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 — 6 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, 6 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 [24] 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.

Evidencias implementadas

De todas esas categorías se han implementado los siguientes informes:

1. Infraestructur y Entorno
    - **1_19 - Actividades de mantenimiento de edificios**
      Se ha desarrollado una solución automatizada que recopila, procesa y organiza información sobre contratos de mantenimiento de edificios universitarios desde la plataforma de contratación de la Universidad de Burgos. El sistema realiza búsquedas automáticas en el portal institucional en función de distintos tipos de mantenimiento (extraídos dinámicamente de un documento .docx). Utiliza técnicas de web scraping y procesamiento de texto (con *BeautifulSoup*) para limpiar el contenido HTML y extraer datos clave como tipo de contrato, expediente, edificio y enlace. Luego, los datos son enviados a un modelo LLM que los clasifica en una estructura normalizada. Finalmente, se genera un informe en Word y PDF con una tabla resumen que incluye hipervínculos, proporcionando evidencia estructurada sobre las actividades de mantenimiento realizadas en el campus durante el año.
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

Los LLMs (Large Language Models) son modelos de inteligencia artificial avanzados entrenados con grandes volúmenes de datos textuales para comprender y generar lenguaje natural. Se basan en redes neuronales profundas, especialmente en la arquitectura Transformer, utilizada en modelos como GPT, PaLM o LLAMA.

Algunas de sus características principales son:

- Procesamiento del lenguaje natural (NLP): Capacidad para interpretar texto con gran precisión.
- Generación de contenido automatizado: Creación de textos estructurados y respuestas en lenguaje humano.
- Aprendizaje a gran escala: Entrenamiento en miles de millones de documentos para comprender el contexto.
- Interacción vía APIs: Integración con plataformas externas para mejorar la accesibilidad y automatización de tareas.

Para mejorar la recopilación de evidencias en el contexto de UI GreenMetric, se ha desarrollado un sistema basado en LLMs (Large Language Models) que permite automatizar la clasificación y análisis de datos relacionados con sostenibilidad.

Por defecto, este proyecto utiliza LLAMA, pero la configuración es flexible y permite la adaptación a otros modelos, como GPT, según las necesidades específicas de procesamiento y generación de información.

3.3 Selenium

Selenium es una herramienta de automatización que permite controlar navegadores web de forma programada, facilitando la interacción con sitios

web como si un usuario real estuviera operando el navegador. Se utiliza principalmente para automatización de pruebas, web scraping y bots que interactúan con plataformas web.

Selenium opera mediante scripts en Python, Java, JavaScript, entre otros, que controlan el navegador a través de un WebDriver específico para cada navegador (Chrome, Firefox, Edge, etc.).

En este proyecto, Selenium se usa junto a LLMs para la recopilación automática de evidencias de GreenMetric, accediendo a plataformas educativas y científicas para obtener datos sobre cursos, publicaciones y eventos relacionados con sostenibilidad.

3.4 API

Una API (Application Programming Interface) [20] es un conjunto de herramientas y definiciones que permite desarrollar software de manera estructurada. Su función principal es establecer una interfaz de comunicación entre diferentes aplicaciones, sin necesidad de conocer los detalles internos de su implementación.

Las APIs juegan un papel fundamental en la simplificación del desarrollo de software, ya que permiten reutilizar funcionalidades existentes y facilitar la interoperabilidad entre sistemas. Algunas de sus ventajas clave incluyen:

- Flexibilidad, permitiendo la integración con múltiples tecnologías.
- Facilidad en el diseño y uso de aplicaciones, al proporcionar interfaces bien definidas.
- Ahorro de tiempo y recursos, al evitar la necesidad de desarrollar desde cero ciertas funcionalidades.

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 [16, 14, 11].

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 [16].

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 [11].

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 [16, 14, 11].

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 [15] 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.

HTML, CSS y JavaScript

Para la construcción del sistema de información y su interfaz web, se han utilizado **HTML, CSS y JavaScript**. A diferencia de otras implementaciones que recurren a frameworks o plantillas predefinidas, en este proyecto se ha realizado el desarrollo de la interfaz de manera **totalmente manual**, asegurando un diseño personalizado y adaptado a las necesidades específicas del sistema.

Las características de estos lenguajes son:

- **HTML**: Lenguaje de marcado para la estructura de la página, permitiendo definir elementos como botones, formularios y contenedores de información.
- **CSS**: Utilizado para la personalización visual, proporcionando estilos a los elementos HTML sin depender de librerías externas.
- **JavaScript**: Lenguaje de programación que facilita la interactividad de la página, incluyendo validaciones de datos, actualización dinámica de contenidos y funcionalidad sin necesidad de recarga de la página.

El hecho de haber desarrollado la interfaz web desde cero sin usar plantillas preconfiguradas ha permitido una **mayor flexibilidad y control sobre el diseño y funcionalidad** del sistema. Este enfoque garantiza que cada componente de la plataforma se ajuste perfectamente a los requisitos del proyecto, optimizando la integración con los módulos de procesamiento automatizado.

Conclusión

El uso de **Python para el procesamiento de datos**, junto con una **implementación manual en HTML, CSS y JavaScript**, ha permitido la creación de un sistema eficiente, optimizado y adaptado completamente a las necesidades de GreenMetrics. La flexibilidad y el control total sobre la arquitectura web han sido factores clave para garantizar la correcta ejecución de los procesos de extracción y análisis de información.

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 [19]. 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, 18].

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 [9, 17].

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 [9]. Además, su **amplia comunidad** proporciona documentación y soporte, lo que lo convierte en una opción ideal para proyectos open-source [17].

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 [9]. Además, aunque ofrece repositorios gratuitos, algunas funcionalidades avanzadas requieren suscripción [17].

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 [9, 17].

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 [3, 12, 5].

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 [3]. Además, su automatización de flujos de trabajo permite la creación de epics, seguimiento de velocidad y generación de informes basados en datos reales [12].

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 [5]. Además, aunque su interfaz es intuitiva, algunas funcionalidades avanzadas requieren tiempo para dominarse [3].

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 [12]. 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 [5].

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 [22, 23, 4].

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 [23].

Por último, **SonarQube** destaca por su enfoque en la **detección de código duplicado, cobertura de pruebas y análisis de calidad**, aspectos fundamentales para el mantenimiento y evolución del software [4]. 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 [21, 13, 10].

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 [21]. 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 [13]. 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 [21].

- **Bajo consumo de recursos**: Ideal para aplicaciones web ligeras que no necesitan manejar grandes volúmenes de datos [13].
- **Portabilidad**: Almacena los datos en un único archivo, lo que facilita la migración y el mantenimiento [10].
- **Fiabilidad en almacenamiento**: Ofrece un mecanismo seguro para el almacenamiento de datos, evitando pérdidas en caso de fallos inesperados [21].

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**.

4.9 Editor de video

Canva

Canva [2] es una plataforma de diseño gráfico que permite crear contenido visual de manera sencilla y accesible. Con Canva, los usuarios pueden diseñar presentaciones, publicaciones para redes sociales, carteles, folletos, vídeos, currículums y mucho más sin necesidad de conocimientos avanzados en diseño.

Canva está disponible en versión gratuita y en planes de pago como Canva Pro y Canva para Equipos, que ofrecen funciones avanzadas para empresas y profesionales.

5. Aspectos relevantes del desarrollo del proyecto

En este apartado se van a recoger los aspectos más importantes del desarrollo del proyecto. Desde las decisiones que se tomaron y sus implicaciones, hasta los numerosos problemas a los que hubo que enfrentarse y como se solucionaron.

5.1. Inicio del Proyecto

El desarrollo del sistema de automatización para la recopilación de evidencias en UI GreenMetric comenzó con un análisis detallado de los indicadores requeridos y la metodología para su evaluación. Se identificaron las principales dificultades en la extracción manual de evidencias y se planteó la implementación de un sistema basado en LLMs (Large Language Models) y Selenium para optimizar el proceso.

5.2. Metodología

Al comienzo del proyecto se propuso utilizar una metodología ágil para la gestión del proyecto. De entre todas las opciones se escogió emplear Scrum, ya que permite entre otras cosas, entregar incrementos de producto funcionales de manera regular. También permite ajustar las prioridades en cada sprint para responder a nuevas necesidades que pudieran darse. Se ha aplicado de manera general aunque no ha podido seguirse al 100 % esta metodología debido a que no se cumplían todos los requisitos necesarios, como ser un equipo de 4 a 8 personas, hacer reuniones diarias. Se siguió una

estrategia de desarrollo incremental a través de sprints (iteraciones). Cada sprint tuvo una duración de dos semanas.

Al final de cada sprint se entregaba una parte funcional del producto. Al finalizar cada sprint se realizaban reuniones de revisión de dicha iteración, además de planificar el nuevo sprint. Durante la planificación del sprint se realizaba una pila de tareas que debían completarse durante esa iteración. Para monitorizar el avance del proyecto se utilizaron gráficos burndown.

5.3. Formación

Durante el desarrollo, se profundizó en técnicas de procesamiento de lenguaje natural (NLP) y automatización web con Selenium, permitiendo la extracción de datos en tiempo real desde plataformas académicas. Se exploraron arquitecturas eficientes para la integración de LLMs, optimizando el análisis de grandes volúmenes de información.

5.4. Flujo de Trabajo del Sistema

El proceso de generación de evidencias sigue un flujo estructurado que permite la extracción, transformación y carga de datos de manera automatizada (ver imagen 5.1):

1. **Usuario solicita una evaluación.**
2. **El sistema extrae datos relevantes de fuentes institucionales.**
3. **Se procesan los documentos y se identifican las evidencias.**
4. **Los resultados son almacenados en la base de datos y presentados en un informe.**

Este flujo es fundamental para la correcta ejecución de la pipeline ETL, garantizando la integridad y precisión de los datos obtenidos.

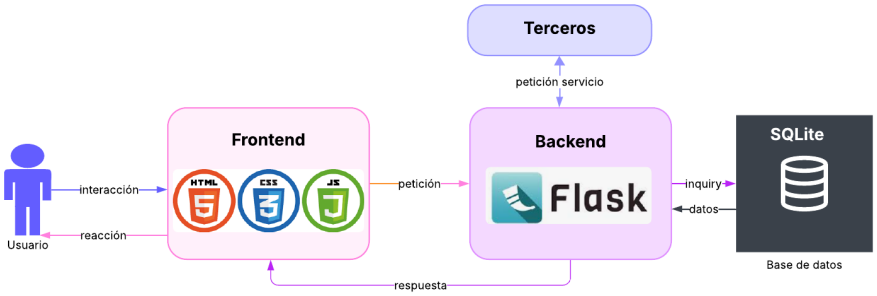

Figura 5.1: Diagrama de componentes

5.5. Implementación del Sistema

Evidencia 6_1 Number of Courses on Environment and Sustainability

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

5.4.1 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.

```python
def obtener_html(host, path):
    """
    Recupera el contenido HTML de una página web usando HTTP/1.1 sobre HTTPS.

    Args:
        host (str): El nombre de host del sitio web (ej. "www.ubu.es").
        path (str): La ruta a la página específica (ej. "/grados-ordenados-por-ramas-de-conocimiento").

    Returns:
        str: El contenido HTML de la página como una cadena decodificada en UTF-8.

    Raises:
        Exception: Si la solicitud HTTP falla (el código de estado no es 200).
    """
    conn = http.client.HTTPSConnection(host)
    conn.request("GET", path)
    response = conn.getresponse()
    if response.status == 200:
        return response.read().decode("utf-8")
    else:
        raise Exception(f"Error al obtener la página: {response.status}")
```

Figura 5.2: Conexión HTTP

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.

```python
# Obtener el HTML
html = obtener_html(host, path)

# Buscar todos los enlaces con una expresión regular
enlaces = re.findall(r'href="([^"]+)"', html)
```

Figura 5.3: Expresiones regulares utilizadas para la obtención de enlaces

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.

```
# Filtrar los enlaces
enlaces_filtrados = [
    enlace for enlace in enlaces
    if any(filtro in enlace for filtro in filtro_incluir) and
    not any(filtro in enlace for filtro in filtro_excluir)
]
```

Figura 5.4: Filtrado de enlaces

4. Normalización de rutas

- Convertimos enlaces relativos en absolutos concatenando el host ("https://www.ubu.es").

```
# Convertir enlaces relativos a absolutos
enlaces_absolutos = [
    enlace if enlace.startswith("http") else f"https://{host}{enlace}"
    for enlace in enlaces_filtrados
]
```

Figura 5.5: Normalización de rutas

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.

```
    # Guardar los enlaces en un archivo Excel dentro de la carpeta "data"
    archivo_completo = os.path.join(ruta_data, archivo_salida)
    try:
        df = pd.DataFrame(enlaces_absolutos, columns=["link"])
        df.to_excel(archivo_completo, index=False, engine="openpyxl")
        print(f"Enlaces guardados en '{archivo_completo}'")
    except Exception as e:
        print(f"Error al guardar el archivo Excel: {e}")

# Procesar los enlaces de los grados
host_grados = "www.ubu.es"
path_grados = "/grados-ordenados-por-ramas-de-conocimiento"
filtro_incluir_grados = ["grado"]
filtro_excluir_grados = ["grados", "acceso-admision", "mailto", "decreto", "servicio-de"]
archivo_salida_grados = "enlaces_filtrados_grados_ubu.xlsx"
procesar_enlaces(host_grados, path_grados, filtro_incluir_grados, filtro_excluir_grados, archivo_salida_grados)

# Procesar los enlaces de los másteres
host_masteres = "www.ubu.es"
path_masteres = "/estudios/oferta-de-estudios/masteres-universitarios-oficiales"
filtro_incluir_masteres = ["master"]
filtro_excluir_masteres = ["masteres", "mailto", "acceso-admision", "decreto", "servicio-de", "international-students"]
archivo_salida_masteres = "enlaces_filtrados_masteres_ubu.xlsx"
procesar_enlaces(host_masteres, path_masteres, filtro_incluir_masteres, filtro_excluir_masteres, archivo_salida_masteres)
```

Figura 5.6: Carga en Dataframe y exportación a excel

5.4.2 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”.

```
def get_excel_data(file_path, program_type=None):
    """Reads the Excel file and adds the program type."""
    data = pd.read_excel(file_path)
    if program_type:
        data['tipo_programa'] = program_type
    return data
```

Figura 5.7: Lectura de enlaces

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’)].

```python
def setup_chrome_driver():
    """Sets up and returns a headless Chrome WebDriver."""
    chrome_options = Options()
    chrome_options.add_argument("--headless")
    chrome_options.add_argument("--disable-gpu")
    chrome_options.add_argument("--no-sandbox")
    return webdriver.Chrome(options=chrome_options)
```

Figura 5.8: Configuración de Selenium

```python
def process_program_data(degrees, anho, tipo_estudio, ruta_data, ruta_guias):
    """Processes data for each program and stores the results."""
    subject_data = []
    for i, degree in degrees.iterrows():
        basic_link = str(degree[0])
        modalidad = "online" if "online" in basic_link else "presencial"
        driver = setup_chrome_driver()
        url = f"{basic_link}/informacion-basica/guias-docentes"
        driver.get(url)

        # Allow page to load
        time.sleep(5)

        # Find all links to the guides
        enlaces = driver.find_elements(By.XPATH, "//a[contains(@href, 'asignatura')]")
```

Figura 5.9: Utilización de Selenium

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)

```
url_descarga = f"https://ubuvirtual.ubu.es/mod/guiadocente/get_guiadocente.php?asignatura={url_asignatura.split('asignatura=')[-1].split('&')[0]}&cursoacademico={anho2}"
```

Figura 5.10: URL de descarga

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().

```python
def download_file(url, file_path):
    """Downloads a file from the given URL and saves it to the specified file path."""
    try:
        response = requests.get(url)
        response.raise_for_status()
        with open(file_path, "wb") as f:
            f.write(response.content)
        print(f"Guía descargada: {file_path}")
    except requests.exceptions.RequestException as e:
        print(f"Error al descargar el archivo: {e}")
```

Figura 5.11: Gestión de peticiones HTTP

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.

```python
def save_to_database(subject_data, anho, degrees, tipo_estudio):
    """Saves the subject data to the database if not already present."""
    for data in subject_data:
        try:
            with app.app_context():
                registro_existente = Busqueda.query.filter_by(
                    anho=anho,
                    codigo_asignatura=data['codigo_asignatura'],
                    modalidad=data['modalidad']
                ).first()

                if not registro_existente:
                    tipo_programa = data.get('tipo_programa', tipo_estudio)

                    nuevo_registro = Busqueda(
                        anho=anho,
                        tipo_programa=tipo_programa,
                        codigo_asignatura=data['codigo_asignatura'],
                        nombre_archivo=data['nombre_archivo'],
                        modalidad=data['modalidad']
                    )
                    db.session.add(nuevo_registro)
                    db.session.commit()
                else:
                    print(f"El registro para {data['codigo_asignatura']} ({data['modalidad']}) ya existe, ignorado.")
        except IntegrityError:
            print(f"Error de integridad para {data['codigo_asignatura']} ({data['modalidad']}).")
```

Figura 5.12: Añadir datos en la base de datos

6. Salida a Excel resumen
   - Generamos un .xlsx con las filas procesadas: enlace original, código, modalidad y nombre de archivo.

```python
def save_to_excel(subject_data, ruta_data, tipo_estudio):
    """Saves the processed data to an Excel file."""
    df_subjects = pd.DataFrame(subject_data)
    output_file = {
        'master': "datos_asignaturas_masteres.xlsx",
        'grado': "datos_asignaturas_grados.xlsx",
        'ambos': "datos_asignaturas_grados_masteres.xlsx"
    }.get(tipo_estudio, "datos_asignaturas_default.xlsx")

    df_subjects.to_excel(os.path.join(ruta_data, output_file), index=False)
    print(f"Datos guardados en {output_file}")

def procesar_guias(anho, tipo_estudio):
    """Main function to process the guides."""
    ruta_data = os.path.join("sostenibilidad", "data")
    ruta_guias = os.path.join(ruta_data, "guias")
    os.makedirs(ruta_guias, exist_ok=True)

    # Load program data
    if tipo_estudio == 'master':
        ruta_excel = os.path.join(ruta_data, "enlaces_filtrados_masteres_ubu.xlsx")
        degrees = get_excel_data(ruta_excel, 'master')
    elif tipo_estudio == 'grado':
        ruta_excel = os.path.join(ruta_data, "enlaces_filtrados_grados_ubu.xlsx")
        degrees = get_excel_data(ruta_excel, 'grado')
    elif tipo_estudio == 'ambos':
        grados = get_excel_data(os.path.join(ruta_data, "enlaces_filtrados_grados_ubu.xlsx"), 'grado')
        masteres = get_excel_data(os.path.join(ruta_data, "enlaces_filtrados_masteres_ubu.xlsx"), 'master')
        degrees = pd.concat([grados, masteres], ignore_index=True)

    subject_data = process_program_data(degrees, anho, tipo_estudio, ruta_data, ruta_guias)
    save_to_database(subject_data, anho, degrees, tipo_estudio)
    save_to_excel(subject_data, ruta_data, tipo_estudio)
```

Figura 5.13: Ficheros excel generados, y función principal

5.4.3 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.

```python
def obtener_ruta_excel(tipo_estudio, ruta_data):
    """
    Obtiene la ruta completa al archivo Excel de datos de asignaturas
    basado en el tipo de estudio.

    Args:
        tipo_estudio (str): El tipo de estudio ('master', 'grado', 'ambos').
        ruta_data (str): La ruta al directorio que contiene los archivos de datos.

    Returns:
        str: La ruta completa al archivo Excel.

    Raises:
        ValueError: Si el tipo de estudio no es válido.
        FileNotFoundError: Si el archivo Excel esperado no se encuentra en la ruta especificada.
    """
    archivos = {
        'master': "datos_asignaturas_masteres.xlsx",
        'grado': "datos_asignaturas_grados.xlsx",
        'ambos': "datos_asignaturas_grados_masteres.xlsx"
    }
    archivo = archivos.get(tipo_estudio)
    if not archivo:
        raise ValueError("Tipo de estudio no válido.")

    ruta_excel = os.path.join(ruta_data, archivo)
    if not os.path.exists(ruta_excel):
        raise FileNotFoundError(f"No se encontró el archivo {ruta_excel}")

    return ruta_excel

def leer_pdf(ruta_pdf):
    """
    Lee el texto de cada página de un archivo PDF.

    Args:
        ruta_pdf (str): La ruta completa al archivo PDF.

    Returns:
        list[str]: Una lista de strings, donde cada string es el texto extraído
                   de una página del PDF. Las páginas sin texto son omitidas.
    """
    with open(ruta_pdf, "rb") as file:
        reader = PdfReader(file)
        return [page.extract_text() for page in reader.pages if page.extract_text()]
```

Figura 5.14: Lectura de ficheros

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.

```python
def extraer_info_pdf(pdf_text):
    """
    Extrae información especifica (titulación, denominación, código y página con asteriscos)
    del texto de un PDF, asumiendo que la información clave está en la primera página.

    Args:
        pdf_text (list[str]): Una lista de strings, donde el primer string
                              contiene el texto de la primera página del PDF.

    Returns:
        tuple[str | None, str | None, str | None, int | None]: Una tupla que contiene:
            - titulacion (str | None): La titulación extraída, o None si no se encuentra.
            - denominacion (str | None): La denominación de la asignatura, o None si no se encuentra.
            - codigo (str | None): El código de la asignatura, o None si no se encuentra.
            - pagina_con_asteriscos (int | None): El número de la primera página que contiene
                                                  al menos un asterisco, o None si ninguna página lo contiene.
    """
    def extraer_patron(patron):
        """
        Función auxiliar para buscar un patrón regex en el texto de la primera página.
        """
        match = re.search(patron, pdf_text[0])
        return match.group().strip() if match else None

    denominacion = extraer_patron(r"(?<=1. Denominación de la asignatura:\n).*")
    titulacion = extraer_patron(r"(?<=Titulación\n).*")
    codigo = extraer_patron(r"(?<=Código\n).*")

    asteriscos = [text.count("*") for text in pdf_text if text]
    pagina_con_asteriscos = max((i + 1 for i, count in enumerate(asteriscos) if count > 0), default=None)

    return titulacion, denominacion, codigo, pagina_con_asteriscos
```

Figura 5.15: Expresiones regulares

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.

```python
def extraer_info_pdf(pdf_text):
    """
    Extrae información específica (titulación, denominación, código y página con asteriscos)
    del texto de un PDF, asumiendo que la información clave está en la primera página.

    Args:
        pdf_text (list[str]): Una lista de strings, donde el primer string
                              contiene el texto de la primera página del PDF.

    Returns:
        tuple[str | None, str | None, str | None, int | None]: Una tupla que contiene:
            - titulacion (str | None): La titulación extraída, o None si no se encuentra.
            - denominacion (str | None): La denominación de la asignatura, o None si no se encuentra.
            - codigo (str | None): El código de la asignatura, o None si no se encuentra.
            - pagina_con_asteriscos (int | None): El número de la primera página que contiene
                                                  al menos un asterisco, o None si ninguna página lo contiene.
    """
    def extraer_patron(patron):
        """
        Función auxiliar para buscar un patrón regex en el texto de la primera página.
        """
        match = re.search(patron, pdf_text[0])
        return match.group().strip() if match else None

    denominacion = extraer_patron(r"(?<=1. Denominación de la asignatura:\n).*")
    titulacion = extraer_patron(r"(?<=Titulación\n).*")
    codigo = extraer_patron(r"(?<=Código\n).*")

    asteriscos = [text.count("*") for text in pdf_text if text]
    pagina_con_asteriscos = max((i + 1 for i, count in enumerate(asteriscos) if count > 0), default=None)

    return titulacion, denominacion, codigo, pagina_con_asteriscos
```

Figura 5.16: Detección de páginas con marcadores

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.

```python
def procesar_fila(index, row, ruta_guias):
    """
    Procesa una única fila del DataFrame de asignaturas.

    Lee el nombre del archivo PDF de la fila, construye la ruta completa,
    lee el contenido del PDF y extrae la información relevante.

    Args:
        index (int): El índice de la fila en el DataFrame.
        row (pd.Series): La fila actual del DataFrame.
        ruta_guias (str): La ruta al directorio que contiene los archivos PDF de las guías.

    Returns:
        tuple[str | None, str | None, str | None, int | None]: Una tupla con la titulación,
            denominación, código y página con asteriscos extraídos del PDF.

    Raises:
        ValueError: Si el nombre del archivo PDF en la fila no es válido.
        FileNotFoundError: Si el archivo PDF especificado no existe.
        ValueError: Si no se pudo extraer texto del PDF.
    """
    nombre_pdf = row.get("nombre_archivo")

    if pd.isna(nombre_pdf) or not isinstance(nombre_pdf, str):
        raise ValueError(f"Nombre de PDF no válido en la fila {index}: {nombre_pdf}")

    ruta_pdf = os.path.join(ruta_guias, nombre_pdf)
    if not os.path.exists(ruta_pdf):
        raise FileNotFoundError(f"El archivo {ruta_pdf} no existe.")

    pdf_text = leer_pdf(ruta_pdf)
    if not pdf_text:
        raise ValueError(f"No se pudo extraer texto del PDF: {ruta_pdf}")

    return extraer_info_pdf(pdf_text)
```

Figura 5.17: Actualización de DataFrame

5. Persistencia final

   - Exportamos un nuevo archivo Excel —p.ej. datos_asignaturas_grados_actualizado.xlsx— con la información completa y lista para análisis o visualización.

```python
def guardar_excel(asignaturas, tipo_estudio, ruta_data):
    """
    Guarda el DataFrame procesado en un nuevo archivo Excel.

    El nombre del archivo de salida depende del tipo de estudio.

    Args:
        asignaturas (pd.DataFrame): El DataFrame con los datos de asignaturas procesados.
        tipo_estudio (str): El tipo de estudio ('master', 'grado', 'ambos').
        ruta_data (str): La ruta al directorio donde se guardará el archivo de salida.

    Raises:
        ValueError: Si el tipo de estudio no es válido para determinar el nombre del archivo de salida.
    """
    rutas_salida = {
        'master': "datos_asignaturas_masteres_actualizado.xlsx",
        'grado': "datos_asignaturas_grados_actualizado.xlsx",
        'ambos': "enlaces_filtrados_grados_masteres_ubu.xlsx"
    }
    salida = rutas_salida.get(tipo_estudio)
    if not salida:
        raise ValueError("Tipo de estudio no válido para guardar archivo.")

    ruta_excel_actualizado = os.path.join(ruta_data, salida)
    asignaturas.to_excel(ruta_excel_actualizado, index=False)

def procesar_asignaturas(tipo_estudio):
    """
    Función principal para procesar las asignaturas.

    Carga el archivo Excel de entrada, inicializa nuevas columnas para la información
    a extraer, itera sobre cada fila, llama a `procesar_fila` para obtener los datos
    del PDF asociado y actualiza el DataFrame. Finalmente, guarda el DataFrame procesado.

    Args:
        tipo_estudio (str): El tipo de estudio ('master', 'grado', 'ambos').
    """
    ruta_data = os.path.join("sostenibilidad", "data")
    ruta_excel = obtener_ruta_excel(tipo_estudio, ruta_data)
    asignaturas = pd.read_excel(ruta_excel)

    asignaturas["titulacion"] = None
    asignaturas["denominacion"] = None
    asignaturas["codigo"] = None
    asignaturas["pagina_con_asteriscos"] = None

    ruta_guias = os.path.join(ruta_data, "guias")

    for index, row in asignaturas.iterrows():
        try:
            titulacion, denominacion, codigo, pagina_con_asteriscos = procesar_fila(index, row, ruta_guias)

            asignaturas.at[index, "titulacion"] = titulacion
            asignaturas.at[index, "denominacion"] = denominacion
            asignaturas.at[index, "codigo"] = codigo
            asignaturas.at[index, "pagina_con_asteriscos"] = pagina_con_asteriscos

        except Exception as e:
            print(f"Error en el archivo para fila {index}: {str(e)}")
            continue

    guardar_excel(asignaturas, tipo_estudio, ruta_data)
```

Figura 5.18: Guardar ficheros y función principal

5.4.4 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.

```python
def ejecutar_script(script_name):
    """
    Ejecuta un script de Python especificado por su nombre y maneja posibles errores.

    Utiliza `subprocess.run` para ejecutar el script como un proceso separado.
    Si el script se ejecuta correctamente, imprime un mensaje de éxito.
    Si el script devuelve un código de salida distinto de cero (indicando un error),
    imprime un mensaje de error y sale del script actual con un código de error.

    Args:
        script_name (str): El nombre del script de Python a ejecutar.
                           Se espera que el script esté en el mismo directorio
                           o en una ruta accesible.
    """
    try:
        # Ejecuta el script y muestra la salida en consola
        print(f"Ejecutando {script_name}...")
        subprocess.run([sys.executable, script_name], check=True)
        print(f"{script_name} ejecutado con éxito.")
    except subprocess.CalledProcessError as e:
        print(f"Error al ejecutar {script_name}: {e}")
        sys.exit(1)

def ejecutar_orden():
    """
    Ejecuta una secuencia predefinida de scripts en un orden especifico.

    Esta función define el flujo de trabajo principal para obtener y procesar
    los datos de asignaturas y guías docentes llamando a `ejecutar_script`
    para cada paso.
    """
    # Paso 1: Ejecutar el script grados.py
    ejecutar_script('grados.py')

    # Paso 2: Ejecutar el script guias_docentes.py
    ejecutar_script('guias_docentes.py')

    # Paso 3: Ejecutar el script procesadoAsignaturas.py
    ejecutar_script('procesadoAsignaturas.py')
```

Figura 5.19: Script gestiona la ejecución ordenada de otros scripts

LLM

El sistema desarrollado permite la **extracción, análisis y clasificación automática de datos** contenidos en guías docentes en formato PDF. Se ha implementado un modelo de procesamiento basado en LLMs para identificar competencias de sostenibilidad y relacionarlas con los Objetivos de Desarrollo Sostenible (ODS) establecidos por la ONU [7].

Para ello, se han diseñado múltiples módulos que garantizan una ejecución eficiente y estructurada del sistema.

5.4.1 Configuración del sistema

El sistema se configura utilizando **Flask-SQLAlchemy** como ORM para gestionar la base de datos y 'load_dotenv()' para cargar variables de entorno desde un fichero '.env' (imagne 5.20. Entre las configuraciones iniciales se encuentran:

- Definición de la conexión con la base de datos mediante `DATABASE_URL`.
- Configuración de `SQLALCHEMY_TRACK_MODIFICATIONS` para mejorar el rendimiento.
- Importación de parámetros de configuración desde `config.py`.

```
# Configurar la base de datos manualmente si es necesario
app.config['SQLALCHEMY_DATABASE_URI'] = os.getenv("DATABASE_URL")
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
```

Figura 5.20: Configuración del sistema

5.4.2 Verificación de directorios y archivos

Antes de procesar los documentos, el sistema realiza una serie de verificaciones de entorno, asegurando la disponibilidad de los archivos PDF y la configuración correcta de rutas (imagen). Las comprobaciones incluyen:

- Validación de la existencia del directorio de almacenamiento de guías docentes (`data/guias`).
- Comprobación de la ruta del archivo de salida (`resultados_guias.xlsx`).
- Listado de archivos PDF disponibles para su procesamiento.

```python
# Definir rutas
directorio = os.path.join("sostenibilidad","data", "guias")  # Carpeta con los archivos PDF
archivo_salida = os.path.join("sostenibilidad","data", "resultados_guias.xlsx")  # Archivo de salida


# Verificar si el directorio de los archivos PDF existe
if not os.path.exists(directorio):
    print(f"El directorio de los archivos PDF no existe: {directorio}")
    sys.exit(1)  # Salir del programa si no existe el directorio

# Verificar si el archivo de salida es una ruta válida
if not os.path.isdir(os.path.dirname(archivo_salida)):
    print(f" La carpeta de salida no existe: {os.path.dirname(archivo_salida)}")
    sys.exit(1)  # Salir si no existe la carpeta
```

Figura 5.21: Verificación de directorios y archivos

5.4.3 Extracción y procesamiento de datos con IA

El sistema emplea modelos de lenguaje grande (LLMs) a través de una API externa para analizar los textos de las guías docentes (imagen . El procedimiento sigue los siguientes pasos:

- Extracción de texto de los documentos PDF con `PyPDF2`.
- Construcción del prompt para la IA, solicitando la identificación de competencias de sostenibilidad.
- Envío de la consulta a la API mediante una solicitud `POST` en formato JSON.
- Filtrado y procesamiento de la respuesta generada por la IA.

```
# Extraer texto del PDF
with open(pdf_path, "rb") as file:
    reader = PdfReader(file)
    pdf_text = " ".join([page.extract_text() for page in reader.pages[:2]])

if not pdf_text.strip():
    print(f"Advertencia: No se pudo extraer texto del archivo {pdf_file}. Saltando...")
    continue  # Si el PDF no tiene texto, pasamos al siguiente


# Construcción del prompt para la IAf
body = {
    "model": myModel,
    "messages": [
        {
            "role": "system",
            "content": (
                "You will receive the data of a teaching guide in Spanish. Do not translate it to English. "
                "Extract and return only the following fields:\n"
                "1. The name of the subject (without commas)\n"
                "2. The degree\n"
                "3. The code\n"
                "4. The curricular sustainability competencies\n"
                "Find section with the following title  Competencias que debe adquirir el alumno/a al cursar la asignatura. Identify \n"
                "only the competencies listed there that have an asterisk (*). \n"
                " If no competencies have an asterisk, return 'None'.\n"
                "If at least one competency has an asterisk, return only those that have one and for each competency, analyze its\n"
                "description and determine which of the following objectives it aligns with based on its content:\n"

                    "1. Fin de la pobreza\n"
                    "2. Hambre cero\n"
                    "3. Salud y bienestar\n"
                    "4. Educación de calidad\n"
                    "5. Igualdad de género\n"
                    "6. Agua limpia y saneamiento\n"
                    "7. Energía asequible y no contaminante\n"
                    "8. Trabajo decente y crecimiento económico\n"
                    "9. Industria, innovación e infraestructura\n"
                    "10. Reducción de las desigualdades\n"
                    "11. Ciudades y comunidades sostenibles\n"
                    "12. Producción y consumo responsables\n"
                    "13. Acción por el clima\n"
                    "14. Vida submarina\n"
                    "15. Vida de ecosistemas terrestres\n"
                    "16. Paz, justicia e instituciones sólidas\n"
                    "17. Alianzas para lograr los objetivos\n"
                "If a competency clearly aligns with one or more objectives, list them next to the competency. SDG X (Y) - Competency Where:\n"
                    "X is the number of the corresponding objective.\n"
                    "Y is the name of the objective.\n"
                    "Competency is the code of the competency\n"
                "If no clear alignment is found, leave that field empty. If no asterisk return None\n"
                "Return the four fields separated by commas in this format:\n"
                "Subject name, degree, code, Curricular sustainability competencies\n"
                "Ensure that you return ONLY these four fields in a single line, without extra text."
            )
        },
        {
            "role": "user",
            "content": pdf_text
        }
    ],
    "temperature": 0.2
}
```

Figura 5.22: Extracción y procesamiento de datos

5.4.4 Validación y almacenamiento de resultados

Los datos extraídos son estructurados y almacenados para facilitar su análisis posterior (imagen . Se garantiza la integridad mediante:

- Verificación de registros previos antes de insertar nuevos datos (`SELECT ... WHERE`).
- Clasificación de asignaturas según su relevancia en sostenibilidad.
- Almacenamiento en un archivo Excel para generar informes y facilitar su revisión.

```python
# Validar si hay al menos 3 elementos (para evitar errores)
if len(contenido_separado) >= 3:
    competences = contenido_separado[3] if len(contenido_separado) > 3 else "None"

    # Aquí detectamos si el curso es sostenible
    es_sostenible = False
    if competences and competences.lower() != "none":
        es_sostenible = True

    # Añadimos sostenibilidad también en el DataFrame
    nuevo_resultado = pd.DataFrame([{
        "Name": contenido_separado[0],
        "Degree_Master": contenido_separado[1],
        "Code": contenido_separado[2],
        "Competences": competences,
        "Sostenibilidad": es_sostenible,
        "FileName": pdf_file.lower()  # <-- AÑADIDO AQUÍ

    }])
    resultados = pd.concat([resultados, nuevo_resultado], ignore_index=True)
```

Figura 5.23: Validación y almacenamiento de resultados

5.4.5 Actualización de registros en la base de datos

Se establece un proceso de actualización dinámica, evitando duplicaciones y manteniendo la coherencia de la información (imagen :

- Comprobación de registros previos para actualizar la sostenibilidad de una asignatura.
- Generación de consultas SQL dinámicas con Flask-SQLAlchemy.
- Validación final antes de sincronizar los datos en la base de datos.

```python
with app.app_context():
    # Verificar las columnas del DataFrame para asegurarse de que coinciden
    print("Columnas del DataFrame:", resultados.columns)
    for _, row in resultados.iterrows():
        # Verificar que las columnas necesarias existen en el DataFrame
        if 'Code' not in row or 'FileName' not in row or 'Sostenibilidad' not in row:
            print("Columna faltante en la fila, asegurándose de que las columnas existan en el DataFrame")
            continue
        print(f"{row['Code']},{row['FileName']}")


        # Verificación de la consulta
        busqueda_existente = Busqueda.query.filter_by(
            codigo_asignatura=row["Code"],
            nombre_archivo=row["FileName"]  # ahora usamos el FileName directamente
        )

        # Imprimir la consulta SQL generada para depuración
        print("Consulta SQL generada:", str(busqueda_existente.statement))

        # Ejecutar la consulta y verificar si existe el resultado
        busqueda_existente = busqueda_existente.first()

        if busqueda_existente:
            busqueda_existente.sostenibilidad = row["Sostenibilidad"]
        else:
            print(f"No encontrado para {row['Code']} {row['Name']}")

    db.session.commit()
```

Figura 5.24: Actualización de registros en la base de datos

Conclusión El desarrollo del sistema ha permitido una optimización significativa en el análisis de datos académicos, mejorando la identificación automática de competencias de sostenibilidad en guías docentes. La implementación de tecnologías avanzadas como **LLMs, Flask-SQLAlchemy y procesamiento de PDFs** ha permitido la creación de una solución eficiente, escalable y adaptable para la evaluación académica de sostenibilidad.

Selenium

La automatización con Selenium permite simular la interacción con páginas web para realizar tareas como la extracción de datos, la navegación por enlaces y la descarga de documentos. En el contexto de este proyecto, Selenium se ha utilizado para la recopilación de guías docentes desde la web institucional de la universidad, para la obtención de las licitaciones o para buscar el número de artículos publicados, entre otras más. Esto facilita el proceso de obtención de información estructurada.

A continuación vamos a explicar el funcionamiento, usando de ejemplo la evidencia 1_19 Annual Operation Maintenance Percentage:

El propósito principal de la automatización con Selenium es facilitar la recopilación de datos sobre licitaciones, evitando la necesidad de realizar

búsquedas manuales en cada plataforma. A través de este sistema, se optimiza la extracción de información relevante sobre oportunidades de contratación pública, lo que resulta en una mejora sustancial en la gestión de datos.

El flujo de trabajo de la automatización con Selenium sigue una serie de pasos estructurados (código 5.25:

1. Acceso a la Plataforma de Licitaciones: El sistema inicia una conexión segura con los sitios web gubernamentales y plataformas de contratación.

2. Extracción de Información: Se identifican los elementos HTML clave, como formularios de búsqueda, enlaces y tablas de resultados.

3. Filtrado y Normalización de Datos: Se descartan registros irrelevantes y se estructuran los datos obtenidos en un formato estándar.

Para lograr esta automatización, se ha desarrollado un script en Python que emplea Selenium con ChromeDriver en modo headless (sin interfaz gráfica). Algunas de las técnicas utilizadas incluyen:

- Simulación de Interacciones de Usuario: Selenium envía solicitudes y completa formularios de búsqueda automáticamente.

- Extracción de Datos con XPath: Se utilizan expresiones XPath para localizar los elementos HTML relevantes dentro de las páginas web.

- Gestión de Errores y Excepciones: Se han implementado mecanismos de control para garantizar la estabilidad del sistema en caso de cambios en la estructura del sitio web.

- La ejecución de múltiples búsquedas de licitaciones puede generar una alta carga de procesamiento. Para manejar esto, se emplea subprocess.run([...], check=True), lo que permite:

  - Ejecutar cada tarea de forma independiente, evitando bloqueos en el sistema.
  - Control de errores global, deteniendo la ejecución si ocurre un fallo crítico.
  - Optimización del rendimiento, asegurando que el sistema pueda procesar grandes volúmenes de información en paralelo.

La integración de Selenium en este proyecto ha permitido mejorar la eficiencia en la búsqueda de licitaciones, reduciendo la intervención manual y garantizando una recopilación de datos más precisa. Entre los principales beneficios destacan: el ahorro de tiempo en la consulta de licitaciones y la minimización de errores humanos en la extracción de información.

```
# El navegador no muestra su interfaz gráfica.
chrome_options = Options()
chrome_options.add_argument("--headless")  # Habilita el modo headless

# Iniciar el navegador
driver = webdriver.Chrome(options=chrome_options)
driver.get("https://contratacion.ubu.es/licitacion/busquedaAvanzConc.do")

# Esperar a que la página cargue
time.sleep(3)

# # Llenar el campo "Expediente"
# campo_expediente = driver.find_element(By.ID, "expediente")
# campo_expediente.send_keys('UBU/2023/0018')
try:
    # Llenar el campo "Objeto"
    campo_objeto = driver.find_element(By.ID, "ObjContrato")
    campo_objeto.send_keys(mantenimiento)
except Exception as e:
    print(f"Error al encontrar el campo de búsqueda: {e}")



# Marcar la casilla "Ver expedientes de organismos dependientes"
casilla_organismos = driver.find_element(By.NAME, "descendientes")
if not casilla_organismos.is_selected():
    casilla_organismos.click()  # Marcar la casilla si no está seleccionada

# Hacer clic en el botón de búsqueda
boton_buscar = driver.find_element(By.NAME, "busquedaFormAvanz")
boton_buscar.click()


 # Encuentra todas las filas de la tabla con la clase 'resultados'
try:
    WebDriverWait(driver, 10).until(
        EC.presence_of_element_located((By.CSS_SELECTOR, ".resultados"))
    )
    resultados = driver.find_elements(By.CSS_SELECTOR, ".resultados tbody tr")
except TimeoutException:
    print(f"No se encontraron resultados para {mantenimiento}")
    resultados = []

enlaces = []

# Verifica si hay resultados
if len(resultados) == 0:
    print("No se encontraron resultados.")
else:
    for resultado in resultados:
        # Extraer el enlace de la primera columna (suponiendo que está en la primera columna)
        enlace = None
        enlace_elemento = resultado.find_element(By.CSS_SELECTOR, "a")
        if enlace_elemento:
            enlace = enlace_elemento.get_attribute("href")  # Obtener el atributo href del enlace
            enlaces.append(enlace)  # Guardar todos los enlaces en la
# Cerrar el navegador cuando termine
driver.quit()
return enlaces
```

Figura 5.25: Código donde se emplea Selenium

5.6. Testing

Se configuro un servicio de integración continua, de tal forma, que cada vez que se realizaba un commit en el repositorio, se ejecutaba SonarQube; que

analizaba el código duplicado, violaciones de estándares, bugs potenciales, etc.

Se consiguió pasar de tener un 13 % de código duplicado, a tener solamente un 2,3 % de código duplicado, no pasando SonarQube, por tener metodo POST y GET en una única función en lugar de tenerlos en dos funciones separadas.

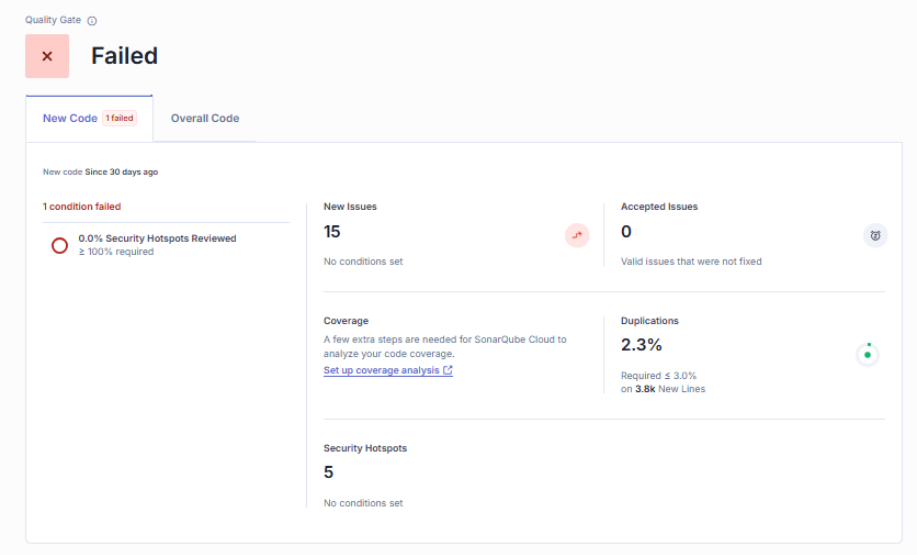

Figura 5.26: Análisis de SonarQube

5.7. Documentación

La documentación del sistema se ha estructurado para garantizar una accesibilidad clara y completa a todos los aspectos técnicos y funcionales del proyecto.

5.7.1 Manual de Usuario

Se han desarrollado manuales de usuario en español e inglés, alojados en la Wiki de GitHub del proyecto. Estos manuales proporcionan una guía detallada sobre el uso del sistema, desde su instalación y configuración hasta la interpretación de los resultados obtenidos.

5.7.2 Manual del Programador

Además, se ha creado un manual del programador, con especificaciones técnicas sobre la arquitectura del sistema, la integración de LLMs (Large Language Models) y la utilización de Selenium para la automatización de evidencias. Este documento facilita la comprensión del código y su futura expansión o adaptación.

5.7.3 Material Audiovisual

Para mejorar la accesibilidad y comprensión del sistema, se han producido videos explicativos en español e inglés , cubriendo los aspectos clave del funcionamiento del software.

- **Videos introductorios en español e inglés**: Explican el uso básico del sistema y sus funcionalidades principales. El vídeo explicativo en español se encuentra disponible en: Video en español.

  Para acceder a la versión en inglés, utiliza el siguiente enlace: Video en inglés.

- **Video detallado en español**: Presenta un análisis técnico más profundo sobre la implementación y optimización del sistema, dirigido a desarrolladores y usuarios avanzados. El vídeo explicativo se encuentra disponible en: Video explicativo.

5.7.4 Plataforma de Documentación

Toda la documentación ha sido organizada de manera estructurada en GitHub Wiki, facilitando la consulta y actualización de contenidos conforme avanza el desarrollo del proyecto.

5.8. Resolución de problemas técnicos

La aplicación no ha podido ser desplegada debido a la utilización de LLMs locales, lo que ha generado errores en la ejecución y ha impedido su correcto funcionamiento. Además, el elevado volumen de archivos generados ha supuesto una limitación adicional para su implementación en un entorno de producción.

Por este motivo, se ha optado por crear una **máquina virtual** que permita gestionar los recursos de manera más eficiente y garantizar la estabilidad del sistema.

6. Trabajos relacionados

En el ámbito de la automatización de la recopilación de evidencias y la generación de información estructurada, existen diversos trabajos que han contribuido al desarrollo de técnicas eficientes en diferentes sectores. A continuación, se presentan algunos proyectos y estudios relevantes en el campo de la generación y gestión de evidencias:

6.1 Aplicaciones de Web Scraping en la Automatización de Evidencias Académicas

Estudios recientes han explorado el uso de Web Scraping para extraer información de portales educativos, facilitando la recopilación de datos sobre asignaturas, publicaciones científicas y eventos relacionados con sostenibilidad. Estos sistemas han demostrado ser efectivos en la estructuración de evidencias dentro de universidades, optimizando la integración con plataformas de evaluación.

Un ejemplo representativo de esta aplicación es el trabajo de *Análisis y generación de evidencias científicas en relación a la salud materna* desarrollado por la **Universitat Autònoma de Barcelona** [6]. Este estudio explora técnicas de extracción automatizada de datos en el ámbito médico y su impacto en la generación de evidencia científica.

6.2 Sistemas de Certificación y Evidencias Digitales en Educación y Ciencia

Los avances en la gestión de evidencias digitales han impulsado el desarrollo de plataformas automatizadas que permiten la certificación de logros académicos y científicos. Algunas investigaciones han utilizado técnicas de

blockchain para garantizar la autenticidad de los datos, mejorando la transparencia en la generación y validación de evidencias universitarias.

El estudio de *Práctica profesional basada en evidencias y organizaciones que aprenden*, publicado en **Scielo** [8], analiza cómo la certificación basada en evidencia digital impacta el aprendizaje organizacional, aportando mejoras en la estructuración de logros profesionales y educativos.

6.3 Conclusión

Los trabajos mencionados reflejan la evolución de los sistemas de recopilación y análisis de evidencias en distintos sectores. La integración de tecnologías como web scraping, blockchain y LLMs ha permitido automatizar y optimizar procesos de evaluación, contribuyendo al desarrollo de herramientas más eficientes en la gestión de información académica y de sostenibilidad.

7. Conclusiones y Líneas de trabajo futuras

Conclusiones

El desarrollo del proyecto ha permitido avanzar en la **automatización de la recopilación de evidencias** en el contexto de UI GreenMetric, optimizando el análisis y clasificación de datos relacionados con sostenibilidad mediante **LLMs y técnicas de Web Scraping**. Entre las principales conclusiones destacan:

- **Eficiencia en la generación de evidencias**: La integración de Large Language Models (LLMs) ha permitido automatizar el procesamiento de grandes volúmenes de información, mejorando la precisión en la identificación de cursos, publicaciones científicas y eventos relacionados con sostenibilidad.
- **Uso de Selenium para extracción de datos**: La implementación de Selenium ha facilitado el acceso dinámico a plataformas educativas y científicas, reduciendo la intervención manual en la recopilación de evidencias.
- Flexibilidad en la configuración del sistema: La arquitectura modular del proyecto permite modificar el modelo de lenguaje utilizado (LLAMA, GPT, entre otros), adaptándose a necesidades específicas.
- **Impacto en la evaluación de sostenibilidad universitaria**: La automatización de evidencias aporta una mejora significativa en la clasificación de instituciones dentro de GreenMetric, reduciendo el margen de error y optimizando la generación de informes.

Se ha aprendido a programar en HTML, CSS y JavaScript, lenguajes que no se han visto en la carrera, ya que para las interfaces no se han utilizado plantillas para su realización, sino que se ha hecho a mano.

También se ha aprendido a utilizar Canvas, para la creación de los videos, aprendido a hacer voz en off.

Líneas de Trabajo Futuras

Para continuar mejorando el sistema y ampliar sus capacidades, se proponen las siguientes líneas de desarrollo:

- **Optimización de la extracción de evidencias con IA avanzada:** Explorar el uso de modelos más sofisticados de NLP para mejorar la identificación de información relevante en documentos institucionales.
- **Integración de bases de datos científicas:** Conectar el sistema con APIs de bases de datos como Scopus y Google Scholar para facilitar la obtención de métricas de impacto en publicaciones.
- **Expansión a otros indicadores de sostenibilidad**: Incorporar nuevos criterios dentro de GreenMetric para evaluar con mayor precisión el compromiso ambiental de las universidades.
- **Mejoras en la interfaz y accesibilidad del sistema**: Desarrollo de una plataforma visual que permita a los usuarios consultar evidencias y realizar ajustes en los criterios de evaluación.
- **Aplicación de técnicas de Blockchain**: Garantizar la integridad y trazabilidad de las evidencias mediante tecnologías descentralizadas para evitar alteraciones en los registros.

Este enfoque permitirá mejorar la eficiencia del sistema y asegurar su escalabilidad para futuras aplicaciones en el ámbito académico y de sostenibilidad.

Bibliografía

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

[2] Canva. Canva - plataforma de diseño gráfico. https://www.canva.com/, 2025. [Internet; consultado junio-2025].

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

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

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

[6] Universitat Autònoma de Barcelona. Análisis y generación de evidencias científicas en relación a la salud materna. https://www.tdx.cat/handle/10803/285365, 2020. [Internet; consultado junio-2025].

[7] Organización de las Naciones Unidas. Objetivos de desarrollo sostenible. https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/, 2025. [Internet; consultado junio-2025].

[8] Red de Revistas Científicas de América Latina. Práctica profesional basada en evidencias y organizaciones que aprenden. https://www.scielo.org.mx/scielo.php?script=sci_arttext&

pid=S0185-26982023000200122, 2023. [Internet; consultado junio-2025].

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

[10] 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. [Internet; consultado mayo-2025].

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

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

[13] Hostinger. Sqlite vs mysql: Análisis a detalle para tu conveniencia. https://www.hostinger.com/es/tutoriales/sqlite-vs-mysql-cual-es-la-diferencia, 2025. [Internet; consultado mayo-2025].

[14] IONOS. Flask vs Django: comparativa de los frameworks de Python. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/flask-vs-django/, 2024. [Internet; consultado mayo-2025].

[15] KeepCoding. Ventajas y desventajas de python. https://keepcoding.io/blog/ventajas-y-desventajas-de-python/, 2023. [Internet; consultado mayo-2025].

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

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

[18] 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 mayo-2025].

[19] 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 mayo-2025].

[20] Red Hat. ¿Qué es una API? https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces, 2020. [Internet; consultado junio-2025].

[21] 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. [Internet; consultado mayo-2025].

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

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

[24] Universitas Indonesia. UI GreenMetric World University Rankings – Official Website. https://greenmetric.ui.ac.id/, 2025. [Internet; consultado diciembre-2024].