UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Sistema de información de apoyo a GreenMetrics

Presentado por Lorena Bueno Porras
en Universidad de Burgos — 2 de junio de 2025
Tutor: José Manuel Galán Ordax y Virginia Ahedo García

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. José Manuel Galán Ordax y Dª Virginia Ahedo García, profesores del departamento de Ingeniería Civil, área de Ingeniería de Organización.

Expone:

Que el alumno D. Lorena Bueno Porras con DNI 71307925J, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Sistema de información de apoyo a GreenMetrics.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 2 de junio de 2025

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. José Manuel Galán Ordax	Dª Virginia Ahedo García

Resumen

Este proyecto desarrolla un sistema automatizado de información que permite recopilar y generar evidencias de manera eficiente, utilizando los datos disponibles en la Universidad de Burgos. Este sistema se alineará con los criterios y métricas solicitadas por el ranking GreenMetrics, facilitando la evaluación de la sostenibilidad y otros indicadores claves exigidos por dicho ranking.

Descriptores

Servidor flask, python, aplicación web, creación de informes.

Abstract

Develop an automated information system that enables the efficient collection and generation of evidence, using the data available at the University of Burgos. This system will align with the criteria and metrics required by the GreenMetrics ranking, facilitating the assessment of sustainability and other key indicators demanded by this ranking.

Keywords

Flask server, python, web aplication, report creation.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

1.1 Contexto y Antecedentes

En el ámbito universitario, la gestión de información juega un papel fundamental en la toma de decisiones estratégicas. La Universidad de Burgos, comprometida con la sostenibilidad y la eficiencia, enfrenta el desafío de recopilar y analizar grandes volúmenes de datos con el fin de evaluar su desempeño en áreas clave. Sin embargo, los métodos tradicionales de recopilación suelen ser manuales, lentos y susceptibles a errores, lo que dificulta la obtención de resultados precisos y una gestión optimizada de la información.

Actualmente, la sostenibilidad es un aspecto clave en la evaluación de universidades a nivel internacional. Diversos rankings, como el GreenMetrics, establecen métricas que permiten medir el grado de sostenibilidad en función del uso de recursos, políticas medioambientales y eficiencia energética. Para mantenerse competitiva y fomentar una gestión eficiente de su impacto ambiental, la Universidad de Burgos necesita una herramienta que facilite la recopilación y análisis de datos de manera automática, fiable y transparente.

1.2 Objetivos del Proyecto

Este proyecto tiene como objetivo desarrollar un sistema automatizado de información, capaz de recopilar y generar evidencias de manera eficiente utilizando los datos disponibles en la universidad. La solución propuesta se alineará con los criterios y métricas del ranking GreenMetrics, facilitando una evaluación más precisa de la sostenibilidad y otros indicadores clave.

Los objetivos específicos del proyecto incluyen:

1. Automatizar la recopilación de información de fuentes institucionales relevantes.
2. Reducir la carga de trabajo manual en la generación de evidencias.
3. Facilitar la toma de decisiones basada en métricas fiables y alineadas con los estándares internacionales de sostenibilidad.

1.3 Enfoque del Proyecto

El enfoque adoptado para el desarrollo del sistema se basa en la necesidad de automatización y accesibilidad en la gestión de información. Dado que la recopilación manual de datos supone una carga significativa, se ha optado por integrar tecnología de procesamiento de información, almacenamiento en bases de datos y herramientas de visualización de datos, permitiendo que la gestión de indicadores de sostenibilidad sea más rápida y eficaz.

Este enfoque está justificado por los siguientes factores:

1. Optimización del tiempo y recursos dentro de la universidad.
2. Fiabilidad de los datos obtenidos mediante sistemas automatizados.
3. Facilitación de la evaluación universitaria en rankings internacionales como GreenMetrics.

1.4 Metodología del Proyecto

Para llevar a cabo el desarrollo del sistema, se ha seguido una metodología basada en principios ágiles (Scrum), permitiendo iteraciones rápidas y mejorando la adaptación del proyecto a los requerimientos de los usuarios.

La metodología concreta incluye:

- Análisis de requisitos: Evaluación de métricas de sostenibilidad.
- Diseño y planificación: Estructuración de la arquitectura del sistema y herramientas utilizadas.
- Implementación: Desarrollo del sistema con integración en bases de datos y plataformas web.
- Pruebas y evaluación: Validación de la funcionalidad y ajuste de los parámetros de recopilación de datos.

1.5 Estructura de la memoria

La memoria sigue la siguente estructura:

- **Introducción** Presentación del problema a abordar y la solución planteada. Además, se detalla la organización de la memoria y los materiales complementarios que se incluyen.
- **Objetivos del Proyecto** Explicación de los propósitos del proyecto, destacando las metas que se buscan alcanzar con su desarrollo.
- **Conceptos Teóricos** Descripción concisa de los fundamentos teóricos esenciales para entender la solución propuesta y su aplicación en el proyecto.
- **Técnicas y Herramientas** Lista de metodologías y herramientas empleadas en la planificación, gestión y desarrollo del proyecto.
- **Aspectos Destacables del Desarrollo** Exposición de los eventos más relevantes ocurridos durante la ejecución del proyecto, resaltando aquellos que influyeron en su evolución.
- **Trabajos Relacionados** Se incluye un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.
- **Conclusiones y Líneas de Trabajo Futuras** Síntesis de los resultados obtenidos tras la realización del proyecto, así como posibles mejoras o expansiones de la solución desarrollada.

1.6 Estructura de los anexos

Los anexos que acompañan la memoria incluyen:

- **Plan del proyecto software:** Análisis de viabilidad y planificación temporal.
- **Especificación de requisitos del software:** Detalles sobre la fase de análisis, objetivos generales, catálogo de requisitos del sistema y especificación de requisitos funcionales y no funcionales.
- **Especificación de diseño:** Explicación de la fase de diseño, incluyendo el alcance del software, arquitectura, estructura de datos y procedimientos implementados.

- **Manual del programador:** Información clave sobre el código fuente, su estructura, proceso de compilación, instalación y ejecución, además de pruebas realizadas.
- **Manual de usuario:** Guía para el correcto uso de la aplicación y sus funcionalidades principales.
- **Sostenibilización curricular:** recoge una reflexión personal sobre los aspectos de sostenibilidad abordados en el proyecto.

1.7 Resumen de Resultados y Conclusiones

Gracias a la implementación del sistema automatizado de información, la Universidad de Burgos podrá evaluar de manera eficiente sus indicadores de sostenibilidad, alineándose con los estándares de GreenMetrics. La automatización reduce errores y optimiza la accesibilidad de los datos, permitiendo una toma de decisiones más informada y estratégica.

Este proyecto no solo mejora la evaluación institucional, sino que también aporta una herramienta clave para el desarrollo de políticas de sostenibilidad dentro de la universidad, facilitando la gestión y análisis de información de manera moderna y eficaz.

2. Objetivos del proyecto

En esta sección se expondrán los objetivos que se busca alcanzar con el proyecto, denominados objetivos generales, así como aquellos requerimientos específicos que permitirán su cumplimiento, los cuales se identificarán como objetivos técnicos.

2.1 Objetivos generales

Los objetivos generales de este proyecto son:

- Desarrollar una aplicación que permita automatizar la creación de evidencias.
- Almacenar las evidencias generadas de forma estructurada.
- Facilitar la visualización de las guías docentes descargadas.
- Asegurar la escalabilidad del sistema, permitiendo su adaptación a futuras necesidades.

2.2 Objetivos técnicos

Los objetivos técnicos de este proyecto son:

- Optimizar la eficiencia del código mediante prácticas de desarrollo limpio y modular.
- Implementar autenticación y gestión de permisos para usuarios dentro de la aplicación.

- Desarrollar una interfaz de usuario amigable basada en principios de usabilidad y diseño responsivo.
- Implementar soporte para la exportación de evidencias en formatos estándar (PDF, CSV, JSON).
- Utilizar Git como sistema de control de versiones distribuido junto con la plataforma GitHub.
- Desarrollar un código que permita generar evidencias de distintos tipo, utilizando LLM, API y web scrapping.
- Hacer uso de herramientas de integracion continua como SonarQube en el repositorio.
- Aplicar la metodología ágil Scrum.
- Utilizar ZenHub como herramienta de gestion de proyectos.
- Realizar una página web para la aplicación.

3. Conceptos teóricos

A lo largo de esta memoria estaremos haciendo referencia a las evidencias utilizadas por GreenMetric. A continuación explicaremos que son y en que consisten.

3.1 Evidencias

GreenMetric clasifica la sostenibilidad universitaria en **seis categorías principales**, cada una con sus propias evidencias y requisitos técnicos:

1. Infraestructura y Entorno

Esta categoría evalúa la **proporción de espacios verdes dentro del campus y las políticas de conservación ambiental**. Las evidencias incluyen:

- Planos y fotografías georreferenciadas de áreas verdes.
- Registros de mantenimiento y conservación de espacios naturales.
- Informes sobre eficiencia en el uso del suelo y planificación urbana.

2. Energía y Cambio Climático

Se analiza el **consumo energético de la universidad y su compromiso con la reducción de emisiones de carbono.** Las evidencias incluyen:

- Datos de consumo energético obtenidos de sensores IoT y sistemas de monitoreo.

- Informes sobre uso de energías renovables y eficiencia energética.
- Modelos predictivos de reducción de huella de carbono, basados en análisis de datos históricos.

3. Gestión de Residuos

Evalúa la implementación de **programas de reciclaje y reducción de residuos**. Las evidencias incluyen:

- Registros de gestión de residuos peligrosos, con trazabilidad mediante blockchain.
- Datos de reciclaje y reducción de desechos, almacenados en bases de datos estructuradas.
- Sistemas de clasificación automática de residuos, utilizando visión artificial.

4. Uso del Agua

Se mide la **eficiencia en el consumo y reutilización del agua** dentro del campus. Las evidencias incluyen:

- Registros de consumo hídrico, obtenidos mediante sensores inteligentes.
- Informes sobre sistemas de reutilización de agua, con simulaciones de impacto ambiental.
- Modelos de optimización del uso del agua.

5. Transporte

Evalúa el **uso de medios de transporte sostenibles** dentro de la universidad. Las evidencias incluyen:

- Datos sobre movilidad estudiantil, obtenidos mediante análisis de patrones de desplazamiento.
- Registros de uso de bicicletas y vehículos eléctricos, con integración en sistemas GIS.
- Simulaciones de impacto ambiental, basadas en modelos de tráfico y consumo energético.

6. Educación e Investigación

Se analiza la **integración de la sostenibilidad en la oferta académica y la producción científica**. Las evidencias incluyen:

- Listado de cursos y programas académicos relacionados con sostenibilidad.
- Publicaciones científicas sobre medio ambiente, indexadas en bases de datos especializadas.
- Indicadores de impacto de la investigación.

De todas esas categorías se han implementado los siguientes informes:

1. Infraestructur y Entorno
    -
2. Educación e Investigación
    - **6_1 - Cursos sobre sostenibilidad** Se ha desarrollado un sistema automatizado para recopilar información sobre cursos y programas académicos relacionados con sostenibilidad. La solución permite extraer datos de los planes de estudio y generar evidencias sobre la oferta educativa en esta área. Se han implementado técnicas de web scraping para obtener información de los sitios web.
    - **6_2 -Número total de cursos/asignaturas ofrecidas** Se ha desarrollado un sistema automatizado para recopilar y estructurar la información sobre la oferta total de asignaturas en la universidad. Esto permitirá calcular con precisión la proporción de cursos dedicados a la sostenibilidad en relación con la oferta total. Para ello, se cuenta el número de asignaturas almacenadas en la base de datos que se generá a medida que se van descargando las guías docentes.
    - **6_3 - Proporción de cursos de sostenibilidad respecto al total de cursos** A partir de los datos obtenidos en los indicadores anteriores, se ha implementado un módulo analítico que calcula la ratio de cursos de sostenibilidad respecto al número total de asignaturas. Este análisis es clave para evaluar el compromiso de la universidad con la educación en sostenibilidad y su alineación con los criterios de GreenMetric.

- **6_4 Fondos de investigación dedicados a la sostenibilidad** Este indicador evalúa la inversión financiera dedicada a la investigación en sostenibilidad dentro de la universidad. Para ello, el sistema procesa la información contenida en un fichero Excel, donde se registran todos los proyectos de investigación y sus respectivos fondos asignados.

  A partir de estos datos, se genera una evidencia estructurada que presenta el monto total destinado a proyectos sostenibles en comparación con el financiamiento global para investigación, desglosado por años. Además, se calcula el ratio de inversión en sostenibilidad, proporcionando una métrica cuantificable que permite evaluar el compromiso institucional con el desarrollo de estudios ambientales y sostenibles.
- **6_7 - Número de publicaciones científicas sobre sostenibilidad** Este indicador evalúa el número de publicaciones científicas en temas ambientales y de sostenibilidad. Se ha desarrollado un sistema de procesamiento de datos bibliográficos, que recopila el número de artículos publicados por la Universidad de Burgos en Google Scholar.
- **6_8 - Número de eventos relacionados con sostenibilidad** Este indicador mide la cantidad de eventos organizados por la universidad en torno a la sostenibilidad, incluyendo conferencias y talleres. Para ello es necesario tener acceso a los informes de UBU Verde, en los cuales viene un resumen de las actividades realizadas. Entonces un LLM, irá leyendo dichas actividades y clasificando aquellas que son sostenibles, para luego contarlas y generar la evidencia.

3.2 LLM

3.3 Selenium

4. Técnicas y herramientas

4.1 Metodología

Scrum

Scrum es un marco de trabajo ágil enfocado en el desarrollo de software que sigue un enfoque iterativo e incremental. Organiza el trabajo en ciclos llamados sprints, en los que se planifican, desarrollan y revisan avances del proyecto, permitiendo una mejora continua y una rápida adaptación a los cambios.

4.2 Tipo de arquitectura

Opciones: aplicación web o aplicación de escritorio.

Elegida: aplicación web.

La elección de una arquitectura web frente a una de escritorio en el desarrollo de este proyecto responde a múltiples ventajas que ofrece este enfoque.

En primer lugar, una aplicación web proporciona mayor impacto visual, ya que permite una interfaz más dinámica y moderna. Su accesibilidad multiplataforma garantiza que los usuarios puedan acceder a la aplicación desde cualquier dispositivo con un navegador sin preocuparse por compatibilidades ni instalaciones específicas.

Además, la arquitectura web facilita la integración de diversas tecnologías tanto en el backend como en el frontend, en el diseño de la base de datos y en la visualización de datos. Esto permite utilizar frameworks avanzados y herramientas especializadas para mejorar la experiencia de usuario y optimizar el rendimiento del sistema.

Finalmente, el proyecto cuenta con un alto grado de escalabilidad, lo que permite su expansión para que otros usuarios puedan utilizarlo o adaptarlo a nuevas necesidades. La arquitectura web favorece la modularidad y la evolución de la aplicación sin depender de entornos específicos, asegurando una solución flexible y preparada para el crecimiento futuro.

4.3 Framework

Opciones: Flask o Django

Elegida: Flask

Tras analizar la comparación entre Flask y Django (tabla 4.1), se ha optado por Flask como el framework más adecuado para este proyecto. La decisión se fundamenta en su simplicidad y flexibilidad, lo que permite desarrollar una aplicación sin una estructura rígida ni una sobrecarga innecesaria de funcionalidades preconfiguradas [12, 10, 7].

Flask destaca por su curva de aprendizaje más accesible, lo que facilita una implementación rápida sin necesidad de comprender una arquitectura compleja. Además, su configuración inicial ligera permite al desarrollador elegir únicamente las herramientas necesarias, evitando la inclusión de componentes innecesarios [12].

Si bien Django ofrece una solución más completa y escalable desde el inicio, Flask proporciona mayor control y adaptabilidad, lo que resulta ideal para proyectos pequeños o medianos que no requieren una infraestructura robusta. Su ecosistema basado en extensiones permite integrar tecnologías específicas en el backend, frontend, diseño de bases de datos y visualización de datos, asegurando una solución personalizada y eficiente [7].

Por último, la elección de Flask también responde a la necesidad de un desarrollo ágil y escalable, permitiendo futuras expansiones sin depender de una estructura predefinida. Gracias a su enfoque minimalista, el proyecto puede evolucionar de manera flexible, adaptándose a nuevas necesidades sin restricciones impuestas por el framework [12, 10, 7].

Aspecto	**Flask**	**Django**
Enfoque	Minimalista y flexible.	Todo integrado.
Curva de aprendizaje	Simple al inicio, poca estructura.	Requiere entender su arquitectura.
Escalabilidad	Ideal para proyectos pequeños/-medianos.	Diseñado para escalar desde el inicio.
Configuración	Ligera, personalizada.	Completa y preconfigurada.
Ecosistema	Requiere añadir extensiones.	Funcionalidades integradas.
Flexibilidad	Total control del diseño.	Convenciones predefinidas.
Uso típico	Startups y proyectos a medida.	Aplicaciones robustas y empresariales.

Tabla 4.1: Comparativa entre Microframework y Framework completo

4.4 Lenguaje de programación

Python

Python [11] es un lenguaje de programación interpretado, de alto nivel y orientado a objetos, ampliamente utilizado en diversos ámbitos del desarrollo de software. Su diseño enfatiza la legibilidad del código y la facilidad de uso, lo que lo convierte en una opción popular tanto para principiantes como para desarrolladores experimentados. Alguna de sus ventajas son

- **Sintaxis clara y sencilla**: Su estructura intuitiva facilita la escritura y comprensión del código, reduciendo la curva de aprendizaje.
- **Multiplataforma**: Compatible con Windows, macOS y Linux, lo que permite desarrollar aplicaciones sin preocuparse por la compatibilidad del sistema operativo.
- **Extensa biblioteca estándar**: Ofrece módulos para manipulación de archivos, redes, bases de datos, inteligencia artificial y más, evitando la necesidad de desarrollar funcionalidades desde cero.
- **Versatilidad**: Se emplea en desarrollo web, ciencia de datos, inteligencia artificial, automatización y análisis de datos, entre otros.

4.5 Documentación

Opciones: LateX, Microsoft Word.

Elegida: LateX.

Se ha optado por LaTeX como la herramienta más adecuada para la redacción del documento debido a su capacidad de generar documentos estructurados y profesionales, especialmente en el ámbito académico y científico. Su integración con Overleaf, un editor en línea, facilita la colaboración y edición sin necesidad de instalar software adicional [15]. Aunque Microsoft Word es más intuitivo y accesible, LaTeX ofrece mayor control sobre el formato, gestión avanzada de referencias y mejor manejo de ecuaciones matemáticas, lo que lo convierte en la opción ideal para este proyecto [1, 14].

4.6 Desarrollo web

Github

GitHub es una de las plataformas más utilizadas para el desarrollo colaborativo de software, basada en el sistema de control de versiones **Git**. Su popularidad se debe a su facilidad de uso, integración con herramientas de desarrollo y comunidad extensa [5, 13].

Entre sus principales ventajas se encuentran la **colaboración eficiente**, que permite que múltiples desarrolladores trabajen en un mismo proyecto sin conflictos, y la **integración con CI/CD**, facilitando la automatización de pruebas y despliegues [5]. Además, su **amplia comunidad** proporciona documentación y soporte, lo que lo convierte en una opción ideal para proyectos open-source [13].

Sin embargo, GitHub también presenta algunas desventajas. Su adquisición por parte de Microsoft generó preocupaciones sobre privacidad y control, lo que llevó a algunos desarrolladores a migrar a alternativas como GitLab [5]. Además, aunque ofrece repositorios gratuitos, algunas funcionalidades avanzadas requieren suscripción [13].

A pesar de estas limitaciones, GitHub sigue siendo la opción preferida para la mayoría de desarrolladores debido a su **popularidad, facilidad de uso y ecosistema robusto**. Su integración con herramientas de desarrollo y su comunidad extensa lo convierten en una plataforma ideal para proyectos colaborativos [5, 13].

ZenHub

ZenHub es una herramienta de gestión de proyectos que se integra directamente con **GitHub**, permitiendo a los equipos de desarrollo organizar tareas,

visualizar el progreso y mejorar la colaboración sin salir de la plataforma [2, 8, 4].

Entre sus principales ventajas se encuentran la **gestión visual de proyectos**, que ofrece tableros Kanban y gráficos de seguimiento para facilitar la planificación y ejecución de tareas [2]. Además, su automatización de flujos de trabajo permite la creación de epics, seguimiento de velocidad y generación de informes basados en datos reales [8].

Sin embargo, ZenHub también presenta algunas desventajas. Su dependencia de GitHub limita su utilización en otros entornos, ya que solo es útil para equipos que ya trabajan en esta plataforma [4]. Además, aunque su interfaz es intuitiva, algunas funcionalidades avanzadas requieren tiempo para dominarse [2].

A pesar de estas limitaciones, ZenHub ha sido elegido para este proyecto debido a su integración directa con GitHub, lo que permite una gestión eficiente del código y las tareas sin necesidad de herramientas externas [8]. Su enfoque en **metodologías ágiles**, junto con su capacidad para automatizar flujos de trabajo y generar informes detallados, lo convierten en una opción ideal para equipos de desarrollo que buscan mejorar la productividad y la colaboración [4].

4.7 Servicios de integración continua

Calidad de código

Opciones: Codeclimate, SonarQube y Codacy.

Elegida: SonarQube.

Tras analizar las opciones disponibles 4.2, SonarQube ha sido seleccionado como la herramienta más adecuada para este proyecto. Su capacidad para realizar análisis estático avanzado, detectar vulnerabilidades de seguridad y ofrecer métricas detalladas lo convierten en la mejor opción para garantizar la calidad del código [17, 18, 3].

Además, su compatibilidad con múltiples lenguajes y su integración con herramientas de desarrollo permiten una implementación eficiente en entornos empresariales y proyectos de gran escala.

Aunque requiere una curva de aprendizaje más pronunciada, su potencia y flexibilidad justifican su elección frente a CodeClimate y Codacy [18].

Por último, **SonarQube** destaca por su enfoque en la **detección de código duplicado, cobertura de pruebas y análisis de calidad**, aspec-

tos fundamentales para el mantenimiento y evolución del software [3]. Su capacidad para integrarse con sistemas de integración continua y monitoreo garantiza que la calidad del código se mantenga en cada fase del desarrollo.

Característica	**CodeClimate**	**SonarQube**	**Codacy**
Enfoque	Análisis de calidad de código y deuda técnica	Análisis estático de código con métricas avanzadas	Revisión automática de código con enfoque en seguridad
Lenguajes soportados	Soporta principalmente Ruby, JavaScript y Python	Compatible con más de 25 lenguajes de programación	Compatible con más de 40 lenguajes
Integración	GitHub, GitLab, Bitbucket	GitHub, GitLab, Bitbucket, Jenkins, Azure DevOps	GitHub, GitLab, Bitbucket, Slack
Análisis de seguridad	Básico	Avanzado, con detección de vulnerabilidades	Enfocado en seguridad y cumplimiento de estándares
Facilidad de uso	Interfaz intuitiva, pero con menos opciones avanzadas	Completo y configurable, requiere más aprendizaje	Fácil de usar, con configuración automática
Escalabilidad	Adecuado para proyectos pequeños y medianos	Ideal para grandes proyectos y entornos empresariales	Funciona bien en proyectos de cualquier tamaño
Costo	Modelo de suscripción	Versión gratuita y de pago con más funcionalidades	Modelo de suscripción con opciones gratuitas

Tabla 4.2: Comparativa de herramientas de análisis de código

4.8 Base de datos

En el desarrollo de aplicaciones web, la elección del sistema de gestión de bases de datos es fundamental para garantizar un rendimiento óptimo y una administración eficiente de los datos. Dos de las opciones más utilizadas son **SQLite** y **MySQL**, cada una con características específicas que las hacen adecuadas para distintos tipos de proyectos [16, 9, 6].

SQLite es un sistema de gestión de bases de datos ligero y embebido que no requiere un servidor dedicado. Su simplicidad y facilidad de uso lo convierten en una opción ideal para aplicaciones con bajo volumen de datos

y acceso limitado [16]. Además, su almacenamiento en un único archivo facilita la portabilidad y la integración en aplicaciones móviles y web ligeras.

MySQL, por otro lado, es un sistema de gestión de bases de datos relacional más robusto, diseñado para manejar grandes volúmenes de datos y múltiples conexiones simultáneas. Es ampliamente utilizado en aplicaciones web que requieren escalabilidad y seguridad avanzada [9]. Su arquitectura cliente-servidor permite una administración eficiente de los datos y una mejor gestión de usuarios.

A continuación, se presenta una tabla comparativa 4.3 con las principales diferencias entre ambos sistemas:

Característica	**SQLite**	**MySQL**
Enfoque	Base de datos ligera y sin servidor	Sistema de gestión de bases de datos robusto y escalable
Instalación	No requiere instalación, funciona con archivos individuales	Requiere instalación y configuración de un servidor
Concurrencia	Limitada, no ideal para múltiples conexiones simultáneas	Diseñado para manejar múltiples conexiones de usuarios
Seguridad	Menos opciones de autenticación y permisos	Seguridad avanzada con autenticación y cifrado
Escalabilidad	Adecuado para aplicaciones pequeñas y móviles	Ideal para aplicaciones web y sistemas empresariales
Consumo de recursos	Bajo, funciona con archivos locales	Mayor consumo de memoria y procesamiento

Tabla 4.3: Comparación entre SQLite y MySQL

Para este proyecto, se ha optado por **SQLite** debido a su facilidad de implementación, bajo consumo de recursos y portabilidad. Las razones principales para su elección son:

- **Simplicidad y facilidad de uso**: No requiere configuración de servidor, lo que reduce la complejidad del desarrollo [16].

- **Bajo consumo de recursos**: Ideal para aplicaciones web ligeras que no necesitan manejar grandes volúmenes de datos [9].
- **Portabilidad**: Almacena los datos en un único archivo, lo que facilita la migración y el mantenimiento [6].
- **Fiabilidad en almacenamiento**: Ofrece un mecanismo seguro para el almacenamiento de datos, evitando pérdidas en caso de fallos inesperados [16].

Aunque **MySQL** es una opción más robusta para aplicaciones con múltiples usuarios y grandes volúmenes de datos, **SQLite** ha sido la mejor opción para este proyecto debido a su **ligereza y facilidad de uso**.

5. Aspectos relevantes del desarrollo del proyecto

La parte del proyecto con mayor complejidad teórica y de ingeniería radica en el diseño de la pipeline ETL (Extract–Transform–Load) que integra múltiples tecnologías para extraer información de la web, procesarla y almacenarla de forma estructurada. Esta pipeline se divide en cuatro fases claramente diferenciadas:

1. Extracción de enlaces de grados y másteres
2. Descarga y registro de guías docentes
3. Procesamiento y extracción de datos de los PDFs
4. Gestión de la ejecución

Extracción de enlaces de grados y másteres

Objetivo: obtener de forma automatizada los enlaces a los planes de estudio publicados en la web de la UBU.

1. Conexión HTTP/1.1 sobre TLS
    - Utilizamos el módulo http.client para establecer una conexión HTTPS sin dependencias externas.
    - Se envía una petición GET, se comprueba el código de estado (200 OK) y se decodifica la respuesta en UTF-8.
2. Parsing mediante expresiones regulares

- Con extraemos todas las URLs contenidas en atributos href.
- Esta aproximación “blind” es muy rápida, pero requiere filtros post-procesado para descartar URLs irrelevantes.

3. Filtrado de enlaces
   - **Inclusión**: sólo mantenemos URLs que contengan términos como "grado.º "master".
   - **Exclusión**: descartamos aquellas que incluyan substrings ("mailto:", .ªcceso-admision", etc.) según listas predefinidas.
   - Esta fase aplica el principio de set theory para operaciones de unión e intersección de conjuntos de cadenas.
4. Normalización de rutas
   - Convertimos enlaces relativos en absolutos concatenando el host (“https://www.ubu.es”).
5. Carga en DataFrame y persistencia
   - Creamos un pandas.DataFrame con los enlaces filtrados.
   - Exportamos a Excel (.xlsx) usando to_excel(engine=’openpyxl’) para facilitar el intercambio de datos.

3.2 Descarga y registro de guías docentes

Objetivo: recorrer cada enlace de asignatura, localizar y descargar la guía docente en formato PDF, y almacenar metadatos en base de datos.

1. Lectura de enlaces
   - Importamos los archivos Excel generados en la fase anterior —uno para grados y otro para másteres— y los concatenamos si se solicita el modo “ambos”.
2. Automatización de navegador sin interfaz
   - Empleamos Selenium con ChromeDriver en modo headless (sin GPU, sin sandbox) para simular interacciones de usuario.
   - El WebDriver navega a la página . . . /informacion-basica/guias-docentes y extrae elementos <a> con XPATH //a[contains(@href,’asignatura’)].
3. Construcción de URL de descarga

- A partir del parámetro asignatura=XXX en la URL se genera la ruta directa al PDF: `re.findall(r"href="([` textquotedbl]+)"", html)

4. Gestión de peticiones HTTP con requests
   - Se descarga el contenido del PDF (bytes) y se escribe en disco con open(..., "wb").
   - Control de errores HTTP mediante response.raise_for_status().
5. Persistencia en base de datos
   - Integramos un modelo Flask-SQLAlchemy (Busqueda) con campos: año, tipo de programa, código, modalidad, nombre de archivo.
   - Antes de insertar, comprobamos existencia para evitar duplicados (cláusula SELECT ... WHERE).
   - En caso de violación de integridad, capturamos IntegrityError.
6. Salida a Excel resumen
   - Generamos un .xlsx con las filas procesadas: enlace original, código, modalidad y nombre de archivo.

3.3 Procesamiento y extracción de datos de los PDFs

Objetivo: a partir de cada guía PDF, extraer campos clave (titulación, denominación, código y páginas marcadas) y generar un dataset enriquecido.

1. Lectura de ficheros PDF
   - Utilizamos PyPDF2.PdfReader para abrir cada archivo y extraer texto de cada página.
   - Omitimos páginas sin contenido (extract_text() is None) para optimizar.
2. Extracción con expresiones regulares
   - Definimos patrones con lookbehind para:
     - Denominación de la asignatura: (?<=1. Denominación de la asignatura:).*
     - Titulación: (?<=Titulación).*
     - Código: (?<=Código).*

- La técnica de regex lookarounds permite capturar sólo el contenido relevante sin delimitadores.

3. Detección de páginas con marcadores
   - Contamos asteriscos (*) en cada página y tomamos la primera página donde se detecte uno.
   - Este método sirve para identificar las páginas donde se mencionan competencias sostenibles, tablas o notas.
4. Actualización de DataFrame
   - Por cada fila del Excel de entrada, se invoca la función de procesamiento de PDF y se rellenan cuatro nuevas columnas.
   - En caso de error (archivo no encontrado, patrones no hallados), se registra en consola y se continúa sin abortar toda la ejecución.
5. Persistencia final
   - Exportamos un nuevo archivo Excel —p.ej. datos_asignaturas_grados_actualizado con la información completa y lista para análisis o visualización.

3.4 Gestión de la ejecución

Objetivo: asegurar que todas las fases se ejecuten en el orden correcto y manejar errores de forma centralizada.

1. .Entry point único
   - Los scripts asignaturas_sostenibles.py o API.py actúan como controladores principales, invocando en secuencia:
     a) grados.py
     b) guias_docentes.py
     c) procesadoAsignaturas.py
2. Gestión de procesos
   - Aprovechamos subprocess.run([...], check=True) para lanzar cada fase como un subproceso independiente.
   - Al incluir check=True, cualquier fallo de un subscript devuelve un CalledProcessError, que captura y detiene toda la pipeline (sys.exit(1)).

3. Ventajas de diseño

    - **Modularidad y desacoplamiento**: cada script maneja una responsabilidad única (SRP).
    - **Reutilización**: es sencillo integrar nuevos pasos o cambiar el orden de ejecución.
    - **Robustez**: control de errores local en cada fase y global en el módulo principal.

6. Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

7. Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía

[1] Baeldung. LaTeX vs. Word: Main Differences. "https://www.baeldung.com/cs/latex-vs-word-main-differences", 2024. [Internet; consultado 30-mayo-2025].

[2] Capterra. ZenHub - Opiniones, precios y características. "https://www.capterra.es/software/141621/zenhub", 2025. [Internet; consultado 30-mayo-2025].

[3] Codacy. Comparación entre Codacy y SonarQube. "https://www.codacy.com/comparison/codacy-vs-sonarqube", 2024. [Internet; consultado 28-mayo-2025].

[4] ComparaSoftware. ZenHub Proyectos: precios, funciones y opiniones. "https://www.comparasoftware.com/zenhub", 2025. [Internet; consultado 30-mayo-2025].

[5] Desde Linux. GitHub vs GitLab: ventajas y desventajas de estas plataformas. "https://blog.desdelinux.net/github-vs-gitlab/", 2025. [Internet; consultado 30-mayo-2025].

[6] Lureo Digital. Sqlite vs mysql: ¿cuál es la mejor opción para tu proyecto? https://lureodigital.com/sqlite-vs-mysql-cual-es-la-mejor-opcion-para-tu-proyecto/, 2025. Disponible en: https://lureodigital.com/sqlite-vs-mysql-cual-es-la-mejor-opcion-para-tu-proyecto/.

[7] Geekflare. Flask vs Django – ¿Qué Framework de Python elegir? "https://geekflare.com/es/flask-vs-django/", 2024. [Internet; consultado 28-mayo-2025].

[8] GetApp. Opiniones de ZenHub. "https://www.getapp.es/reviews/110953/zenhub", 2025. [Internet; consultado 30-mayo-2025].

[9] Hostinger. Sqlite vs mysql: Análisis a detalle para tu conveniencia. https://www.hostinger.com/es/tutoriales/sqlite-vs-mysql-cual-es-la-diferencia, 2025. Disponible en: https://www.hostinger.com/es/tutoriales/sqlite-vs-mysql-cual-es-la-diferencia.

[10] IONOS. Flask vs Django: comparativa de los frameworks de Python. "https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/flask-vs-django/", 2024.

[11] KeepCoding. Ventajas y desventajas de python. https://keepcoding.io/blog/ventajas-y-desventajas-de-python/, 2023. Accedido el 29 de mayo de 2025.

[12] Kinsta. Flask vs Django: ¿Cuál deberías usar para tu aplicación web? "https://kinsta.com/es/blog/flask-vs-django/", 2024. [Internet; consultado 28-mayo-2025].

[13] Kinsta. GitLab vs GitHub: Descubre Sus Principales Diferencias y Similitudes. "https://kinsta.com/es/blog/gitlab-vs-github/", 2025. [Internet; consultado 30-mayo-2025].

[14] Microsoft Community. Using LaTeX codes in Word. "https://answers.microsoft.com/en-us/msoffice/forum/all/using-latex-codes-in-word/c40aed4a-70d0-4929-a9f4-ebd912248d53", 2024. [Internet; consultado 30-mayo-2025].

[15] Orvium. ¿Por qué debería utilizar LaTeX en lugar de Word para escribir mi investigación? "https://blog.orvium.io/es/latex-sobre-word/", 2024. [Internet; consultado 30-mayo-2025].

[16] Solo Software Libre. Sqlite vs mysql: ¿qué base usar en apps web ligeras? https://solosoftwarelibre.com/sqlite-vs-mysql-que-base-usar-en-apps-web-ligeras/, 2025. Disponible en: https://solosoftwarelibre.com/sqlite-vs-mysql-que-base-usar-en-apps-web-ligeras/.

[17] SourceForge. Comparación entre SonarQube, Codacy y CodeClimate. "https://sourceforge.net/software/compare/Codacy-vs-Code-Climate-vs-SonarQube/", 2024. [Internet; consultado 28-mayo-2025].

[18] StackShare. Codacy vs Code Climate vs SonarQube. "https://stackshare.io/stackups/codacy-vs-code-climate-vs-sonarqube", 2024. [Internet; consultado 28-mayo-2025].