Propuesta de proyecto

Universidad Europea - https://universidadeuropea.com/

Sistema de Soporte a la Decisión (DSS) para la Inferencia de Vertidos en Ríos, basado en Simulación HPC, DeepONet y API de Servicio

Duo Xu Ye, Andrei Ionut Hrisca, Héctor Omar Hernández Rodríguez, Roberto García Casado, (Tadeo Adrián Troncoso Taraborreli y Víctor Pablo Velayos Velayos)

Universidad Europea de Madrid, C/Tajo, s/n, Villaviciosa de Odón, Madrid 28670, España

INFORMACIÓN

Palabras clave:

Problema Inverso (Inverse Problem)

DeepONet (Deep Operator Network)

Gemelo Digital (Digital Twin)

Detección De Vertidos

Problemas Mal Condicionados (Ill-Posed Problems)

Simulación Hidrodinámica

Computación De Alto Rendimiento (HPC)

CUDA/GPGPU

Aprendizaje De Operadores (Operator Learning)

Sistema De Soporte A La Decisión (DSS)

JavaFX

API REST

Docker

FastAPI

Java/JNI

RESUMEN

La presente propuesta describe un proyecto dual que integra Computación de Alto Rendimiento (HPC) e Ingeniería de Software (Informática) para crear un sistema de soporte a la decisión (DSS) capaz de identificar y localizar eventos de vertido no autorizados en sistemas fluviales.

El núcleo de Computación (OE1-OE4) se centra en resolver el problema inverso, intrínsecamente mal condicionado (32 sensores para +1800 fuentes). Para ello, se desarrollará un "gemelo digital" (simulador hidrodinámico C++/CUDA) capaz de generar un dataset sintético a gran escala (>100.000 muestras). Este dataset entrenará una Red de Operador Profundo (DeepONet) para que aprenda el operador inverso: mapear la respuesta temporal de los sensores a un "mapa de calor" espacial que identifique el origen del vertido.

El núcleo de Informática (OE5-OE6) consiste en construir el producto de software que consume esta tecnología. Se desarrollará una API REST de alto rendimiento (Python/FastAPI, dockerizada) que servirá como backend único, exponiendo las predicciones del DeepONet y la gestión del simulador. Esta API será consumida por una aplicación de escritorio nativa (JavaFX), el DSS final para el usuario. Esta aplicación permitirá a los operadores monitorizar sensores en tiempo real, recibir alertas, visualizar los mapas de calor de predicción y, a los administradores, configurar nuevos escenarios de simulación.

Duo Xu Ye – duo@pemail.es

Revisada el 9 de noviembre 2025

Índices

Índice de contenidos

Objetivo General y Objetivos Específicos

Objetivo General y Objetivos Específicos

Objetivo general

Desarrollar e implementar un modelo de aprendizaje profundo, basado en una arquitectura DeepONet (Red de Operador Profundo), capaz de resolver el problema inverso de la detección de vertidos en sistemas fluviales. El sistema inferirá un "mapa de calor" de la localización e intensidad de un evento de vertido en tiempo casi real, utilizando como única entrada los datos de series temporales de la red de sensores de calidad del agua existente (32 estaciones).

Objetivos específicos

OE1: Desarrollo del Gemelo Digital. Construir un simulador de alto rendimiento que acople un modelo hidrodinámico (basado en la ecuación de Manning) con un modelo de transporte de contaminantes (advección-difusión-reacción). La arquitectura combinará Java (orquestación), C++ (lógica nativa) y CUDA (paralelización de cómputo en GPU) vía JNI.

OE2: Adquisición y Análisis de Datos para Calibración. Obtener y procesar datos reales de la Confederación Hidrográfica del Tajo (CHT), incluyendo series temporales de calidad del agua (SAICA) y aforos (SAIH) mediante web scraping, y datos geoespaciales del Censo de Vertidos Autorizados (+1800 puntos). Estos datos se analizarán para calibrar el estado base del gemelo digital.

OE3: Generación de Dataset Sintético. Utilizar el gemelo digital acelerado por GPU para generar un dataset sintético a gran escala (target: >100.000 muestras). Cada muestra consistirá en un par: (a) la función de respuesta temporal de los 32 sensores y (b) la función espacial del vertido que la originó.

OE4: Entrenamiento y Validación del DeepONet. Diseñar, implementar y entrenar el modelo DeepONet sobre el dataset sintético. El modelo aprenderá el mapeo del operador inverso (de lecturas de sensor a mapa de vertido). Se validará su precisión y capacidad de generalización contra un conjunto de prueba reservado.

OE5: Desarrollo de la Aplicación de Escritorio (Proyecto de Informática). Desarrollar una aplicación de escritorio nativa multiplataforma utilizando JavaFX (OpenJFX 25). Esta aplicación servirá como la interfaz de usuario principal para los operadores y gestores de cuenca.

La aplicación no tendrá lógica de simulación, sino que consumirá los datos y resultados de una API REST centralizada (ver OE6). Sus funciones principales incluirán:

- Visualización de datos de sensores en tiempo real.
- Presentación del estado actual del río.
- Sistema de alertas y visualización de histórico de avisos.
- Renderización de los "mapas de calor" (predicciones) generados por el modelo DeepONet.
- Interacción (configuración y lanzamiento) con el simulador a través de la API.

OE6: Definición y Despliegue de la API de Inferencia y Simulación. Diseñar, implementar y desplegar una API REST de alto rendimiento (usando Python/FastAPI) que actúe como *endpoint* único para el "Proyecto de Informática" (OE5).

Arquitectura y Despliegue: Toda la infraestructura de *backend* (la API de FastAPI, la base de datos como PostgreSQL, etc.) estará dockerizada. La orquestación de estos servicios se gestionará mediante Docker Compose, asegurando un entorno de desarrollo, pruebas y producción coherente y reproducible.

Funcionalidad de la API: Esta API expondrá de forma segura y controlada los siguientes *endpoints*:

- Autenticación y Configuración:
  - Endpoints para la autenticación de la aplicación cliente (ej. `POST /auth/token`).
  - Endpoints para obtener datos de configuración (ej. lista, ubicación y metadatos de los sensores, mapa base del río).
- Datos e Inferencia:
  - Endpoints para recibir datos en tiempo real de los sensores (vía sondeo HTTP o WebSockets).
  - Endpoints para solicitar inferencias al modelo DeepONet (pasando datos de sensores y recibiendo un mapa de calor).
- Consulta de Históricos:
  - Endpoints para consultar el histórico de alertas y estados.
  - Endpoints para consultar datos estáticos (ej. Censo de Vertidos Autorizados del OE2).
- Gestión del Simulador:
  - Endpoints para gestionar (lanzar, detener) la ejecución de nuevas simulaciones en el "Gemelo Digital".
  - Endpoints para consultar el estado (`GET /simulacion/estado/{id}`) de una simulación en curso.

Descripción del Problema y Contexto

Un Desafío Inverso y Mal Condicionado

La identificación y localización de eventos de vertido no autorizados o accidentales en grandes cuencas fluviales es un problema inverso de alta complejidad técnica. Cuando una estación de medición aguas abajo detecta una anomalía en la calidad del agua (el efecto), el contaminante (la causa) ya ha sido transportado, mezclado y difundido por el flujo del río durante un periodo de tiempo desconocido.

Este desafío se magnifica al ser un problema intrínsecamente mal condicionado (*ill-posed*)**.** En el caso de estudio de la cuenca del Tajo, se dispone de un número muy limitado de puntos de observación (aprox. 32 estaciones de calidad del agua) para inferir un campo de fuentes de alta dimensionalidad (más de 1.800 puntos de vertido autorizado conocidos, más un número indefinido de focos ilegales). Con los métodos actuales, es computacional y logísticamente inviable determinar el origen del vertido con la rapidez necesaria.

Relevancia y Justificación del Proyecto

La relevancia de este proyecto es triple:

- Relevancia Social y Ambiental: La contaminación fluvial tiene un impacto directo e inmediato en la salud pública (afectando la potabilidad del agua) y en la integridad de los ecosistemas acuáticos. Una detección rápida es fundamental para activar alertas y proteger los puntos de captación de agua.
- Relevancia Operativa (Utilidad): Los organismos de gestión de cuencas carecen de herramientas proactivas. Actualmente, la detección es reactiva, lo que retrasa las medidas correctoras, incrementa el coste de la remediación y dificulta enormemente la imputación de responsabilidades
- Relevancia Técnica y Científica: Abordar un problema físico *ill-posed* mediante el aprendizaje de operadores (DeepONet), en lugar de con métodos de asimilación de datos tradicionales (que serían demasiado lentos), representa un enfoque novedoso y puntero en la aplicación de la IA.

Solución Esperada y Beneficiarios

La solución esperada es un **sistema de soporte a la decisión (DSS)** basado en el modelo DeepONet entrenado. El sistema se activará automáticamente mediante alertas (por ejemplo, al detectar anomalías o valores fuera de rango en la telemetría de los sensores).

Una vez activado, el sistema ingerirá las series temporales recientes de las 32 estaciones y generará, en tiempo casi real, un "mapa de calor" probabilístico sobre el dominio del río. Este mapa identificará las zonas de origen del vertido más probables y su intensidad estimada, permitiendo una respuesta inmediata y focalizada.

Los **beneficiarios directos** de esta solución son:

1. **Organismos de Gestión de Cuencas:** Como la **Confederación Hidrográfica del Tajo (CHT)**, que podrá optimizar sus recursos de inspección, dirigiéndolos inmediatamente a la zona de alta probabilidad tras recibir una alerta.
2. **Servicios de Protección Ambiental:** Como el **SEPRONA**, al disponer de una herramienta que agiliza la investigación y la recopilación de pruebas.
3. **Empresas de Suministro de Agua:** Que podrán ser alertadas proactivamente de un riesgo aguas arriba de sus puntos de captación.

Fuentes de datos y justificación de su elección

Para abordar este proyecto, se utiliza un enfoque híbrido que combina datos observados del mundo real (para análisis y calibración) con un dataset sintético a gran escala (para el entrenamiento del modelo).

Datos Observados (Para Calibración y Contextualización)

Estos datos son la base para entender el problema y asegurar que nuestro simulador ("gemelo digital") opere bajo condiciones realistas.

- **Fuente y Método de Adquisición:**
  - **Datos de Calidad del Agua (SAICA) y Aforo (SAIH):** Los datos de series temporales de las 32 estaciones de medición se obtienen de los portales de datos abiertos de la Confederación Hidrográfica del Tajo (CHT). Para ello, se han desarrollado *scripts* de *web scraping* en Python que, mediante ingeniería inversa de las peticiones XHR, acceden directamente a los *endpoints* JSON de la plataforma, permitiendo una descarga eficiente de la serie de los últimos 10 días

    - **Censo de Vertidos Autorizados:** Se utiliza el censo oficial de la CHT (`Censo vertidos.-xlsx`), que identifica más de 1.800 puntos de vertido autorizados en la cuenca.

- Enlaces Exactos a los Datasets:
    - *Portal SAIH (Aforos): https://saihtajo.chtajo.es/index.php?url=/tr/mapas/ambito:PL/mapa:H1#nav*
    - *Portal SAICA (Calidad del Agua): https://saihtajo.chtajo.es/index.php?url=/tr/mapas/ambito:PL/mapa:H1#nav*
    - *Censo de Vertidos: https://www.chtajo.es/censo-de-vertidos*
- *Descripción del Contenido:*
    - ***Datos de Sensores:*** *Se obtienen múltiples variables clave (pH, Temperatura, Amonio, Nitratos, O2 Disuelto, Conductividad, etc.) en formato JSON. Los ficheros brutos se almacenan en* `data/datasets/raw/`
    - ***Censo de Vertidos:*** Fichero Excel con +1.800 registros, incluyendo (en su mayoría) coordenadas y tipología industrial del vertido.
- Transformación y Limpieza Estimada
    - **Datos de Sensores:** El análisis exploratorio inicial (EDA) y la limpieza implicarán la interpolación de datos faltantes, el tratamiento de valores atípicos y la agregación temporal y por estación.
    - **Censo de Vertidos:** Este dataset requiere un **enriquecimiento de datos** significativo. Se están utilizando flujos de trabajo automatizados (n8n) y herramientas de búsqueda web (Perplexity) para verificar el estado de las empresas y completar la información faltante.

Datos Sintéticos (Para Entrenamiento del DeepONet)

Fuente y Justificación

- **Fuente:** Este dataset será generado íntegramente por nuestro **gemelo digital** (el simulador hidrodinámico y de transporte Java/JNI/CUDA).
- **Justificación:** El objetivo del proyecto es detectar vertidos *ilegales*, de los cuales, por definición, **no existen datos etiquetados**. Es imposible saber cuándo, dónde y cuánta cantidad se vertió en un evento real pasado.
- Además, para que el DeepONet aprenda a resolver el **problema *ill-posed*** (inferir +1.800 fuentes desde 32 sensores), se requiere un conjunto de entrenamiento masivo (>100.000 muestras) que cubra una vasta combinación de escenarios (diferentes ubicaciones, intensidades, duraciones, caudales del río, etc.). Esto solo es factible mediante la simulación de alto rendimiento.

Descripción del Contenido

- **Formato:** Pares de `(Input_Function, Output_Function)`.
- **Tamaño Estimado:** > 100.000 muestras.
- **Variables de Entrada (Branch Net):** Series temporales simuladas de las 32 estaciones de sensores (ej. vectores de `[32 sensores x 48 horas]`).

- **Variables de Salida (Trunk Net):** La función o "mapa de calor" que describe la ubicación e intensidad del vertido que generó esas lecturas.

Metodología prevista

La metodología del proyecto sigue un proceso estructurado en fases, diseñado para abordar el desafío central: la escasez de datos de entrenamiento y la naturaleza *ill-posed* del problema.

Fases del Proceso

El proyecto se divide en las siguientes etapas clave, alineadas con los objetivos específicos:

1. **Adquisición y Análisis (OE2):** Obtención de datos reales (sensores y censo) mediante los *scrapers* de Python y análisis exploratorio (utilizando *notebooks* como `Análisis del Censo de autorizados a verter.ipynb`) para comprender el dominio y obtener los parámetros de calibración.
2. **Desarrollo del Gemelo Digital (OE1):** Creación del simulador físico. Esta es la etapa más crítica y se detalla a continuación.
3. **Generación de Datos Sintéticos (OE3):** Uso del simulador para crear el dataset de entrenamiento.
4. **Modelado y Evaluación (OE4):** Entrenamiento y validación del modelo DeepONet.
5. **Prototipado (OE5):** Integración del modelo en un sistema funcional.

Metodología del Gemelo Digital (OE1)

El "gemelo digital" es un sistema híbrido que simula la hidrodinámica (cómo fluye el agua) y el transporte de contaminantes (cómo se mueven los vertidos).

- **Arquitectura:** La orquestación de la simulación se maneja en **Java**, que se comunica vía **JNI (Java Native Interface)** con un *solver* (solucionador) numérico de alto rendimiento escrito en **C++/CUDA**.
- **Simulación Física:**
    1. **Hidrodinámica:** Se resuelve la **ecuación de Manning** para calcular el calado (H) y la velocidad (V) del agua en cada sección del río.
    2. **Transporte:** Sobre este flujo, se simula el transporte de contaminantes mediante la ecuación de **advección-difusión-reacción**.
- **Metodología de Optimización (Viabilidad):** La generación de >100.000 escenarios es un desafío computacional masivo (Riesgo: Tiempo de Cómputo). La metodología para mitigar esto se basa en la optimización extrema del *solver* de CUDA (`manning_kernel.cu`):
    1. Se explota la naturaleza **"vergonzosamente paralelizable"** del problema.
    2. **Eliminación de Divergencia de Ramas:** Se ha reescrito el código del kernel para operar **sin sentencias `if`** en el bucle principal. Las comprobaciones (ej. división por cero) se manejan mediante aritmética segura (añadiendo `epsilon`) y saneamiento de datos en la entrada.
    3. **Precisión de Datos:** Se reduce la precisión de los cálculos de `FP64` (doble, estándar en CPU) a `FP32` (simple), que es mucho más rápida en GPU (64:1 en arquitecturas Blackwell RTX5090), sin pérdida significativa de precisión para este problema físico (se estima en 10E-4)

4. **Optimización de Transferencia de Datos:** Para minimizar la latencia CPU-GPU, se implementarán **copias de memoria asíncronas** y se usará *pinned memory*. Esto permite que el motor de **Acceso Directo a Memoria (DMA)** solape cómputo y comunicación.

5. **Profiling y Escalabilidad de Cómputo:** Se utiliza **Nvidia Nsight** para perfilar el kernel y determinar la configuración óptima de hilos y *batch size* para la GPU local (RTX 5090). Si los tiempos de generación lo requieren, el plan contempla la **escalabilidad en la nube**, alquilando tiempo de cómputo en GPUs de última generación (ej. **Nvidia B200**) para asegurar la finalización del dataset.

Metodología de Modelado (OE4)

- **Modelo de Aprendizaje Automático:** Se utilizará una **Red de Operador Profundo (DeepONet)**.
- **Justificación de la Elección:**
    - **Problema *Ill-Posed*:** Un modelo tradicional fracasaría al intentar mapear un vector de 32 sensores a uno de +1.800 posibles fuentes.
    - **Aprendizaje de Operadores:** El DeepONet está diseñado para aprender **operadores** (un mapeo entre espacios funcionales). En nuestro caso:
        1. La **Branch Net** recibirá la *función de respuesta del sensor* (los datos de las 32 series temporales).
        2. La **Trunk Net** recibirá la *función de consulta espacial* (coordenadas del río).
        3. La salida del modelo será la *función de mapa de calor* del vertido.

- **Procesamiento y Entrenamiento Distribuido:** Dado el volumen masivo del dataset sintético (cientos de miles de muestras de alta dimensionalidad), la metodología contempla el uso de **Apache Spark** para el procesamiento distribuido (ETL) de los datos. Asimismo, si el entrenamiento del DeepONet se convierte en un cuello de botella, se utilizarán *frameworks* de entrenamiento distribuido (ej. Horovod sobre Spark) para paralelizar la carga de trabajo en un clúster.

- **Evaluación y *Baselines*:** La robustez del modelo DeepONet se evaluará rigurosamente sobre un conjunto de prueba sintético, usando el Error Cuadrático Medio (MSE) entre el mapa de calor predicho y el real. Para contextualizar su rendimiento, el modelo se comparará contra dos modelos de referencia (*baselines*):
    - **Heurística Simple:** Un algoritmo básico que simule una estrategia de "remontar el río", como una triangulación basada en los tiempos de llegada de la anomalía a los diferentes sensores.
    - **Modelo SOTA Alternativo (PINN):** Se entrenará una **Red Neuronal Informada por la Física (PINN)**. A diferencia del DeepONet (que aprende el operador a partir de los datos), el PINN intentará resolver el problema inverso incorporando las Ecuaciones Diferenciales Parciales (EDPs) del transporte de contaminantes directamente en su función de pérdida. Esta comparación determinará la eficiencia y precisión de nuestro enfoque de aprendizaje de operador (DeepONet) frente a un enfoque basado en la física de la pérdida (PINN) para este problema específico.

Herramientas y Tecnologías

Herramientas y Tecnologías

La naturaleza híbrida del proyecto, que combina simulación física de alto rendimiento, gestión de bases de datos, desarrollo web *full-stack* y *deep learning*, requiere un conjunto de herramientas especializado y moderno.

- **Arquitectura y Simulación (Gemelo Digital):**
    - **Java (JDK 21):** Lenguaje principal para la orquestación del simulador, gestión de configuraciones y la lógica de negocio del *backend*.
    - **C++ (C++17):** Utilizado para escribir el *solver* numérico nativo.
    - **NVIDIA CUDA 12+:** La tecnología clave para la aceleración del *solver* hidrodinámico en GPU.
    - **JNI (Java Native Interface):** El "pegamento" que permite la comunicación de alto rendimiento entre la JVM (Java) y el código nativo (C++/CUDA).
    - **Nvidia Nsight Systems:** Herramienta de *profiling* para analizar el rendimiento del kernel de CUDA.
    - **Gradle:** Sistema de automatización de la compilación (`build.gradle`) que maneja el complejo proceso de compilación de Java 21, C++ y el enlazado de las librerías nativas.
- **Stack de Backend y Datos:**
    - **Python (FastAPI & Uvicorn):** Utilizado como microservicio de inferencia de alto rendimiento para servir los modelos DeepONet/PINN entrenados.
    - **PostgreSQL:** Sistema gestor de base de datos relacional (datos del censo, resultados, etc.).
    - **Redis:** Base de datos en memoria de alta velocidad, usada para el cacheo de resultados de simulación, gestión de sesiones y como *message broker* si es necesario.
    - **Kong API Gateway:** Actúa como punto de entrada único (*Single Point of Entry*), gestionando, securizando y enrutando las peticiones de la UI a los microservicios correspondientes (el *backend* de Java y el *backend* de inferencia de FastAPI).
    - **JDBI3 & HikariCP:** Stack de persistencia de datos de alto rendimiento para Java.
    - **SLF4J/Logback & Jackson:** Utilizados para un logging robusto y la serialización/deserialización de datos y configuraciones JSON.
- **Interfaz de Usuario (UI) y Prototipado (OE5):**
    - **JavaFX (OpenJFX 25):** Stack tecnológico principal para el "Proyecto de Informática" (OE5). Se utilizará para construir la **aplicación de escritorio nativa** que consumirá la API (OE6). La estructura principal de la aplicación, la gestión de ventanas, los menús de navegación, los controles de administrador (RF-ADMIN) y los *dashboards* operativos principales (RF-OP) se implementarán con componentes nativos de JavaFX (controles, *charts*, layouts) para garantizar un rendimiento óptimo y una experiencia de usuario fluida.
    - **React.js (HTML5/CSS3/JavaScript):** Se utilizará para desarrollar un componente web específico, por ejemplo, un *dashboard* de análisis de históricos o un visualizador geoespacial avanzado.

    - **JavaFX WebView:** Como prueba de concepto de integración de tecnologías, la aplicación nativa de JavaFX utilizará su componente `WebView` para cargar y renderizar la vista desarrollada en React.js. Se implementará el puente de comunicación nativo (Java-a-JavaScript) para permitir que la vista web interactúe con el resto de la aplicación JavaFX.
    - **Nginx:** Servidor web y *reverse proxy* de alto rendimiento utilizado para servir la aplicación React.js que será consumida por el `WebView`.
    - **VPS Hosting:** La aplicación web de React.js (y potencialmente la API) estará alojada en un Servidor Privado Virtual (VPS) (con 99.99% uptime).
- **Stack de Ciencia de Datos (Python):**
    - **Python 3.10+:** Lenguaje central para *scraping*, análisis exploratorio y modelado de ML.
    - **Pandas / Geopandas:** Librería fundamental para la manipulación y análisis de datos tabulares (ej. el Censo de Vertidos y los resultados de los *scrapers*).
    - **Matplotlib / Seaborn:** Librerías de visualización para el Análisis Exploratorio de Datos (EDA).
    - **Jupyter Notebooks:** Entorno interactivo para el análisis de datos.
    - **Requests :** Herramientas de elección para *parsear* sitios estáticos y para realizar ingeniería inversa de *endpoints* de API/XHR (el método más eficiente, ya validado en el proyecto).
        - **Playwright / Selenium:** *Frameworks* de simulación de navegador (headless browser)
    - **n8n (N-node-N):** Plataforma *low-code* para el flujo automatizado de enriquecimiento de datos.
    - **Apache Spark:** Contemplado para el procesamiento (ETL) distribuido del dataset sintético.
- **Machine Learning (Modelado):**
    - **Python (TensorFlow / PyTorch):** *Frameworks* de elección para el diseño y entrenamiento de los modelos DeepONet y PINN.
    - **Plataformas de Cómputo en la Nube (AWS/GCP):** Contempladas para el alquiler de GPUs de alto rendimiento (ej. **Nvidia B200**).
- **Testing y Calidad:**
    - **JUnit 5, Mockito & AssertJ:** Stack estándar para *testing* unitario y de integración en el ecosistema Java.
    - **Testeo de GPU (`@Tag("GPU")`):** El `build.gradle` define una tarea personalizada (`gpuTest`) para aislar los tests que requieren hardware de GPU nativo.

Justificación de la Elección (No-Python)

Aunque la asignatura se centra en Python, la arquitectura del proyecto utiliza un enfoque políglota de "la herramienta adecuada para el trabajo adecuado":

1. **Python (Ciencia de Datos y MLOps):** Se usa donde es el líder indiscutible: EDA (*Pandas*), *scraping*, *notebooks* (*Jupyter*), entrenamiento de ML (*TensorFlow/PyTorch*) y servicio de modelos de inferencia (*FastAPI*).

2. **Java/C++/CUDA (HPC y Backend):** Esta combinación se elige para el núcleo del proyecto (generación de datos y orquestación) por razones que Python no puede cubrir:

- **Rendimiento Extremo (HPC):** La generación de >100.000 escenarios físicos es inviable en Python. Se requiere C++/CUDA para optimizaciones de bajo nivel (código *branchless*, `FP32`, DMA).
- **Backend Robusto:** Java 21 y su ecosistema (JDBI, HikariCP) proporcionan una base robusta y concurrente para la lógica de negocio principal.

3. **Prototipo Integrado:** El uso de **JavaFX WebView** permite que el prototipo de escritorio (OE5) cargue la interfaz web principal (React.js), proporcionando un puente de comunicación nativo y entregando una experiencia de usuario moderna.

Plan de Trabajo y Roles del Equipo

Contexto Multi-Asignatura del Proyecto

Es importante destacar que este es un proyecto de gran escala y ambición, concebido para ser desarrollado y presentado en el contexto de **múltiples asignaturas** del Grado. Su arquitectura completa abarca áreas como Computación de Alto Rendimiento, Desarrollo de Backend/Frontend e Inteligencia Artificial.

Para la presente asignatura, **"Proyecto de Computación I"**, el alcance de la evaluación se centrará exclusivamente en el **componente de Ciencia y Análisis de Datos**. Las fases de desarrollo de software (el *solver* CUDA, la arquitectura de microservicios, la UI en React, etc.) se presentan como contexto para justificar la generación y el uso de los datos, pero no forman parte del trabajo a evaluar aquí.

Definición del Proyecto de Informática (Entregables de Gestión)

Mientras que el "Proyecto de Computación I" se centra en la Ciencia de Datos y el HPC (OE1-OE4, OE6), el "Proyecto de Informática" se enfoca en el desarrollo de la aplicación cliente (OE5). La entrega preliminar de este proyecto de informática debe incluir la siguiente documentación:

1. Necesidad y Caso de Negocio

- **Necesidad:** Los organismos de gestión de cuencas (ej. CHT) y los servicios de protección ambiental (ej. SEPRONA) carecen de una herramienta unificada y en tiempo real para reaccionar ante eventos de vertido. La detección actual es reactiva y fragmentada.
- **Caso de Negocio:** Este proyecto proporciona un Sistema de Soporte a la Decisión (DSS) que centraliza la monitorización (sensores), la inferencia inteligente (DeepONet) y la gestión de alertas. El valor reside en reducir drásticamente el tiempo de respuesta (de días a minutos), permitiendo una actuación inmediata, minimizando el daño ambiental y facilitando la imputación de responsabilidades. La aplicación de escritorio (OE5) es la herramienta tangible que entrega este valor al usuario final.

2. Requisitos del Sistema

Se dividen en Requisitos Funcionales (RF) y No Funcionales (RNF).

Autenticación y Roles (RF):

- **RF-01:** El sistema debe solicitar autenticación (usuario y contraseña) al iniciarse.
- **RF-02:** El sistema debe diferenciar entre dos roles de usuario: **Operador** (usuario estándar) y **Administrador** (usuario avanzado).

Requisitos Funcionales (Rol: Operador):

- **RF-OP-01:** El sistema debe mostrar un mapa interactivo de la cuenca del río.

- **RF-OP-02:** El sistema debe mostrar la ubicación y el estado (Online/Offline/Alerta) de los 32 sensores sobre el mapa.
- **RF-OP-03:** Al seleccionar un sensor, el sistema debe mostrar sus datos históricos y en tiempo real (pH, Tª, Amonio, etc.) en gráficos claros.
- **RF-OP-04:** El sistema debe mostrar una notificación/alerta visual y sonora prominente cuando la API reporte un evento de vertido.
- **RF-OP-05:** El sistema debe permitir consultar un histórico filtrable de alertas pasadas (por fecha, sensor, tipo de alerta).
- **RF-OP-06:** Tras una alerta, el sistema debe solicitar y renderizar el "mapa de calor" de predicción del DeepONet sobre el mapa del río.
- **RF-OP-07:** El Operador debe poder ejecutar simulaciones de escenarios (interactuando con el OE1 vía API) *solo* sobre "Gemelos Digitales" existentes y pre-configurados.

Requisitos Funcionales (Rol: Administrador):

- **RF-ADMIN-01:** El sistema debe mostrar una "Sección de Administración" visible *únicamente* para los usuarios con rol de Administrador.
- **RF-ADMIN-02:** Esta sección debe permitir a los Administradores crear, ver, modificar y eliminar (CRUD) "Gemelos Digitales" (nuevos ríos o tramos).
- **RF-ADMIN-03:** El formulario de creación/edición de "Gemelo Digital" debe permitir al Administrador definir y "jugar" con todos los parámetros de base del simulador (ej. geometría del cauce, coeficientes de Manning, ubicación de sensores, topología de la red fluvial, puntos de vertido autorizados).
- **RF-ADMIN-04:** Los Administradores deben poder gestionar las cuentas de usuario (crear/eliminar/modificar usuarios Operadores).

Requisitos No Funcionales (RNF):

- **RNF-01:** La aplicación debe ser una aplicación de escritorio nativa desarrollada en JavaFX (OpenJFX 25).
- **RNF-02:** Toda la comunicación de datos (sensores, mapas, alertas, simulaciones) se realizará exclusivamente a través de la API REST (OE6). La aplicación no debe tener acceso directo a la base de datos ni a los *solvers*.
- **RNF-03:** La actualización de datos en tiempo real debe tener una latencia perceptible para el usuario inferior a 5 segundos (dependiente de la API).
- **RNF-04:** La interfaz debe ser intuitiva, limpia y operable por personal técnico no especializado en computación (para el rol Operador). La interfaz de Administrador puede ser técnicamente más compleja.
- **RNF-05:** El sistema debe ser seguro, gestionando el token de autenticación (obtenido del OE6) y aplicando un **Control de Acceso Basado en Roles (RBAC)** para ocultar/deshabilitar la funcionalidad de administrador a los operadores.

3. User Story (Historias de Usuario)

- **Como** Gestor de Cuenca (Usuario Operativo), **Quiero** ver un dashboard centralizado con el estado de todos los sensores de un vistazo, **Para que** pueda detectar anomalías de forma inmediata.

- **Como** Técnico de Intervención (Usuario Táctico), **Quiero** que, cuando salte una alerta de contaminación, el sistema me muestre un mapa de calor con el origen más probable del vertido, **Para que** pueda dirigir al equipo de inspección a la ubicación correcta sin demora.
- **Como** Analista de Datos (Usuario Analítico), **Quiero** poder consultar el histórico de alertas y los mapas de calor asociados, **Para que** pueda identificar patrones de vertidos recurrentes en ciertas zonas.
- **Como** Administrador del Sistema (Usuario Avanzado), **Quiero** un panel de configuración para definir la geometría y los parámetros hidrodinámicos de un nuevo tramo de río, **Para que** pueda crear un nuevo "Gemelo Digital" que los Operadores puedan usar para sus simulaciones.

4. Procedimiento de Trabajo / Actas de Trabajo

- **Metodología:** El equipo del "Proyecto de Informática" (desarrolladores JavaFX) seguirá una metodología Ágil Kanban.
- **Coordinación:** Se establecerá una reunión de sincronización semanal entre "Computación" (Backend/API) y "Informática" (Frontend/App).
- **Definición de la API:** El entregable inicial clave es la definición de la especificación de la API (ej. OpenAPI/Swagger), que servirá como "contrato" entre ambos equipos y permitirá el desarrollo en paralelo. La API debe incluir *endpoints* de gestión (para Admins) y de consulta (para Operadores).
- **Actas:** Todas las reuniones de coordinación y las decisiones de diseño de la API se documentarán en un sistema compartido para mantener la trazabilidad. Se decide utilizar milestones y issues de github en el repositorio https://github.com/theDuoXu/Project-Stalker

Roles del Equipo (Para "Proyecto de Computación I")

El equipo para esta asignatura está enfocado en el rol de Científicos de Datos

- **Duo Xu Ye:**
  - **Rol Principal (Fuera de esta asignatura):** Arquitecto del Sistema y desarrollador principal del *stack* tecnológico completo (HPC, backend, frontend).
  - **Rol en esta asignatura:** Líder del equipo de datos. Responsable de la integración de los datos con el simulador y del diseño de los experimentos de modelado.
- **Tadeo Adrián Troncoso Taraborreli:**
  - **Rol en esta asignatura:** Analista de Datos y Especialista en ETL. Responsable de la adquisición, limpieza y enriquecimiento de los datos de la CHT y de los nuevos conjuntos de datos (embalses, cartografía).
- **Víctor Pablo Velayos Velayos:**
  - **Rol en esta asignatura:** Analista de Datos y Modelado. Responsable del Análisis Exploratorio de Datos (EDA), la visualización y el desarrollo de los modelos de inferencia *baseline*.

Ampliación del Alcance de Análisis de Datos

Para enriquecer el componente de ciencia de datos de este proyecto, se incorporarán las siguientes tareas y fuentes de datos adicionales:

1. **Integración de Nuevos Datasets:** Además de los datos de la CHT, se analizarán:

    - **Datos de Embalses:** Se obtendrán y analizarán los datos de nivel y volumen de los embalses principales de la cuenca. El objetivo es modelar su impacto en el caudal y en parámetros de calidad del agua aguas abajo (ej. temperatura, dilución).

    - **Datos Cartográficos y Geológicos (IGN):** Se utilizarán capas de información geográfica del Instituto Geográfico Nacional para analizar la correlación entre la geología del terreno, las zonas de inundación y los puntos de vertido, enriqueciendo el análisis de riesgos.

2. **Desarrollo de Modelos de Inferencia *Baseline*:** Antes de abordar los modelos complejos (DeepONet/PINN), el equipo desarrollará y evaluará modelos más simples (ej. **Gradient Boosting, Redes Neuronales Densas**) para resolver sub-problemas concretos, como:

    - Predecir la temperatura del agua en un punto a partir del nivel del embalse y datos meteorológicos.

    - Estimar la concentración de un contaminante en un sensor basándose en el caudal y las lecturas de sensores aguas arriba. Estos modelos servirán como una valiosa referencia para contextualizar el rendimiento de los modelos más avanzados.

Plan de Trabajo (Cronograma de 8 Semanas para Análisis de Datos)

Este cronograma se centra exclusivamente en las tareas de Ciencia de Datos (el alcance de esta asignatura) a realizar por el equipo, asumiendo que el *solver* de HPC (OE1) y el *scraper* de contaminantes ya están operativos.

- **Semanas 1-2: Finalización de la Adquisición de Datos**
    - **Objetivo:** Completar la capa de adquisición de datos para tener todos los *datasets* necesarios.
    - **Tareas:**
        - Desarrollar el *scraper* para datos de Aforos (SAIH). (Responsable: Tadeo)
        - Desarrollar el *scraper* para datos de Embalses. (Responsable: Tadeo)
        - Desarrollar el *scraper* para datos Cartográficos y Geológicos (IGN). (Responsable: Víctor)
        - Desarrollar el *scraper* para la conexión a datos en tiempo real. (Responsable: Duo)
    - **Hito:** Todos los datos brutos (Contaminantes, Aforos, Embalses, Censo, IGN) están consolidados en la base de datos PostgreSQL.
- **Semanas 3-4: Análisis Exploratorio de Datos (EDA) Intensivo**
    - **Objetivo:** Comprender profundamente los datos, encontrar correlaciones y preparar el *feature engineering*. **Esta es la tarea central que está pendiente.**
    - **Tareas:**
        - EDA y visualización de series temporales (contaminantes, aforos, embalses). Búsqueda de correlaciones, estacionalidad y latencias. (Responsable: Víctor)
        - EDA geoespacial: Cruce de datos del Censo de Vertidos con las capas cartográficas del IGN (zonas inundables, geología). (Responsable: Tadeo)
        - Creación de *notebooks* maestros de análisis y documentación del proceso de limpieza de datos. (Responsable: Duo)

    - **Hito:** Informe de EDA completado, con las principales hipótesis y variables identificadas.
- **Semanas 5-6: Desarrollo de Modelos *Baseline***
    - **Objetivo:** Evaluar la predictibilidad de los sub-problemas identificados.
    - **Tareas:**
        - *Feature Engineering* basado en los hallazgos del EDA. (Responsable: Tadeo)
        - Entrenar y evaluar los modelos de inferencia *baseline* (ej. Gradient Boosting, Redes Densas) para predecir variables clave (ej. Tª del agua, nivel de embalse). (Responsable: Víctor)
    - **Hito:** Modelos *baseline* entrenados y evaluados.
- **Semana 7: Preparación de Datasets Finales**
    - **Objetivo:** Preparar los conjuntos de datos limpios para el gemelo digital y los modelos avanzados.
    - **Tareas:**
        - Crear los *datasets* de calibración para el gemelo digital. (Responsable: Duo)
        - Documentar el *data lineage* y las transformaciones finales. (Responsable: Tadeo, Víctor)
    - **Hito:** *Datasets* de calibración y modelado generados y documentados.
- **Semana 8: Documentación y Entrega Final**
    - **Objetivo:** Consolidar el trabajo de la asignatura.
    - **Tareas:**
        - Redacción del informe final del proyecto (enfocado en el análisis de datos y los modelos *baseline*).
        - Limpieza y entrega de los *notebooks* de análisis y modelado.
    - **Hito:** Entrega final del proyecto de la asignatura

Roles del Equipo (Para "Proyecto de Informática")

Este equipo trabajará en paralelo al de "Computación", enfocándose en el desarrollo de la aplicación cliente (OE5) bajo una metodología ágil Kanban.

- **Duo Xu Ye:**
    - **Rol en esta asignatura:** Jefe de Proyecto / Product Owner. Responsable de la **gestión ágil del proyecto**: define los *milestones*, crea y prioriza las *issues* (historias de usuario, *features*, *bugs*) en el *backlog* y valida las entregas. Actúa como nexo técnico y funcional entre la API (OE6) y la aplicación cliente (OE5).
- **Andrei Ionut HRISCA / Héctor Omar HERNÁNDEZ RODRÍGUEZ / Roberto GARCÍA CASADO:**
    - **Rol en esta asignatura:** Desarrolladores de Aplicación (JavaFX).

- **Responsabilidad (Compartida/Difusa):** El equipo de desarrollo trabajará de forma flexible sobre el *backlog* definido por el Jefe de Proyecto. Dado que la API (OE6) define una arquitectura modular, los desarrolladores **implementarán las *issues* asignadas** según surja la necesidad, abarcando todas las áreas de la aplicación cliente:
    - Implementación de vistas y controles de UI/UX (JavaFX).
    - Lógica de negocio del lado del cliente.
    - Gestión de estado y navegación.
    - Integración de los *endpoints* de la API (autenticación, consumo de datos, etc.).

Riesgos del Proyecto y Estrategias de Mitigación

A continuación, se identifican los riesgos potenciales del proyecto, divididos por su área de impacto, y las estrategias de mitigación correspondientes.

1. Riesgos de Simulación y Modelado (Proyecto de Computación)

- **Riesgo (Mitigado): Complejidad del Solver Hidrodinámico (OE1).**
    - *Descripción:* El desarrollo del *solver* físico (ecuación de Manning) podría ser un cuello de botella técnico.
    - *Mitigación:* **Este riesgo se considera MITIGADO.** El *solver* hidrodinámico principal ya ha sido implementado, validado y optimizado en C++/CUDA, demostrando ser una tarea de complejidad manejable (completada en una semana).
- **Riesgo: Calidad de la Calibración del Gemelo Digital (OE2).**
    - *Descripción:* El rendimiento del DeepONet depende al 100% de la calidad del dataset sintético (OE3), y este, a su vez, depende de una calibración precisa del "Gemelo Digital" con los datos reales de la CHT (OE2). Si los datos reales son insuficientes o ruidosos, la calibración será pobre.
    - *Mitigación:* 1) Realizar un Análisis Exploratorio de Datos (EDA) exhaustivo para limpiar y entender las limitaciones de los datos reales. 2) Enriquecer los datos de la CHT con otras fuentes (geología, meteorología). 3) Documentar la precisión de la calibración como el límite superior de la precisión del modelo de IA.
- **Riesgo: No Convergencia del DeepONet (OE4).**
    - *Descripción:* Es el principal riesgo técnico. Existe la posibilidad de que la arquitectura DeepONet no logre aprender a resolver el problema inverso (un problema *ill-posed*) con la precisión necesaria, incluso con un gran dataset sintético.
    - *Mitigación:* 1) Desarrollo de modelos *baseline* (Gradient Boosting, PINN) para establecer un suelo de rendimiento. 2) El modelo PINN (Red Neuronal Informada por la Física) se considera el plan de contingencia principal, ya que su función de pérdida basada en las EDPs puede guiar el aprendizaje si el enfoque de DeepONet (basado solo en datos) falla.
- **Riesgo: Tiempos de Generación del Dataset Sintético (OE3).**
    - *Descripción:* Aunque el *solver* es rápido, generar +100.000 escenarios de simulación (cubriendo múltiples parámetros) podría requerir un tiempo de cómputo prohibitivo en hardware local.

- *Mitigación:* 1) Perfilar el *kernel* de CUDA (Nvidia Nsight) para optimizar el *batch size* y el uso de memoria. 2) Paralelizar la generación de escenarios. 3) Plan de escalado para alquilar tiempo de cómputo en GPUs de nube (AWS/GCP) si los tiempos de generación locales superan una semana.

2. Riesgos de Desarrollo y Despliegue (API y App JavaFX)

- **Riesgo: Desacoplamiento entre la API (OE6) y la Aplicación (OE5).**
  - *Descripción:* Es el riesgo de gestión más común en arquitecturas de microservicios. El equipo de JavaFX (Informática) puede quedar bloqueado si la API (Computación) no está lista, tiene *bugs* o sufre cambios constantes en su "contrato".
  - *Mitigación:* 1) Definición de un contrato de API formal (OpenAPI/Swagger) *antes* de comenzar el desarrollo de la aplicación. 2) **Duo Xu Ye actúa como Jefe de Proyecto y Product Owner (nexo) de ambos equipos**, garantizando la alineación. 3) Creación de un *servidor mock* de la API para que el equipo de JavaFX pueda desarrollar y probar la UI sin depender del *backend* real.
- **Riesgo: Complejidad de la Interfaz de Administrador (RF-ADMIN).**
  - *Descripción:* El requisito de "jugar con los parámetros" y crear nuevos "Gemelos Digitales" (RF-ADMIN-03) es funcionalmente muy complejo y podría consumir la mayoría de las horas de desarrollo, retrasando las funciones clave del Operador.
  - *Mitigación:* Gestión ágil y priorización del *backlog*. Las funcionalidades del **Operador** (visualización de alertas, mapas de calor) se priorizarán como Producto Mínimo Viable (MVP) sobre las del **Administrador** (creación de ríos).
- **Riesgo: Rendimiento de la Visualización en JavaFX (OE5).**
  - *Descripción:* La aplicación debe renderizar múltiples gráficos en tiempo real y, sobre todo, un "mapa de calor" sobre un mapa geográfico interactivo. Esto puede ser un desafío de rendimiento en JavaFX.
  - *Mitigación:* 1) Usar librerías de gráficos optimizadas. 2) Evaluar si el renderizado del mapa de calor se realiza de forma nativa o (como se contempla en el OE5) mediante el componente `WebView` y una librería JS (ej. Leaflet), que está altamente optimizada para ello.

3. Riesgos de Gestión

- **Riesgo: Dependencia de Personal Clave (Key-Person Dependency).**
  - *Descripción:* El proyecto tiene una alta dependencia de Duo Xu Ye, quien actúa como nexo, arquitecto de la API, desarrollador del HPC y Product Owner de la aplicación. Una ausencia de esta persona bloquearía ambos proyectos.
  - *Mitigación:* **Documentación rigurosa.** La especificación de la API (Swagger), la documentación del código del *solver* (HPC) y un *backlog* de *issues* claro y bien definido son cruciales para que el resto del equipo pueda operar con autonomía.