma de decisiones a los productores, en la optimización de las prácticas de alimentación y del suministro de oxígeno, y el manejo de la población de peces. Para ello se necesita manejar las siguientes actuaciones: El control sobre el crecimiento de los peces, la salud de los peces y el control ambiental. Este control se basa en modelos matemáticos integrados que se alimentan con datos obtenidos, *in situ*, de sensores y de fuentes externas, y permite simular varias dinámicas de procesos, permitiendo la estimación de parámetros clave que describen el ambiente y los peces. Un ejemplo de aplicación conceptual es la dirigida a ciclos de cría de trucha arcoíris (“*Oncorhynchus mykiss*”) en una granja acuática (piscifactoría) terrestre operativa en Italia [15], la granja está equipada con grandes piscinas que se abastecen de agua dulce procedente del vecino río Sarca. Debido a las características morfológicas de los peces, no es posible encerrarlos en espacios muy pequeños, principalmente porque necesitan nadar para conseguir oxígeno mediante la filtración del agua que los rodea. Si no se mantienen en movimiento, no fluye el agua a través de sus branquias, y como resultado, no obtendrían el oxígeno necesario. Por este motivo, normalmente las piscifactorías

cuentan con piscinas muy grandes que les permiten, hasta cierto grado, libre movilidad a los peces. El gemelo digital tiene en cuenta los distintos niveles de automatización y control que se encuentran dentro de esta granja.



Ilustración 5 Tanques de canalización, el tanque de oxígeno líquido (tanque blanco a la izquierda) y el pórtico (que contiene la pasarela móvil que cruza las pistas de rodadura).

El gemelo digital se ha diseñado en base un flujo de información distribuido a lo largo de las cuatro fases de gestión de la granja, es decir, observar, interpretar, decidir y actuar. Cada una de estas cuatro fases en el contexto del gemelo digital es descrito en las siguientes subsecciones:

2.2.1.1 La fase de observación.

El objetivo esencial de la fase de observación es extraer información cuantitativa y cualitativa sobre los signos vitales de los peces, que se lleva a cabo mediante observaciones directas y herramientas de adquisición de datos. La fase de observación, junto con la fase de actuación, está directamente ligada al objeto físico de forma que, en un gemelo digital con un grado de madurez de control totalmente autónomo, los vínculos entre los objetos físicos con sus datos de información están desacoplados de las operaciones manuales. En el concepto actual esto sólo se realiza parcialmente, ya que se prevé entradas manuales de datos.

2.2.1.2 La fase de interpretación.

El modelo de granja simula la evolución de la biomasa de peces y de un conjunto de variables de estado relacionadas con el metabolismo de los peces. La solución numérica se obtiene acoplando un módulo que simula la evolución de la biomasa de peces y un módulo que predice la de un conjunto de variables de calidad del agua afectadas por el metabolismo de los peces.

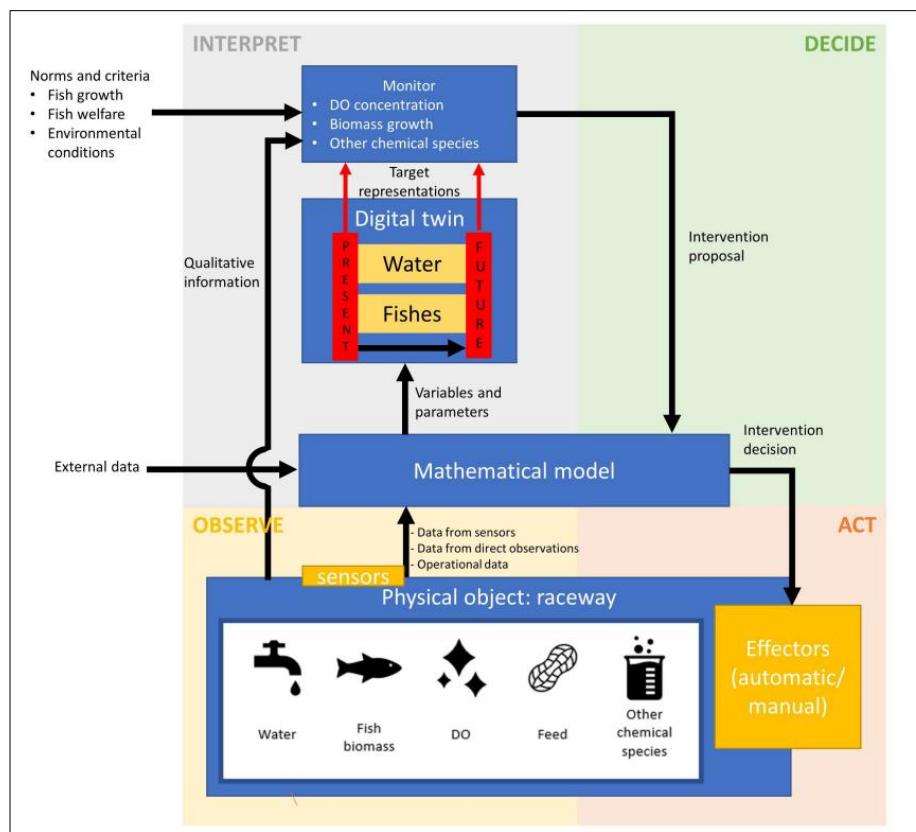


Ilustración 6 Las cuatro fases del funcionamiento del gemelo digital.

El modelo bioenergético simula el crecimiento de la trucha arco iris según la temperatura del agua, la calidad del agua y el régimen de alimentación, tanto a nivel individual como

a nivel de población, según lo deseé el usuario. El efecto de la alimentación en el metabolismo de los peces tiene en cuenta la cantidad real de alimento proporcionado por el piscicultor, y su composición bioquímica en términos de contenido de proteínas crudas, lípidos crudos y carbohidratos. La temperatura del agua afecta tanto el metabolismo como a la ingestión de alimento de los peces.

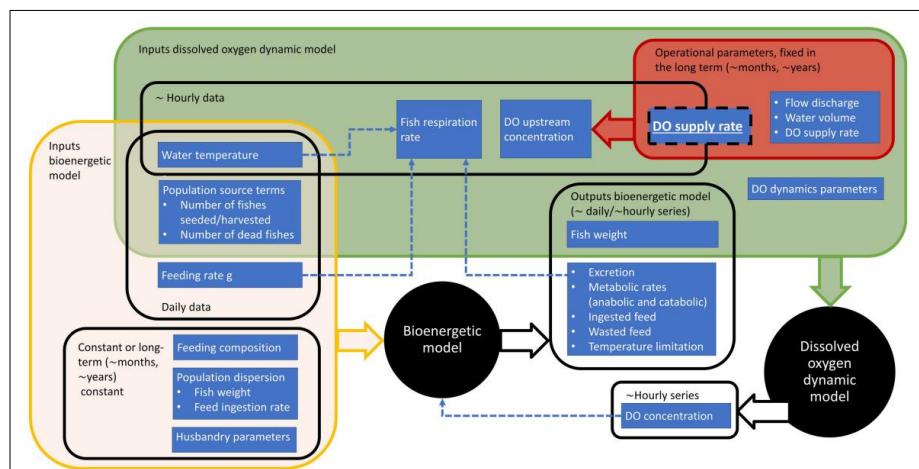


Ilustración 7 Esquema de entradas y salidas a los modelos dinámicos bioenergético y DO (oxígeno disuelto)

La serie temporal del peso de la biomasa de peces calculada a partir del modelo bioenergético [19] se ingresa en el modelo dinámico DO (Concentración de oxígeno disuelto) con el objetivo de predecirlo en las piscinas de la piscifactoría.

2.2.1.3 La fase de decisión.

A partir de las propuestas de intervención de la fase de interpretación anterior, se construyen estados futuros alternativos (Escenarios) del gemelo digital utilizando el modelo matemático. En los que se proponen distintas estrategias de cosecha (captura) de los peces en función de distintas cantidades de peces y del momento en que se realiza. Y como influye en la serie temporal de peso de la biomasa y demás variables (oxígeno disuelto, etc.).

2.2.1.4 La fase de actuación

Actualmente, en la piscifactoría piloto, los controles de suministro de alimento y oxígeno son completamente manuales o activados mediante controles manuales. Esta condición asociada con el apoyo a las decisiones de las herramientas proporcionadas a partir del modelo matemático configuraría un gemelo digital de nivel prescriptivo. Para evolucionar hacia el nivel autónomo, donde los operadores están desacoplados de los controles físicos, el desarrollo preliminar más plausible en la piscifactoría piloto sería la instalación de válvulas automáticas para controlar el suministro de oxígeno líquido. Para ello, y con pronósticos confiables de la tasa de respiración de los peces y la concentración de DO dentro las piscinas, el sistema de control debe convertirse del tipo de bucle abierto actual a bucle cerrado. En el caso de bucle abierto, el DO real en la piscina no influye en la decisión del oxígeno a suministrar, es decir, el suministro de oxígeno es, en principio, independiente del DO observado en la piscina. Por otro lado, para un sistema de bucle cerrado, la válvula se ajusta en función del error entre el DO real y el DO deseado. Por lo tanto, el DO real en la piscina tiene retroalimentación sobre el sistema de control.

3 Análisis de requisitos

Se especifican los requisitos funcionales y no funcionales para la construcción de un prototipo de gemelo digital de nivel 2, predictivo y desconectado del proceso productivo. Quedará fuera del alcance el nivel 3 de validación de datos predictivos contra los datos del proceso. El enfoque principal es apoyar la toma de decisiones para la organización de la planta de acuicultura, específicamente indicando la fecha óptima de recogida de la biomasa.

3.1 Requisitos funcionales

RF1 Gestión de Jaulas Marinas.

- RF1.1 Configuración de balsas: Gestión completa de parámetros (ubicación, fechas, número de peces).
 - **Historia de Usuario:** Como gestor de las jaulas marinas, quiero poder configurar nuevas balsas y modificar los parámetros de las existentes (como ubicación, fechas de inicio y fin de ciclo, y el número inicial de peces) para mantener la información actualizada y precisa en el sistema.
- RF1.2 Persistencia de datos: Almacenamiento y recuperación de configuraciones de balsas.
 - RF1.2.1 Persistencia de datos de temperatura y previsión de temperatura.
 - **Historia de Usuario:** Como gestor de las jaulas marinas, necesito que el sistema guarde automáticamente la configuración de las balsas y me permita recuperarlas en cualquier momento para no perder información y asegurar la continuidad de la planificación.
 - RF1.2.2 Persistencia de datos de precios y previsión de precios.
 - **Historia de Usuario:** Como gestor de las jaulas marinas, necesito que el sistema guarde los datos históricos de precios y las previsiones de precios del salmón para poder analizar tendencias y

optimizar el momento de la cosecha basándome en el valor de mercado.

- RF1.3 Selección de balsa activa: permite elegir y visualizar datos de balsas específicas.
 - **Historia de Usuario:** Como gestor de las jaulas marinas, quiero seleccionar una balsa específica para visualizar sus datos detallados y predicciones asociadas, con el fin de enfocar mi análisis en una unidad productiva particular.

RF2 Modelos predictivos sobre series temporales.

- RF2.1 Temperatura del mar.
 - RF2.1.1 Gestión del modelo predictivo de temperatura.
 - **Historia de Usuario:** Como gestor de las jaulas marinas, quiero que el sistema utilice un modelo predictivo para estimar la temperatura futura del agua del mar, ya que este factor influye directamente en el crecimiento de la biomasa.
- RF2.2 Precio de venta del producto.
 - RF2.2.1 Gestión del modelo predictivo de precios.
 - **Historia de Usuario:** Como gestor de las jaulas marinas, necesito un modelo predictivo para estimar los precios futuros de venta del salmón, lo que me permitirá identificar los momentos más rentables para la cosecha.
 - RF2.2.2 Optimizador del modelo predictivo de precios.
 - **Historia de Usuario:** Como gestor de las jaulas marinas, quiero que el sistema busque automáticamente las configuraciones del modelo predictivo de precios para asegurar que las previsiones sean lo más precisas posible.
 - RF2.2.3 Gestión y persistencia de los mejores parámetros del modelo.
 - **Historia de Usuario:** Como gestor de las jaulas marinas, quiero que el sistema guarde las configuraciones óptimas del modelo de

precios para reutilizarlas en futuras predicciones, evitando la necesidad de re-optimizar cada vez.

- RF2.3 Crecimiento de biomasa.
 - RF2.3.1 Gestión del modelo predictivo de crecimiento.
 - **Historia de Usuario:** Como gestor de las jaulas marinas, quiero que el sistema simule el crecimiento de la biomasa en las balsas utilizando el modelo de Thyholdt que estamos calculando manualmente, considerando factores como la temperatura del agua, para estimar la cantidad de producto disponible en el futuro.

RF3 Cuadro de mandos que muestre los datos al usuario.

RF3.1 Representación de datos.

- RF3.1.1 Representación de datos de temperatura.
 - **Historia de Usuario:** Como gestor de las jaulas marinas, quiero visualizar gráficos claros de las temperaturas históricas y pronosticadas para cada balsa, lo que me ayudará a entender el impacto de este factor en el ciclo productivo.
- RF3.1.2 Representación de datos de precios.
 - **Historia de Usuario:** Como gestor de las jaulas marinas, deseo ver la evolución histórica y previsiones futuras de los precios de venta del salmón en un gráfico, para tomar decisiones informadas sobre el momento de la venta.
- RF3.1.3 Representación de crecimiento de biomasa.
 - **Historia de Usuario:** Como gestor de las jaulas marinas, quiero observar el crecimiento proyectado de la biomasa en un gráfico, para saber cuándo se alcanzará el peso óptimo de los peces.

RF3.2 Detalle de datos de la balsa actual.

Historia de Usuario: Como gestor de las jaulas marinas, necesito acceder a una vista detallada de los datos actuales de la balsa seleccionada, incluyéndola región del mar y su ubicación, profundidad, temperatura media, fecha de inicio y de fin, número inicial de peces, para tener una visión completa de su estado.

RF3.3 Representación de la balsa actual mostrando la biomasa.

Historia de Usuario: Como gestor de las jaulas marinas, quiero una representación visual de la balsa que muestre la biomasa estimada de forma intuitiva, para comprender el estado actual de los peces.

RF3.4 Listado de balsas con detalle de valores óptimos de cosecha.

Historia de Usuario: Como gestor de las jaulas marinas, quiero ver un listado de todas mis balsas, junto con la fecha y los valores óptimos de cosecha, para planificar la recogida y venta de la biomasa de manera eficiente.

RF3.5 Control de tiempo actual y futuro.

Historia de Usuario: Como gestor de las jaulas marinas, quiero poder desplazarme a través del tiempo, tanto al pasado como al futuro, para visualizar cómo los cambios en las variables (temperatura, precios, crecimiento) afectan las proyecciones y escenarios de cosecha.

RF3.6 Importación de datos.

- RF3.6.1 Importación de datos de temperatura.
 - **Historia de Usuario:** Como gestor de las jaulas marinas, quiero importar datos de temperatura desde archivos externos (CSV) para alimentar los modelos predictivos y asegurar que las previsiones se basen en la información más reciente.
- RF3.6.2 Importación de datos de precios.
 - **Historia de Usuario:** Como gestor de las jaulas marinas, quiero importar datos de precios de mercado desde archivos externos, lo que me permitirá actualizar el modelo predictivo de precios y mejorar la precisión de las estimaciones.

RF3.7 Gestión de predicciones.

- RF3.7.1 Gestión de predicción de temperatura.
 - **Historia de Usuario:** Como gestor de las jaulas marinas, quiero poder iniciar y gestionar las predicciones de temperatura para una balsa específica, y ver los resultados en el cuadro de mandos.
- RF3.7.2 Gestión de predicción de precio.
 - **Historia de Usuario:** Como gestor de las jaulas marinas, quiero poder ejecutar y actualizar las predicciones de precios para una balsa, de modo que pueda tomar decisiones de venta oportunas.
 - RF3.7.2.1 Gestión de optimización de precios.
 - **Historia de Usuario:** Como gestor de las jaulas marinas, quiero controlar el proceso de optimización de los parámetros del modelo de precios y visualizar su progreso en tiempo real.
 - RF3.7.2.2 Elección de parámetros de modelo predictivo.
 - **Historia de Usuario:** Como gestor de las jaulas marinas, quiero poder seleccionar los parámetros del modelo predictivo (por ejemplo, los mejores encontrados por el

optimizador) para aplicar la configuración que me ofrezca las predicciones más fiables.

- o RF3.7.3 Gestión de predicción de biomasa.
 - **Historia de Usuario:** Como gestor de las jaulas marinas, quiero activar la predicción del crecimiento de la biomasa para una balsa y ver cómo se proyecta el peso y número de peces a lo largo del tiempo.

Requisitos funcionales fuera de alcance de este TFM (Oportunidades)

Alertas inteligentes: Notificaciones automáticas

Las alertas inteligentes son importantes para la operación eficiente de una planta acuícola, en cuanto a la automatización de tareas manuales, para implementar este sistema se necesita desarrollar algoritmos que puedan monitorear continuamente los datos en tiempo real y modelos predictivos. Estos algoritmos deben ser capaces de detectar situaciones críticas, anomalías y oportunidades que requieran atención inmediata. La clave aquí es la precisión y la rapidez en la detección, así como la capacidad de enviar notificaciones automáticas al operador de la planta o en su caso al sistema de control automático de la planta. Esto puede involucrar el uso de técnicas de "machine learning" y análisis de datos en tiempo real.

Historia de Usuario: Como gestor de las jaulas marinas, quiero recibir notificaciones automáticas (alertas inteligentes) cuando se detecten situaciones críticas (ej. desviación significativa en el crecimiento de biomasa) u oportunidades (ej. pico de precio de mercado), para poder actuar de forma proactiva.

3.2 Requisitos no funcionales

RNF 1. Ejecución multiplataforma.

La ejecución multiplataforma es esencial para garantizar que el software pueda funcionar en diferentes sistemas operativos sin necesidad de modificaciones significativas. Esto implica diseñar soluciones que sean independientes del sistema operativo, utilizando tecnologías como contenedores (Docker), "frameworks" multiplataforma (como Qt o Xamarin), y lenguajes de programación que sean compatibles con múltiples plataformas (como Java o Python). La clave aquí es la portabilidad y la flexibilidad del software.

Historia de Usuario: Como administrador de sistemas, quiero que la aplicación pueda ejecutarse en diferentes sistemas operativos (Windows, macOS, Linux) sin problemas de compatibilidad, para que sea flexible a los cambios y evolucione las políticas sobre sistema operativos de la empresa.

RNF 2. Uso de librerías de ciencia de datos estándar.

El uso de librerías de ciencia de datos estándar asegura que el software pueda aprovechar las herramientas y técnicas más avanzadas en el campo de la ciencia de datos. Esto incluye librerías como NumPy, Pandas, StatsForecast, dentro del ecosistema de Python. Estas librerías proporcionan funcionalidades robustas para el análisis de datos, aprendizaje automático, y procesamiento de datos en general. Además, el uso de librerías estándar facilita la colaboración y la integración con otros sistemas y proyectos. Otras alternativas serían el lenguaje R y su ecosistema. Como motor de análisis que permite grandes volúmenes de datos, se podría usar Apache Spark, aunque esto último podría plantearse en futuras ampliaciones.

Historia de Usuario: Como desarrollador de software de la empresa, quiero que el sistema utilice librerías de ciencia de datos estándar (como NumPy, Pandas, StatsForecast) para asegurar la fiabilidad, mantenibilidad y la posibilidad de futuras extensiones del código.

RNF 3. Uso de software libre.

El uso de software libre ofrece varias ventajas, incluyendo la reducción de costos, la flexibilidad para modificar y adaptar el software según las necesidades específicas del proyecto, y la posibilidad de aprovechar una comunidad activa de desarrolladores para soporte y mejoras continuas. Esto implica seleccionar herramientas y plataformas que sean de código abierto y que tengan una licencia compatible con los objetivos del proyecto. Ejemplos de software libre incluyen sistemas operativos como Linux, bases de datos como PostgreSQL, MySQL, SQLite para aplicaciones pequeñas. Bases de datos como TimescaleDB e InfluxDB especializadas en series temporales y herramientas de desarrollo integrado (IDE) como Eclipse, Visual Studio Code o Spyder que facilitan la programación y depuración de aplicaciones escritas en Python.

En cuanto a repositorios de código:

1. **GitHub:** Es uno de los repositorios de código más conocidos y utilizados. Ofrece una amplia gama de herramientas para la colaboración, el control de versiones y la integración continua. GitHub es gratuito para proyectos públicos y también ofrece planes de pago para proyectos privados con características adicionales.
2. **GitLab:** Similar a GitHub, GitLab es una plataforma de DevOps completa que proporciona control de versiones, integración y entrega continuas (CI/CD). GitLab ofrece una versión gratuita con muchas características útiles, así como planes de pago para funcionalidades avanzadas.
3. **Bitbucket:** Es otra plataforma popular para alojar repositorios de código. Bitbucket es gratuito para equipos pequeños y proyectos públicos, y ofrece planes de pago para equipos más grandes y proyectos privados.
4. **SourceForge:** Es una plataforma de alojamiento de código que ha existido durante mucho tiempo. SourceForge es gratuito y es conocido por alojar una gran cantidad de proyectos de código abierto.

Historia de Usuario: Como gestor de proyecto de la empresa, quiero que el desarrollo del prototipo se base en software libre para reducir costos de licencia, garantizar la flexibilidad de modificación y aprovechar el soporte de la comunidad.

Requisitos No Funcionales fuera del alcance de este TFM (Oportunidades)

Conectividad (Futura)

Sensores IoT para Integración con datos en tiempo real: La integración de sensores IoT es fundamental para obtener datos precisos y en tiempo real. Esto requiere el desarrollo de una infraestructura robusta que permita la recolección, transmisión y procesamiento de datos de múltiples sensores distribuidos en la planta acuícola. La conectividad debe ser fiable y segura para garantizar la integridad de los datos.

Historia de Usuario: Como gestor de las jaulas marinas, me gustaría que el sistema se conectara a sensores IoT para obtener datos en tiempo real de las jaulas marinas (temperatura, oxígeno disuelto, etc.), mejorando la precisión de las predicciones y el control operativo.

Actuadores IoT para el mando de control en tiempo real: Los actuadores IoT permiten el control remoto y en tiempo real de diversos dispositivos en la planta. Esto implica la implementación de protocolos de comunicación eficientes y seguros que permitan enviar comandos a los actuadores desde un sistema centralizado. La latencia mínima y la fiabilidad son aspectos críticos que considerar.

Historia de Usuario: Como gestor de las jaulas marinas, desearía que el sistema pudiera enviar comandos a actuadores IoT para controlar procesos físicos (ej. alimentación, suministro de oxígeno), permitiendo una automatización y optimización en tiempo real.

APIs externas para acceso externo a datos meteorológicos y datos de precios de mercado: La integración de APIs externas para acceder a datos meteorológicos y de precios de mercado puede proporcionar información valiosa para la toma de decisiones. Esto requiere el desarrollo de interfaces de programación que puedan interactuar con estas APIs de manera eficiente y segura. Además, es importante manejar adecuadamente la autenticación y la autorización para proteger los datos.

Historia de Usuario: Como gestor de las jaulas marinas, me gustaría que el sistema se conectara a APIs externas para obtener datos meteorológicos y de precios de mercado de forma automática, enriqueciendo los modelos predictivos con información externa.

Bases de datos

Base de datos para persistencia empresarial e integración con ERP empresarial: La persistencia de datos es esencial para el análisis histórico y la toma de decisiones informadas. La integración con un sistema ERP empresarial permite una gestión más eficiente de los recursos y procesos de la planta. Esto implica el diseño y la implementación de una base de datos que pueda manejar grandes volúmenes de datos y proporcionar acceso rápido y seguro a la información almacenada.

Historia de Usuario: Como gestor de las jaulas marinas, quiero que el sistema se integre con nuestra base de datos empresarial y ERP para una gestión centralizada y eficiente de todos los datos productivos y financieros.

3.3 Roles de las Historias de Usuario

Basado en las historias de usuario proporcionadas en el punto tres análisis de requisitos, se identifican los siguientes roles principales:

- **Gestor de Jaulas Marinas:** Es el usuario principal del sistema, encargado de la planificación, el monitoreo y la toma de decisiones relacionadas con la producción de biomasa en las jaulas marinas. Este rol necesita acceso a configuraciones de balsas, datos históricos, predicciones y alertas.
- **Administrador de Sistemas:** Este rol se enfoca en la infraestructura técnica del software, asegurando su compatibilidad y funcionamiento en diferentes entornos.
- **Desarrollador de Software de la Empresa:** Este rol está interesado en la elección de tecnologías y librerías que garanticen la fiabilidad, mantenibilidad y extensibilidad del código.
- **Gestor de Proyecto de la Empresa:** Este rol se centra en la eficiencia del desarrollo, la reducción de costos y el aprovechamiento de recursos (como el software libre).

3.4 Casos de uso del TFM

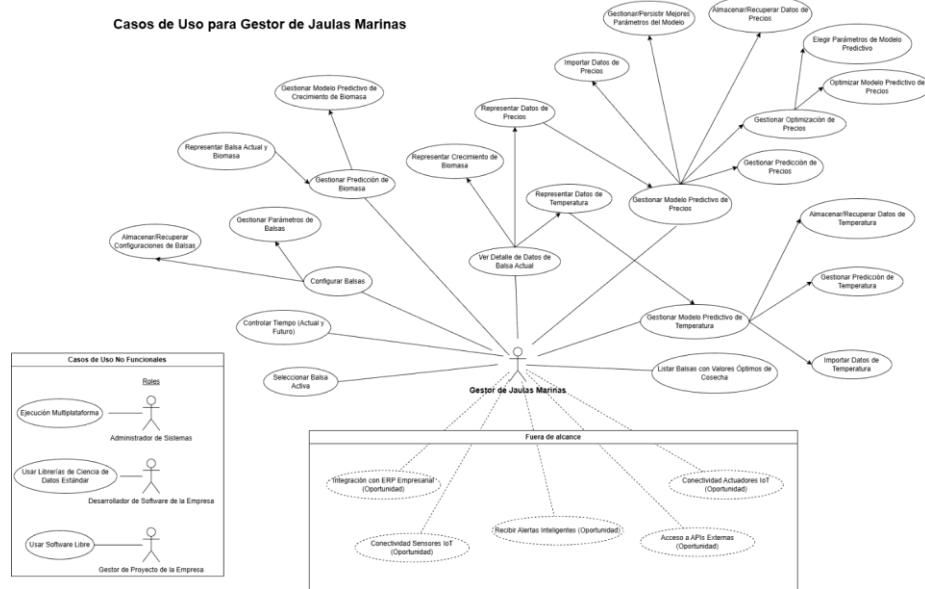


Ilustración 8 Diagrama de Casos de Uso

Casos de Uso Implementados (Línea Continua en el diagrama anterior):

El sistema permite al **Gestor de Jaulas Marinas** realizar las siguientes acciones:

- **Gestión de Balsas:** Configurar, gestionar parámetros, y almacenar/recuperar configuraciones de balsas, incluyendo datos de temperatura y precios. Además, puede seleccionar una balsa activa para visualizar sus datos.
- **Modelos Predictivos:** Gestionar los modelos predictivos de temperatura, precios (incluyendo optimización y persistencia de parámetros) y crecimiento de biomasa.
- **Cuadro de Mandos:** Representar visualmente datos de temperatura, precios y crecimiento de biomasa. También puede ver detalles de la balsa actual,

representar la biomasa visualmente, listar balsas con valores óptimos de cosecha, controlar el tiempo (pasado y futuro), importar datos (temperatura y precios), y gestionar las predicciones de temperatura, precios (incluyendo optimización y elección de parámetros) y biomasa.

- **Requisitos No Funcionales:** El sistema está diseñado para una **Ejecución Multiplataforma** (para el Administrador de Sistemas), utilizando **Librerías de Ciencia de Datos Estándar** (para el Desarrollador de Software) y **Software Libre** (para el Gestor de Proyecto).

Casos de Uso No Implementados / Oportunidades (Línea Discontinua):

Estas funcionalidades se consideran para futuras fases del proyecto y no forman parte del prototipo actual:

- **Alertas Inteligentes:** El sistema no enviará notificaciones automáticas ante situaciones críticas u oportunidades.
- **Conectividad IoT:** No habrá integración con sensores IoT para datos en tiempo real ni con actuadores IoT para control en tiempo real.
- **APIs Externas:** No se conectarán a APIs externas para obtener datos meteorológicos o de precios de mercado de forma automática.
- **Bases de Datos Empresariales:** No se integrará con bases de datos empresariales ni sistemas ERP.

En resumen, el prototipo se enfoca en la recolección de datos, la aplicación de modelos predictivos y la visualización de resultados a través de un cuadro de mandos, apoyando la toma de decisiones sin una conexión en tiempo real con los procesos físicos de la acuicultura.

4 Materiales y métodos

4.1 Series temporales

Las series temporales, también conocidas como series de tiempo, son secuencias de datos observados en intervalos regulares de tiempo y ordenados cronológicamente. Estos datos pueden ser anuales, mensuales, diarios, por horas, segundos, etc. y se representan comúnmente en gráficos para analizar la evolución temporal de los datos.

Se utilizan para:

1. Analizar tendencias, patrones y relaciones entre variables a lo largo del tiempo.
2. Predecir el comportamiento futuro de una variable basándose en sus valores pasados (históricos).
3. Identificar componentes como la tendencia (evolución a largo plazo), variación estacional (movimientos a corto plazo periódicos), variación cíclica (oscilaciones periódicas con amplitud superior a un año) y variación aleatoria (variaciones debidas a fenómenos puntuales).

En estadística y aprendizaje automático (“machine learning”), las series temporales son fundamentales para realizar pronósticos y tomar decisiones basadas en datos históricos.

Cuando hablamos de datos de series temporales, hay algunos conceptos clave que se deben enumerar:

- La Tendencia T_t . Existe una tendencia cuando vemos un aumento o disminución a largo plazo en los datos. La tendencia no necesita ser lineal; también puede ser exponencial. También puede haber un cambio de dirección. Primero, la tendencia puede ser creciente y, después de cierto punto, la tendencia puede ser decreciente.

Comentado [MLG7]: NO hay que introducir las series temporales en la introducción... y mucho menos las metodologías en la introducción.

Tendrás que tener un apartado más delante de las metodologías a aplicar y entre ellas tiene que estar todo esto de series temporales. Bien contado. Técnico

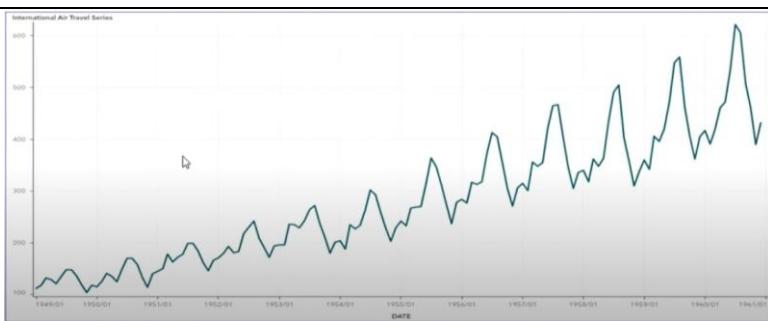


Ilustración 9 Serie de Viajes Aéreos Internacionales; los datos aumentan continuamente desde 1949 hasta 1961. La tendencia aumenta.

- La Estacionalidad E_t . Se determina porque existe un patrón periódico debido al calendario (por ejemplo, trimestral, mensual, diario). La estacionalidad puede ser aditiva o multiplicativa y con tendencia o sin ella.

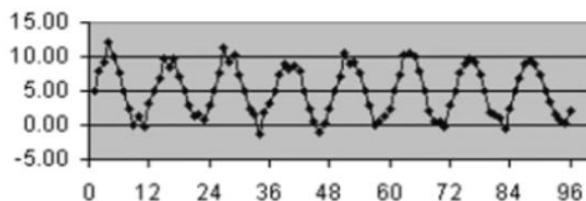


Ilustración 10 Estacionalidad aditiva sin tendencia. La altura entre picos y valles permanece constante.

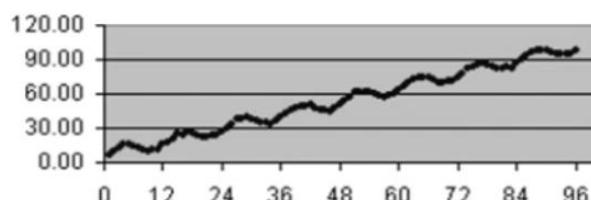


Ilustración 11 Estacionalidad aditiva con tendencia creciente. La altura entre picos y valles permanece constante siguiendo una tendencia lineal creciente.

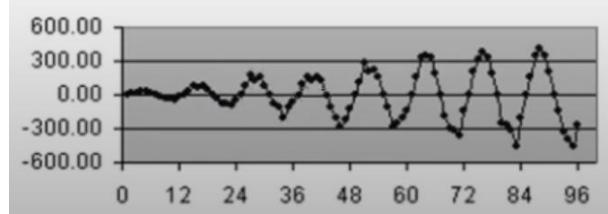


Ilustración 12 Estacionalidad Multiplicativa sin tendencia. La altura entre picos y valles aumenta siguiendo una constante multiplicativa. No se observa tendencia.

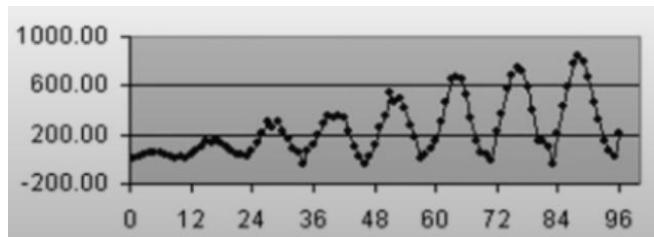


Ilustración 13 Estacionalidad Multiplicativa con tendencia. La altura entre picos y valles aumenta siguiendo una constante multiplicativa. Se observa una tendencia creciente.

- La Variación Cíclica C_t . Es el componente de la serie que recoge las oscilaciones periódicas que tienen una duración superior a un año.

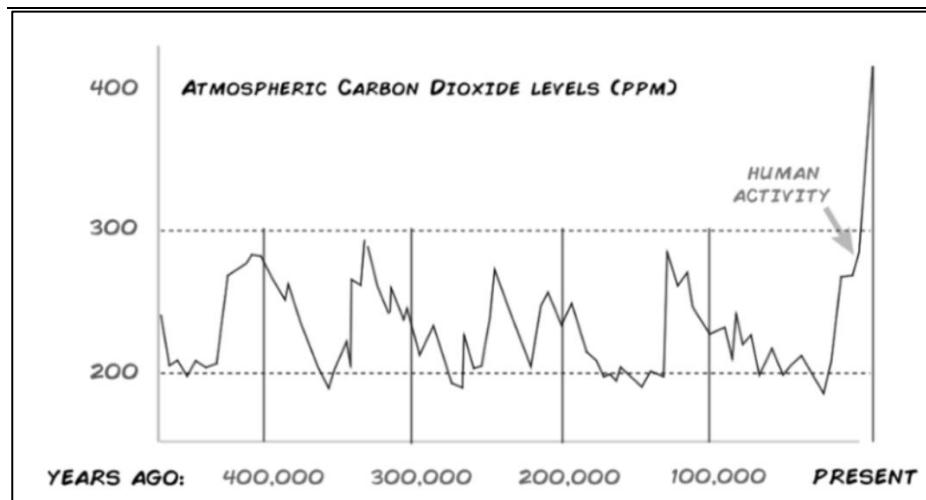


Ilustración 14 En el siguiente gráfico, puede ver los patrones que se repiten en cada 100.000 años.

- Variación Irregular o aleatoria (Ruido o error) I_t . Se debe a variaciones como fenómenos puntuales o aleatorios como tormentas, inundaciones, huelgas, guerras, avances tecnológicos, errores de medida, etc. Difícilmente predecibles.

Las series de tiempo se pueden clasificar atendiendo a los diferentes modelos que se construyen para explicar sus valores observados [20]:

1. Serie de tiempo aditiva: Se suman la tendencia, la estacionalidad, la variación cíclica y la variación irregular para obtener el modelo.

$$x_t = T_t + E_t + C_t + I_t$$

2. Serie de tiempo multiplicativa: Se multiplican la tendencia, la estacionalidad, la variación cíclica y la variación irregular para obtener el modelo.

$$x_t = T_t \cdot E_t \cdot C_t \cdot I_t$$

3. Serie de tiempo mixta: Se combinan los componentes de las series temporales con sumas y multiplicaciones para obtener el modelo.

$$x_t = T_t + E_t \cdot C_t \cdot I_t$$

$$x_t = T_t + E_t \cdot C_t + I_t$$

$$x_t = T_t \cdot E_t \cdot C_t + I_t$$

4.2 Análisis de series temporales.

El análisis de series temporales es una técnica estadística que estudia datos recopilados en intervalos de tiempo regulares para identificar patrones, tendencias, estacionalidades y fluctuaciones. Su objetivo principal es comprender el comportamiento de una variable a lo largo del tiempo y hacer predicciones futuras. Se utiliza en campos como economía, meteorología, finanzas, salud, y más.

Un resumen de los modelos más utilizados podría ser:

- **ARIMA (AutoRegressive Integrated Moving Average):** Modelo clásico que combina auto regresión, diferencias (para hacer la serie estacionaria) y medias móviles. Útil para datos lineales sin mucha estacionalidad.
- **SARIMA (Seasonal ARIMA):** Ampliación de ARIMA que incorpora componentes estacionales, útil si la serie muestra patrones que se repiten en ciclos regulares (por ejemplo, ventas mensuales).
- **Prophet:** Desarrollado por Facebook, está diseñado para manejar datos con estacionalidades fuertes, efectos por días festivos y cambios abruptos de tendencia.
- **Modelos de Gradient Boosting:** Son algoritmos de aprendizaje automático (“machine learning”), que construyen modelos combinando árboles de decisión, de forma secuencial, corrigiendo los errores del anterior. Por ejemplo:
 - **XGBoost:** Veloz y preciso, útil para grandes volúmenes de datos.
 - **LightGBM:** Optimizado para rendimiento, puede manejar grandes conjuntos de datos con bajo consumo de memoria.
- **Redes Neuronales Recurrentes (RNN, LSTM):** Modelos de aprendizaje profundo que capturan relaciones a largo plazo en series temporales, especialmente útiles para datos complejos y no lineales.

4.3 Modelo Prophet

Prophet es una herramienta de código abierto, disponible en los lenguajes Python y R, diseñada para abordar los desafíos de generar pronósticos de alta calidad. Su desarrollo busca solucionar dos problemas clave que se presentan en la práctica de la previsión empresarial. En primer lugar, las técnicas de previsión completamente automáticas suelen ser difíciles de ajustar y carecen de la flexibilidad necesaria para incorporar supuestos o heurísticas útiles. En segundo lugar, los analistas encargados de la ciencia de datos en una organización a menudo poseen una vasta experiencia en el dominio específico de los productos o servicios que apoyan, pero pueden carecer de formación especializada en la previsión de series de tiempo. [21].

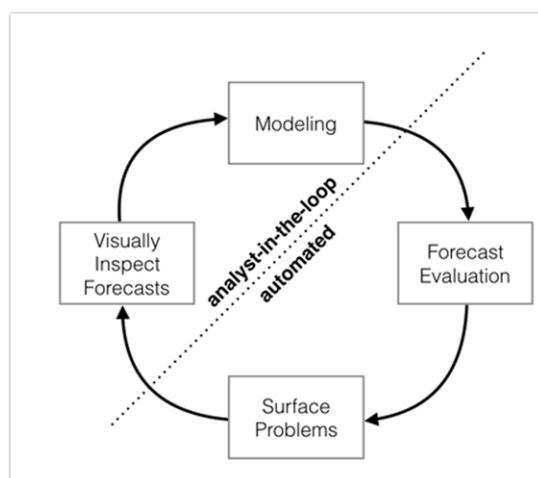


Ilustración 15 Proceso de refinamiento de la previsión

La figura anterior esquematiza el proceso iterativo de refinamiento de la previsión empresarial. Este proceso comienza con la modelización de series temporales mediante una especificación flexible, lo que facilita la interpretación humana de cada parámetro. Posteriormente, se generan pronósticos basados en este modelo, comparándolos con un conjunto de valores de referencia obtenidos de fechas históricas simuladas para evaluar su rendimiento. Si se identifica un rendimiento insatisfactorio o si otros

elementos del pronóstico requieren intervención manual, los problemas se comunican al analista. El analista puede entonces examinar el pronóstico y decidir si es necesario ajustar el modelo en función de esta retroalimentación. [21].

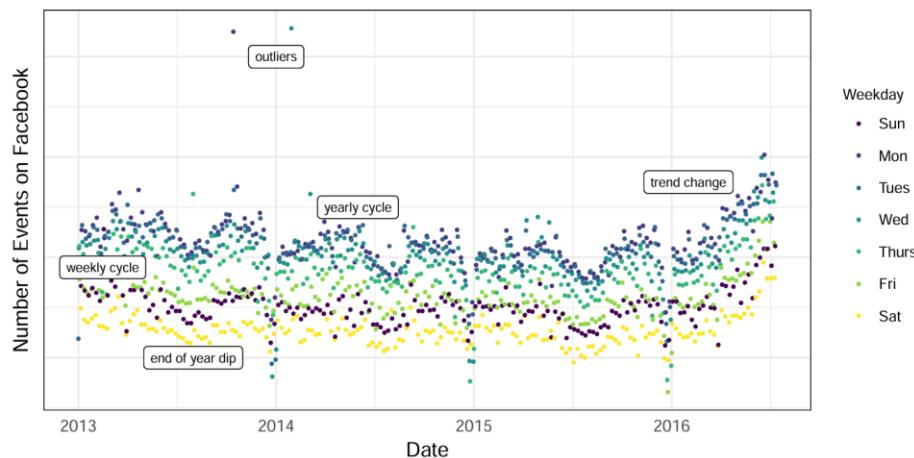


Ilustración 16 Número de eventos creados en Facebook.

La figura anterior, muestra el número de eventos creados en Facebook. Hay un punto para cada día, y los puntos están codificados por colores según el día de la semana para mostrar el ciclo semanal. Las características de esta serie temporal son representativas de muchas series temporales empresariales: múltiples estacionalidades marcadas, cambios de tendencia, valores atípicos y eventos festivos. Hay varios efectos estacionales claramente visibles en esta serie temporal: ciclos semanales y anuales, y una caída pronunciada alrededor de Navidad y Año Nuevo. Estos efectos estacionales surgen de forma natural y son previsibles en series temporales generadas por acciones humanas. La serie temporal también muestra una clara tendencia de cambio en los últimos seis meses, que puede surgir en series temporales afectadas por nuevos productos o cambios en el mercado. Finalmente, los conjuntos de datos reales suelen presentar valores atípicos, y esta serie temporal no es la excepción [21].

Se implementa un modelo de series temporales descomponible (Harvey y Peters, 1990) con tres componentes principales: tendencia, estacionalidad y festivos. Estos se combinan en la siguiente ecuación:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t.$$

$g(t)$ es la función de tendencia que modela cambios no periódicos en el valor de la serie temporal, $s(t)$ representa cambios periódicos (estacionalidad semanal y anual), y $h(t)$ representa los efectos de los días festivos que ocurren en horarios potencialmente irregulares durante uno o más días. El término de error, ϵ_t , representa cualquier cambio que no sea acomodado por el modelo. Esta especificación es similar a un modelo aditivo generalizado (GAM) (Hastie y Tibshirani 1987), una clase de modelos de regresión con suavizadores potencialmente no lineales aplicados a los regresores. Modelar la estacionalidad como un componente aditivo es el mismo enfoque adoptado por el suavizado exponencial (Gardner 1985). La estacionalidad multiplicativa, donde el efecto estacional es un factor que multiplica $g(t)$, se puede lograr mediante una transformación logarítmica [21].

4.4 Modelo Gradient Boosting y LightGBM

Los modelos de potenciación de gradiente han ganado popularidad en la comunidad de aprendizaje automático gracias a su capacidad para lograr excelentes resultados en una amplia gama de casos de uso, incluyendo tanto regresión como clasificación. Algunas de las principales ventajas de usar modelos de potenciación de gradiente para la predicción son [22]:

- La facilidad con la que se pueden incorporar al modelo variables exógenas (factores externos al modelo que pueden ayudar a explicar el comportamiento de la variable principal), además de variables autorregresivas.
- La capacidad de capturar relaciones no lineales (no proporcionales) entre variables.
- Alta escalabilidad, que permite a los modelos gestionar grandes volúmenes de datos.

Existen varias implementaciones populares de potenciación de gradiente en Python; cuatro de las más populares son XGBoost, LightGBM, HistGradientBoostingRegressor de scikit-learn y CatBoost. Todas estas bibliotecas utilizan la API de scikit-learn, lo que las hace compatibles con skforecast. LightGBM es un framework de potenciación de gradientes que utiliza algoritmos de aprendizaje basados en árboles. Está diseñado para ser distribuido y eficiente [23][24].

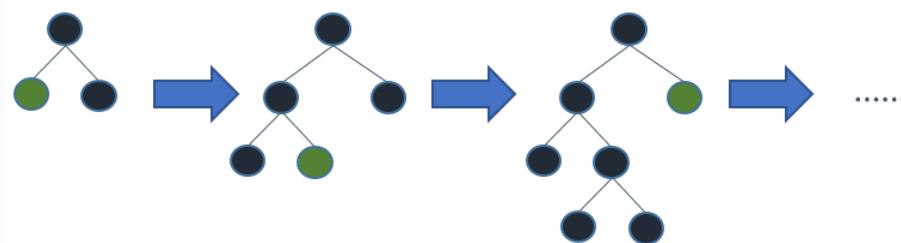


Ilustración 17 Ejemplo de Optimización Leaf-wise tree growth

LightGBM utiliza algoritmos basados en histogramas [25, 26, 27] que agrupan los valores continuos de las características (atributos). Esto acelera el entrenamiento y reduce el uso de memoria. Las ventajas de los algoritmos basados en histogramas incluyen las siguientes:

- Costo reducido para calcular la ganancia de cada división.
- Uso de la sustracción de histogramas para una mayor velocidad. Para obtener los histogramas de una hoja en un árbol binario, usa la sustracción de histogramas de su padre y su vecino.
- Reduce el uso de memoria. Reemplaza valores continuos con contenedores discretos. No es necesario almacenar información adicional para pre ordenar los valores de las características.
- Reduce el costo de comunicación para el aprendizaje distribuido.

4.5 Análisis de crecimiento de Biomasa

El análisis de crecimiento de biomasa es una herramienta fundamental en la biología y la ecología, ya que permite entender cómo los organismos acumulan masa a lo largo del tiempo. Este análisis es especialmente relevante en la acuicultura, donde el crecimiento eficiente de los peces es crucial para la sostenibilidad y rentabilidad de la producción.

4.5.1 Crecimiento de los Peces y función logística

El crecimiento de los peces es un proceso intrincado, influenciado por la genética, la alimentación, la calidad del agua y las condiciones ambientales. En la acuicultura, el objetivo es optimizar estos factores para potenciar el crecimiento y la salud de los peces. Habitualmente, el crecimiento se cuantifica mediante el aumento de peso y longitud a lo largo del tiempo. La función logística, por su parte, es una herramienta matemática comúnmente empleada para modelar el crecimiento de poblaciones y la acumulación

Comentado [MLG8]: Esto es información que meterás al modelo no? Hay mucho mucho lio en esta parte

Comentado [PL9R8]: El modelo implementara la función de Thyholdt en función de la temperatura del mar pronosticada y el paso del tiempo. A partir del número inicial de peces.

de biomasa en entornos con recursos finitos. Esta función describe un crecimiento sigmoidal, con forma de "S", que es típico de numerosos procesos biológicos.

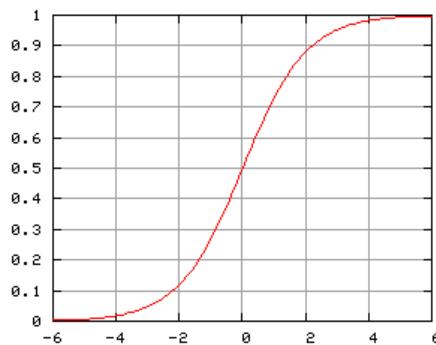


Ilustración 18 función logística, curva logística o curva en forma de S

La ecuación logística básica se expresa como [31]:

$$\frac{dN}{dt} = rN \left(1 - \frac{N}{K}\right)$$

donde:

- **N** es la biomasa o la población en el tiempo *t*.
- **r** es la tasa de crecimiento intrínseca.
- **K** es la capacidad de carga del entorno, es decir, el límite máximo de biomasa que el entorno puede soportar.

La función logística tiene tres fases principales:

1. **Crecimiento Exponencial Inicial:** Cuando la biomasa es pequeña en comparación con la capacidad de carga, el crecimiento es casi exponencial.

2. **Disminución de la Tasa de Crecimiento:** A medida que la biomasa se acerca a la capacidad de carga, la tasa de crecimiento disminuye debido a la competencia por recursos.
3. **Saturación:** Finalmente, la biomasa se estabiliza en torno a la capacidad de carga, y el crecimiento neto se detiene.

Este modelo es útil para predecir el comportamiento de la biomasa en diferentes condiciones ambientales y para optimizar la gestión de recursos en la agricultura, la silvicultura y la acuicultura.

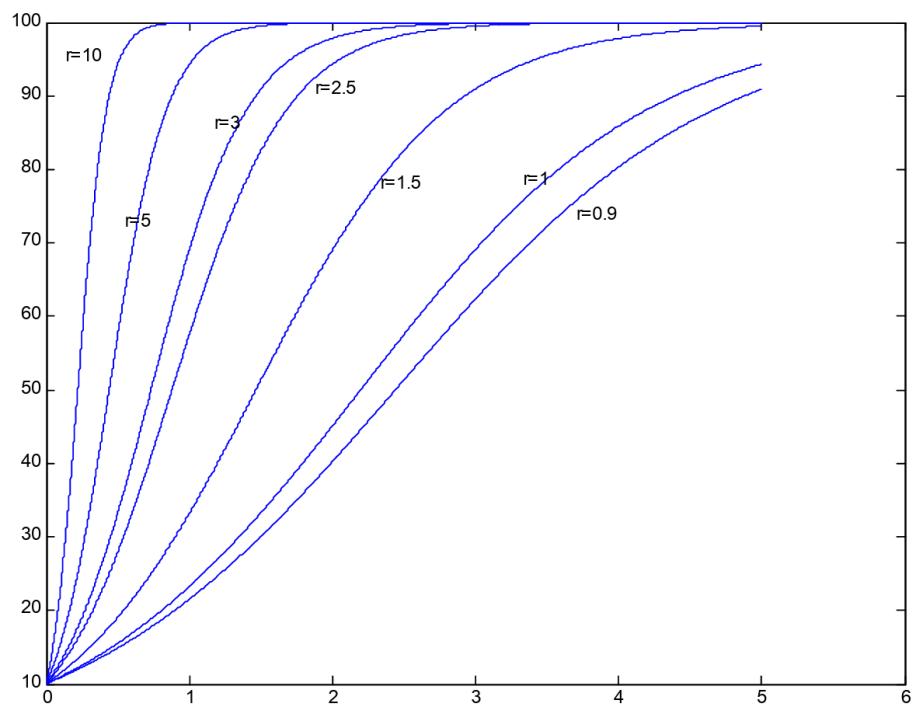


Ilustración 19 curvas logísticas para diversos valores de r con $N_0 = 10$ y $K = 100$

La figura anterior muestra varias curvas logísticas para diversos valores de r con $N_0 = 10$ y $K = 100$. Cuanto mayor sea r , más rápido alcanzará la curva la capacidad de carga K .

4.5.2 Modelo de Crecimiento de Thyholdt (2014) para el Salmón Noruego

Uno de los modelos más utilizados para analizar el crecimiento del salmón noruego es el propuesto por Thyholdt en 2014. Este modelo se basa en datos empíricos y considera variables como la temperatura del agua, la densidad de población y la calidad del alimento. Thyholdt desarrolló una fórmula matemática que permite predecir el crecimiento del salmón en diferentes condiciones, lo que es de gran utilidad para los productores de salmón.

Este modelo se basa en una función de crecimiento logística que utiliza datos agregados de tres regiones diferentes de Noruega, estimados para cinco generaciones distintas de salmón de cultivo [28,29,30]. La función de crecimiento considera variables como la temperatura del agua, la densidad de población y la calidad del alimento, lo que permite a los productores de salmón optimizar sus estrategias de cultivo.

El estudio de Thyholdt destaca la importancia de la temperatura del agua en el crecimiento del salmón. Se encontró que un aumento en la temperatura del mar acelera el crecimiento en las regiones del norte y centro de Noruega, mientras que, en la región sur, un aumento en la temperatura tiene un efecto negativo en el crecimiento [28]. Este modelo es particularmente útil para los productores, ya que les permite ajustar sus estrategias de alimentación y manejo en función de las condiciones ambientales específicas de cada región.

Como Thyholdt indica [28]: "El aumento del crecimiento de los peces está determinado principalmente por la intensidad de la alimentación y la temperatura (Aasheim et al., 2011). Sin embargo, dado que los patrones de alimentación presentan poca variación, solo se utiliza la temperatura como variable explicativa en el modelo. Por lo tanto, el modelo específico estimado para cada región i y cohorte c en el mes t es":

$$W_{i,c,t} = \frac{\alpha}{[1+e^{-(\beta T_{i,t})(t-\mu)}]}$$

Ilustración 20 Thyholdt (2014) función de crecimiento

en la que $W_{i,c,t}$ es el peso promedio de los peces en la región i en la cohorte c en el mes t , y $T_{i,t}$ es la temperatura en la región i en el mes t . α , β , and μ son parámetros por estimar; α es el peso máximo asintótico, β es el coeficiente de pendiente y μ es el punto de inflexión.

Región	Especie	T (°C)	β	α (g)	μ (meses)
Mediterráneo [28]	Dorada, Lubina	18-24	0,018-0,025	1.000-2.500	18-24
Atlántico Norte [33]	Salmón Atlántico	12-18	0,012-0,018	5.000-15.000	18-30
Báltico/Nórdico [33]	Salmón Atlántico	4-14	0,008-0,014	5.000-12.000	20-32
Mar del Norte [35]	Salmón Atlántico	8-17	No estándar	6.000-14.000	20-30
Sudeste Asiático [36]	Tilapia	24-32	No estándar	500-1.500	5-7
Latinoamérica [34]	Salmón, Tilapia	20-33	No estándar	5.000-12.000	18- 24

Tabla 1 Constantes de crecimiento para varias especies

Los valores específicos de (α , β , μ) de los parámetros de crecimiento de la tabla anterior se calculan en estudios científicos [37][38][39][40][41] que son ejemplos representativos del tipo de investigación de donde se extraen dichos valores para las especies mencionadas.

Con los datos de la tabla y las referencias [37][38][39][40][41], un resumen de las constantes de crecimiento para el salmón noruego (*Salmo salar*) son:

- **α (alfa - Peso Máximo):** El rango típico se sitúa entre **5.000 y 12.000 gramos**. Este es el peso teórico máximo que alcanzaría el pez en un ciclo de cultivo.
- **β (beta - Tasa de Crecimiento):** El rango reportado es de **0,008 a 0,018**. Este coeficiente indica la rapidez con la que el pez se acerca a su peso máximo.
- **μ (mu - Punto de Inflexión):** Generalmente ocurre entre los **18 y 31 meses** de edad. Este es el momento en el que la tasa de ganancia de peso del salmón es más alta.

La temperatura óptima para el cultivo de salmón Atlántico (*Salmo salar*) en jaulas marinas se encuentra entre 8 y 14°C, donde los peces se alimentan y crecen mejor. Temperaturas superiores a 16°C pueden causar estrés, disminución del apetito y crecimiento reducido, mientras que temperaturas por encima de 23°C pueden ser letales. En su hábitat natural, el salmón puede tolerar temperaturas entre 7 y 20°C.

4.5.3 Conclusión

Este modelo será fundamental para el prototipo de Gemelo Digital, ya que deberá ser codificado e implementado en el sistema para simular el crecimiento de la biomasa de salmón. La implementación de la función de Thyholdt permitirá al prototipo realizar predicciones precisas del peso y la cantidad de peces en las balsas a lo largo del tiempo.

Para ello, el modelo de crecimiento implementará la función de Thyholdt, utilizando la temperatura del mar pronosticada por el modelo de temperatura, el transcurso del

tiempo, y el número inicial de peces (biomasa inicial). Esto es crucial para apoyar la toma de decisiones, especialmente en la determinación de la fecha óptima de cosecha.

4.6 Planificación

En este proyecto, se adoptará una planificación ágil para el desarrollo del prototipo de Gemelo Digital, lo que nos permitirá responder rápidamente a los cambios y aprender de cada iteración. Nos enfocaremos en ciclos de trabajo cortos (Sprints) que incluirán planificación, ejecución, revisión y adaptación. La investigación será una actividad continua, especialmente relevante en las fases iniciales y de modelado.

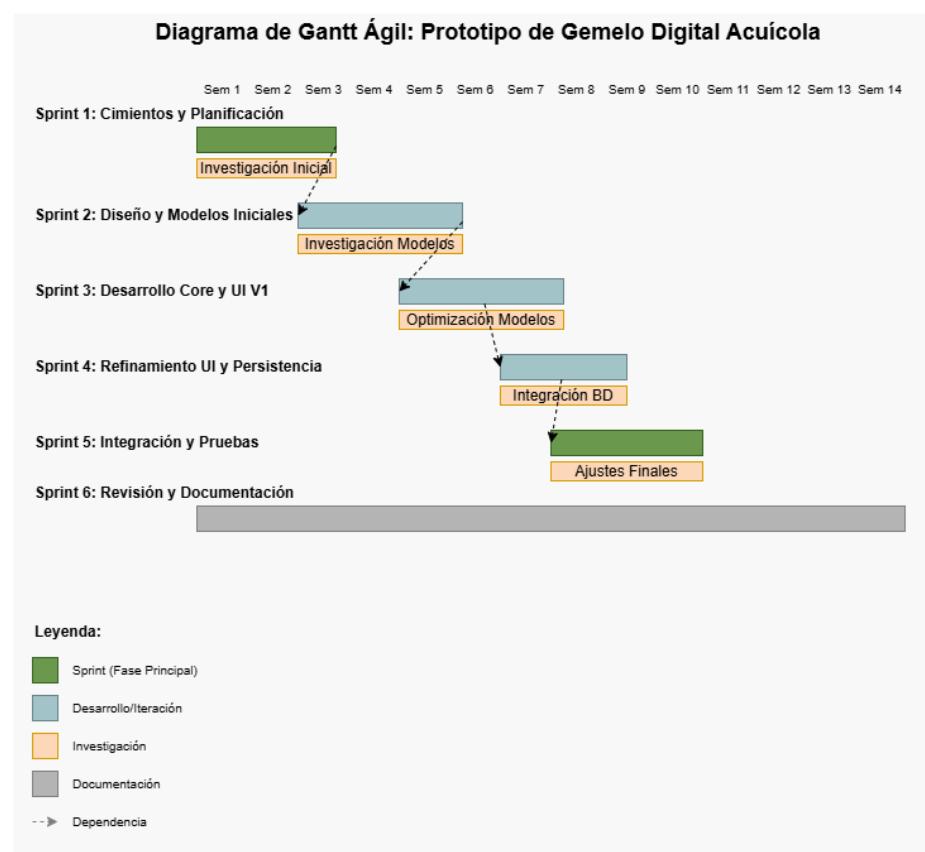


Ilustración 21 Gantt del Prototipo

Pedro López Treitiño

En el diagrama anterior, se presenta un diagrama de Gantt que visualiza nuestros Sprints, incluyendo las fases clave y la dedicación a la investigación. Cada "Sem" representa una semana de trabajo.

5 Diseño

5.1 Introducción

El prototipo se implementará como un gemelo digital de nivel 1 a 3 desconectado del proceso productivo de la planta de acuicultura. No se implementará, el sistema de comunicaciones, sensores y actuadores IoT que podrán plantearse para futuras ampliaciones. Y tampoco de validarán los datos pronosticados con los datos del proceso.

El prototipo se centrará en el desarrollo para la ayuda a la toma de decisiones de la organización de la planta de acuicultura. Se plantea un cuadro de mandos que visualice los modelos predictivos de crecimiento de biomasa, temperatura del agua y precio de venta del producto.

En la solución propuesta para el prototipo, denominada "SalmonTwin", se optará por una aplicación de escritorio multiplataforma con un único usuario encargado de configurar y analizar los datos proporcionados por la aplicación, el gestor de las jaulas marinas, en futuras ampliaciones se podrán plantear múltiples roles de usuarios. Se utilizará Python como lenguaje de programación para poder integrarse sencillamente con el ecosistema de librerías de ciencia de datos para Python; y se usará Pyside6 como "framework" gráfico para la aplicación.

El código se va a implementar en su totalidad con el IDE de Visual Studio Code. Aunque no es un IDE en sí, es un editor de código muy potente y extensible. Con las extensiones adecuadas, de Python, se convierte en un entorno de desarrollo muy completo para Python, que incluye depuración de código e integración con GitHub como repositorio de código. Está desarrollado por Microsoft y es gratuito y de código abierto. Existiendo versiones para varios sistemas operativos. Se puede usar en Windows, macOS y Linux.

En resumen, se usarán las siguientes librerías y entornos:

Librería	Versión	Descripción
<i>PySide6</i>	6.9.0	PySide6 es el conjunto de enlaces oficiales de Python para el framework Qt, que permite desarrollar aplicaciones gráficas de usuario (GUI) de alta calidad y multiplataforma.
<i>pyqtgraph</i>	0.13.7	pyqtgraph es una librería de gráficos y visualización de datos para Python, diseñada para ser rápida y fácil de usar, ideal para aplicaciones científicas y de ingeniería.
<i>pandas</i>	2.3.0	pandas es una librería de análisis de datos que proporciona estructuras de datos flexibles y expresivas, como DataFrames, para manipular y analizar datos de manera eficiente.
<i>numpy</i>	2.2.6	numpy es una librería fundamental para la computación científica en Python, que ofrece soporte para matrices y operaciones matemáticas de alto rendimiento.
<i>statsforecast</i>	2.0.1	statsforecast es una librería de Python para el análisis de series temporales, que incluye modelos estadísticos y algoritmos de machine learning para realizar predicciones precisas.
<i>prophet</i>	1.1.7	prophet es una herramienta de análisis de series temporales desarrollada por Facebook, que facilita la creación de modelos predictivos robustos y precisos.
<i>python</i>	3.11.9	Python es un lenguaje de programación de alto nivel, interpretado y de propósito general, conocido por su simplicidad y legibilidad, ampliamente utilizado en desarrollo web, análisis de datos, inteligencia artificial, etc.

5.2 Arquitectura del sistema

El proyecto “SalmonTwin” implementa una arquitectura MVC (Model-View-Controller) adaptada específicamente para PySide6, donde se ha personalizado el patrón tradicional para aprovechar las características del “framework Qt” y las necesidades específicas de la aplicación de gemelo digital.

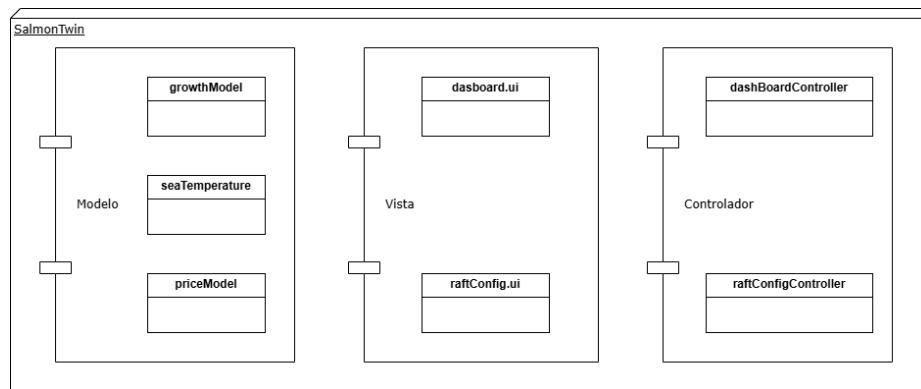


Ilustración 22 Diagrama de componentes

Las vistas “dashboard.ui” y “raftConfig.ui” se van a diseñar con “Qt Creator 15.0.1 (Community)” que permite hacer un prototipado rápido de componentes visuales mediante “drag and drop” (arrastrar y soltar) de componentes gráficos. Esta herramienta genera dos archivos “.ui” que se cargan desde Pyside6 para añadirles funcionalidad desde el controlador de la vista.

Para almacenar los datos de cada balsa de la planta de acuicultura se utilizará la clase “seaRaft”. La persistencia de datos se realizará sobre archivos JSON (JavaScript Object Notation) son un formato de texto ligero para el intercambio de datos. Son fáciles de leer y escribir tanto para humanos como para máquinas, lo que los hace ideales para almacenar y transmitir datos estructurados.

5.3 Diseño de componentes

Explicación detallada de cada componente del sistema, incluyendo su funcionalidad, tecnologías utilizadas, y cómo se integran con otros componentes.

5.3.1 Modelo

El modelo se compone de tres clases “growthModel”, “seaTemperature” y “priceModel”. Que se encargan respectivamente del modelo de crecimiento de la biomasa, el de predicción de temperatura y el de predicción de precio de venta de la materia prima.

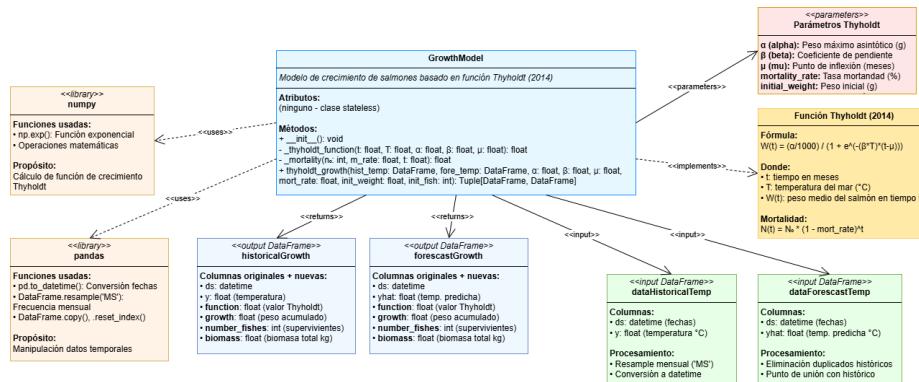


Ilustración 23 Diagrama de clase del GrowthModel

5.3.1.1 Modelo GrowthModel

Descripción General

El diagrama muestra la **clase GrowthModel**, que implementa el modelo de crecimiento de salmones basado en la función matemática de **Thyholdt (2014)** [28]. Esta clase **GrowthModel**, es un componente fundamental del sistema "Salmon Twin", encargado de simular y predecir el crecimiento de la biomasa de salmón en jaulas marinas. Este modelo se basa en la función matemática de Thyholdt (2014) que considera la temperatura del agua marina como factor principal para estimar la evolución temporal del peso, la biomasa total y la supervivencia de los salmones.

Estructura de la Clase

Métodos Principales

La clase GrowthModel se implementa con una estructura bien definida para asegurar la robustez y precisión de sus cálculos.

- `__init__()`: Constructor vacío que inicializa la clase sin parámetros específicos. Internamente, se apoya en dos métodos privados clave:

- `_thyholdt_function()`: Método privado que implementa la función

$$\text{matemática de crecimiento } W(t) = \frac{\alpha}{1+e^{-(\beta T)(t-\mu)}}$$

de Thyholdt, que calcula el peso del salmón en función del tiempo en meses (t), la temperatura del mar en grados Celsius (T), el peso máximo asintótico en gramos (α), el coeficiente de pendiente (β), y el punto de inflexión en meses (μ).

- t : tiempo en meses.
 - T : temperatura del mar en grados Celsius.
 - α : peso máximo asintótico en gramos.
 - β : coeficiente de pendiente.
 - μ : punto de inflexión en meses.
- `_mortality()`: Por su parte, el método privado que calcula la mortandad de los salmones usando la fórmula:

$$N(t) = N_0 \times (1 - mortality_percent)^t$$

se encarga de determinar la mortandad de los salmones, utilizando una fórmula exponencial que considera el número inicial de salmones (N_0), el porcentaje de mortandad por unidad de tiempo (mortality_percent), y el tiempo en meses (t).

- N_0 : número inicial de salmones.
- $mortality_percent$: tasa de mortandad en tanto por ciento de la población por unidad de tiempo (por mes)
- t: tiempo en meses.
- `thyholdt_growth()`: Método público principal que orquesta todo el proceso de cálculo.

Para su funcionamiento, el `GrowthModel` depende de librerías científicas estándar de Python como NumPy para operaciones matemáticas, especialmente para la función exponencial de Thyholdt (`np.exp()`), y Pandas para la manipulación eficiente de datos temporales, incluyendo la conversión de fechas y el remuestreo mensual.

La clase espera datos de entrada en forma de DataFrames de Pandas, que incluyen temperaturas históricas (`dataHistoricalTemp`) y temperaturas pronosticadas (`dataForecastTemp`), cada uno con columnas específicas para fechas (`ds`) y valores de temperatura (`y` o `yhat`). Además, requiere parámetros configurables como el peso máximo (α), el coeficiente (β), el punto de inflexión (μ), la tasa de mortalidad, el peso y el número de peces iniciales. Como resultado, `GrowthModel` produce dos DataFrames de salida: `historicalGrowth` y `forecastGrowth`, que contienen los cálculos históricos y las predicciones futuras, respectivamente. Ambos DataFrames se enriquecen con columnas que detallan el valor de la función de Thyholdt, el peso acumulado del salmón en gramos, el número de peces supervivientes y la biomasa total en kilogramos.

El proceso de cálculo dentro del `GrowthModel` sigue una secuencia lógica. Primero, se realiza un preprocesamiento de datos que incluye la conversión de fechas a formato

datetime y el remuestreo a una frecuencia mensual ('MS') para uniformar la temporalidad de los datos. También se eliminan duplicados entre los datos históricos y de pronóstico. Posteriormente, se lleva a cabo un cálculo iterativo donde los días se convierten a meses (aproximadamente 30,44 días/mes, basado en la duración media del año gregoriano) [32]. Para cada período mensual, se aplica la función de Thyholdt para calcular el factor de crecimiento y se determina la mortandad. Se incorpora una limitación biológica para asegurar que el peso no supere el máximo asintótico (α). Finalmente, la biomasa se calcula multiplicando el peso por el número de supervivientes y convirtiendo el resultado a kilogramos. Para asegurar la continuidad gráfica y la consistencia en las predicciones, el crecimiento se acumula mes a mes, tomando el estado del período anterior como base, y se añade un punto de unión entre los datos históricos y las predicciones. El factor de crecimiento del último período histórico se utiliza como base para las proyecciones futuras.

Desde un punto de vista matemático, la función de Thyholdt es una curva sigmoidea que modela el crecimiento de los peces en función de la temperatura. Los parámetros α , β y μ definen el peso máximo, la sensibilidad del crecimiento a la temperatura y el punto de inflexión del crecimiento, respectivamente. La temperatura del agua (T) actúa como un factor limitante en este modelo. Este modelo es particularmente relevante para la acuicultura marina, ya que integra la dependencia térmica del crecimiento biológico con la mortalidad natural, lo que permite obtener predicciones realistas de biomasa y, consecuentemente, optimizar las decisiones de cosecha.

Resumen de dependencias y relaciones

Librerías Científicas Utilizadas

- **NumPy:** Para operaciones matemáticas, especialmente `np.exp()` en la función exponencial de Thyholdt.
- **Pandas:** Para manipulación de datos temporales, conversión de fechas y remuestreo mensual.

Datos de Entrada

- `dataHistoricalTemp`: DataFrame con temperaturas históricas (columnas: 'ds' para fechas, 'y' para temperatura).
- `dataForecastTemp`: DataFrame con temperaturas predichas (columnas: 'ds' para fechas, 'yhat' para temperatura pronosticada).
- **Parámetros del modelo:** α (peso máximo), β (coeficiente), μ (punto inflexión), tasa de mortalidad, peso inicial, número inicial de peces.

Datos de Salida

- `historicalGrowth`: DataFrame enriquecido con cálculos históricos.
- `forecastGrowth`: DataFrame enriquecido con predicciones futuras.

Ambos DataFrames de salida incluyen las columnas originales más:

- `function`: Valor de la función Thyholdt para cada período.
- `growth`: Peso acumulado del salmón en gramos.
- `number_fishes`: Número de peces supervivientes.
- `biomass`: Biomasa total en kilogramos.

Resumen del proceso de cálculo.

1. Preprocesamiento de Datos

- Conversión de fechas a formato datetime usando pd.to_datetime().
- Remuestrear a frecuencia mensual ('MS') para uniformidad temporal.
- Eliminación de duplicados entre datos históricos y de pronóstico.

2. Cálculo Iterativo

- **Conversión temporal:** Días se convierten a meses usando la aproximación 30,44 días/mes. ($365,24 / 12 = 30,44$). El valor de 365,24 días proviene de la duración media de un año en el calendario gregoriano, que es el calendario utilizado en la mayoría de los países del mundo. Este valor se obtiene al considerar que un año tiene 365 días en un año común y 366 días en un año bisiesto [32].
- **Aplicación de Thyholdt:** Para cada período mensual se calcula el factor de crecimiento.
- **Cálculo de mortalidad:** Aplicación exponencial de la tasa de supervivencia.
- **Limitación biológica:** El peso no puede superar el máximo asintótico (α).
- **Cálculo de biomasa:** Multiplicación (peso \times supervivientes) / 1000 (conversión a kg).

3. Continuidad Temporal

- El crecimiento se acumula mes a mes, llevando el estado del período anterior.
- Se añade un punto de unión entre datos históricos y predicciones para continuidad gráfica.

- El factor de crecimiento del último período histórico se usa como base para las predicciones.

Resumen del fundamento matemático

La **función Thyholdt** es una curva sigmoidea que modela el crecimiento de peces en función de la temperatura, donde:

- **α (alpha)**: Define el peso máximo que puede alcanzar el salmón.
- **β (beta)**: Controla la sensibilidad del crecimiento a la temperatura.
- **μ (mu)**: Punto de inflexión temporal donde el crecimiento es máximo.
- **T**: Temperatura del agua como factor limitante del crecimiento.

Este modelo es especialmente relevante para acuicultura marina porque integra la **dependencia térmica** del crecimiento biológico con la **mortalidad natural**, proporcionando predicciones realistas de biomasa para la optimización de decisiones de cosecha.

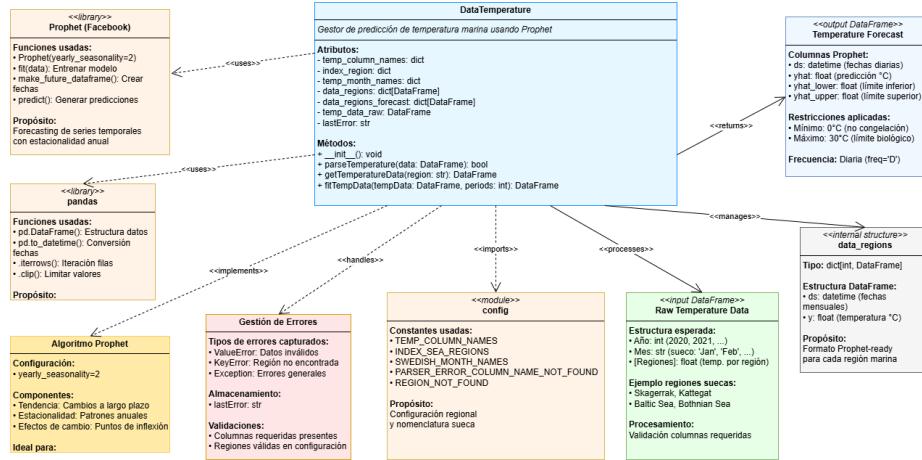


Ilustración 24 Diagrama de clase de DataTemperature

5.3.1.2 Modelo seaTemperature

Resumen del Diagrama de Clase DataTemperature

Descripción General

El diagrama ilustra la **clase DataTemperature**, un componente especializado del sistema SalmonTwin que gestiona la predicción de temperatura marina para las aguas suecas. Esta clase actúa como un “**wrapper**” **inteligente** alrededor de la librería “**Prophet**” de Facebook, adaptándolo específicamente para el contexto de acuicultura marina con consideraciones regionales y biológicas.

Arquitectura de la Clase

La arquitectura de la clase DataTemperature está definida por una serie de atributos de configuración regional y métodos principales. Entre los atributos se incluyen "temp_column_names", un diccionario que contiene los nombres de las columnas esperadas en los datos de entrada; "index_region", un mapeo de índices a nombres de regiones marinas suecas como Finnmark o Troms; "temp_month_names", para la conversión de nombres de meses suecos a formato numérico; "data_regions", un diccionario que almacena DataFrames de Pandas por región en formato Prophet (columnas 'ds' para fechas, 'y' para temperatura); y data_regions_forecast, que guarda las predicciones de temperatura por región. Además, cuenta con lastError, un sistema centralizado para la gestión de errores. Los métodos principales incluyen parseTemperature(), que procesa los datos brutos multiregionales y los convierte al formato utilizado por Prophet; getTemperatureData(), para recuperar datos históricos de una región específica; y fitTempData(), que ejecuta la predicción utilizando Prophet con una configuración optimizada.

Resumen de atributos de configuración regional

- temp_column_names: Diccionario con nombres de columnas esperadas en los datos de entrada.
- index_region: Mapeo de índices a nombres de regiones marinas suecas (Finnmark, Troms, Nordland, Nord-Trøndelag, Sør-Trøndelag, Møre og Romsdal, Sogn og Fjordane, Hordaland, Rogaland og Agder, etc).
- temp_month_names: Conversión de nombres de meses suecos a formato numérico.
- data_regions: Diccionario que almacena DataFrames por región en formato Prophet (ds, y).
- data_regions_forecast: Diccionario con predicciones por región.
- lastError: Sistema de gestión de errores centralizado.

Dependencias e Integración

En cuanto a las dependencias e integración, DataTemperature se apoya en Prophet de Facebook como motor de predicción, configurado con `yearly_seasonality=2` para capturar patrones estacionales complejos de temperatura marina a lo largo de los años. Utiliza Pandas para la manipulación de datos temporales, la conversión de fechas y la estructuración de DataFrames. También depende de un módulo "Config" que centraliza los nombres de las regiones suecas, las columnas esperadas y los mensajes de error. La clase está específicamente diseñada para el contexto acuícola sueco, incorporando regiones marinas específicas como "Finnmark" y "Troms", la nomenclatura local para los nombres de los meses en sueco, y una configuración regionalizada donde cada región marina tiene sus propios DataFrames y predicciones.

Resumen de dependencias externas

- **Prophet (Facebook)**: Motor de predicción con `"yearly_seasonality"=2` para capturar patrones estacionales complejos de temperatura marina, en los años anteriores.
- **Pandas**: Manipulación de datos temporales, conversión de fechas y estructuración de DataFrames.
- **Config**: Módulo de configuración que centraliza nombres de regiones suecas, columnas esperadas y mensajes de error.

Resumen de especialización regional sueca

La clase está específicamente diseñada para el **contexto acuícola sueco**, incorporando:

- **Regiones marinas específicas**: "Finnmark", "Troms", "Nordland", "Nord-Trøndelag", "Sør-Trøndelag", "Møre og Romsdal", "Sogn og Fjordane", "Hordaland", "Rogaland og Agder", etc.

-
- **Nomenclatura local:** Procesamiento de nombres de meses en sueco.
 - **Configuración regionalizada:** Cada región marina tiene sus propios DataFrames y predicciones.

Flujo de Procesamiento de Datos

El flujo de procesamiento de datos comienza con la entrada de datos brutos, que se espera en un DataFrame con columnas de Año, Mes (en sueco) y temperatura por región, y se valida automáticamente la presencia de las columnas requeridas. Luego, se transforman al formato Prophet, convirtiendo el Año y el Mes sueco a formato datetime ('YYYY-MM') y estructurando los datos en DataFrames individuales por región con columnas 'ds' (fecha) y 'y' (temperatura). La predicción con Prophet se realiza utilizando una configuración optimizada (yearly_seasonality=2) para capturar la tendencia a largo plazo, la estacionalidad anual y los efectos de cambio, generando predicciones diarias (freq='D') para alta granularidad. Finalmente, el post-procesamiento incluye la aplicación de la limitación biológica con .clip(lower=0, upper=30) para mantener valores realistas de temperatura, evitando temperaturas de congelación (0°C) y superando el límite biológico de supervivencia del salmón (30°C). Esta limitación se aplica tanto a la predicción central (yhat) como a las bandas de confianza (yhat_lower, yhat_upper).

El algoritmo Prophet especializado, con la siguiente configuración: "yearly_seasonality=2", permite capturar patrones estacionales complejos, variaciones de temperatura no lineales, posibles dobles ciclos anuales y adaptarse a los patrones térmicos específicos de las aguas suecas. Sus componentes modelan la tendencia (cambios graduales por el cambio climático), la estacionalidad (ciclos anuales predecibles) y los changepoints (detección automática de puntos de inflexión térmicos). La gestión de errores y robustez se asegura mediante una validación multinivel que comprueba el esquema de datos, la validez del formato de fecha, la suficiencia de datos (mínimo 10 registros, 5 para entrenamiento) y la coherencia de los rangos temporales. Los errores se almacenan en lastError, un sistema centralizado que permite un diagnóstico posterior, con mensajes configurables y un retorno coherente de "None" en caso de error.

Resumen de funcionamiento:

1. Entrada de Datos Brutos

- **Formato esperado:** DataFrame con columnas Año, Mes (sueco), y temperatura por región.
- **Validación:** Verificación automática de presencia de columnas requeridas.
- **Ejemplo:** {'Año': 2023, 'Mes': 'Jan', 'Skagerrak': 4.2, 'Kattegat': 3.8}

2. Transformación a Formato Prophet

- **Conversión temporal:** Año + Mes sueco → datetime formato 'YYYY-MM'
- **Estructuración:** Separación por región en DataFrames individuales con columnas 'ds' (fecha) y 'y' (temperatura)
- **Normalización:** Un DataFrame Prophet por cada región marina.

3. Predicción con Prophet

- **Configuración optimizada:** "yearly_seasonality"=2 para patrones estacionales de temperatura marina.
- **Componentes capturados:** Tendencia a largo plazo + estacionalidad anual + efectos de cambio.
- **Frecuencia:** Predicciones diarias (freq='D') para granularidad alta.

4. Post-procesamiento y Restricciones

- **Limitación biológica:** Aplicación de ".clip(lower=0, upper=30)" para mantener valores realistas de temperatura.
- **Justificación:** 0°C evita temperaturas de congelación, 30°C es el límite superior biológico para supervivencia de salmón.

- **Aplicación completa:** Limitación en predicción central ($yhat$) y bandas de confianza ($yhat_lower$, $yhat_upper$).

Resumen de gestión de errores y robustez

Validación de Entrada

- **Verificación de esquema:** Comprobación automática de columnas requeridas.
- **Manejo de regiones:** Validación de existencia de regiones solicitadas.
- **Captura de excepciones:** Manejo robusto de “ValueError”, “KeyError” y excepciones generales.

Almacenamiento de Errores

- **lastError:** Sistema centralizado que permite diagnóstico posterior.
- **Mensajes configurables:** Errores parametrizados desde el módulo “config”.
- **Retorno coherente:** “None” en caso de error con información diagnóstica disponible.

Relevancia para Acuicultura Marina

Esta clase es fundamental para el sistema "Salmon Twin" por varias razones: permite una precisión regional al considerar las características térmicas específicas de cada zona marina sueca, lo cual impacta directamente el crecimiento del salmón; sus predicciones estacionales son cruciales para determinar los ciclos de crecimiento y los momentos óptimos de cosecha; las predicciones de temperatura alimentan directamente el modelo de Thyholdt para calcular la biomasa futura; las restricciones biológicas de 0-30°C garantizan predicciones realistas dentro del rango de supervivencia del salmón; y la granularidad temporal de las predicciones diarias facilita decisiones operativas precisas. En resumen, el

DataTemperature es un componente científico robusto que combina el aprendizaje automático avanzado (Prophet) con el conocimiento específico del dominio acuícola sueco, proporcionando la base térmica para todas las predicciones de crecimiento y decisiones económicas del sistema "Salmon Twin".

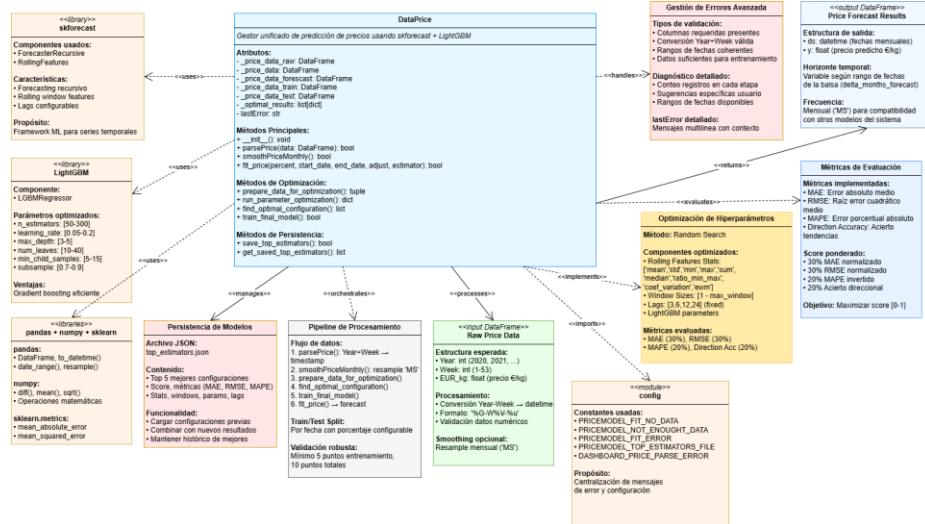


Ilustración 25 Diagrama de Clase DataPrice

5.3.1.3 Modelo priceModel

Resumen del Diagrama de Clase DataPrice

Descripción General

El diagrama ilustra la **clase DataPrice**, un componente avanzado del sistema “SalmonTwin” que implementa un **sistema de predicción de precios** basado en “machine learning”. Esta clase combina la potencia de la librería **skforecast** con el uso del algoritmo **LightGBM** para generar predicciones precisas de precios de salmón, incorporando optimización automática de “hiperparámetros” y persistencia inteligente de configuraciones.

Pedro López Treitiño

Arquitectura de la Clase

La arquitectura de la clase DataPrice se estructura en torno a una gestión de estados de datos rigurosa y métodos de procesamiento definidos. Los estados de datos incluyen _price_data_raw para los datos originales (Año, Semana, EUR_kg), _price_data para los datos procesados y validados en formato temporal, y _price_data_forecast para las predicciones generadas. Para la validación y depuración, los datos se dividen en _price_data_train y _price_data_test. Los "optimal_results" se utilizan para almacenar las mejores configuraciones de "hiperparámetros" encontradas durante la optimización. Los métodos de procesamiento de datos incluyen parsePrice() para convertir "Año+Semana" a un timestamp, smoothPriceMonthly() para aplicar un remuestreo mensual ('MS') que asegura la compatibilidad con otros modelos del sistema, y prepare_data_for_optimization() para dividir los datos en conjuntos de entrenamiento y prueba, ofreciendo un diagnóstico detallado sobre la suficiencia de los datos.

Stack Tecnológico

5.3.1.3 Modelo priceModel

El DataPrice es un componente avanzado del sistema "SalmonTwin" que implementa un sistema de predicción de precios basado en el aprendizaje automático. Esta clase integra la potencia de la librería skforecast con el algoritmo LightGBM para generar predicciones precisas de precios del salmón. Un aspecto clave es la incorporación de optimización automática de hiperparámetros y una gestión inteligente de la persistencia de configuraciones.

La arquitectura de la clase DataPrice se estructura en torno a una gestión de estados de datos rigurosa y métodos de procesamiento definidos. Los estados de datos incluyen “_price_data_raw” para los datos originales (Año, Semana, EUR_kg), “_price_data” para los datos procesados y validados en formato temporal, y “_price_data_forecast” para las predicciones generadas. Para la validación y depuración, los datos se dividen en “_price_data_train” y “_price_data_test”. Los optimal_results se utilizan para almacenar las mejores configuraciones encontradas durante la optimización.

Los métodos de procesamiento de datos incluyen parsePrice() para convertir "Año+Semana" a un timestamp, smoothPriceMonthly() para aplicar un remuestreo mensual (MS) que asegura la compatibilidad con otros modelos del sistema, y prepare_data_for_optimization() para dividir los datos en conjuntos de entrenamiento y prueba, ofreciendo un diagnóstico detallado sobre la suficiencia de los datos.

El *stack* tecnológico del DataPrice se compone de herramientas clave. Para el *framework* especializado, se utiliza skforecast, que proporciona un Forecaster Recursive para la predicción "multi-step" recursiva, optimizada para series temporales. También emplea "RollingFeatures" para la ingeniería de características con ventanas deslizantes, permitiendo configurar estadísticas como la media, desviación estándar, valores mínimos, máximos, suma, mediana, relación min-max, coeficiente de variación y media móvil ponderado exponencialmente (ewm). Como motor de predicción, se integra LightGBM a través de LGBMRegressor, un algoritmo de *gradient boosting*.

Pedro López Treitiño

eficiente con parámetros optimizados. El espacio de hiperparámetros para la optimización incluye `n_estimators`, `learning_rate`, `max_depth` y `num_leaves`. Para prevenir el *overfitting*, se consideran parámetros como `min_child_samples`, `subsample` y `colsample_bytree`.

Resumen de uso de skforecast: Framework Especializado

- `ForecasterRecursive`: Predicción “multi-step” (en varios pasos) recursiva optimizada para series temporales.
- `RollingFeatures`: Ingeniería de características con ventanas deslizantes.
- **Estadísticas configurables:**
 - `'mean'`: La **media** o promedio.
 - `'std'`: La **desviación estándar**, que mide la dispersión de los datos.
 - `'min'`: El **valor mínimo**.
 - `'max'`: El **valor máximo**.
 - `'sum'`: La **suma** total de los valores.
 - `'median'`: La **mediana**, el valor central de un conjunto de datos ordenado.
 - `'ratio_min_max'`: La **relación entre el valor mínimo y el máximo**.
 - `'coef_variation'`: El **coeficiente de variación**, que compara la desviación estándar con la media.
 - `'ewm'`: La **media móvil ponderado exponencialmente**, que da más peso a los datos más recientes.

LightGBM: Motor de Predicción

- `LGBMRegressor`: “Gradient boosting” eficiente con parámetros optimizados.
- **Espacio de hiperparámetros**: `n_estimators` [50-300], `learning_rate` [0,05-0,2], `max_depth` [3-5], `num_leaves` [10-40].

-
- **Prevención de overfitting:** min_child_samples[5-15], subsample[0,7-0,9], colsample_bytree[0,7-0,9].

Sistema de Optimización Automática

El sistema de optimización automática se basa en una búsqueda aleatoria multiobjetivo (`find_optimal_configuration()`), que realiza una búsqueda exhaustiva en cuatro espacios de parámetros simultáneamente. Esto abarca la selección aleatoria de estadísticas, la generación de tamaños de ventana adaptativos al tamaño de los datos, la configuración del regresor (hiperparámetros y prevención de *overfitting*), y el uso de *lags* en grupos de cuatro para capturar patrones estacionales y tendencias (por ejemplo, [3,6,12,24] para trimestral, semestral, anual y bianual). La evaluación se realiza mediante una multimétrica ponderada que incluye el Error Absoluto Medio (MAE) con un peso del 30% para la precisión general, la Raíz del Error Cuadrático Medio (RMSE) con un 30% para penalizar errores grandes, el Error Porcentual Absoluto Medio (MAPE) con un 20% para la precisión relativa, y la Precisión de Dirección (Direction Accuracy) con un 20% para el acierto en la predicción de tendencias, un factor crítico para las decisiones de cosecha. El objetivo es maximizar un “*score*” ponderado definido por la fórmula:

$$score = 0,30 \times \left(1,0 - \frac{mae}{5,0}\right) + 0,30 \times \left(1,0 - \frac{rmse}{8,0}\right) + 0,20 \times \left(1,0 - \frac{mape_{value}}{100}\right) + 0,20 * \left(\frac{dir_acc}{100}\right)$$

La persistencia inteligente del conocimiento se implementa mediante un sistema de memoria organizacional que permite almacenar las 5 mejores configuraciones históricas en formato JSON (`save_top_estimators()`) y recuperarlas para su reutilización (`get_saved_top_estimators()`). Los nuevos resultados se integran de manera inteligente con los históricos, manteniendo la diversidad. Estas configuraciones se guardan en el archivo `estimators.json`, que contiene el `score`, las métricas (MAE, RMSE, MAPE, Direction Accuracy), las estadísticas, los tamaños de las ventanas, los parámetros y los *lags*.

Pipeline de Procesamiento

El *pipeline* de procesamiento se inicia con `parsePrice()` para la conversión de "Año+Semana" a `datetime` con validación por fila. Luego, `smoothPriceMonthly()` realiza el remuestreo mensual utilizando Pandas para reducir el ruido. `prepare_data_for_optimization()` divide los datos de forma inteligente, validando la suficiencia de estos. La orquestación de la búsqueda de parámetros se lleva a cabo mediante `run_parameter_optimization()` con "callbacks" de progreso. Un "callback" es una función o método que se pasa como argumento a otra función (en este caso, `run_parameter_optimization()`) y que se ejecuta en un momento específico durante la ejecución de esa función principal. En este escenario, los "callbacks" de progreso se utilizan para recibir actualizaciones sobre el estado y los avances de la búsqueda de los parámetros óptimos del modelo de precios, sin necesidad de que la función principal (`run_parameter_optimization()`) detenga su ejecución. Esto permite que, mientras la búsqueda de parámetros se ejecuta en segundo plano (en un hilo separado), el sistema pueda actualizar la interfaz de usuario con información en tiempo real, como el progreso de la búsqueda, los resultados parciales o los mejores parámetros encontrados hasta el momento.

El entrenamiento final del modelo se realiza con `train_final_model()` utilizando los mejores parámetros encontrados, y la predicción final para el horizonte específico de la balsa se ejecuta con `fit_price()`.

La adaptabilidad dinámica del modelo se manifiesta en su capacidad para ajustar el horizonte de las predicciones al rango temporal específico de cada balsa, escalar automáticamente los tamaños de las ventanas (`window_sizes`) al tamaño de los datos disponibles, y adaptar los *lags* para evitar el *overfitting* con datos limitados. La integración con el ecosistema "SalmonTwin" se asegura a través de la compatibilidad temporal, generando salidas en formato mensual ('MS') compatibles con los modelos de temperatura y crecimiento, y coordinándose con las fechas de balsa y los calendarios de cosecha, equilibrando la precisión con la usabilidad operativa. Este modelo soporta decisiones críticas al permitir la predicción de tendencias de precios (Direction

Accuracy), lo que ayuda a anticipar subidas o bajadas, y a identificar periodos óptimos de cosecha para maximizar ingresos. También ofrece gestión de riesgo a través de los intervalos de confianza implícitos en el conjunto de configuraciones.

Resumen del flujo de Datos

1. **[parsePrice()]:** Year+Week → datetime con validación individual por fila.
2. **[smoothPriceMonthly()]:** Remuestreo mensual usando pandas para reducir ruido.
3. **[prepare_data_for_optimization()]:** División temporal inteligente con validación de suficiencia.
4. **[run_parameter_optimization()]:** Orquestación de búsqueda con “callbacks” de progreso.
5. **[train_final_model()]:** Entrenamiento con mejores parámetros encontrados.
6. **[fit_price()]:** Predicción final para horizonte específico de balsa.

Relevancia Estratégica para Acuicultura

En definitiva, la clase DataPrice representa un activo tecnológico diferenciador y de relevancia estratégica para la acuicultura. Su capacidad de inteligencia acumulativa permite que cada optimización mejore la base de conocimiento organizacional. Ofrece precisión financiera al traducir directamente las predicciones en decisiones de cosecha multimillonarias. Su adaptabilidad regional le permite ajustarse a diferentes mercados y estacionalidades, y su robustez operativa le permite manejar datos limitados o de baja calidad. Finalmente, su arquitectura está diseñada para la escalabilidad, pudiendo gestionar múltiples balsas y regiones simultáneamente. Este sistema de predicción de precios combina las mejores prácticas de aprendizaje automático con la ingeniería de software, proporcionando la base cuantitativa para optimizar el valor económico de las decisiones de cosecha en el sistema "Salmon Twin".

5.3.2 Vista

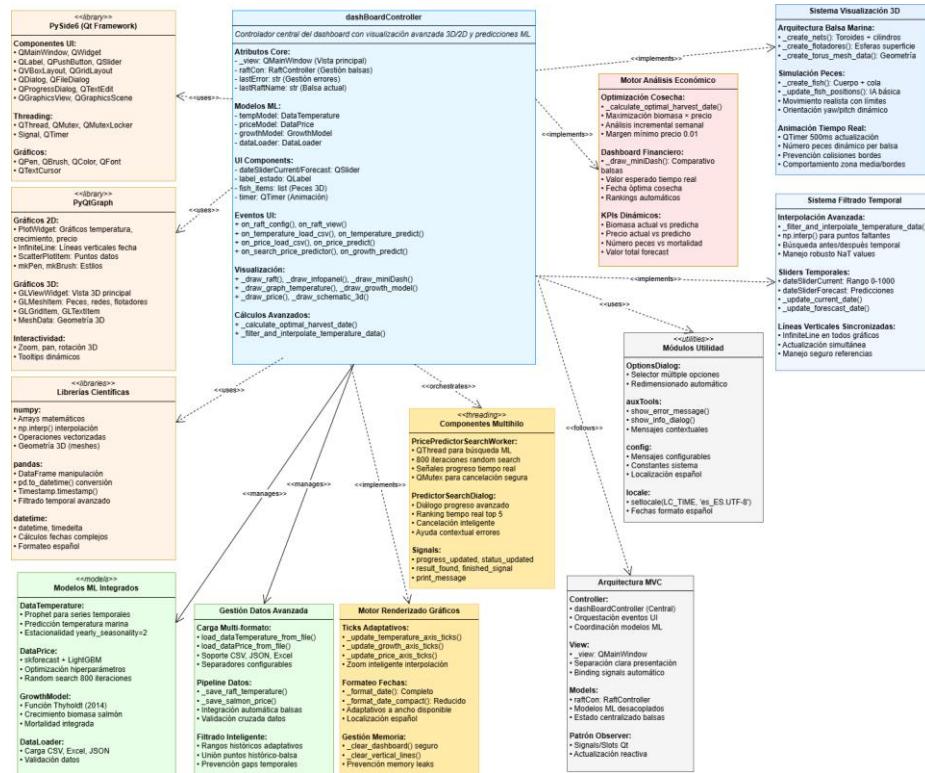


Ilustración 26 Diagrama de clase de dashBoardController

5.3.3 Controlador

Resumen del Diagrama de Clase dashBoardController

Descripción General

El diagrama ilustra la **clase dashBoardController**, el **núcleo neurálgico** del sistema “SalmonTwin” que actúa como **controlador central del “dashboard” (cuadro de mandos)** con capacidades avanzadas de visualización 2D/3D y orquestación de predicciones ML (“Machine Learning”). Esta clase implementa el patrón **MVC (Model-View-Controller)**.

Pedro López Treitiño

View-Controller) y coordina todos los componentes del sistema para proporcionar una experiencia de usuario unificada e intuitiva para la gestión de balsas marinas de acuicultura. En su arquitectura, gestiona el estado de los datos y las referencias, incluyendo la ventana principal (`_view`), el RaftController para la gestión de balsas, un sistema de errores (`lastError`) y un `label` en la barra de estado para *feedback* al usuario.

También orquesta los modelos de ML:

DataTemperature (Prophet) para la predicción de temperatura, DataPrice (skforecast + LightGBM) para la predicción de precios con optimización, y GrowthModel (Thyholdt 2014) para el crecimiento de la biomasa de salmón. Además, utiliza una clase “DataLoader” para la carga de datos en múltiples formatos (CSV, JSON, Excel).

La clase incorpora componentes interactivos de la interfaz de usuario, como deslizadores para la navegación temporal (`dateSliderCurrent`, `dateSliderForecast`) y un sistema de animación 3D para los peces, aunque el seguimiento IoT en tiempo real de la biomasa no está implementado, se podrá conectar en futuras revisiones. También utiliza un “QTimer” un reloj para animaciones en tiempo real.

El controlador maneja eventos de la interfaz de usuario para la gestión de balsas (abrir configurador, seleccionar balsa) y para la carga de datos (importar CSV de temperatura y precios). En cuanto a la orquestación de predicciones ML, gestiona el inicio de las predicciones de temperatura (Prophet), precios (skforecast con selección de los 5 mejores estimadores guardados), y crecimiento (Thyholdt con parámetros optimizados).

Un aspecto destacado es la búsqueda de optimización de parámetros del modelo de precios, que se realiza de forma asíncrona mediante un sistema de *threading* avanzado para no bloquear la interfaz gráfica. El sistema de *threading* utiliza un PricePredictorSearchWorker (QThread) para la búsqueda asíncrona de parámetros, con mecanismos de cancelación segura y señales especializadas para la comunicación entre hilos (actualización de progreso, estado, resultados, mensajes, etc.). Un PredictorSearchDialog (QDialog) integrado muestra el progreso mediante una barra y

un ranking en tiempo real de las mejores configuraciones, ofreciendo además manejo inteligente de errores y prevención de cierres inesperados durante la ejecución.

El motor de visualización 3D avanzado crea una arquitectura de balsa marina realista con toroides, cilindros y flotadores, utilizando geometría 3D optimizada con *shaders* y materiales realistas. También simula peces con un comportamiento y movimiento realista, incluyendo prevención de colisiones y orientación dinámica.

Los gráficos son adaptativos y multi-capa sincronizados, con renderizado inteligente de los *ticks* de los ejes, interpolación avanzada de datos para fechas que caen entre puntos mensuales, y deslizadores temporales coordinados para actualizar el histórico y las predicciones.

Finalmente, el controlador incluye un motor de análisis económico para la optimización inteligente de la cosecha, maximizando la biomasa y el precio en intervalos semanales. También presenta un mini-*dashboard* financiero multi-balsa con comparativas visuales, de KPIs sobre cosecha optima). Implementa un manejo de memoria inteligente para prevenir “fugas de memoria” y asegura la integración con el ecosistema SalmonTwin a través de una interfaz coherente, visualización profesional, análisis en tiempo real, optimización económica, *threading* robusto y escalabilidad visual.

Resumen de la arquitectura del controlador central

Gestión de Estado y Referencias

- _view: QMainWindow principal que aloja toda la interfaz gráfica.
- raftCon: RaftController para gestión centralizada de balsas marinas.
- lastError/lastRaftName: Sistema de gestión de errores y tracking de estado actual.
- label_estado: QLabel en statusbar para feedback permanente al usuario.

Orquestación de Modelos ML

- tempModel: DataTemperature (Prophet) - Predicción temperatura marina.
- priceModel: DataPrice (skforecast + LightGBM) - Predicción precios con optimización.
- growthModel: GrowthModel (Thyholdt 2014) - Crecimiento biomasa salmón.
- dataLoader: DataLoader - Carga multi-formato (CSV, JSON, Excel).

Componentes UI Interactivos

- dateSliderCurrent/dateSliderForecast: QSlider (0-1000) para navegación temporal. Para marcar la fecha actual y otro para marcar la fecha de predicción.
- fish_items/fish_count: Sistema de animación 3D de peces. En una posible ampliación se podría hacer un seguimiento IoT real de la biomasa de peces (no implementado).
- timer: QTimer (500ms) para animación tiempo real.

Sistema de Eventos UI

Eventos de Gestión de Balsas

- on_raft_config(): Abre configurador de balsas y actualiza contador.
- on_raft_view(): Selector de balsa con ventana de dialogo para elegir balsa.

Pipeline de Carga de Datos

- on_temperature_load_csv(): QFileDialog → DataLoader → DataTemperature.parseTemperature(). Importa csv con datos de temperatura.
- on_price_load_csv(): QFileDialog → DataLoader → DataPrice.parsePrice() → smoothPriceMonthly(). Importa csv con datos de precios con remuestreo mensual.
- load_dataTemperature_from_file() / load_dataPrice_from_file(): Métodos unificados multi-formato para carga desde fichero.

Orquestación de Predicciones ML

- on_temperature_predict(): Prophet con datos históricos ampliados (mínimo 6 meses)
- on_price_predict(): skforecast con selector de estimadores top-5 guardados.
- on_growth_predict(): Thyholdt con parámetros optimizados ($\alpha=7000g$, $\beta=0.018$, $\mu=19.0$, $mortality=0.015$)
- on_search_price_predictor(): Búsqueda exhaustiva 800 iteraciones con threading avanzado para no bloquear el interfaz gráfico.

Resumen del sistema de “Threading” (multihilo) Avanzado

PricePredictorSearchWorker (QThread)

- **Búsqueda asíncrona:** 800 iteraciones “random search” sin bloquear UI.
- **QMutex:** Cancelación segura con “stop()” y QMutexLocker.
- **“Signals” (QT) especializados:** “progress_updated”, “status_updated”, “result_found”, “finished_signal”, “print_message”. Para tener mensajería entre hilos diferentes.
- **Callback de progreso:** _progress_callback() con actualización tiempo real.

PredictorSearchDialog (QDialog)

- **Progreso integrado:** QProgressBar dentro del diálogo (no popup).
- **Ranking tiempo real:** Top-5 mejores configuraciones con métricas detalladas.
- **Manejo inteligente de errores:** Ayuda contextual y sugerencias específicas.
- **Prevención cierre:** closeEvent() personalizado durante ejecución.

Motor de Visualización 3D Avanzado

Arquitectura Balsa Marina Realista

- **_create_nets():** Toroides + cilindros con **_create_torus_mesh_data()**
- **_create_flotadores():** Esferas distribuidas en perímetro superficie.
- **GLMeshItem:** Geometría 3D optimizada con shaders ('shaded', smooth=True).
- **Materiales realistas:** Colores diferenciados (redes azul, flotadores verde, estructuras grises).

Simulación Inteligente de Peces

- `_create_fish()`: Cuerpo (escala 2.5x1.0x1.0) + cola (1.2x0.1x0.8) con “offset” (desplazamiento) calculado.
- `_update_fish_positions()`: IA básica con movimiento realista.
- **Comportamiento adaptativo**: Zonas media/bordes con parámetros diferenciados.
- **Prevención colisiones**: “Push” (empuje) hacia centro de la balsa cuando alcanza “net_radius” (radio de la balsa).
- **Orientación dinámica**: “Yaw/pitch” (guiñada/cabeceo) de los peces con cambios graduales y respuesta a bordes.

Sistema de Gráficos Adaptativos

Renderizado Inteligente de Ticks

- `_update_temperature_axis_ticks()`: Interpolación para zoom alto + datos existentes para zoom normal.
- `_update_growth_axis_ticks()/_update_price_axis_ticks()`: Algoritmos adaptativos por tipo de datos.
- **Formato contextual**: `_format_date()(completo)` frente a `_format_date_compact()(reducido)`. Para adaptarse al tamaño de la ventana.
- **Cálculo dinámico**: `max_ticks` basado en ancho píxeles (150px por tick) de los separadores de las gráficas.

Gráficos Multi-capas Sincronizados

- **InfiniteLine:** Líneas verticales sincronizadas en todos los gráficos (temperatura, crecimiento, precio).
- **Manejo robusto referencias:** `_clear_vertical_lines()` previene RuntimeError.
- **Datos históricos ampliados:** Concatenación inteligente con punto puente entre histórico y balsa.
- **ScatterPlotItem:** Puntos de datos “overlay” superpuestos.

Interpolación Avanzada de Datos

- `_filter_and_interpolate_temperature_data(): np.interp()` cuando fecha predicción cae entre puntos mensuales.
- **Búsqueda antes/después:** Manejo robusto de gaps temporales con validación.
- **Creación punto sintético:** DataFrame interpolado integrado en histórico.
- **Formato uniforme:** `pd.to_datetime()` con eliminación NaT values.

Sliders Temporales Coordinados

- **Rango 0-1000:** Mapeo lineal a días disponibles por balsa
- `_update_current_date():` Actualización histórico + líneas verticales.
- `_update_forecast_date():` Actualización predicción + KPIs dinámicos.

Motor de Análisis Económico

Optimización de Cosecha Inteligente

- `_calculate_optimal_harvest_date()`: Maximización biomasa × precio con intervalos semanales. Para cálculo de punto de recogida óptimo.
- **Parámetros calibrados:** `deltaMinPrice=0.01` para cambios significativos, `dateOffset=7días`
- **Análisis incremental:** Evaluación continua hasta “`end_date`” (fecha final) con “early stopping” (la fecha optima más cercana).
- **Manejo de anomalías:** Validación `biomasa>0` y `precio>0` en cada iteración.

Dashboard Financiero Multi-balsa

- `_draw_miniDash()`: Comparativo visual con balsa actual destacada.
- **Escalado adaptativo:** Tamaños diferenciados (350x165 vs 270x140) según importancia.
- **KPIs tiempo real:** Valor esperado, fecha óptima, biomasa predicha.
- **Estados visuales:** N/A en rojo, valores válidos en verde.

Gestión de Datos Empresarial

Pipeline de Datos Robusto

- `_save_raft_temperature() / _save_salmon_price()`: Integración automática con RaftController para almacenar los datos de cada balsa de manera unificada.
- **Datos históricos ampliados:** Extensión automática con mismo rango temporal.
- **Validación cruzada:** Verificación coherencia temporal y suficiencia datos

Manejo de Memoria Inteligente

- `_clear_dashboard()`: Limpieza segura con `timer.stop()` y desconexión signals.
- `_clear_vertical_lines()`: Prevención memory leaks en references PyQtGraph.
- **Tracking conexiones**: `timer_connected` flag para prevenir múltiples conexiones.
- **Gestión widgets**: `deleteLater()` en orden reverso del layout.

Stack Tecnológico Integrado

PySide6/Qt Framework

- **QMainWindow**: Ventana principal con statusbar y layout central.
- **QSlider**: Componentes temporales con rango 0-1000 y `valueChanged` signals.
- **QThread**: Threading asíncrono con QMutex para operaciones ML pesadas.
- **QDialog**: Interfaces modales (OptionsDialog, PredictorSearchDialog).

PyQtGraph Especializado

- **PlotWidget**: Gráficos 2D con títulos, leyendas y grid.
- **GLViewWidget**: Visualización 3D con cámara configurable y background personalizable.
- **InfiniteLine**: Líneas verticales móviles sincronizadas.
- **GLMeshItem**: Objetos 3D complejos (toroides, esferas, cilindros).

Librerías Científicas

- **numpy**: Interpolación (`np.interp`), operaciones vectorizadas, geometría 3D.
- **pandas**: Manipulación DataFrames, conversiones datetime, filtrado temporal.

Patrones de Diseño Implementados

MVC (Model-View-Controller)

- **Model:** Modelos ML desacoplados (DataTemperature, DataPrice, GrowthModel)
- **View:** _view (QMainWindow) con separación clara presentación.
- **Controller:** dashBoardController coordinando toda la lógica.

Observer Pattern

- **Signals/Slots Qt:** Actualización reactiva automática.
- **Event-driven:** Respuesta a cambios sliders, selecciones usuario.
- **Callback system:** Progress callbacks para threading ML.

Factory Pattern

- **Generación widgets:** Métodos especializados (*draw**) para cada tipo visualización.
- **Configuración contextual:** Parámetros adaptativos según tipo balsa.

Relevancia para Ecosistema SalmonTwin

Esta clase representa el **cerebro operativo** del sistema SalmonTwin porque:

1. **Integración total:** Coordina todos los modelos ML en una interfaz coherente.
2. **Visualización profesional:** Combina 2D científico + 3D inmersivo para comprensión completa.
3. **Análisis tiempo real:** Sliders temporales permiten exploración interactiva de escenarios.
4. **Optimización económica:** Identificación automática de fechas cosecha óptimas.

Pedro López Treitiño

-
- 5. **Threading robusto:** Operaciones ML pesadas sin bloqueo de interfaz.
 - 6. **Escalabilidad visual:** Soporte múltiples balsas con comparativas automáticas.

El diagrama muestra un **controlador de clase empresarial** que implementa las mejores prácticas de ingeniería de software mientras proporciona capacidades avanzadas de visualización y análisis específicamente diseñadas para maximizar la rentabilidad y eficiencia operativa en acuicultura marina de salmón.

5.4 Diseño de la interfaz de usuario (UI)

El sistema SalmonTwin presenta una interfaz gráfica de usuario (GUI) diseñada específicamente para la gestión de balsas marinas de acuicultura. La aplicación combina elementos de un cuadro de mandos, visualización 3D y herramientas de análisis predictivo en una experiencia unificada e intuitiva. La tecnología elegida como se ha comentado anteriormente es PySide6, que es el conjunto de enlaces oficiales de Python para el framework Qt, que permite desarrollar aplicaciones gráficas de usuario (GUI) de alta calidad y multiplataforma y que se integra perfectamente con el lenguaje Python con el que están desarrollados los modelos predictivos. También se ha utilizado “QT Creator Community” como diseñador “drag and drop” como ayuda al diseño de la interfaz.



Ilustración 27 QT Creator Community

5.4.1 Diseño preliminar

El objetivo del diseño es producir una visualización clara y estructurada para que sea útil para el usuario. Para esto se dividirá la pantalla en tres zonas funcionales, un menú de control en la parte superior de color gris, en tabla inferior, y una barra de notificaciones en la parte inferior en naranja en la tabla inferior:

Zona funcional 1 (verde en la tabla inferior): Datos de la balsa actual y controles de tiempo, en la parte izquierda. En la parte derecha, mini-cuadro de mandos con varias balsas e Información sobre fecha optima de recogida.

Zona 2 (azul) – Predicción de variables exógenas: incluye predicciones de temperatura y precios.

Zona 3 (roja) – Resultado esperado para la granja. Presenta salidas de la biomasa estimada (número de peces, tamaño y tasa de mortalidad). Debajo animación 3D de visualización actual de los peces.

Balsas Temperatura Precio Crecimiento	
Datos de Balsa Actual	Mini-cuadro de mando con varias balsas
Controles de tiempo	Información sobre fecha optima de recogida
Predicción Temperatura	Biomasa esperada (tamaño, mor)
Predicción Precios	Número de peces (gráfico dinámico jaula)
Balsa Actual	Notificaciones

5.4.2 Implementación Final

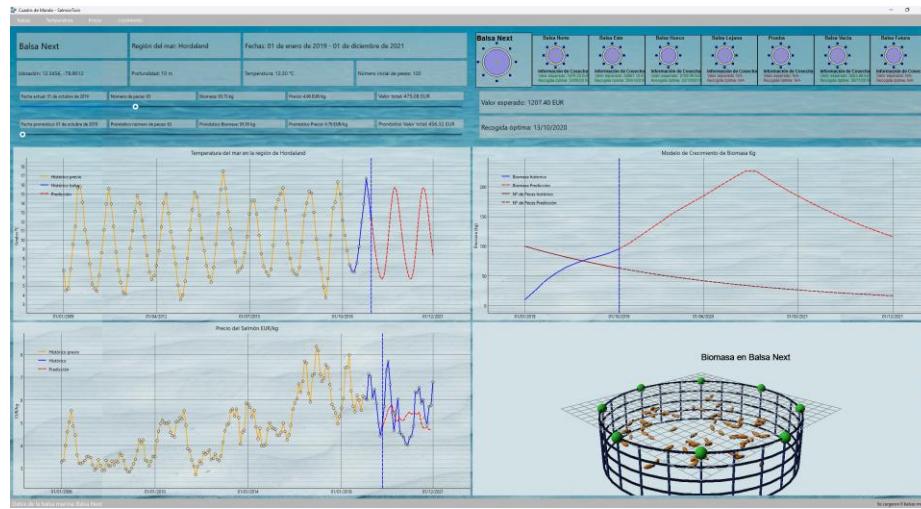


Ilustración 28 Implementación final de la ventana del cuadro de mando.

La implementación final ha quedado como la imagen que se puede observar más arriba. Además, se han añadido varias ventanas más para tareas auxiliares de configuración y de optimización que se describen a continuación.



Ilustración 29 Pantalla de configuración y creación de balsas

La pantalla de creación y configuración de balsas que se ve en la ilustración 27 arriba.

Pedro López Treitiño

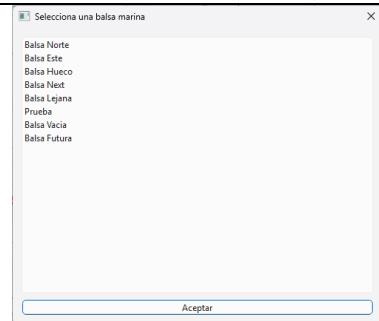


Ilustración 30 Pantalla de selección de balsas

La pantalla de selección de balsas arriba se utiliza para distintas funcionalidades, para ver los detalles de una balsa, para cargar datos de temperatura o precio en una balsa, para predecir temperatura, precio o crecimiento de biomasa en una balsa. Es decir, siempre que se realice una acción que se deba realizar sobre una balsa esta pantalla mostrará las balsas disponibles para elegir la balsa sobre la que se realizará la acción.

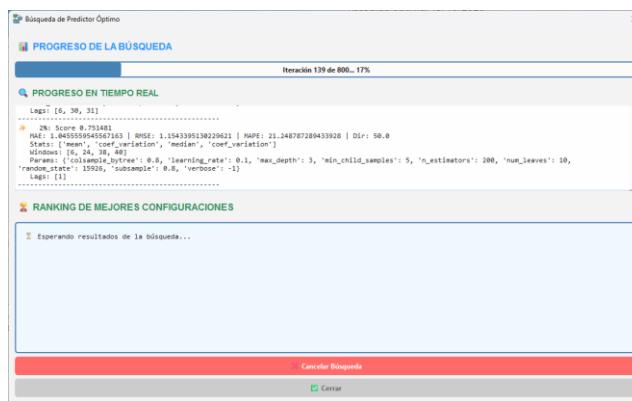


Ilustración 31 Pantalla de búsqueda de párametros del modelo de precios

La pantalla que se ve en la ilustración 29 muestra el progreso de la búsqueda automática de los parámetros del modelo de precios.

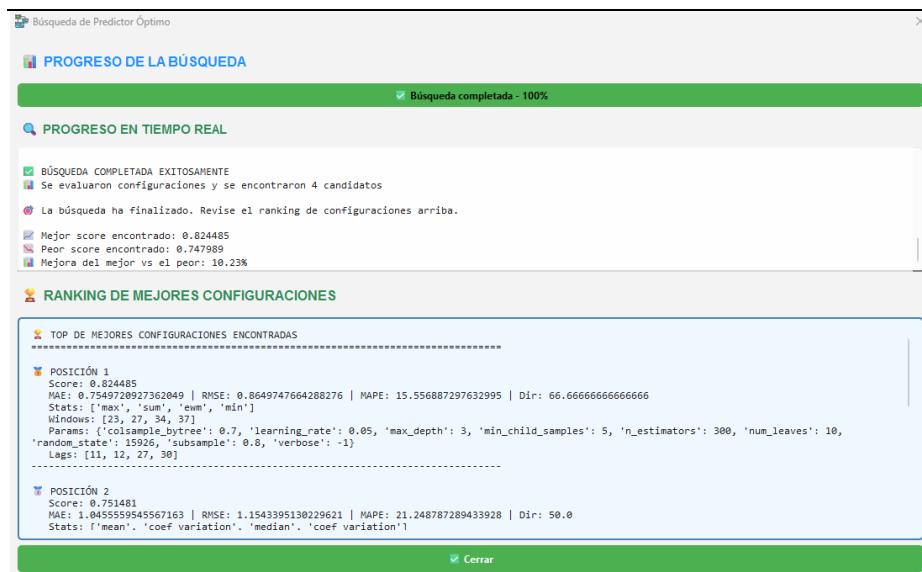


Ilustración 32 Pantalla de búsqueda de parámetros finalizada

En la ilustración 30, arriba, se puede ver la misma pantalla de la ilustración 29 con la búsqueda terminada.

Además, se generan varias ventanas de información o de error según se va trabajando con la aplicación. Como las siguientes:

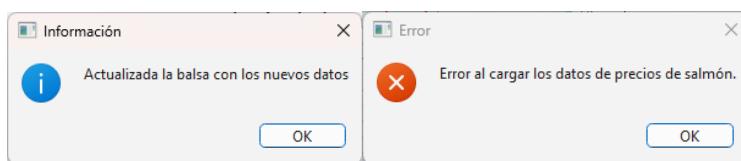


Ilustración 33 Ventanas de información y error

5.5 Diseño de sistema de persistencia de datos

El sistema SalmonTwin implementa un **modelo de persistencia híbrido** que combina archivos JSON para configuraciones y estructuras de datos complejas con un diseño preparado para migración futura hacia bases de datos relacionales y especializadas en series temporales. Esta arquitectura refleja las necesidades específicas de un sistema de gemelo digital para acuicultura marina.

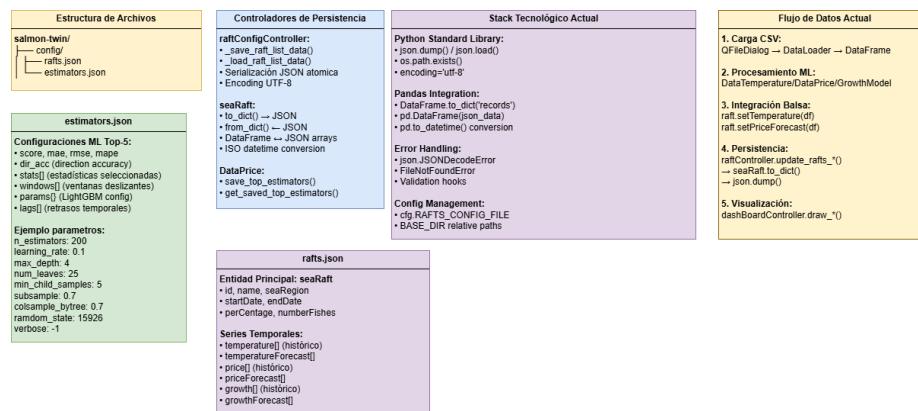


Ilustración 34 Sistema de persistencia actual

5.5.1 Resumen del Diagrama: Sistema de persistencia actual de SalmonTwin

Estructura de Archivos

El sistema organiza la configuración en el directorio “config” con dos archivos JSON principales:

- **rafts.json:** Datos de balsas y series temporales tanto históricas cargadas como las pronosticadas por los modelos.
- **estimators.json:** Configuraciones del modelo de precios.

Almacenamiento de Datos

rafts.json gestiona la entidad principal “seaRaft” que incluye:

- Metadatos básicos: id, nombre, región marina, fechas de inicio-fin, porcentaje que indica la fecha actual, entre fecha inicial y fecha final, y número de peces iniciales.
- Series temporales históricas y de pronóstico para: temperatura, precios y crecimiento

estimators.json almacena las 5 mejores configuraciones de los hiperparámetros del modelo de precios:

- Métricas de rendimiento (score, mae, rmse, mape, dir_acc).
- Parámetros de configuración de LightGBM.
- Estadísticas seleccionadas, ventanas deslizantes y lags (retrasos) temporales.

Controladores de Persistencia

- **raftConfigController**: Maneja serialización JSON atómica con encoding UTF-8.
- **seaRaft**: Proporciona conversión bidireccional JSON-objeto y manejo de DataFrames.
- **DataPrice**: Desde esta clase se gestiona el guardado y recuperación de estimadores principales para el modelo de precios.

5.5.2 Flujo de Datos

1. **Carga:** CSV → DataLoader → DataFrame.
2. **Procesamiento:** Modelos ML especializados por tipo de dato.
3. **Integración:** Asignación de datos procesados a balsas.
4. **Persistencia:** Serialización automática a JSON.
5. **Visualización:** Renderizado en dashboard.

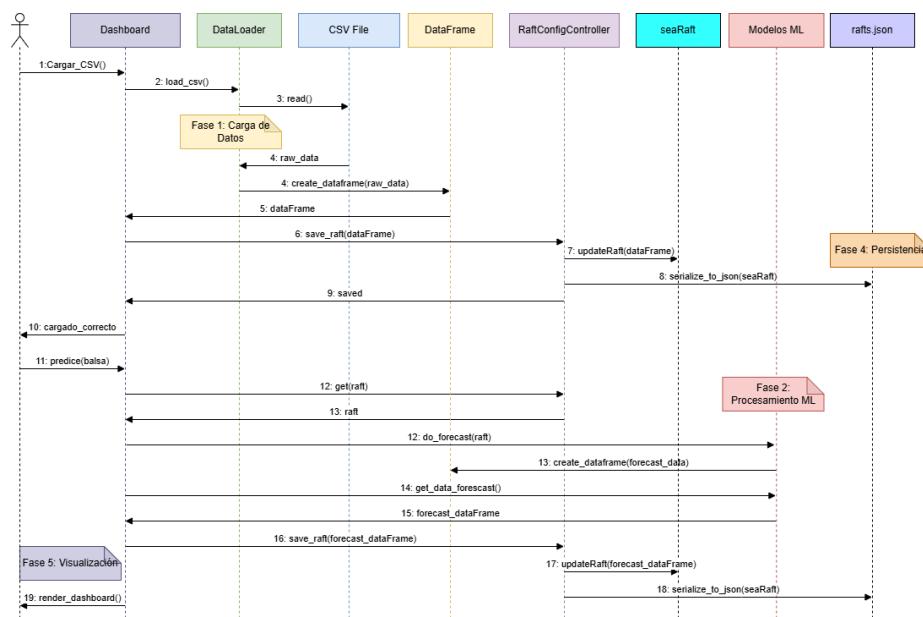


Ilustración 35 Flujo de datos de persistencia

El diagrama de secuencia anterior muestra el flujo completo de datos del sistema SalmonTwin, que se inicia cuando el usuario solicita cargar un archivo CSV a través del dashboard (Cuadro de mandos), el cual delega la operación al “DataLoader” (Clase encargada de leer datos en ficheros, bases de datos, etc) que lee y procesa los datos del archivo CSV creando un DataFrame (Panda) estructurado. Posteriormente, el

dashboard coordina la integración de estos datos con el sistema de balsas mediante el RaftConfigController (Controlador para las balsas), que actualiza la entidad seaRaft (Clase para balsas) y la serializa a formato JSON para persistencia. A continuación, se ejecuta la fase de procesamiento de “Machine Learning” donde el dashboard solicita predicciones a los modelos especializados, los cuales obtienen los datos de la balsa para generar “forecasts” (predicciones) y crear nuevos DataFrames con las predicciones. Finalmente, estas predicciones se integran nuevamente en la balsa a través del RaftConfigController, se serializan al archivo rafts.json, y el dashboard renderiza toda la información actualizada en la interfaz gráfica, completando así un ciclo que transforma datos CSV en predicciones visualizadas mediante una arquitectura MVC que separa claramente las responsabilidades de carga, procesamiento, persistencia y presentación de datos.

5.5.3 Stack Tecnológico

Utiliza la biblioteca estándar de Python (“json”, “os.path”), integración nativa con Pandas para conversión “DataFrame-JSON”, manejo robusto de errores y gestión centralizada de configuración con rutas relativas.

6 Resultados y pruebas

6.1 Introducción

Tras la fase de análisis de requisitos y el diseño detallado de la arquitectura y componentes del sistema "SalmonTwin" en los capítulos anteriores, este capítulo se centra en la verificación y validación del prototipo. El diseño, por muy robusto que sea, es una especificación teórica; las pruebas son el proceso empírico que garantiza que la implementación de software se adhiere a dicho diseño, cumple con los requisitos funcionales y no funcionales, y se comporta de manera predecible y fiable.

El objetivo de esta fase no es solo demostrar que el software "funciona", sino cuantificar su calidad, robustez y corrección. Se ha diseñado un plan de pruebas sistemático que abarca múltiples niveles, desde la validación de los algoritmos de predicción en el nivel más bajo (pruebas unitarias) hasta la verificación de los flujos de trabajo completos del usuario a través de la interfaz gráfica (pruebas de sistema).

Este enfoque estructurado nos permite aislar y corregir defectos de manera eficiente, asegurar que la integración de los distintos módulos (Modelos, Vista, Controlador) sea cohesiva y verificar que la lógica de negocio, especialmente la relacionada con los modelos predictivos y el análisis económico, se ha implementado correctamente. Los resultados de estas pruebas proporcionarán una medida tangible de la calidad del prototipo y su preparación para futuras iteraciones y despliegues.

6.1.1 Tecnología utilizada

Pytest es un "framework" de testing para Python que permite escribir pruebas de software de manera simple y escalable. Se destaca por su sintaxis clara y concisa, eliminando la necesidad de código repetitivo que otros frameworks requieren.

Una de las características principales de Pytest es su sistema de **descubrimiento de pruebas**. Automáticamente encuentra y ejecuta las pruebas en del proyecto, siempre

que se sigan unas convenciones de nombrado sencillas, como nombrar los archivos “test_*.py” o “*_test.py” y las funciones de prueba con el prefijo “test_”.

Pytest utiliza la palabra clave “assert” de Python para verificar los resultados, lo que hace que las aserciones sean muy legibles e intuitivas. Si una aserción falla, Pytest proporciona un informe detallado sobre los valores que causaron el fallo, facilitando enormemente la depuración.

Características Clave

- **Sintaxis Sencilla:** Escribir pruebas es tan fácil como escribir una función que comienza con “test_”. No se requieren clases complejas.

```
def test_thyholdt_function_basic_calculation(self, growth_model):  
    # Arrange - Preparación de datos  
    t_months = 12  
    temperature = 10.0  
    alpha = 7000.0  
  
    # Act - Ejecución de la función  
    result = growth_model._thyholdt_function(t_months, temperature, alpha, beta, mu)  
  
    # Assert - Verificación de resultados  
    assert isinstance(result, (int, float))  
    assert result > 0
```

Ilustración 36 Ejemplo de función de prueba

- **Fixtures:** Permite definir y reutilizar componentes de prueba, como la configuración inicial o la conexión a una base de datos. Las “fixtures” se gestionan de forma modular y eficiente mediante inyección de dependencias.

```
@pytest.fixture
def sample_temperature_data(self):
    """Datos de temperatura histórica"""
    dates = pd.date_range(start='2023-01-01', periods=12, freq='MS')
    return pd.DataFrame({
        'ds': dates,
        'y': [8.5, 8.2, 9.1, 10.5, 12.3, 14.8, 16.2, 15.9, 13.7, 11.4, 9.8, 8.9]
    })
```

Ilustración 37 Ejemplo de Fixture de Pytest

- **Informes Detallados:** En caso de fallo, ofrece una salida muy explícita que ayuda a identificar rápidamente la causa del error.

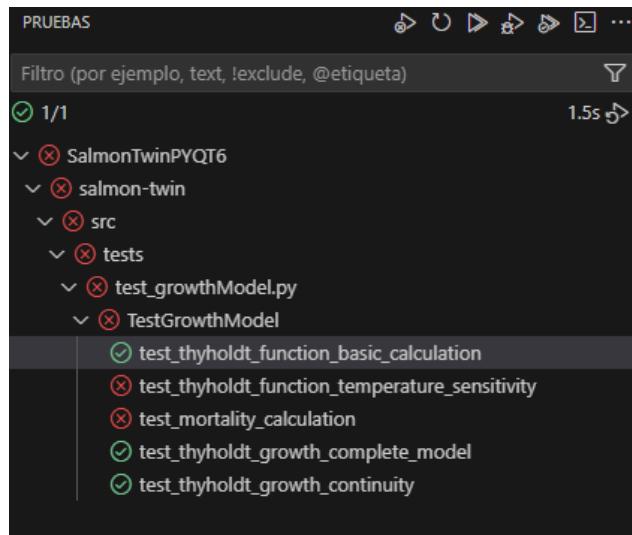


Ilustración 38 Árbol de ejecución de pruebas

```
===== FAILURES =====
____ TestGrowthModel.test_thyholdt_function_temperature_sensitivity ____

self = <test_growthModel.TestGrowthModel object at 0x000001AD01CEB8D0>
growth_model = <model.growthModel.GrowthModel object at 0x000001AD01CF5310>

def test_thyholdt_function_temperature_sensitivity(self, growth_model):
    """
    UT-GM-002: Verificar sensibilidad a cambios de temperatura
    """
    # Arrange
    t_months = 12
    alpha = 7000.0
    beta = 0.02
    mu = 17.0
    temp_low = 5.0
    temp_high = 15.0

    # Act
    growth_low = growth_model._thyholdt_function(t_months, temp_low, alpha, beta, mu)
    growth_high = growth_model._thyholdt_function(t_months, temp_high, alpha, beta, mu)

    # Assert
>   assert growth_high > growth_low # Mayor temperatura = mayor crecimiento
E   assert np.float64(1.2769786666444944) > np.float64(2.642784681587018)
```

Ilustración 39 Ejemplo de mensaje de fallo en la prueba

- **“Plugins”:** Su arquitectura es altamente extensible a través de un vasto ecosistema de “plugins”, que añaden funcionalidades como reportes de cobertura de código (“pytest-cov”) o la ejecución de pruebas en paralelo (“pytest-xdist”).
- **Parametrización:** Facilita la ejecución de una misma prueba con diferentes conjuntos de datos, lo que es ideal para probar múltiples casos límite.

En resumen, Pytest se ha convertido en el estándar de facto para el testing en Python gracias a su simplicidad, potencia y flexibilidad, siendo una herramienta fundamental para desarrolladores que buscan asegurar la calidad y robustez de su código.

6.2 Plan de Pruebas

A continuación, se detalla el plan de pruebas para el prototipo "SalmonTwin". No se detallarán las pruebas falladas ya que se corregirá el software para que se pasen todas las pruebas.

Nomenclatura para ID:

Se forma de la siguiente manera XX-TT-NNN, donde XX:

- UT: Prueba unitaria.
- IT: Prueba integración.
- ST: Prueba de sistema.

Y TT es:

- GM: Modelo de crecimiento.
- DT: Modelo de temperatura.
- DP: Modelo de precios.

Y NNN es un Número correlativo.

Para la función de Thyholdt, en el modelo de crecimiento, se usarán los siguientes parámetros si no se especifican:

- 'alpha': 7000.0, # Peso máximo asintótico (gramos)
- 'beta': 0.018, # Tasa de crecimiento
- 'mu': 19.0 # Punto de inflexión temporal (mes 19)

6.2.1 Resumen de pruebas realizadas

6.2.1.1 *Modelo de crecimiento*

Cobertura General

- **Número de Pruebas:** 13 pruebas.
- **Funciones Principales Probadas:** _thyholdt_function(), _mortality(), thyholdt_growth()
- **Enfoque:** Validación matemática robusta del modelo de crecimiento de salmón.

Pruebas Unitarias (11 tests)

ID	Función Probada	Descripción	Casos Cubiertos
UT-GM-001	thyholdt function()	Cálculo básico de función Thyholdt	Parámetros estándar, verificación de rangos
UT-GM-002	thyholdt function()	Sensibilidad a temperatura	Temperatura baja vs alta (5°C vs 15°C)
UT-GM-003	mortality()	Cálculo de mortalidad exponencial	Fórmula discreta: $N(t) = N_0 \times (1-r)^t$
UT-GM-006	thyholdt function()	Valores cero en parámetros	"Alpha"=0, "beta"≈0, "tiempo"=0, "temperatura"=0
UT-GM-007	thyholdt function()	Valores negativos	Tiempo, "alpha", "beta", "temperatura", "mu" negativos
UT-GM-008	thyholdt function()	Valores extremos	Valores muy grandes/pequeños, "overflow" control.

UT-GM-009	<u>thyholdt function()</u>	Casos extremos matemáticos	Punto de inflexión (mu), “overflow”, valores cercanos
UT-GM-010	<u>thyholdt function()</u>	Manejo NaN/Infinito	NaN en parámetros, infinitos, validación robusta
UT-GM-011	<u>mortality()</u>	Casos extremos mortalidad	Mortalidad 0%, 100%, tiempo=0, población=0
UT-GM-012	<u>thyholdt function()</u>	Validación de tipos	Strings, None, tipos incorrectos
UT-GM-013	<u>thyholdt function()</u>	Condiciones de frontera	Antes/después de mu, temperaturas límite

Pruebas de Integración (2 tests)

ID	Descripción	Componentes Integrados
IT-GM-004	Modelo completo Thyholdt	Temperatura histórica + forecast + crecimiento + mortalidad
IT-GM-005	Continuidad temporal	Verificación de unión entre datos históricos y predicción

Características Destacadas

Robustez Matemática:

- Validación de fórmula:** $W(t) = \alpha / (1 + e^{(-\beta \times T \times (t - \mu))})$
- Casos extremos:** “NaN”, infinitos, “overflow”, “underflow”

- **Validación de tipos:** Entrada incorrecta → “None”
- **Rangos físicos:** Peso $\leq \alpha$, crecimiento ≥ 0

Cobertura de Escenarios:

- **Temperaturas:** -5°C a 30°C, “NaN”, infinito
- **Tiempo:** 0 a 1000 meses, negativo, punto de inflexión
- **Parámetros:** Valores realistas y extremos
- **Mortalidad:** 0% a 100%, poblaciones grandes

6.2.1.2 *Modelo de temperatura*

Cobertura General

- Número de Pruebas: 19 pruebas
- Funciones Principales: parseTemperature(), getTemperatureData(), fitTempData()
- Enfoque: Procesamiento de datos de temperatura marina y predicción con Prophet

Pruebas Unitarias (7 tests)

ID	Función Probada	Descripción	Validaciones
UT-DT-001	Inicialización	Verificación de atributos y estructura inicial	hasattr(), DataFrames vacíos
UT-DT-002	Inicialización	DataFrames vacíos al inicio	.empty = True

UT-DT-005	<u>parseTemperature()</u>	Columnas faltantes	Error registrado, DataFrames vacíos
UT-DT-006	<u>parseTemperature()</u>	Datos vacíos	result = False, error registrado
UT-DT-007	<u>parseTemperature()</u>	Fechas malformadas	Rechazo de meses inválidos
UT-DT-009	<u>getTemperatureData()</u>	Región inválida	return None, error registrado
UT-DT-013	<u>fitTempData()</u>	Error en Prophet	Manejo de ValueError
UT-DT-014	<u>fitTempData()</u>	Entrada inválida	DataFrame vacío/malformado

Pruebas de Integración (10 tests)

ID	Flujo Probado	Componentes	Verificaciones
IT-DT-003	Parseo completo	Validación + conversión + almacenamiento	Estructura, tipos, contenido
IT-DT-004	Validación contenido	"Parseo" + verificación de valores	Rangos 0-30°C, fechas correctas
IT-DT-008	Obtención de datos	"Parseo" + get para región válida	DataFrame con 'ds', 'y'

IT-DT-010	Todas las regiones	Parse + get para 9 regiones	Datos disponibles para todas
IT-DT-011	Predicción Prophet	Mock Prophet + configuración	yearly_seasonality=2, predicción
IT-DT-012	Clipping temperaturas	Prophet + limitación 0-30°C	Valores extremos controlados
IT-DT-015	Flujo completo	Parse → get → validación	Workflow end-to-end
IT-DT-016	Persistencia errores	Múltiples operaciones + errores	Estado de error consistente
IT-DT-019	Registro único	Parse + validación de un registro	Fecha correcta, contenido válido

Pruebas de Sistema (2 tests)

ID	Descripción	Métricas	Validaciones
ST-DT-017	Rendimiento dataset grande	1000 registros, < 5 segundos	Procesamiento masivo
ST-DT-018	Valores extremos	Temperaturas -6°C a 30°C	Robustez con datos límite

Pedro López Treitiño

Características Destacadas

Procesamiento de Datos:

- **9 regiones noruegas:** Finnmark, Troms, Nordland, etc.
- **Conversión temporal:** Año+Mes → datetime64[ns]
- **Validación de estructura:** Columnas requeridas, tipos correctos
- **Manejo de errores:** Error estado consistente

Integración con Prophet:

- **Configuración automática:** yearly_seasonality=2
- **“Clipping” de temperaturas:** Rango 0-30°C realista
- **Manejo de excepciones:** Prophet errors → None + error log
- **Formato de salida:** DataFrame con yhat, yhat_lower, yhat_upper

Escalabilidad y Robustez:

- **Dataset grande:** 1000 registros en < 5 segundos
- **Valores extremos:** -6°C a 30°C sin fallas
- **Recuperación de errores:** Estado limpio después de fallos
- **“Memory management”:** DataFrames independientes por región

6.2.1.3 *Modelo de precios*

Pruebas Unitarias (17 tests):

- **UT-DP-001 a UT-DP-017:** Prueban funcionalidades específicas aisladas
- Inicialización, validaciones, manejo de errores individuales
- Getters/setters simples
- Casos de borde específicos

Pruebas de Integración (6 tests):

- **IT-DP-001 a IT-DP-006:** Prueban flujos completos
- Parseo + suavizado + optimización
- Integración con bibliotecas externas (LGBMRegressor, ForecasterRecursive)
- Flujos de trabajo end-to-end

Pruebas de Sistema (7 tests):

- **ST-DP-001 a ST-DP-007:** Prueban rendimiento y robustez
- Datasets grandes, uso de memoria
- Valores extremos, condiciones límite
- Concurrencia, escalabilidad, recuperación de errores

Características Destacadas

Cobertura Completa:

- **Funciones públicas:** parsePrice(), smoothPriceMonthly(), fit_price()

Pedro López Treitiño

- **Funciones de optimización:** prepare_data_for_optimization(), run_parameter_optimization()
- **Gestión de archivos:** save_top_estimators(), get_saved_top_estimators()
- **Getters/setters:** getPriceData(), setPriceData()

Casos de Error:

- Datos sin columnas requeridas
- Formatos inválidos (año, semana, precio)
- Archivos JSON corruptos
- Falta de datos para optimización

Casos Límite:

- Datasets grandes (rendimiento)
- Valores extremos de precios
- Fechas límite (años 1900-2100)
- Acceso concurrente

Mocking Apropriado:

- **Unitarias:** Minimal mocking, enfoque en lógica
- **Integración:** Mock de dependencias externas (LGBMRegressor, etc.)
- **Sistema:** Mock para pruebas de escalabilidad controladas

Total: 30 pruebas que cubren la funcionalidad principal, desde validaciones básicas hasta rendimiento del sistema completo.

6.2.2 Detalle de pruebas realizadas

El objetivo es probar las unidades de código más pequeñas (clases y métodos) de forma aislada para validar su lógica interna, pruebas unitarias, de sistema para validar varios componentes y de sistema para hacer pruebas de rendimiento. La siguiente tabla muestra las pruebas unitarias realizadas:

ID	Descripción	Resultado Esperado	Resultado Obtenido
UT-GM-001	Validar cálculo básico de la función Thyholdt.	No debe exceder el peso máximo asintótico. Se ha usado un periodo de 6 años 72 meses. # Parámetros de la función Thyholdt temperature=10.0 alpha = 7000.0 beta = 0.02 mu = 17.0	No ha excedido el peso máximo asintótico.
UT-GM-002	Verificar sensibilidad a cambios de temperatura en Thyholdt	A mayor temperatura mayor crecimiento. Se usará temperatura 5,0 como baja y 15,0 como alta.	A mayor temperatura ha habido mayor crecimiento

Pedro López Treitiño

UT-GM-003	Verificar cálculo de mortandad exponencial	<p>Debe haber mortalidad, pero algunos deben sobrevivir.</p> <p>Debe verificarse la fórmula de mortandad:</p> $N(t) = N_0 \times (1 - r)^t$ <p>modelo discreto</p>	<p>Hubo mortandad y algunos han sobrevivido.</p> <p>Formula verificada para 1,5% mensual y 12 meses.</p>
IT-GM-004	<p>Verificar modelo completo de crecimiento Thyholdt</p>	<ul style="list-style-type: none"> • Datos entrada: <ul style="list-style-type: none"> ◦ alpha = 7000.0 ◦ beta = 0.02 ◦ mu = 17.0 ◦ mortality_rate = 0.015 ◦ initial_weight = 100.0 ◦ initial_number_fishes = 1000 • Verificar estructuras de datos de salida no son vacías: <ul style="list-style-type: none"> ◦ historical_growth ◦ forecast_growth • Verificar columnas requeridas: <ul style="list-style-type: none"> ◦ ['ds', 'function'] 	<ul style="list-style-type: none"> • estructuras de datos de salida no son vacías • columnas requeridas presente. • Valores lógicos mayores que cero. • Tendencia creciente

		<pre>'growth', 'number_fishes', 'biomass'] • Verificar valores lógicos ○ historical_growth['growth'] >= 0 ○ historical_growth['biomass'] >= 0 ○ historical_growth['number_fishes'] > 0) • Verificar tendencia de crecimiento ○ Creciente.</pre>	
IT-GM-005	Verificar continuidad entre datos históricos y de predicción	Se debe verificar que el punto de unión del último punto de la serie histórica coincide con el primer punto de la predicción en cuanto a fechas.	Coinciden los puntos de unión.
UT-GM-006	Verificar manejo de valores cero en parámetros	Con alpha=0, el crecimiento debería ser cero y con beta=0 el crecimiento debería ser	Con alpha=0, el crecimiento es cero y con beta=0 el

		mínimo y con temperatura cero debería ser muy bajo.	crecimiento es mínimo y con temperatura cero es muy bajo.
UT-GM-007	Verificar manejo de valores negativos	Verificar que si tiempo negativo, Alpha, beta, temperatura y mu negativo. El crecimiento es no negativo.	Comprobado que si tiempo negativo, Alpha, beta, temperatura y mu negativo. El crecimiento es no negativo.
UT-GM-008	Verificar manejo de valores extremos	Verificar funcionamiento con valores muy grandes y valores muy pequeños. El resultado no devuelve valores NaN, ni infinitos ni crecimientos negativos.	Verificado funcionamiento con valores muy grandes y valores muy pequeños. El resultado no devuelve valores NaN, ni infinitos ni crecimientos negativos.
UT-GM-009	Verificar casos extremos matemáticos	Verificar manejo de “overflow” correctamente, Debe controlar adecuadamente el punto de inflexión (tiempo igual a mu y valores cercanos).	Se verifica manejo de “overflow” correctamente, Se controla adecuadamente el punto de inflexión (tiempo igual a mu y valores cercanos).
UT-GM-010	Verificar manejo de NaN e infinitos	Verificar Resultado con NaN en tiempo, NaN en	Verificado Resultado con NaN en tiempo, NaN en

		<p>temperatura, NaN en Alpha, debe ser cero.</p> <p>Verificar infinito en tiempo debe ser igual a Alpha e infinito en temperatura debe ser cero.</p>	<p>temperatura, NaN en Alpha, debe ser cero.</p> <p>Verificado infinito en tiempo debe ser igual a Alpha e infinito en temperatura debe ser cero.</p>
UT-GM-011	Verificar casos extremos en función de mortalidad	<p>Verificar:</p> <ul style="list-style-type: none"> • Sin mortalidad, población debe mantenerse. • Con mortalidad del 100%, no debe haber peces sobrevivientes. • Sin tiempo, no hay mortalidad. • Sin población inicial, resultado debe ser cero. • Valores negativos en la entrada debe devolver cero en la salida. 	<p>Verificado:</p> <ul style="list-style-type: none"> • Sin mortalidad, población debe mantenerse. • Con mortalidad del 100%, no debe haber peces sobrevivientes. • Sin tiempo, no hay mortalidad. • Sin población inicial, resultado debe ser cero. <p>Valores negativos en la entrada debe devolver cero en la salida.</p>
UT-GM-012	Verificar validación de tipos de entrada	Verificar que valores de entrada que no sean	Verificado que valores de entrada que no sean números

		números devuelven “None” como salida.	devuelven “None” como salida.
UT-GM-013	Verificar condiciones de frontera específicas	Verificar valores cerca del punto de inflexión y manejo de temperaturas altas y bajas.	Verificado valores cerca del punto de inflexión y manejo de temperaturas altas y bajas.
UT-DT-001	Verificar inicialización correcta de DataTemperatur e	Verificar: <ul style="list-style-type: none"> • Que los atributos se inicializan correctamente • data_regions tiene la estructura correcta • data_regions_forecast tiene la estructura correcta 	Verificado: <ul style="list-style-type: none"> • Que los atributos se inicializan correctamente • data_regions tiene la estructura correcta data_regions_forecast tiene la estructura correcta
UT-DT-002	Verificar que los DataFrames se inicializan vacíos	Todos los DataFrames deben estar vacíos al inicio	Todos los DataFrames están vacíos al inicio
IT-DT-003	Verificar parseo (Procesado) correcto de datos válidos	Se deben poder “parsear” (Procesar) correctamente los datos de ejemplo:	Se han procesado correctamente los datos de ejemplo.

		<pre>def test_fisiche(): """Prueba la función temperatura_data(self) que recibe un DataFrame con datos de temperatura de ejemplo valídos""" return self.assertEqual(temperatura_data(self), pd.DataFrame([{'Year': 2003, 'Month': 1, 'Day': 1, 'Temp': 10}, {'Year': 2003, 'Month': 1, 'Day': 2, 'Temp': 11}, {'Year': 2003, 'Month': 1, 'Day': 3, 'Temp': 12}, {"Year": 2003, "Month": 1, "Day": 4, "Temp": 13}, {"Year": 2003, "Month": 1, "Day": 5, "Temp": 14}, {"Year": 2003, "Month": 1, "Day": 6, "Temp": 15}, {"Year": 2003, "Month": 1, "Day": 7, "Temp": 16}, {"Year": 2003, "Month": 1, "Day": 8, "Temp": 17}, {"Year": 2003, "Month": 1, "Day": 9, "Temp": 18}, {"Year": 2003, "Month": 1, "Day": 10, "Temp": 19}, {"Year": 2003, "Month": 1, "Day": 11, "Temp": 20}, {"Year": 2003, "Month": 1, "Day": 12, "Temp": 21}, {"Year": 2003, "Month": 1, "Day": 13, "Temp": 22}, {"Year": 2003, "Month": 1, "Day": 14, "Temp": 23}, {"Year": 2003, "Month": 1, "Day": 15, "Temp": 24}, {"Year": 2003, "Month": 1, "Day": 16, "Temp": 25}, {"Year": 2003, "Month": 1, "Day": 17, "Temp": 26}, {"Year": 2003, "Month": 1, "Day": 18, "Temp": 27}, {"Year": 2003, "Month": 1, "Day": 19, "Temp": 28}, {"Year": 2003, "Month": 1, "Day": 20, "Temp": 29}, {"Year": 2003, "Month": 1, "Day": 21, "Temp": 30}, {"Year": 2003, "Month": 1, "Day": 22, "Temp": 31}, {"Year": 2003, "Month": 1, "Day": 23, "Temp": 32}, {"Year": 2003, "Month": 1, "Day": 24, "Temp": 33}, {"Year": 2003, "Month": 1, "Day": 25, "Temp": 34}, {"Year": 2003, "Month": 1, "Day": 26, "Temp": 35}, {"Year": 2003, "Month": 1, "Day": 27, "Temp": 36}, {"Year": 2003, "Month": 1, "Day": 28, "Temp": 37}, {"Year": 2003, "Month": 1, "Day": 29, "Temp": 38}, {"Year": 2003, "Month": 1, "Day": 30, "Temp": 39}, {"Year": 2003, "Month": 1, "Day": 31, "Temp": 40}], columns=['Year', 'Month', 'Day', 'Temp'])</pre>	
IT-DT-004	Verificar que el contenido de los datos se parsea correctamente	<p>Verificar que fecha:'2023-01' está en la primera fila.</p> <p>Verificar que las temperaturas están entre 0°C y 30°C.</p>	<p>Verificado que fecha:'2023-01' está en la primera fila.</p> <p>Verificado que las temperaturas están entre 0°C y 30°C.</p>
UT-DT-005	Verificar manejo de datos con columnas faltantes	<p>Verificar:</p> <ul style="list-style-type: none"> • El parseo de datos inválidos debe retornar False • Debe haber un error registrado • El error debe tener formato 'Error:...' • Los DataFrames deben seguir vacíos tras error • 	<p>Verificado:</p> <ul style="list-style-type: none"> • El parseo de datos inválidos debe retornar False • Debe haber un error registrado • El error debe tener formato 'Error:...' • Los DataFrames deben seguir vacíos tras error •
UT-DT-006	Verificar manejo de datos vacíos	El parseo de datos vacíos debe retornar error.	El parseo de datos vacíos retorna error.

<i>UT-DT-007</i>	Verificar manejo de fechas malformadas	Datos con fechas malformadas deben ser rechazados	Datos con fechas malformadas se rechazan y devuelven error.
<i>IT-DT-008</i>	Verificar obtención de datos para región válida.	Debe retornar datos para región válida.	Se retornan datos para región válida.
<i>UT-DT-009</i>	Verificar manejo de región inválida	No se deben retornar datos para región inválida y se devolverá mensaje de error.	No retornan datos para región inválida y se devuelve mensaje de error.
<i>IT-DT-010</i>	Verificar que se pueden obtener datos de todas las regiones	Se deben poder obtener todas las regiones disponibles.	Se obtienen todas las regiones disponibles.
<i>IT-DT-011</i>	Simula predicción exitosa con "Prophet" usando mocks	Resultado no nulo con "DataFrame" válido, 10 períodos y "Prophet" configurado.	Resultado no nulo con "DataFrame" válido, 10 períodos y "Prophet" configurado con "yearly_seasonality=2".
<i>IT-DT-012</i>	Verificar que las temperaturas se	Límites aplicados:	$yhat \geq 0 \text{ y } \leq 30$

	limitan entre 0°C y 30°C.	Temperaturas >= 0°C y Temperaturas <= 30°C.	yhat_lower >= 0 y <= 30 yhat_upper >= 0 y <= 30
UT-DT-013	Verificar manejo de errores en Prophet	Debería devolver None cuando hay error en Prophet y el error.	Retornó None cuando hay error en Prophet y el error.
UT-DT-014	Verificar manejo de datos de entrada inválidos	DataFrame vacío de entrada y datos de entrada mal formados deben devolver None.	DataFrame vacío de entrada y datos de entrada mal formados devolvió None.
IT-DT-015	Verificar flujo completo de trabajo	Verificar parseo exitoso, datos obtenidos y estructura válida.	Verificado parseo exitoso, datos obtenidos y estructura válida.
IT-DT-016	Verifica que errores se almacenan y persisten entre operaciones	Verificar que operaciones fallan y los errores persisten.	Verificado que operaciones fallan y los errores persisten.
ST-DT-017	Verificar rendimiento con datasets grandes	Debe procesar datasets de 1000 registros en menos de 5 segundos.	Se procesó datasets de 1000 registros en menos de 5 segundos

ST-DT-018	Verificar manejo de valores extremos de temperatura	Debe manejar temperaturas extremas válidas.	Se manejaron temperaturas extremas válidas.
IT-DT-019	Verificar procesamiento de un solo registro	Debe procesar un solo registro correctamente	Se procesa un solo registro correctamente.
UT-DP-001	Verificar inicialización correcta de DataPrice	Verificar que los atributos se inicializan correctamente.	Verificado que los atributos se inicializan correctamente.
UT-DP-002	Verificar validación de columnas requeridas	Debe devolver "True" para datos con columnas válidas y debe registrar error.	Devuelve "True" para datos con columnas válidas y debe registrar error.
UT-DP-003	Verificar manejo de columnas faltantes	Debe devolver "False" para columnas inválidas y debe registrar error	Devuelve "False" para columnas inválidas y debe registrar error

UT-DP-004	Verificar manejo de formato de año inválido	Debe fallar con año inválido y debe registrar error.	Falla con año inválido y registra error.
UT-DP-005	Verificar manejo de formato de semana inválida	Debe fallar con semana inválida y debe registrar error.	Falla con semana inválida y registrar error.
UT-DP-006	Verificar manejo de formato de precio inválido	Debe fallar con precio inválido y debe registrar error.	Falla con precio inválido y registra error.
UT-DP-007	Verificar manejo de suavizado sin datos	Debe fallar sin datos	Falla sin datos
UT-DP-008	Verificar obtención de datos vacíos	Debe devolver “None” cuando no hay datos	Devuelve “None” cuando no hay datos

<i>UT-DP-009</i>	Verificar obtención de predicción vacía	Debe devolver “None” cuando no hay predicción	Devuelve “None” cuando no hay predicción
<i>UT-DP-010</i>	Verificar establecimiento de datos	Debe devolver datos establecidos.	Devuelve datos establecidos.
<i>UT-DP-011</i>	Verificar preparación de datos sin datos iniciales.	Debe devolver “None” sin datos.	Devuelve “None” sin datos.
<i>UT-DP-012</i>	Verificar optimización sin datos preparados	Debe devolver “None” sin datos	Devuelve “None” sin datos
<i>UT-DP-013</i>	Verificar entrenamiento sin datos	Debe devolver “None” sin datos	Devuelve “None” sin datos
<i>UT-DP-014</i>	Verificar entrenamiento sin resultados de optimización	Debe devolver “None” sin resultados de optimización.	Devuelve “None” sin resultados de optimización.

<i>UT-DP-015</i>	Verificar obtención de estimadores cuando no existe archivo	Debe devolver lista vacía cuando no existe archivo y no debe haber error.	Devuelve lista vacía cuando no existe archivo y no hay error.
<i>UT-DP-016</i>	Verificar manejo de JSON inválido en estimadores	Debe devolver lista vacía con JSON inválido	Devuelve lista vacía con JSON inválido
<i>UT-DP-017</i>	Verificar ajuste de precio sin datos	Debe devolver “False” sin datos	Devuelve “False” sin datos
<i>IT-DP-001</i>	Verificar flujo completo de “parseo” de precios	El parseo debe ser exitoso	El parseo fue exitoso.
<i>IT-DP-002</i>	Verificar flujo de “parseo” y suavizado	El “parseo” y el suavizado debe ser exitoso.	El “parseo” y el suavizado son exitosos.

IT-DP-003	Verificar preparación completa de datos para optimización	Debe tener al menos 6 meses de datos y se deben preparar exitosamente.	Tiene al menos 6 meses de datos y se prepararon exitosamente.
IT-DP-004	Verificar integración de optimización de parámetros	Debe devolver resultados de optimización y debe guardar resultados	Devolvió resultados de optimización y guardó resultados
IT-DP-005	Verificar guardado completo de estimadores	Debe guardar una lista de 5 estimadores que deben tener "score".	Guarda una lista de 5 estimadores que tienen "score".
IT-DP-006	Verificar flujo completo de ajuste de precios	Debe ajustar modelo exitosamente y debe generar predicción.	Ajusta modelo exitosamente y genera predicción.
ST-DP-001	Verificar rendimiento con dataset grande	Debe procesar dataset grande exitosamente en menos de 3 segundos.	Procesado dataset grande exitosamente en menos de 3 segundos.

ST-DP-002	Verificar uso eficiente de memoria	Crecimiento de memoria debe ser razonable (< 10MB)	Crecimiento de memoria es razonable (< 10MB)
ST-DP-003	Verificar robustez con valores extremos de precios	Debe tratar valores extremos correctamente	Trata valores extremos correctamente
ST-DP-004	Verificar manejo de condiciones límite de fechas	Debe tratar fechas límite correctamente	Tratar fechas límite correctamente
ST-DP-005	Verificar seguridad en acceso concurrente	No debe haber errores de concurrencia	No hubo errores de concurrencia
ST-DP-006	Verificar escalabilidad de optimización	La escalabilidad debe ser aproximadamente lineal ($1000/100 = 10x$)	La escalabilidad es aproximadamente lineal ($1000/100 = 10x$)
ST-DP-007	Verificar recuperación	Debe tener datos válidos después de recuperación	Tiene datos válidos después de recuperación

Pedro López Treitiño

después de errores		
--------------------	--	--



UNIVERSIDAD DE OVIEDO

Escuela Politécnica de Ingeniería de Gijón

Hoja 139 de 142

6.2.3 Pruebas de usuario desde UI de la aplicación.

Pedro López Treitiño

7 Referencias

- [1] Gelernter, David (1991). *Mirror Worlds: or the Day Software Puts the Universe in a Shoebox...How It Will Happen and What It Will Mean.* Oxford University Press. ISBN 9780195344851– via Google Books.
- [2] 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 23 April 2012-26 April 2012 Honolulu, Hawaii <https://doi.org/10.2514/6.2012-1818>. *Structures, Structural Dynamics, and Materials Conference*
- [3] [Digital Twins and Living Models at NASA - NASA Technical Reports Server \(NTRS\)](#)
- [4] [What Is a Digital Twin? | IBM](#)
- [5] A. Udugama, Isuru & Öner, Merve & Cabaneros, Pau & Beenfeldt, Christian & Bayer, Christoph & Huusom, Jakob & Gernaey, Krist & Sin, Gürkan. (2021). Towards Digitalization in Bio-Manufacturing Operations: A Survey on Application of Big Data and Digital Twin Concepts in Denmark. *Frontiers in Chemical Engineering*. 3. 10.3389/fceng.2021.727152.
- [6] Gomes Alves, Rafael & Souza, Gilberto & Maia, Rodrigo & Tran, Anh & Kamienski, Carlos & Soininen, Juha-Pekka & Aquino Junior, Plínio & Lima, F. (2019). A Digital Twin for Smart Farming. 10.1109/GHTC46095.2019.9033075.
- [7] C. Kamienski et al., "SWAMP: an IoT-based Smart Water Management Platform for Precision Irrigation in Agriculture," 2018 Global Internet of Things Summit (GloTS), Bilbao, Spain, 2018, pp. 1-6, doi: 10.1109/GIOTS.2018.8534541. keywords: {Irrigation;Internet of Things;Water resources;Computer architecture;Sensors;Meteorology;Internet of Things;Smart Water Management;Precision Irrigation},
- [8] [Fiware-Orion](#)
- [9] [Introduction to MongoDB — MongoDB Manual](#)
- [10] [Home - fiware-draco](#)
- [11] [NGSI-LD Smart Farm Tutorials \(ngsi-ld-tutorials.readthedocs.io\)](#)
- [12] Díaz Fernández, R. (2019). Agente IoT de FIWARE para el control de un dispositivo de monitorización de actividad física. (Trabajo Fin de Grado Inédito). Universidad de Sevilla, Sevilla. <https://hdl.handle.net/11441/91376>
- [13] [MySQL :: MySQL Documentation](#)

Con formato: Español (España)

Con formato: Español (España)

Con formato: Español (España)

Pedro López Treitiño

[14] [Grafana | Query, visualize, alerting observability platform](#)

Con formato: Español (España)

[15] Lima AC, Royer E, Bolzonella M and Pastres R. Digital twins for land-based aquaculture: A case study for rainbow trout (*Oncorhynchus mykiss*) [version 2; peer review: 2 approved]. Open Res Europe 2023, 2:16 (<https://doi.org/10.12688/openreseurope.14145.2>)

Con formato: Español (España)

[16] [Operational technology - Wikipedia](#)

[17] [Qué son las piscifactorías - Acuario de Lanzarote \(aquariumlanzarote.com\)](#)

[18] [8. Pescado | OCDE-FAO Perspectivas Agrícolas 2021-2030 | OECD iLibrary \(oecd-ilibrary.org\)](#)

[19] Matteo Bolzonella, Adriano Coutinho de Lima, Edouard Royer, & Roberto Pastres. (2021). adrianocl/DTRT: Mathematical Model - Digital Twin for Rainbow Trout (*Oncorhynchus mykiss*) land-based aquaculture (v1.0.0). Zenodo. <https://doi.org/10.5281/zenodo.5638971>

[20] [RPubs - Descomposición de Series Temporales \(Modelo Aditivo y Multiplicativo Clásicos\)](#)

[21] Taylor SJ, Letham B. 2017. Forecasting at scale. PeerJ. Preprints 5:e3190v2 <https://doi.org/10.7287/peerj.preprints.3190v2>

[22] [Forecasting with XGBoost and LightGBM - Skforecast Docs](#)

[23] [Features — LightGBM 4.6.0.99 documentation](#)

[24] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree." *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pp. 3149-3157.

[25] Ranka, Sanjay, and V. Singh. "CLOUDS: A decision tree classifier for large datasets." Proceedings of the 4th Knowledge Discovery and Data Mining Conference. 1998.

[26] Machado, F. P. "Communication and memory efficient parallel decision tree construction." (2003).

[27] Li, Ping, Qiang Wu, and Christopher J. Burges. "Mcrank: Learning to rank using multiple classification and gradient boosting." *Advances in Neural Information Processing Systems 20 (NIPS 2007)*.

[28] Thyholdt, S. B. (2014). "The Importance of Temperature in Farmed Salmon Growth: Regional Growth Functions for Norwegian Farmed Salmon." *Marine Resource Economics*, 29(4), 339-350.

[29] Brækkan, E. H., & Thyholdt, S. B. (2014). "The Bumpy Road of Demand Growth — An Application to Atlantic Salmon." *Marine Resource Economics*, 29(4), 339-350.

[30] Wang, C.-D., & Olsen, Y. (2023). "Quantifying regional feed utilization, production and nutrient waste emission of Norwegian salmon cage aquaculture." *Aquaculture Environment Interactions*, 15, 231-249.

[31] Tsoularis, A. (2025). "Analysis of Logistic Growth Models." Massey University.

[32] [Calendar Calculations](#)

[33] **Estudios Noruegos / Granjas Suecas/Finlandesas.** No es una referencia específica. Representa informes técnicos de institutos como el [Instituto Noruego de Investigación Marina](#) o el [Instituto de Recursos Naturales de Finlandia \(Luke\)](#).

[34] **Organización de las Naciones Unidas para la Alimentación y la Agricultura (FAO).** [FAO - Pesca y Acuicultura](#)

[35] **Consejo Internacional para la Exploración del Mar (ICES).** [Página Principal de ICES](#)

[36] **Centro de Desarrollo de la Pesca en el Sudeste Asiático (SEAFDEC).** [Página Principal de SEAFDEC](#)

[37] **García-Launay, F., et al. (2014).** A full life cycle and spatially explicit model to evaluate the effects of farming practices on sea bass (*labrax*) production. *Aquaculture*.

[38] [DOI: 10.1016/j.aquaculture.2014.07.032](#) **Assefa, A., et al. (2019).** Genetic parameters for growth and survival of Atlantic salmon (*Salmo salar*) in a nucleus breeding population in Norway. *Aquaculture*.

[39] [DOI: 10.1016/j.aquaculture.2018.12.046](#) **Araneda, M., et al. (2008).** White shrimp *Penaeus vannamei* culture in freshwater at three densities: condition, growth and survival. *Aquaculture*.

[40] [DOI: 10.1016/j.aquaculture.2008.06.006](#) **Karakaplan, S., & Dikel, S. (2020).** Growth performance of Nile

[41] Tilapia (*Oreochromis niloticus*) in a biofloc system. *Journal of Applied Animal Research*. [DOI: 10.1080/09712119.2020.1749870](#)

[37][38][39][40][41]