Bachillerato en Ingeniería en Ciencia de Datos
Proyecto Final – Curso de Programación Web
Profesor: Alejandro Zamora

Objetivo

Desarrollar, en equipos, una aplicación web completa que integre todos los conocimientos vistos en el curso:

- Frontend: HTML, CSS, JavaScript
- Backend: Python con FastAPI (o Flask)
- Orquestación de Pipelines: Prefect 2.x o Apache Airflow
- Seguridad de datos: HTTPS, validación de entradas, manejo de credenciales
- Base de datos: SQL (MySQL, PostgreSQL o similar)
- DevOps/Despliegue: Docker, CI/CD (GitHub Actions, GitLab CI, etc.), hosting (Heroku, Render, AWS, GCP…)

La aplicación debe ofrecer un CRUD completo, consumir al menos una API externa, exponer su propia API interna para lecturas/escrituras, y utilizar un pipeline de datos que limpie, registre logs y genere backups automáticos.

Tecnologías y herramientas disponibles

- Frontend:
    - HTML5 semántico
    - CSS3 (Flexbox o Grid)
    - JavaScript ES6+ (Fetch API, módulos)

- Backend:
    - Python 3.9+ con FastAPI
    - Pydantic para validación de modelos
    - mysql-connector-python o SQLAlchemy ORM

- Pipelines (elige una):
    - Prefect 2.x (Flows, Tasks, Orion, Agent)
    - Apache Airflow (DAGs, Operators, Scheduler)

- Seguridad de datos:
    - HTTPS (certificados TLS)
    - Sanitización/validación de inputs (inyección SQL, XSS)
    - Manejo de secretos (Vault, variables de entorno, Prefect Secrets)

- Base de datos:
    - MySQL o PostgreSQL
    - Diseño de esquemas normalizados
    - Scripts SQL para migraciones y carga inicial

- DevOps / Despliegue:
    - Docker / Docker Compose
    - Pipelines CI/CD (GitHub Actions, GitLab CI)
    - Despliegue en la nube (Heroku, Render, AWS EC2, GCP Cloud Run)

Requerimientos funcionales

1. CRUD principal
    - Usuarios podrán crear, leer, actualizar y eliminar recursos (por ejemplo, tareas, productos, leads, registros de clima, etc.).
2. API externa
    - Consumir datos de al menos una API pública (PokeAPI, OpenWeatherMap, TheMealDB, etc.) y mostrarlos en el frontend.
3. API interna
    - Exponer rutas REST (`/api/...`) para realizar operaciones contra la base de datos local.
4. Pipeline de datos
    - ETL automático que:
        - Extrae registros crudos (RAW) desde la base de datos y/o API externa.
        - Transforma/limpia (elimina nulos, formatea, valida rangos).
        - Carga en tablas ‘cleaned’.
        - Genera un log (TXT o JSON) con métricas de calidad (registros leídos, limpiados, removidos).
        - Crea backups periódicos en CSV (carpeta `backups/`).
    - El pipeline deberá programarse (cron o intervalo fijo) con Prefect u Airflow.
5. Seguridad
    - Toda conexión HTTP debe usar HTTPS.

- Validar y sanitizar entradas de usuario.
- Proteger credenciales de la base de datos y de APIs externas.

6. Despliegue & DevOps
    - Contenerización con Docker.
    - CI/CD que ejecute pruebas básicas y despache a un entorno de staging o producción.
    - Documentar pasos de despliegue en un `README.md`.

Entregables

1. Código fuente completo en repositorio Git (GitHub/GitLab):
    - Carpeta `frontend/` (HTML, CSS, JS).
    - Carpeta `backend/` (Python, FastAPI, scripts de pipeline).
    - Carpeta `pipeline/` (Flows o DAGs, backups, logs).
    - Scripts SQL para crear tablas y cargar datos de prueba.
    - Workflow de CI/CD (`.github/workflows/…` o similar).
2. Diagrama de arquitectura (puede ser en Lucidchart, draw.io, PowerPoint):
    - Componentes frontend, backend, base de datos, orquestador de pipelines, CI/CD, hosting.
    - Flujos de datos y dependencias.
3. Documentación en `README.md`:
    - Introducción al proyecto y objetivo.
    - Instrucciones de instalación y ejecución local (Docker o manualmente).
    - Cómo acceder a la UI y a la documentación de la API (Swagger UI).
    - Cómo disparar el pipeline manualmente.
4. Presentación final (10 minutos por grupo):
    - Demostración en vivo de la aplicación (CRUD, consumo de API externa, datos limpios).
    - Explicación del pipeline: definición del Flow/DAG, programación, logs y backup.
    - Despliegue: CI/CD y entorno de producción.
    - Diagrama de infraestructura y rol de cada componente.