Enunciado del Proyecto: WebServices + WebScraping

Descripción General

El objetivo de este proyecto final es integrar y demostrar las competencias adquiridas durante el curso en el desarrollo de aplicaciones web en el entorno servidor. El alumnado deberá diseñar e implementar una solución completa que abarque el ciclo de vida del dato: **adquisición (Web Scraping), almacenamiento (Base de Datos Relacional), gestión (Aplicación Web MVC) y distribución (API REST)**.

El desarrollo se realizará utilizando **PHP nativo (versión 8.1 o superior)**, sin el uso de frameworks *full-stack* (como Laravel o Symfony), para evidenciar el dominio de los fundamentos del lenguaje, la programación orientada a objetos (POO) y los patrones de diseño.

Temática del Proyecto

La temática es de **elección libre** por parte del alumno. El proyecto debe aportar valor mediante la consolidación de datos dispersos o el enriquecimiento de los mismos. A continuación, se proponen **10 casos de uso** de diferentes ámbitos para servir de inspiración, incluyendo ejemplos de uso de Inteligencia Artificial (IA) para la generación o mejora de datos:

1. **Mercado Inmobiliario:** Extracción de listados de viviendas de diferentes portales. Uso de IA para generar una descripción comercial atractiva o estimar una calificación de "oportunidad de inversión" basada en precio/metro cuadrado.
2. **Análisis de Productos Tecnológicos:** Scraping de precios y especificaciones de componentes de PC. Uso de IA para generar un resumen automático de las reseñas de los usuarios (análisis de sentimiento positivo/negativo).
3. **Agregador de Noticias Temáticas:** Recopilación de titulares de prensa sobre un tema específico (ej. "Ciberseguridad"). Uso de IA para categorizar automáticamente la noticia (ataques, defensa, legislación) y generar un breve resumen ejecutivo.
4. **Portal de Empleo IT:** Extracción de ofertas de trabajo para perfiles técnicos. Análisis de las tecnologías más demandadas y cálculo de salarios medios por tecnología y experiencia.
5. **Recetario Inteligente:** Scraping de listas de precios de supermercados e ingredientes básicos. El sistema sugiere recetas posibles basadas en los ingredientes más económicos de la semana.

6. **Agenda Cultural:** Recopilación de eventos (conciertos, teatro) de varias fuentes locales. Uso de una API de IA para recomendar eventos similares basándose en la descripción del artista o la obra.
7. **Monitorización Financiera/Cripto:** Obtención de valores bursátiles o criptomonedas en tiempo real. Generación de alertas cuando se superan ciertos umbrales y cálculo de medias móviles.
8. **Automoción y Segunda Mano:** Extracción de datos de vehículos usados. Comparativa del precio del vehículo respecto a la media del mercado para ese año y kilometraje.
9. **Investigación Académica:** Recopilación de *abstracts* de artículos científicos (Open Access) sobre una materia. Uso de IA para "traducir" el lenguaje técnico a un resumen divulgativo comprensible para el público general.
10. **Turismo y Clima:** Extracción de datos meteorológicos y disponibilidad hotelera de una región. El sistema puntúa qué municipios son mejores para visitar el fin de semana basándose en el clima y el precio.

Arquitectura y Requisitos Técnicos

El proyecto debe cumplir estrictamente con los siguientes requisitos técnicos para garantizar su viabilidad y mantenibilidad:

1. **Estructura de Directorios:** Se debe seguir una organización lógica y profesional, separando claramente la parte pública de la lógica privada:

codeText

```
/nombre-proyecto
├── /public        # Punto de entrada web (index.php, assets: css, js, img)
├── /src           # Lógica de la aplicación (Modelos, Controladores, Clases)
├── /views         # Plantillas HTML/PHP (Vistas)
├── /tests         # Pruebas Unitarias (PHPUnit)
├── /logs          # Ficheros de log generados por la aplicación
├── /docs          # Documentación técnica
├── vendor/        # Librerías gestionadas por Composer
├── composer.json  # Definición de dependencias
└── README.md      # Instrucciones de despliegue y uso
```

2. **Estándares de Código:** Adherencia al estándar **PSR-12**. Uso de tipado estricto (declare(strict_types=1);) en clases y funciones.

3. **Base de Datos:** Motor MySQL o MariaDB. La interacción desde PHP debe realizarse exclusivamente mediante **PDO** (PHP Data Objects).

Fases del Desarrollo

El proyecto se divide en cinco fases funcionales que deben estar integradas en la entrega final:

Fase 1: Adquisición de Datos (Web Scraping)

Desarrollo de scripts para la obtención automatizada de información desde fuentes externas.

- Se deben utilizar librerías como **Guzzle** (cliente HTTP) y **DOMCrawler** o **Symfony Panther**.
- Los datos extraídos deben ser procesados, limpiados y normalizados antes de su almacenamiento.
- *Opcional/Valorado:* Integración con APIs de IA (ej. OpenAI, Hugging Face) para enriquecer o transformar los datos extraídos.

Fase 2: Persistencia y Seguridad (Base de Datos)

Diseño e implementación del modelo de datos.

- Diseño de una base de datos relacional normalizada.
- Implementación de la capa de acceso a datos (Modelos) en PHP.
- **Seguridad:** Uso obligatorio de sentencias preparadas en PDO para la prevención de Inyección SQL.

Fase 3: Aplicación Web (Interfaz de Usuario)

Desarrollo de la interfaz visual para la consulta y gestión de los datos.

- Implementación del patrón **MVC (Modelo-Vista-Controlador)** con un *Front Controller* en public/index.php.
- Funcionalidades requeridas: Listados paginados, filtrado de datos, visualización gráfica (Chart.js o similar) y operaciones CRUD (Crear, Leer, Actualizar, Borrar).

Fase 4: Interoperabilidad (API REST)

Creación de un servicio web para exponer los datos a terceros.

- Desarrollo de endpoints que devuelvan la información en formato **JSON**.
- La API debe manejar correctamente los códigos de respuesta HTTP (200, 404, 500, etc.).
- Implementación de un pequeño script cliente de prueba que consuma esta API.

Fase 5: Calidad y Mantenimiento (Logs, Tests y Docs)

Implementación de herramientas para el control y calidad del software.

- **Logs:** Integración de **Monolog** para registrar eventos del sistema (ej. inicio del scraping, errores de conexión, accesos).
- **Documentación:** Uso de DocBlocks en el código y generación de documentación técnica mediante **phpDocumentor**.
- **Testing:** Creación de pruebas unitarias con **PHPUnit** para validar la lógica de negocio crítica.

Rúbrica de Evaluación

La evaluación del proyecto se realizará sobre un total de **10 puntos**, distribuidos según la implementación de las fases y la calidad técnica:

Fase / Criterio	Descripción Detallada	Puntuación (Max 10)
Fase 1: Scraping y Datos (15%)	Correcta implementación de la extracción de datos (Guzzle/Crawler). Limpieza y estructuración de la información obtenida. Integración efectiva de IA o lógica compleja de generación de datos.	**1.5 pts**
Fase 2: BBDD y Seguridad (20%)	Diseño relacional correcto y normalizado. Conexión robusta mediante PDO. **Uso estricto de sentencias preparadas** (seguridad).	**2.0 pts**
Fase 3: Aplicación Web MVC (25%)	Implementación correcta del patrón MVC. Separación de lógica y vistas. Funcionalidad del Front Controller. Usabilidad de la interfaz (filtros, paginación, gráficos).	**2.5 pts**
Fase 4: API REST (15%)	Estructura correcta de los endpoints. Respuestas JSON válidas. Manejo adecuado de verbos HTTP y códigos de estado. Script cliente funcional.	**2 pts**
Fase 5: Calidad del Código (25%)	**POO Avanzada:** Uso de herencia, interfaces y código limpio (PSR-12). **Herramientas:** Implementación de Monolog (Logs), phpDocumentor y gestión con Composer. **Testing:** Existencia de al menos 3 tests unitarios significativos con PHPUnit.	**2 pts**

Puntuación Adicional (Opcional)

Se podrá obtener hasta 1 punto extra (sin superar la nota máxima de la asignatura) cumpliendo los siguientes requisitos de presentación:

Criterio Adicional	Descripción	Puntuación Extra
Memoria y Vídeo	Entrega de una memoria en PDF (análisis funcional, diagrama E-R, decisiones técnicas) **Y** un vídeo demostrativo (screencast de 2-5 min) explicando el funcionamiento y el código.	**+1.0 pto**

Entregables

La entrega se realizará exclusivamente a través de un sistema de control de versiones. El alumno deberá proporcionar:

Enlace al Repositorio (GitHub/GitLab): El repositorio debe ser público o accesible por el docente.

Fichero **README.md**: Debe ubicarse en la raíz del proyecto y contener:

- Descripción del proyecto.
- Requisitos del servidor.
- Instrucciones paso a paso para la instalación y despliegue (composer install, configuración de credenciales, etc.).

Script SQL: Fichero .sql necesario para regenerar la estructura de la base de datos y cargar datos iniciales si fuera necesario.

Memoria del Proyecto: (En caso de optar a la puntuación adicional) Documento PDF incluido en el repositorio.